Technical University of Crete
Electrical and Computer Engineering Department
Microprocessor & Hardware Laboratory

COMPOSITION AND ANALYSIS OF SECURITY, PRIVACY, AND DEPENDABILITY ON EMBEDDED SYSTEMS

SPD-SAFE: SECURITY, PRIVACY, AND DEPENDABILITY MANAGEMENT ON EMBEDDED SYSTEMS IN SAFETY-CRITICAL APPLICATIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
OF THE TECHNICAL UNIVERSITY OF CRETE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

George Hatzivasilis

Supervisor: Ioannis Papaefstathiou

Chania 2017

This page intentionally left blank.

ABSTRACT

Specifying and measuring the properties that a system provides, plays an important role for risk analysis during the development and management processes. Large organizations, like NASA and the FORD motor company, apply formal methodologies to guarantee that the developed systems fulfil the design requirements and accomplish the desired mission critical goals.

The effectiveness of cyber-attacks raises security as a main system concern. Privacy also becomes important, as high volumes of personal data are processed by modern systems. Still, simple establishment of security and privacy defence mechanisms do not guarantee protection. The dependability of the solution must be also verified. Safety management is crucial, as any incident can lead to potential damage or even personal injury.

Tackling the overall security, privacy, and dependability (SPD) calculation in a practical and systematic manner is difficult. The problem hardens when we deal with a composed system. In the era of pervasive and ubiquitous computing several systems are dynamically composed, with high volumes of heterogeneous embedded and mobile devices exchanging information.

This thesis examines SPD and safety-related issues on Internet-of-Things (IoT) in corporation with Artificial Intelligence (AI). Formal methods are applied for system composition and SPD validation on evaluating systems. Then, the AI process can manage the system in real-time to protect the system itself and the users of the ambient environment. For example, a smart campus setting can assist living conditions during normal operation and counter cyber-attacks. In case of emergency, like fire or earthquake, the AI manages the surrounding smart equipment to assist the timely and safe evacuation of all evacuees. In other smart city scenarios, the system can detect physical tampering of critical railway infrastructure or car accidents and inform the involving authorities to take actions. Information regarding the incident and the passengers' health condition are distributed. The goal is to achieve fast response with adequate rescue means.

The main achievements of this work include a formal framework which describes the SPD properties of a composed system and its sub-components, and how these features are affected by changes in the state-architecture. The implementation can be used for composition verification, security validation, comparison of different system settings, and evaluation of the impact of a change in a system. Moreover, the proposed framework can be used as a middleware for real-time monitoring and management of a system. Technically, the framework is modelled as the reasoning process of JADE agents and ported in the OSGi middleware platform. The network layer is further fortified by a novel trust-based secure routing protocol that provides enhanced security and performance, surpassing the current solutions. Lightweight cryptographic primitives are implemented at the device end to ensure authentication, confidentiality, and integrity.

The deployment is applied and demonstrated in five main scenarios:

- A smart home application to evaluate and manage embedded devices with ambient intelligence capabilities for assisting living.
- An IoT system for precision agriculture deployments with wireless sensor networks (WSNs) for monitoring olive groves or forests while detecting and countering cyber-attacks.

- A smart campus setting for disaster mitigation planning that manages the surrounding smart equipment and assist the timely and safe evacuation of all evacuees in case of emergency, like fire.
- A railway cyber-physical system (CPS) for smart transportation with in-carriage and on-route WSNs that continuously monitor the critical infrastructure for safety-related incidents while providing protection against cyber-attacks.
- A smart vehicle fleet management where the system monitors the underlying vehicles at runtime, protecting against cyber-attacks. The system can also detect car accidents and inform the involving authorities to take actions.

ΠΕΡΙΛΗΨΗ

Ο προσδιορισμός και η μέτρηση των ιδιοτήτων που παρέχει ένα σύστημα διαδραματίζει σημαντικό ρόλο στην ανάλυση κινδύνου κατά τη διάρκεια των διαδικασιών ανάπτυξης και διαχείρισης. Μεγάλοι οργανισμοί, όπως η NASA και η αυτοκινητοβιομηχανία FORD, εφαρμόζουν τυπικές μεθόδους για να διασφαλίσουν ότι τα παραγόμενα συστήματα πληρούν τις απαιτήσεις σχεδίασης και επιτυγχάνουν τους επιθυμητούς στόχους που είναι κρίσιμης σημασίας για την επίτευξη της αποστολή.

Η αποτελεσματικότητα των επιθέσεων στον κυβερνοχώρο αναδεικνύει την ασφάλεια ως κύρια ιδιότητα ενός συστήματος. Η ιδιωτικότητα γίνεται επίσης σημαντική, καθώς τα σύγχρονα συστήματα επεξεργάζονται μεγάλους όγκους προσωπικών δεδομένων. Ωστόσο, η απλή δημιουργία μηχανισμών ασφάλειας και προστασίας της ιδιωτικής ζωής δεν εγγυάται προστασία. Πρέπει επίσης να επαληθευτεί η αξιοπιστία της λύσης. Η διαχείριση της φυσικής ασφάλειας είναι ζωτικής σημασίας, καθώς κάθε περιστατικό μπορεί να οδηγήσει σε πιθανή ζημιά ή ακόμα και σε τραυματισμό.

Η αντιμετώπιση του υπολογισμού της συνολικής ασφάλειας, της ιδιωτικότητας, και της αξιοπιστίας (ΑΙΑ) με πρακτικό και συστηματικό τρόπο είναι δύσκολη. Το πρόβλημα δυσκολεύει όταν αντιμετωπίζουμε ένα σύνθετο σύστημα. Στην εποχή της διάχυτης και πανταχού παρουσίας υπολογιστικής, πολλά συστήματα είναι δυναμικά συντεθειμένα, με μεγάλους όγκους ετερογενών ενσωματωμένων και κινητών συσκευών να ανταλλάσσουν πληροφορίες.

Αυτή η εργασία εξετάζει θέματα ΑΙΑ και φυσικής ασφάλειας που σχετίζονται με το Διαδίκτυο των Πραγμάτων (ΔτΠ) σε συνεργασία με την Τεχνητή Νοημοσύνη (ΤΝ). Τυπικές μέθοδοι εφαρμόζονται για τη σύνθεση του συστήματος και την επικύρωση του επιπέδου ΑΙΑ στα συστήματα αξιολόγησης. Στη συνέχεια, η διαδικασία ΤΝ μπορεί να διαχειριστεί το σύστημα σε πραγματικό χρόνο για να προστατεύσει το ίδιο το σύστημα και τους χρήστες του περιβάλλοντα χώρου. Για παράδειγμα, μια εφαρμογή έξυπνης πανεπιστημιούπολης μπορεί να βοηθήσει τις συνθήκες διαβίωσης κατά τη διάρκεια της κανονικής λειτουργίας και να καταπολεμήσει τις κυβερνο-επιθέσεις. Σε περίπτωση έκτακτης ανάγκης, όπως η πυρκαγιά ή ο σεισμός, η ΤΝ διαχειρίζεται τον περιβάλλοντα έξυπνο εξοπλισμό για να βοηθήσει στην έγκαιρη και ασφαλή εκκένωση όλων των ανθρώπων. Σε άλλα σενάρια έξυπνης πόλης, το σύστημα μπορεί να ανιχνεύσει την φυσική παραβίαση της κρίσιμης σιδηροδρομικής υποδομής ή τροχαία ατυχήματα και να ενημερώσει τις εμπλεκόμενες αρχές για τη λήψη μέτρων. Πληροφορίες σχετικά με το περιστατικό και την κατάσταση της υγείας των επιβατών διανέμονται στις αρχές. Ο στόχος είναι να επιτευχθεί γρήγορη ανταπόκριση με τα κατάλληλα μέσα διάσωσης.

Τα κύρια επιτεύγματα αυτής της μελέτης περιλαμβάνουν ένα τυπικό πλαίσιο που περιγράφει τις ιδιότητες ΑΙΑ ενός σύνθετου συστήματος και των υποσυστημάτων του και τον τρόπο με τον οποίο αυτά τα χαρακτηριστικά επηρεάζονται από τις αλλαγές στην αρχιτεκτονική δομή. Η εφαρμογή μπορεί να χρησιμοποιηθεί για την επαλήθευση της σύνθεσης, την επικύρωση της ασφάλειας, τη σύγκριση των διαφορετικών ρυθμίσεων του συστήματος και την αξιολόγηση των επιπτώσεων μιας αλλαγής σε ένα σύστημα. Επιπλέον, το προτεινόμενο πλαίσιο μπορεί να χρησιμοποιηθεί ως ενδιάμεσο λογισμικό για την παρακολούθηση και διαχείριση σε πραγματικό χρόνο ενός συστήματος. Από τεχνικής πλευράς, το πλαίσιο διαμορφώνεται ως διαδικασία συλλογιστικής των πρακτόρων JADE και μεταφέρεται στην πλατφόρμα

middleware OSGi. Το επίπεδο δικτύου προστατεύεται επιπλέον από ένα καινοτόμο ασφαλές πρωτόκολλο δρομολόγησης βασισμένο σε εμπιστευτικότητα το οποίο παρέχει ενισχυμένη ασφάλεια και επίδοση, ξεπερνώντας τις σημερινές λύσεις. Ελαφριά δομικά στοιχεία κρυπτογραφίας υλοποιήθηκαν σε επίπεδο συσκευής ώστε να εξασφαλιστούν η αυθεντικότητα, η εμπιστευτικότητα, και η ακεραιότητα.

Η ανάπτυξη εφαρμόζεται και επιδεικνύεται σε πέντε κύρια σενάρια:

- Μια εφαρμογή για έξυπνο σπίτι για την αξιολόγηση και τη διαχείριση ενσωματωμένων συσκευών με δυνατότητες περιβαλλοντικής ευφυΐας για την παροχή βοήθειας στις συνθήκες διαβίωσης.
- Ένα σύστημα ΔτΠ για την ανάπτυξη γεωργικών μηχανισμών ακριβείας με ασύρματα δίκτυα αισθητήρων (ΑΔΑ) για την παρακολούθηση των ελαιώνων ή των δασών καθώς και τον εντοπισμό και την αντιμετώπιση των κυβερνο-επιθέσεων.
- Μια εφαρμογή έξυπνης πανεπιστημιούπολης για τον σχεδιασμό μετριασμού καταστροφών που διαχειρίζεται τον περιβάλλοντα έξυπνο εξοπλισμό και βοηθά στην έγκαιρη και ασφαλή εκκένωση όλων των εκτοπισμένων σε περίπτωση έκτακτης ανάγκης, όπως η πυρκαγιά.
- Ένα κυβερνο-φυσικό σύστημα (ΚΦΣ) σιδηροδρόμων για έξυπνες μεταφορές με ΑΔΑ στα μέσα μεταφοράς και κατά μήκος της διαδρομής που παρακολουθούν συνεχώς την υποδομή για συμβάντα που σχετίζονται με την φυσική ασφάλεια, παρέχοντας ταυτόχρονα προστασία από επιθέσεις στον κυβερνοχώρο.
- Μια διαχείριση στόλου έξυπνων οχημάτων, όπου το σύστημα παρακολουθεί τα υποκείμενα οχήματα κατά το χρόνο εκτέλεσης, προστατεύοντας από επιθέσεις στον κυβερνοχώρο. Το σύστημα μπορεί επίσης να ανιχνεύσει τροχαία ατυχήματα και να ενημερώσει τις εμπλεκόμενες αρχές για την ανάληψη ενεργειών.

KEYWORDS – ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Software Measurement Techniques, Dynamic System Composition, Security Validation, Privacy Protection, Dependability Verification, Ambient Intelligence, Pervasive Computing, Internet of Things (IoT), Lightweight Cryptography, Smart Environments, Cyber-Physical Systems (CPS), Embedded Systems, Sensor Networks, Event Calculus, JESS, JADE, OSGi.

Τεχνικές Μέτρησης Λογισμικού, Δυναμική Σύνθεση Συστήματος, Πιστοποίηση Ασφάλειας, Προστασία Ιδιωτικότητας, Επιβεβαίωση Αξιοπιστίας, Περιβάλλων Νοημοσύνη, Διάχυτη Υπολογιστική, Διαδίκτυο των Πραγμάτων, Ελάσσων Κρυπτογραφία, Ευφυή Περιβάλλοντα, Κυβερνο-Φυσικά Συστήματα, Ενσωματωμένα Συστήματα, Δίκτυα Αισθητήρων, Λογισμός Γεγονότων, JESS, JADE, OSGi.

Acknowledgements

I would like to thank my advisor, Dr. Ioannis Papaefstathiou, for allowing me to conduct my research and providing remarkable support, and Dr. Charalampos Manifavas for the excellent collaboration during my PhD studies.

My sincere thanks also go to all my colleagues and co-authors who helped design and/or implement various parts of the work presented herein. Their contributions have been instrumental in reaching the goal I set out to achieve.

I would also like to thank the committee members for offering their valuable expertise and precious time, helping bring this dissertation to its final form.

Finally, I wish to extend my deepest gratitude to my loved ones, family and friends; they stood by me throughout the years of my studies, having to endure my long working hours, nights and weekends, my sometimes-erratic mood due to tiredness and stress, and, most importantly, my absence. Nevertheless, they were always there for me, each helping me in their own way, and for that, I will always be grateful.

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

“You cannot manage what you cannot measure. If you cannot measure it, you cannot improve it”.

Lord Kelvin

COMMITEES

SUPERVISING

Assoc. Professor **Ioannis Papaefstathiou**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Professor **Apostolos Dollas**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Assoc. Professor **Charalampos Manifavas**, *Dept. of Electrical Engineering & Computing Sciences, Rochester Institute of Technology, Dubai, United Arab Emirates*

EXAMINING (JURY)

Assoc. Professor **Ioannis Papaefstathiou**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Professor **Apostolos Dollas**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Professor **Dionisios Pnevmatikatos**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Assoc. Professor **Antonios Deligiannakis**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Assist. Professor **Vasilis Samoladas**, *Dept. of Electrical & Computer Engineering, Technical University of Crete, Greece*

Assoc. Professor **Charalampos Manifavas**, *Dept. of Electrical Engineering & Computing Sciences, Rochester Institute of Technology, Dubai, United Arab Emirates*

Researcher A’ **Sotiris Ioannidis**, *Institute of Computer Science, Foundation for Research and Technology – Hellas, Greece*

PUBLICATIONS

Bellow follow the journal and conference publications stemming from the research work presented in this dissertation, listed in reverse chronological order.

JOURNAL PUBLICATIONS

1. "SCOTRES: Secure Routing for IoT and CPS", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, IEEE Internet of Things Journal (IoT), IEEE, 2017. (DOI: 10.1109/JIOT.2017.2752801) [IF: 7.596]
2. "AmbISPDM: Managing Embedded Systems in Ambient Environment and Disaster Mitigation Planning", G. Hatzivasilis, I. Papaefstathiou, D. Plexousakis, C. Manifavas, N. Papadakis, Applied Intelligence, Springer, August, 2017, pp. 1-21. (DOI: 10.1007/s10489-017-1030-0) [IF: 1.904]
3. "Password-Hashing Status", G. Hatzivasilis, Cryptography, MDPI Open Access Journal, vol. 1, issue 2, number 10, 2017. (DOI: 10.3390/cryptography1020010) [IF: -]
4. "A Review of Lightweight Block Ciphers", G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, C. Manifavas, Journal of Cryptographic Engineering, Springer, vol. 7, 2017, pp. 1-44. (DOI: 10.1007/s13389-017-0160-y) [IF: 1.88]
5. "Lightweight Authenticated Encryption for Embedded On-Chip Systems", G. Hatzivasilis, G. Floros, I. Papaefstathiou, C. Manifavas, Information Security Journal: A Global Perspective, Taylor & Francis, vol. 25, issue 4-6, 2016, pp. 151-161. (DOI: 10.1080/19393555.2016.1209259) [IF: 0.38]
6. "Software Security, Privacy and Dependability: Metrics and Measurement", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, IEEE Software, IEEE, vol. 33, issue 4, 2016, pp. 46-54. (DOI: 10.1109/MS.2016.61) [IF: 2.19]
7. "RtVMF – A secure Real-time Vehicle Management Framework with critical incident response", K. Fysarakis, G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, IEEE Pervasive Computing Magazine (PVC) – Special Issue on Smart Vehicle Spaces, IEEE, vol. 15, issue 1, 2016, pp. 22-30. (DOI: 10.1109/MPRV.2016.15) (DOI:) [IF: 3.25]
8. "A survey of lightweight stream ciphers for embedded systems", C. Manifavas, G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, Security and Communication Networks, Wiley, 21 December, 2015, issue 9, pp. 1226-1246. (DOI: 10.1002/sec.1399) [IF: 1.067]

CONFERENCE PUBLICATIONS

1. "SecRoute: End-to-End Secure Communications for Wireless Ad-Hoc Networks", G. Hatzivasilis, I. Papaefstathiou, K. Fysarakis, I. Askoxylakis, 22nd IEEE Symposium on Computers and Communications (ISCC 2017), IEEE, Heraklion, Crete, Greece, 03-06 July, 2017.
2. "Real-time management of railway CPS", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, 5th EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems (ECyPS 2017), IEEE, Bar, Montenegro, 11-15 June, 2017.
3. "Lightweight password hashing scheme for embedded systems", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, I. Askoxylakis, 9th WG 11.2 International Conference on Information Security Theory and Practice (WISTP), IFIP, Heraklion, Crete, Greece, 24-25 August, 2015, Springer, LNCS, 9311, pp. 249-259. (DOI: 10.1007/978-3-319-24018-3_17)
4. "RT-SPDM: Real-time Security, Privacy & Dependability Management of Heterogeneous Systems", K. Fysarakis, G. Hatzivasilis, I. G. Askoxylakis, C. Manifavas, Human Aspects

of Information Security, Privacy and Trust (HCI International 2015), 2-7 August, 2015, Los Angeles, CA, USA, Springer, LNCS, vol. 9190, pp. 619-630. (DOI: 10.1007/978-3-319-20376-8_55)

5. "ModConTR: a Modular and Configurable Trust and Reputation-based system for secure routing", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, 11th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'2014), IEEE, Doha, Qatar, 10-13 November, 2014, pp. 56-63. (DOI: 10.1109/AICCSA.2014.7073179)
6. "An efficient anti-malware intrusion detection system implementation, exploiting GPUs", I. Papaefstathiou, A. Bilanakos, K. Fysarakis, G. Hatzivasilis, C. Manifavas, International Conference on Advanced Technology & Sciences (ICAT 2014), 12-15 August, 2014, Antalya, Turkey, pp. 1-9.
7. "DSAPE – Dynamic Security Awareness Program Evaluation", C. Manifavas, K. Fysarakis, K. Rantos, G. Hatzivasilis, Human Aspects of Information Security, Privacy and Trust (HCI International 2014), 22-27 June, 2014, Creta Maris, Heraklion, Crete, Greece, Springer, LNCS, vol. 8533, pp. 258-269. (DOI: 10.1007/978-3-319-07620-1_23)
8. "A reasoning system for composition verification and security validation", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, N. Papadakis, 6th IFIP International Conference on New Technologies, Mobility & Security (NTMS 2014), IFIP, 30 March – 2 April, 2014, Zayed University, Dubai, UAE, pp. 1-4. (DOI: 10.1109/NTMS.2014.6814001)
9. "ULCL: an Ultra-Lightweight Cryptographic Library for embedded systems", G. Hatzivasilis, E. Gasparis, A. Theodoridis, C. Manifavas, Measurable security for Embedded Computing and Communication Systems (MeSeCCS 2014), within the 4th International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2014), 7-9 January, 2014, Lisbon, Portugal, pp. 11-18. (DOI: 10.5220/0004900602470254)
10. "Embedded systems security challenges", K. Fysarakis, G. Hatzivasilis, K. Rantos, A. Papanikolaou, C. Manifavas, Measurable security for Embedded Computing and Communication Systems (MeSeCCS 2014), within the 4th International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2014), 7-9 January, 2014, Lisbon, Portugal, pp. 1-10. (DOI: 10.5220/0004901602550266)
11. "Lightweight cryptography for embedded systems – a comparative analysis", C. Manifavas, G. Hatzivasilis, K. Fysarakis, K. Rantos, 6th International Workshop on Autonomous and Spontaneous Security (SETOP 2013), in conjunction with the 18th annual European research event in Computer Security (ESORICS 2013) symposium, 12-13 September, 2013, RHUL, Egham, U.K., Springer, LNCS, vol. 8247, pp. 333-349. (DOI: 10.1007/978-3-642-54568-9_21)
12. "Building trust in ad hoc distributed resource-sharing networks using reputation-based systems", G. Hatzivasilis, C. Manifavas, 16th Panhellenic Conference on Informatics (PCI 2012), IEEE, 5-7 October, 2012, Piraeus, Greece, pp. 416-421. (DOI: 10.1109/PCi2012.28)

AWARDS

1. "SCOTRES: Secure Routing for IoT and CPS", G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, Cyber Security Awareness Week (CSAW'17), Applied Research Competition – Best Research Paper Finalist, NYU Trandon School of Engineering, Abu Dhabi, UAE, 09-11 November, 2017.
2. "RtVMF – A secure Real-time Vehicle Management Framework with critical incident response", K. Fysarakis, G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, Cyber Security Awareness Week (CSAW'16), Applied Research Competition – Best Research Paper

Finalist (Top 5), NYU Trandon School of Engineering, Abu Dhabi, UAE, 10-12 November, 2016.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

In the ongoing 4th Industrial Revolution, embedded devices exchange high volumes of information regarding the users and the ambient environment. The protection of these assets is crucial as successful attacks can harm the business operation or cause injuries and even depths. Novel and efficient security, privacy, and dependability (SPD) management on Internet-of-Things (IoT) and safety-critical applications utilizing Artificial Intelligence (AI) is possible. The thesis contributes in this domain with the following achievements:

- **A systematic and practical measurement methodology of SPD:** risk assessment methods evaluate a distinct system in order to capture the main protection properties that are provided.
- **System composition and SPD validation on evaluating systems in the IoT domain:** after assessing the individual SPD metrics of each subsystem, we examine the defense level that is accomplished by the finally composed setting.
- **Combination with AI:** the AI process manages the system in real-time to protect the system itself and the users of the ambient environment.
- **Safety-critical reasoning and disaster mitigation planning:** Except from defending cyber-assets, our solution safeguards the physical environment of the application and the users as well. The system can detect safety-critical events, like fire or earthquake, and protect the users by imposing disaster mitigation plans and strategies.
- **The network layer security is enhanced with a secure and energy-efficient routing mechanism:** the trust-based secure routing scheme improves the performance and load-balancing of wireless ad hoc networks during normal operation, while protecting from several threats, like jamming, blackhole, On-Off, and Sybil attacks.
- **Novel lightweight cryptographic mechanisms are deployed at the embedded device end:** lightweight cryptography protects the digital assets that are maintained at the end nodes of the system. The implemented primitives provide the main cryptographic functionality for authentication, confidentiality, and integrity, and offer enhance functionality for authenticated encryption and password-hashing.
- **Five application scenarios:** a smart building example presents the main methods for SPD evaluation and composition validation. A smart campus setting demonstrates the capability of our solution to manage safety-critical events and assist the users to timely evacuate the campus in case of fire. A precision agriculture application presents the ability to detect and counter a high variety of cyber-attacks. The intelligent transportation scenario further examines the protection of the critical infrastructure of a railway system. Finally, the social mobility scenario demonstrates the full potentials of the proposed system, where smart vehicles and on-road equipment are managed at runtime in order to deploy a smart city application and implement a real-time emergency response service in case of car crashes.

1.1 The 4th Industrial Revolution

In the era of the Internet of Things (IoT) massive numbers of interconnected embedded devices are deployed in critical infrastructures and cyber-physical systems (CPS). The Internet of Everything has an economic impact of more than 14 trillion by 2020. Pervasive and ubiquitous computing devices, featuring sensors and actuators, are already deployed in a variety of domains (smart grid, city infrastructure, transportation, residential/home automation, industrial systems, military, and healthcare, among others [1], [2], [3]). Figure 1 depicts the industrial revolution milestones, based on DFKI GMBH data.

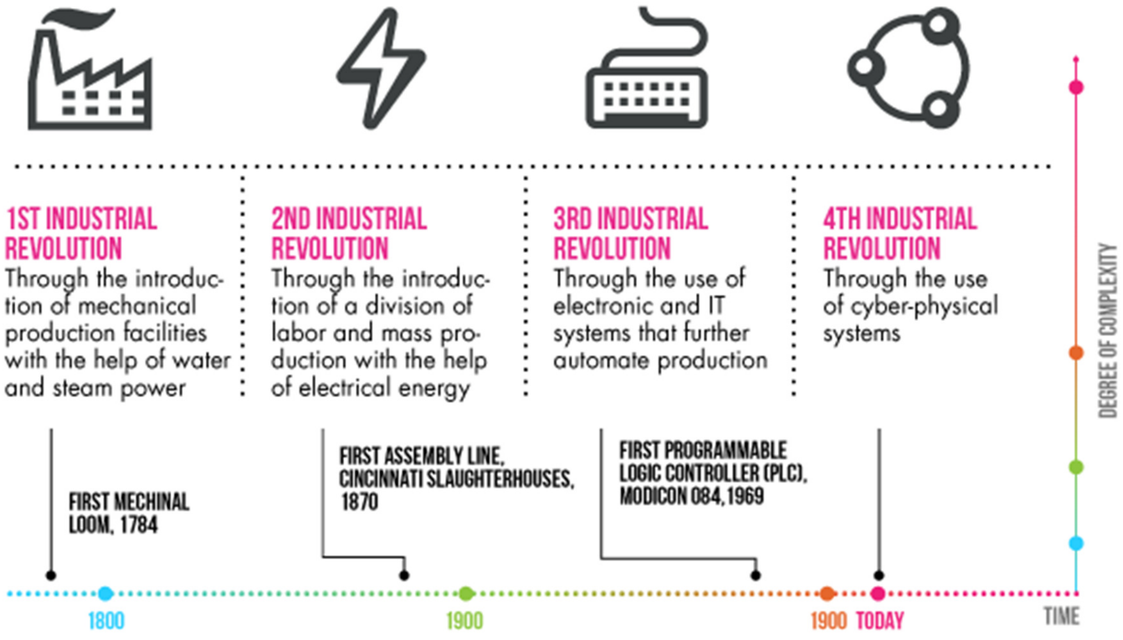

1ST INDUSTRIAL REVOLUTION
Through the introduction of mechanical production facilities with the help of water and steam power
2ND INDUSTRIAL REVOLUTION
Through the introduction of a division of labor and mass production with the help of electrical energy
3RD INDUSTRIAL REVOLUTION
Through the use of electronic and IT systems that further automate production
4TH INDUSTRIAL REVOLUTION
Through the use of cyber-physical systems
FIRST MECHINAL LOOM, 1784
FIRST ASSEMBLY LINE, CINCINNATI SLAUGHTERHOUSES, 1870
FIRST PROGRAMMABLE LOGIC CONTROLLER (PLC), MODICON 084, 1969
1800
1900
1900
TODAY
TIME
DEGREE OF COMPLEXITY

Figure 1 From industry 1.0 towards 4.0: the 4th industrial revolution of CPS

In 2015, 10 billion IoT devices were deployed and their number is going to reach the 34 billion devices in 2020. Figure 2 illustrates the connectivity forecasts of IoT devices, based on Cisco data[1]. Research and industry struggle to integrate this evolving technology and the exponential growth of IoT.

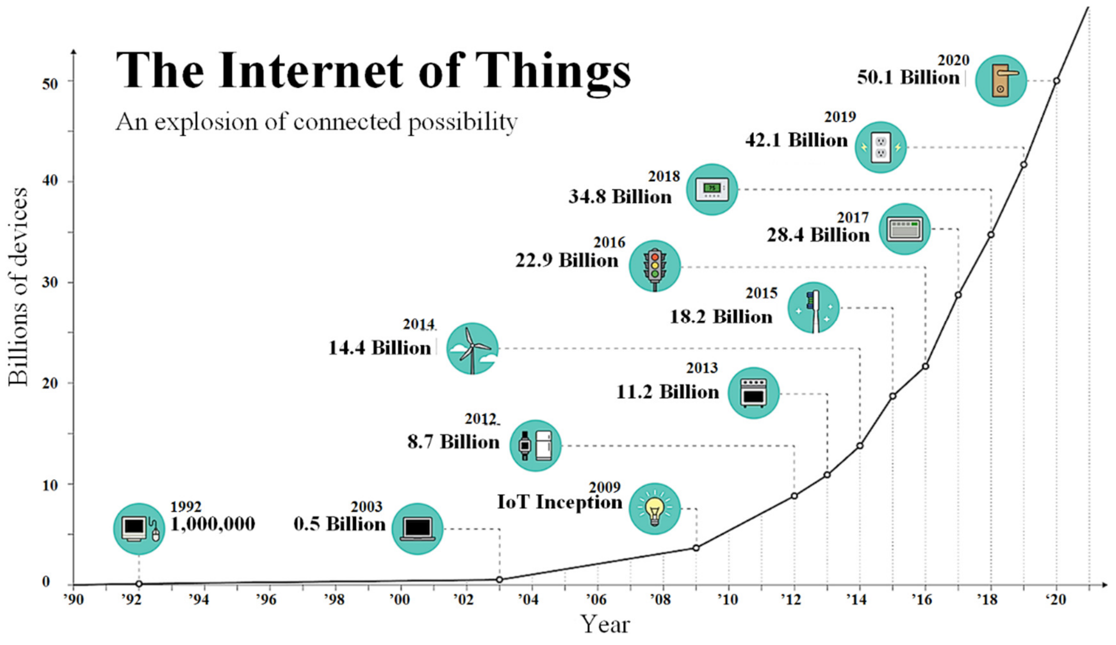


Figure 2 The IoT connectivity forecasts based on Cisco data

[1] The Connectivist: http://www.theconnectivist.com

Existing networking [3] and security mechanisms [4] are adapted to handle the vast population of IoT devices. Seamless and higher level machine to machine (M2M) and human-to-machine (H2M) interactions, are another important requirement in order to effectively monitor and manage the infrastructure, allowing the use of its full potential. Typically, end-users do not possess the skills to configure and setup the devices that may be found in smart environments; in large-scale deployments, individually setting up devices is not even feasible. From the perspective of implementers, there is a need for rapid development and deployment, while simultaneously tackling issues of scaling and inherent limitations in terms of resources (CPU, memory, power etc.). However, at its current state, the ubiquitous computing landscape is segregated, consisting of numerous proprietary solutions, which are typically incompatible with each other. This makes setting up, managing and securing a smart device ecosystem, significantly challenging. Various *"IoT protocols"* proposed by researchers aim to address these issues, while standardization initiatives try to guarantee interoperability, through the wide and structured deployment of the proposed mechanisms.

1.2 Motivation

Embedded systems (ESs) permeate our lives in various forms, from avionics to e-textiles, automobiles, home automation and wireless sensor nodes. In terms of their physical size, they range from miniature wearable or sensor nodes (i.e. motes) to large industrial deployments of programmable logic controllers (PLCs).

The various intrinsic and application specific characteristics of ESs complicate the task of guaranteeing the security, namely handling the confidentiality, integrity, and availability aspects of their applications and the data they handle. Their characteristics habitually include resource constraints (namely computational capabilities, storage capacity, memory, and power), dynamically formulated, remotely-managed networking and even unattended operation in hostile environment and time-critical applications. Therefore, while securing networked computer systems is not a novel concern, the techniques developed for personal and enterprise systems are often unsatisfactory or even inapplicable in the case of embedded devices.

In addition to the above, ES applications often feature direct interaction with the physical world. This further differentiates ES security as a related incident in a critical application may lead to asset damage or even personal injury and death. In [5] researchers demonstrated that it is feasible to manipulate all critical sub-systems in modern automobiles using a wireless-enabled MP3 player connected to the vehicle's embedded control network. The presented attacks include accessing the brake controller, thus disabling or forcibly activating the brakes and consequently compromising the safety of the driver and passengers, as well as injecting malicious code to erase any evidence of tampering after a crash.

Furthermore, since ESs are often responsible for vital, time-critical applications, a delay or a speedup of even a fraction of a second in system's response or reaction could have dire consequences. A recent and widely publicized example of such a case is the worm Stuxnet, a highly specialized malware which was designed to target the specific Siemens Supervisory Control and Data Acquisition (SCADA) systems installed in Iran's uranium enrichment infrastructure. The purpose of the worm was to take control of the PLCs causing periodic variations in the uranium enrichment centrifuges' rotor speed, thus destroying the devices. Indeed, because of Stuxnet, Iranian scientists were forced to replace approximately 1000 centrifuges over a few months when, prior to the attack, normal failure rates were approximately 800 per year [6]. The abovementioned differentiating factors in embedded computing security must be taken under consideration during ESs design and implementation.

However, estimating a system's properties is not a trivial task, especially in the era of IoT where several heterogeneous technologies are integrated and work together. Metrics for evaluating aspects like security and performance are becoming an integral feature in system development. They provide a quantitative assessment of a system's compliance with the application's requirements. This also enables comparisons between different system settings based on objective factors, facilitating the selection of the ones that are more appropriate with respect to the design and/or specific deployment criteria. Moreover, it enables metric-driven management procedures for real-time systems.

1.2.1 Standardized Measurement Methodologies

Several organizations have defined methodologies to capture the security properties of a computer system. The most popular of them include: TCSEC (USA's Orange book), ITSEC (Europe's Orange book), CTCPEC (Canada's Orange book), Common Criteria (universal Orange book), SSE-CMM, NIST FIPS-140 series, and NIST SP 800-55.

Regarding software security metrics, they are context sensitive, architecture dependent, and their aggregation may not always lead to increased protection. A good metric must be:

- Quantitative
- Objective
- Obtainable
- Repeatable
- Inexpensive
- Based on a formal method and
- Has ground truth and a time dimension

Measurement is an observation that results in information about a quantity. The goal is to reduce uncertainty in decision making. The right metrics help us to identify the greatest risks in our system and reduce the critical vulnerabilities.

Metrics are going to become an integral component of future systems due to the increased consumers' demands for better security and the government involvement for more effective regulation and public protection.

1.2.2 System Security Properties and Composition

Safety and system dependability are crucial, as any incident can lead to potential damage or even personal injury [7]. In 2005, an early smart home installation in Greece was liable for the death of its owner. The home was equipped with magnetic blinds in the windows and the outer doors that was managed by the ambient system. At night, the blinds were locked as an anti-theft mechanism. When the house caught fire, the blinds remain close, trapping the resident inside.

Privacy is also an important factor. Home is commonly sensed as a private space and is protected by current regulations [8]. Service providers do not intentionally collect private data on the server-side. However, the absence of adequate protection on the device-side (e.g. [9], [10], [11], [12], [13]) means that data collection might be relatively easy, even by attackers with low skills.

Security is always a concern. ENISA surveys the cyber-security challenges for building a smart home system [8]. The need for security is still underestimated by both vendors and users. Attacks target the weakest element to capture credentials and enable more powerful attacks, taking control of the smart infrastructure.

Specifying and measuring the properties that a system provides, plays an important role for risk analysis during the development and management processes. Considering the aforementioned aspects of these applications, context-aware technologies are required in order to specify the ambient environment conditions ([14], [15], [16], [17]). Semantic ontologies model the ambient domain, presenting it in machine-readable format. Then, logic languages can process this knowledge to reason about and react to the current state of the ambient environment. Software agents are a form of intelligent entity that incorporates these features, providing reasoning and management capabilities on the underlying system (e.g. [7], [16], [17], [18]).

However, as multi-agent systems (MAS) cooperate and exchange information conflicts may arise regarding diverse beliefs about the real system state. Such affairs are originated either by contrasting goals or by inconsistent pieces of knowledge ([19], [20], [21]). The problem occurs in settings where the agents are autonomous and strongly motivated by their own interests or in heterogeneous systems where agents with different reasoning and computational capabilities coexist in a dynamic environment. Thus, conflict resolution processes are required to check and regulate a disputation.

1.3 The Framework

Motivated by the above, this thesis presents SPD-Safe – a formal framework which describes the security, privacy, and dependability (SPD) properties of a composed system and its sub-components, and how these features are affected by changes in the state-architecture. SPD-Safe can be used for composition verification, security validation, comparison of different system settings, and evaluation of the impact of a change in a system. Moreover, the proposed implementation of the SPD-Safe framework can be used as a middleware for safety-critical systems, featuring real-time monitoring and management of the underlying system component.

SPD-Safe addresses three main issues of system development:

- Extension of the composition and management processes with metrics allowing measurement of the SPD level of the final system. We consider that metrics will become an integral feature in system development as they can be used to compare different system settings and help quantify the impact of an incoming change in an existing system.
- Verification that different components can be composed and form the system under examination. This is especially useful for heterogeneous systems (systems composed of different types of devices) and systems-of-systems (systems that are composed from several sub-systems).

- Validation of the properties that hold as the result of the composition of two systems. This feature is imperative for assuring that the composed system works properly and features the claimed specifications.

Technically, SPD-Safe is modelled as the reasoning process of JADE agents and is ported in the OSGi middleware platform. The deployment is applied in several smart applications to evaluate and manage embedded devices with ambient intelligence capabilities for assisting living conditions and mission-critical features.

SPD-Safe consists of two main components: **CompoSecReasoner** and **AmbISPDM**. The former models the composition validation and SPD verification, while the latter uses the management theory of the ambient environment and manages the system in real-time. CompoSecReasoner reasons about the system composition and SPD validation while AmbISPDM models the SPD and safety related management strategy and the system's administration through real-time technologies. The whole reasoning process is transformed into a JADE agent's reasoning behavior and implemented as a multi-agent epistemic reasoner with conflict resolution capabilities.

CompoSecReasoner models technologies and protocols as attributes. A SPD analysis is performed for every developed attribute and a relevant SPD is defined for each protection level that is provided. Then, the evaluated metrics and properties of the system are determined, including how they may be affected by the underlying attributes at runtime. This information is encoded in the evaluation functions of the relevant metrics and properties. For the system composition, verification, and security validation, the system is considered as a set of components of four layers (i.e. node, network, middleware, overlay) and the components of each layer are composed of sub-components of the adjacent lower layer. When composition is successful, the composition verification and security validation process are revisited.

AmbISPDM implements the core reasoning engine. The AI reasoning process is an event-based model checker that extends Event Calculus (EC) and is implemented in the Jess rule engine. The context theory is aware of the SPD aspects of the underlying embedded system (through CompoSecReasoner) and the safety goals and strategies.

Regarding the network layer, the secure routing protocol SCOTRES safeguards the underlying devices from malicious activity, while enhancing load-balancing and performance. It consists of five metrics that rate the nodes' participation and evaluate the correct operation. The topology metric enhance load-balancing and protects isolated and distant nodes from being cut off. The energy metric is aware of the nodes' energy levels and protects the network from energy dissipation and the related attacks. The channel-health metric detects jamming attacks in the wireless channel and constraints their effects. The core reputation and trust metrics evaluate each node cooperation and fair use of the networks resources and counters the main attacks on routing.

1.4 Use Cases

This subsection describes in brief the use cases where the SPD-Safe is deployed. The totally five scenarios cover a high variety of application settings with each one of them focusing on specific system aspects, highlighting the different capabilities of the proposed framework.

1.4.1 Smart Home
Smart homes install electronic building equipment to ease the living conditions of the residents and decrease the resource consumption (i.e. electric energy or water). They involve, among others, automation and controls for lighting, air condition, heating, and physical security. Investments on smart home appliances, such as washers, ovens, and refrigerators, increase the offered functionality.

IoT-enabled devices monitor the surrounding environment and control the underlying equipment. Ambient intelligence functionality assists living conditions and enables e-health services, like independent living of elders and personalized healthcare applications. Physical protection can be also achieved by implementing anti-theft mechanisms in the physical entry/exit points of the house (i.e. doors and windows). In terms of cyber-security, privacy protection is a main concern while security bridges can expose the resident's sensitive personal information.

SPD-Safe can be used for the real-time monitoring and management of the embedded systems that are applied in a smart home setting (Section 7). Moreover, the proposed solution can verify that the different technologies are composable and estimate the security and privacy properties of the finally composed system.

1.4.2 Precision Agriculture
Novel farming management techniques involve observation, measurement, and response to field variability in crops. Through electronic devices that are deployed in the farm environment, precision agriculture enhances decision making support and enables the on-line management of the cyber-physical equipment.

Bad weather conditions can disrupt communication or damage the installed devices. The wide operational area hardens the physical inspection of the system, with dependable and green solutions becoming imperative. Moreover, such settings are vulnerably to attacks due to the open communication medium.

SPD-Safe is deployed on a rural application that monitors olive oil groves through WSNs (Section 9). The overall setting enables the monitoring of the environmental parameters of the grove and the management of the IoT-enabled devices through cloud. The protection mechanisms are energy efficient and provide protection against both outer attackers (i.e. jammers) and compromised equipment (e.g. blackhole or Sybil attacks).

1.4.3 Smart Campus
Modern campuses run on new technologies that have fundamentally changed how educational institutions work. Advanced energy consumption monitoring, unified collaboration and management of mobile office equipment (e.g. mobile phones and tablets), and intelligent surveillance services that enhance safety, are some of the most common achievements, bringing great convenience, efficiency, and flexibility to campus life.

For SPD-Safe, smart campus is an enhanced version of the smart home setting, including many buildings and higher diversity of ambient services (Section 10). The system eases educational and smart classroom services (e.g. classroom lighting or wireless access to printers), while the electronic services of the campus, like the e-mail server, are safeguarded against on-line attacks.

Except from these ambient functionality at normal operation, the system can detect cases of emergency, such as fire. Then, the deployed equipment is managed by an AI process in order to support disaster mitigation plans and assist the safe evacuation of all evacuees. The emergency services are timely informed and coordinated to accomplish the most efficient and effective actions.

1.4.4 Smart Transportation

Smart transportation includes passenger services and critical infrastructure protection. The main goals are higher efficiency and productivity, green operation, and improved safety and security.

Railways constitute a main mean of mass transportation, with signaling systems directing the train traffic. They use several telecommunication technologies between the track and the trains in order to enable management and infrastructure control. Security is imperative while the protection of the critical infrastructure and distant shelters is important and is achieved via intelligent surveillance services.

SPD-Safe controls a railway CPS and counters cyber-attacks that try to compromise the infrastructure equipment or disrupt communication (Section 11). Periodic malfunctioning is also considered. WSNs installed in-carriage or along the train's route, defend carriages that contain dangerous cargo or prevent crashes with objects that block the train's route (e.g. cars that are stuck on railway crossing). Smart cameras are also deployed in order to enhance the physical security of the critical infrastructure.

1.4.5 Social Mobility

Modern smart cities collect data from environmental and social sensors and provide innovative and sustainable transport and mobility services. Through data mining techniques, useful information is determined, like traffic flow, parking lots, bad road conditions or repair activities, and accidents. In the smart transportation context, social mobility stands for the movement of people and resources in an informed manner that accomplishes positive social outcomes. Information and communication technologies facilitate the organization between goods, services, and human capital.

Smart vehicles comprise electronic and electromechanical equipment in conjunction with computer-controlled devices and radio communications. They sense their environment and exchange information with the driver and other vehicles or smart city services. However, hackers can exploit these techniques and perform on-line cyber-attacks, taking control of the underlying equipment and threatening the passengers' safety.

The abovementioned smart transportation scenario is extended in specialized CPSs (Section 12) that include smart vehicles, like cars. SPD-Safe preserves user privacy and location. Cyber-protection against modern types of attacks on smart cars is imposed. The system provides mobile and real-time services to drivers and passengers. The main achievement is the materialization of a critical incident response operation that informs emergency authorities in case of car-crashes in order to provide timely services with appropriate and adequate rescue means.

1.4.6 Main Demonstration Goals of Each Use Case

The five case studies examine different aspects of system modelling and management. Table 1 summarizes the main demonstration goals of each scenario.

Table 1 Demonstration goals of the five use cases

Use case	Demonstration goals
Smart home	- Assist living conditions - Modelling of the SPD metrics and the composition validation and verification processes
Precision agriculture	- Enhanced security analysis against cyber-attacks by outsiders or compromised nodes - Basic integration and communication with cloud
Smart campus	- Management of the composed system at normal operation (i.e. assist smart classroom services) - Protection against main cyber-attacks - Assist disaster mitigation planning in case of emergency
Smart transportation	- Protection against enhanced cyber-attacks - Surveillance and physical protection of critical infrastructure
Social mobility	- Retrofitting in existing vehicles - Protection against modern cyber-attacks in smart vehicles - Real-time critical incident response in case of road accidents

2. Background & Related Work

2.1 System Composition

In secure system design it is imperative to prove that a system is correct and works properly. This proof of concept can be modeled with UML or with a logic-based method. UML-based approaches are constrained as they mainly describe static instances of a system. On the other hand, logic-based approaches are more appropriate for dynamic system verification as they can better model the different states of a system and how they change as time progresses.

Model-based approaches meet these challenges of system design. Their key advantage is the abstraction of the application domain. Thus, they provide a system designer with a set of concepts that are tailored for a specific application domain. The Model Integrated Computing (MIC) [22] can perform formal analysis, verification, validation, and generation of embedded systems. It uses the Domain Specific Modeling Language (DSML), which is defined by UML meta-models, to achieve the abstraction of the application domain.

Several implementations of DSMLs frameworks for embedded systems design are proposed. General composition methods are examined in [23], [24]. For modeling and analysis of systems, the Society of Automotive Engineers (SAE) standardized the architecture analysis and design language (AADL) [23]. AADL models software and hardware layers of embedded systems. It assures about real-time requirements, like schedulability, power consumption, safety, and fault tolerance. In AADL, a system is modeled as system, hardware, and software components. An interface declaration defines a component's externally observable attributes, like ports that are used for connecting with other components. An implementation declaration defines the components' inner structure, like the component's sub-components and how they are connected. A formalism was presented in [24], that abstractly translates the system configurations and allow system verification through model checking. Similarly, the Simple Modeling Language for Embedded Systems (SMoLES) [24] utilizes DSML to offer a concise syntax for constructing embedded systems. It models a system's components as objects that communicate and synchronized with each other. The component is the building block of the system and contains input/output ports for receiving/sending data. An assembly contains components and describes how components are connected.

However, those general composition methodologies do not consider the special requirements of secure system design. In [25], a framework is proposed that incorporates security modeling into existing embedded system design tools. Security extensions of DSML are embodied to a meta-model composition mechanism. As a case study, the Security Model Analysis Language (SMAL) [25] is integrated to SMoLES to construct a combined system; called SMoLES-SEC. SMAL is a DSML that models access control policies for embedded systems. This constraint expressiveness capability of SMAL, bounds the reasoning capabilities of the resulting SMoLES-SEC. Moreover, this approach fails to determine the security properties that hold after the composition process and the final security level of the system.

In the area of action and change (Knowledge Representation & Reasoning) event-driven model-based methods have been proposed to better describe the behavior of a dynamic system in a formal manner. A system is dynamic if it changes due to the execution of an action/event as time progresses. Such formal methods are described in [26], [27]. In [26], a semantic domain for model-based analysis is introduced that verify the preemptive schedulability and composition of distributed real-time embedded systems. As a case study, the framework is applied on a mission-critical avionics system. They use the Uppaal model checker to model a system and perform schedulability analysis. A composition event between two components is expressed as a set of priorities and dependencies that must be granted. Systems are formed as timed automata and their specifications are described in timed computational tree logic (TCTL) (temporal logic that formalizes statements about system models). In [27], an interface-based event-driven system is proposed for modularization and composition of distributed systems. A component's reactions are described as interfaces (a variation of Harel statecharts). Then, an interface encapsulates the component's behavior and reflects the component's composition capabilities. A meta-model is proposed for expressing the structural, control, and runtime aspects of distributed event systems. Such event-driven frameworks can describe dynamic systems and reason about their properties as time progresses.

Theoretical work in secure system composition is discussed in [28], [29]. In [28], the authors provide a set of composition theorems for proving the security of protocols under composition and standard stand-alone definitions of security. In [29], the methods of universal composability and concurrent general composition are studied and applied on public-key models in scenarios with no trust setup. However, the theoretical work presented in [28], [29] is impractical for immediate appliance in a real dynamic system with many states and technologies. Yet, the derived security and composition results can be taken into account by the implemented tools for more robust formal documentation.

2.2 Security Validation

Two main types of validation techniques are proposed in system design. The prior-composition validation type imposes rules or policies during the composition process (e.g. security-by-contract and runtime enforcement). If these rules and policies are satisfied, the composition is performed. Thus, the composed system should exhibit some desirable properties. The post-composition validation type analyzes the system after the composition to determine the final properties that hold (e.g. security). In some cases, both types of validation can be applied.

The composition verification methods [25], [26], [27], which were discussed in the previous subsection, apply prior-composition validation techniques that are based on pre-conditions, policies, or interface checking. Other prior-composition methods are discussed in [30], [31]. In [30], a method for deriving the security properties of a composed system using contracts is described. Compositional security Contracts (CsC) refer to the security properties that must be accomplished between two components at compositional-level. Several CsCs establish a System-level security Contract (SsC), which summarizes the functionalities that a composed system offers to the external world. Then, the method reasons about the final security properties and goals that are achieved by the system. In [31], a framework for runtime enforcement mechanisms is proposed. The examined system is modeled as an automaton and the framework traces a sequence of events and states to determine if constraints specified by security policies are violated at runtime. The enforcement mechanism performs two main functionalities: permits legal operation without changes and amends illegal operations.

Tools for post-composition security validation are discussed in [32], [33]. In [32], a visualization and modeling tool for security metrics and measurement management is described. The authors apply threat modeling to evaluate metrics at the architectural level of a composed system. Then, they perform a hierarchical presentation of security metrics that is based on decomposition techniques of measured security objectives and determine the overall security properties. In [33], the authors present Multihost multistage Vulnerability Analysis (MulVAL) – an end-to-end framework for analyzing network vulnerabilities. MulVAL uses the language Datalog to model the Red Hat bugs that are reported in Open Vulnerability Assess Language (OVAL – a formal vulnerability definition language). The framework scans for software programs that are running on the network nodes and alerts about the relevant bugs that are reported in OVAL. The tool was tested on a real network to detect policy violations caused by software vulnerabilities.

Nonetheless, these tools ([30], [31], [32], [33]) manage only security validation methods. None of them deals with the composition process.

Theoretical work is presented in [34], [35]. In [34], the authors propose a framework for deriving secure protocols that are based on two core ones, whose security properties are known. They use formal composition, refinements, and transformations to derive families of new protocols that extend the functionality of these two core protocols. Thus, the new protocols inherit the known security properties. Their method is restrictive and deduces only protocols with similar structure and at least the same level of security. In [35], a formal validation of automated policy refinement is presented. The method is applied in the management of network security systems. Policy hierarchies for model-based management are formally validated and propagating from an abstract level to its immediate lower neighbor. Finally, two theorems are proved to ensure compliance between the abstract policies and the actual system behavior.

2.3 Quantifying Security

To quantify the security of a composed system, one has to quantify the security of its sub-components and then measure the overall security. A main method in quantifying the security level in such systems is the medieval castle approach [36]. A system is modeled as a medieval castle with security doors, which are the target of an attacker. An attack is successful if it breaks through the doors and reaches a treasure room inside the castle – the resources that are protected by the security mechanisms. The difficulty in passing through the security doors and reaching a castle's inner treasure room indicates the castle/system security level. Each door is a security mechanism of the system and its resistance in attacks is measured by relevant metrics. The medieval castle approach is appropriate for measuring the security level of a static instance of a system. Thus, it cannot be straightforwardly applied to dynamic system analysis.

Newer methods are the attack surface metric [37] and the multi-metric approach [38]. The method in [37] considers that an attacker uses a system's methods, channels, and data items to attack the system. These features at the entry and exit points of a system that are potentially exploiting by attacks, determine the attack surface of the system. Quantitative metrics are used to calculate the damage potential-effort radio and define the attack surface size. In [38], multiply metrics are combined to form the systems security. The different metrics are modeled as a triple vector of $\langle Security, Privacy, Dependability \rangle$. Each metric takes measurable values from a set. All metrics' values are mapped in the SPD formation to easy the comparison and composition of different metrics. Then, a hybrid meta-heuristic fuzzy aggregation and composition method is applied to evaluate and visualize the security parameters of the system.

In [39], the authors go one step further and propose a metric-driven management framework for e-health digital ecosystems. They focus on digital ecosystems on chronic diseases and suggest three types of metrics. The risk-driven security engineering and assurance metrics are mainly set at deployment-time to offer early visibility of security effectiveness. The continuous security monitoring metrics are assessed at operational-time and contribute to security correctness evaluation, better systematization, and traceability between the different metric results and product requirements. The automated adaptive decision-making metrics are also evaluated at operational-time and enable higher quality security effectiveness understanding in operational security monitoring and future version of the target ecosystem. The framework supports continuous security monitoring and automated or semi-automated security decisions based on these metrics.

Theoretical work is presented in [40]. The authors propose a model for formal description and analysis of security metrics and formalize a number of security metrics. Thus, they investigate relations to define more secure metrics. In contrast to the aforementioned approaches (e.g. [36], [37], [38], [39]) that mainly consider the structural factors of a system in order to evaluate security, the authors conclude that a metric can be good or bad based on the attacker model.

2.4 Discussion

Regarding system composition and security validation, in this work, SPD-Safe is presented – a framework which has the ability to estimate the new level of security when a change takes place in the system. SPD-Safe adopts an event and model-based approach to model a dynamic system and its properties. No such methodology has been implemented so far that combines the three aforementioned features. The proposed methodologies are either general composition methods ([23], [24], [41]) or deal with specific security issues ([25], [28], [29], [32], [33], [34], [36], [37], [38]) none of which can meet our requirements.

In most of these methodologies (e.g. [25], [23], [24], [37], [38]), the composition verification is modeled as inputs/outputs or interfaces of the components that are composed. After this process, security properties are validated based on theorems or polices (e.g. [28], [33], [34]). Methods for quantifying the security level are applied to the final system state afterwards (e.g. [36], [37], [38]).

For the composition verification process, SPD-Safe introduces a novel structure which is called *"operation"* (see Section 5). An operation is a set of actions (e.g. technologies and protocols) that two components must be able to perform in order to be composed and function together. The abstraction context that is provided by the operation formalism is adequate for modeling a real system in contrast to the input/output and the interface approaches that are very modeling-intensive in the domain of heterogeneous embedded systems.

In security validation, SPD-Safe utilizes both types of validation by imposing rules during the composition process and model-checking after it. The security validation process is derived by a set of functions that are invoked for each examined property. The process is enriched with the metrics feature.

The modeling of the metrics is partially based on a combination of the medieval castle approach [36], the attack surface [37], and a variation of the multi-metric formalism [38] (see Section 3, Section 4, and Section 5). The formal definition of the metrics is based on the theoretical work that is conducted in [40]. Moreover, we accede to the conclusion in [40] that a system's security is strongly affected by the attacker model. Thus, apart from the structural analysis of a system, we also consider the parameters of the environment setting and the attacker model.

3. Security, Privacy, and Dependability Metrics

This section presents the main measurement methodologies and the proposed SPD multi-metric approach. The application on the social mobility scenario and the evaluation of the relevant software components are also described.

3.1 Core Metrics

Software measurement is important in risk management during software development [42]. For example, NASA applies software inspection throughout development to corroborate design and requirements fulfillment and to enhance product dependability, especially in mission-critical systems [43]. Measurement techniques that comply with the relevant standards help indicate the validity of the solution and the business [44].

In an industrial setting, it is imperative to develop measurement methods that evaluate many attributes at the same time and present useful information about the overall system. For example, Ford has established a research methodology for modeling automotive software security, privacy, usability, and reliability [45]. This methodology analyzes the software functionality on the basis of these properties and assigns qualitative values (low, medium, or high) to each one. Ford has applied this methodology to real automotive software, such as for an antilock braking system and a valet key.

One particularly useful approach is to measure software's *attack surface*, which indicates its susceptibility to attack. Examples of metrics for this include the *attack surface metric* (ASM) [37] and the *Relative Attack Surface Quotient* (RASQ) [46]. The ASM assembles the related parameters to estimate the attack surface, assuming that the smaller the surface (the exposure of the system's resources), the higher the security. Not all resources contribute equally to the attack surface; attackers benefit more from specific resources (for example, attacking applications with root privilege to gain higher access rights to the exploited assets). To account for this, the ASM estimates the damage potential–effort ratio for each entry or exit point. Similarly, Microsoft models the RASQ to quantify the relative attackability of the Windows server OS platforms. This approach assigns a vulnerability level to each potential exploit point. RASQ is the summation of the effective attack surface values of all root attack vectors (system features that can affect security). Attack biases are aggregated, representing the risk of exploiting each attack vector. The metric is also validated against real exploits, on the basis of relevant reports from CVE (Common Vulnerabilities and Exposures) and US-CERT (US Computer Emergency Readiness Team) databases. However, up to this point, attack surface metrics such as these have considered only security, not privacy or dependability. Moreover, these metrics reveal only a system's attackability, not its protection level.

To alleviate this situation, this thesis proposes a multi-metric methodology to evaluate and quantify software system security, privacy, and dependability (SPD). This SPD evaluation is based on the ASM, but it has enhanced the security specification on the basis of the Common Criteria Evaluation Methodology (CEM) [47] and the *Open Source Security Testing Methodology Manual* [48]. For privacy, it employs ISO/IEC standards 29100 [49] and 27018 [50], and the European Data Protection Directive 95/46/EC. The dependability is derived from IEC standard 60300 [51]. The basic surface calculation is also enhanced with a risk analysis of the applied protection mechanisms' effectiveness. The metric calculates the real protection level of the potentially attackable points.

3.2 The SPD Multi-Metric Approach

The SPD multi-metric employs two methodologies. First, we determine the SPD surface, similarly as in the ASM and RASQ. At this point, we perform relevant security analysis employing those two metrics, extended by the privacy and dependability perspectives. The results reveal the system's attackability and the potential damage.

Second, we employ systematic risk assessment to determine the effectiveness of the system's protection mechanisms. The overall analysis estimates the real protection level by aggregating the risk analysis of the attackable points and the protection mechanisms' effectiveness.

In effect, the SPD multi-metric measures the separation between the potential threats and the protected assets. Figure 1 illustrates the SPD methodology's main features.

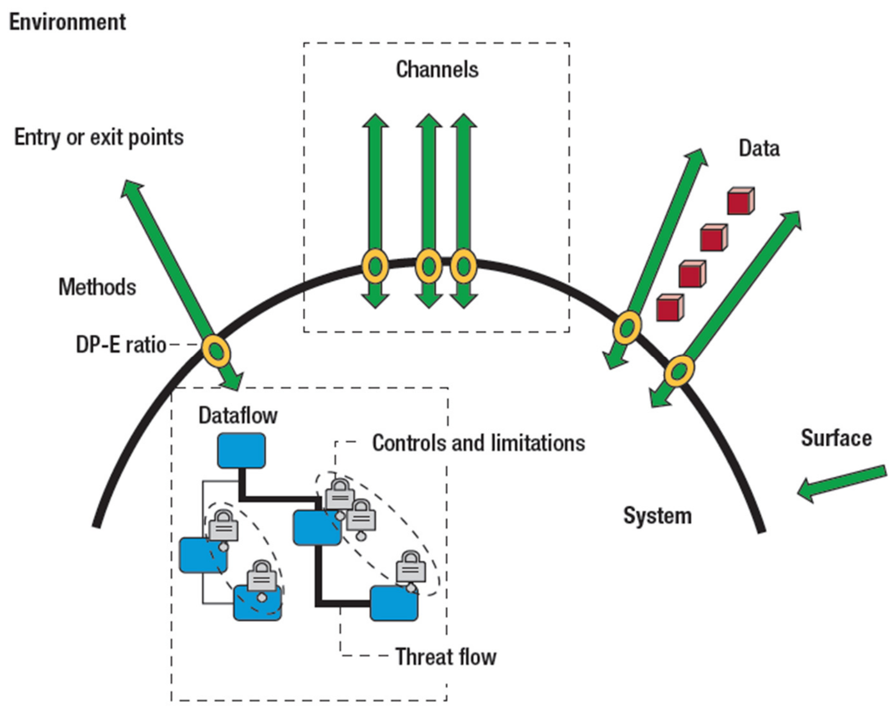


Figure 3 The SPD methodology

3.2.1 The SPD Surface

When analyzing the SPD state, we determine whether possible threats exist. In the case of security, a threat is the potential for abuse of protected assets resulting from malicious human activity [47], [48]. In the case of privacy, a threat is a malicious or non-malicious event that affects protected assets related to personal identifiable information (PII) [49], [50]. In the case of dependability, a threat is a non-malicious event in the fault chain (fault-error-failure) [51].

Not all threats contribute equally because not all interactions are equally likely to cause harm or cause the same damage. So, we identify the possible *threat flows* (TFs) that each involved system resource introduces. Then, we assign a *damage potential–effort* (DP-E) ratio to each TF. The DP-E ratio (which is based on the ASM) takes into account:

- The potential damage to the system in case of disclosure and
- The effort the attacker is willing to devote to attack the resource.

The higher the potential damage or the lower the effort, the higher the TF's contribution to the attack surface.

We estimate the damage potential and effort in tandem because these two factors are usually related in real attack scenarios. For example, an attacker might be willing to spend effort in gaining high privilege levels to gain more access rights and cause more damage.

As we mentioned before, our SPD surface considers privacy and dependability in addition to security. Although the general notion that attackers would target a high-privilege method seems correct in many cases, attackers could exploit a lower-privilege method that has access to sensitive personal data. Nevertheless, according to surface metrics that focus just on security, the latter method would be less likely to be attacked, with a lower contribution to the attack surface. Similarly, for dependability, a mission-critical method or a component with high dependency in the overall system architecture might be more valuable to an attacker than a method with administrator access rights.

Individually analyzing the three SPD properties could also produce inaccurate conclusions regarding attackability because an attacker could formulate an attack involving all three properties. So, the SPD surface evaluates the aggregated effect of the three properties, resulting in more accurate system analysis and better design.

The DP-E ratio for each TF integrates three underlying ratios for security, privacy, and dependability. For security-related interaction, the damage potential is determined by evaluating the method's privilege and the attacker's effort is defined by the method's access rights. For privacy, the damage potential is based on the PII type and the effort is based on the actuator type that processes that data. For dependability, the potential damage is derived from the component's criticality and the effort is derived from the component's dependency on other components. Table 2 summarizes the DP-E ratio parameters and the values that are used in this study.

Table 2 DP-E (damage potential–effort) ratio parameters and values for the SPD multi-metric.

SPD property	Damage potential	Effort
Security	Privilege [47] -Root: 5 -Debugger: 4 -Authenticated user: 3	Access rights [47] -Administrator: 4 -Authenticated user: 3 -Anonymous user: 1 -Unauthenticated user: 1

Privacy	PII (personal identifiable information) type [49] -Sensitive personal data: 5 -Personal data: 4 -Statistical data:1	PII actuator [49] -PII principal: 4 -Contracted PII processor: 3 -Third party: 2
Dependability	Criticality [51] -Mission critical: 4 -Business critical: 3 -Business operational: 2 -Business supporting: 1	Dependency level [52] -Coupling: 4 -Outgoing dependency: 3 -Incoming dependency: 2 -Independent operation: 1

The DP-E ratio for a TF is the summation of the relevant DP values divided by E:

$$TF_{DP-E} = \left(S_{DP} + P_{DP} + D_{DP}\right) / \left(S_E + P_E + D_E\right),$$

The system surface is the summation of the underlying DP-E ratios:

$$Surface_{sys} = \sum TF_{DP-E} .$$

3.2.2 The SPD Value

Determining the SPD value involves analyzing the software's *porosity*, *controls*, and *limitations*.

Porosity

Porosity refers to the set of TFs for which interaction between assets and threats is possible. We break down each TF into *access*, *trust*, and *complexity pores* representing the effect on security, privacy, or dependability, respectively.

Controls

Controls are the means to influence a threat's impact when interaction occurs. They constitute the protection mechanisms that aim to prevent interactions of assets and threats, and enhance their separation.

There are two types of controls. *Interactive controls* directly affect operations when interaction occurs. *Process controls* indirectly affect operations by protecting assets after interaction and creating defensive strategies.

The SPD multi-metric approach considers 12 interactive-control types (five for security, four for privacy, and three for dependability) and 14 process control types (four for security, six for privacy, and four for dependability), as described in the relevant standards. Table 3 and Table 4 define these controls.

Table 3 SPD Interactive controls.

SPD property	Aspect	Control	Description
Security	Confidentiality	Authentication	Challenges credentials on the basis of identification and authorization.
		Resilience	Retains protection in the case of corruption or failure.
	Integrity	Subjugation	Ensures that interactions occur according to a defined process, removing freedom of choice and liability in the case of disclosure.
	Availability	Continuity	Retains interactivity in the case of corruption or failure.
		Indemnification	Involves a contract between the asset owner and an interacting entity. It might include warnings as a precursor of legal action and public legislative protection.
Privacy	Collection	Consent	Involves the PII (personal identifiable information) principal's freely given, specific, and informed agreement to the PII's processing. The PII should not be disclosed or shared with third parties without the PII principal's consent.
		Opt-in	Involves a process or policy in which the PII principal agrees explicitly to the PII's processing, before relevant consent.
	Access	Indentifiability	Results in identifying, directly or indirectly, the PII principal on the basis of a given set of PII. It might include complete indentifiability, pseudonymization, or anonymity.
		Notification	Notifies PII principals that their data is being collected.
Dependability	Reliability	Survivability	Provides degraded but useful operations that are acceptable to users for a specified period during a failure.
		Performability	Ensures how well the system will perform over a specified period in the presence of faults.
	Maintainability	Removal during use	Records and removes faults through the maintenance cycle, after production.

Table 4 SPD Process controls.

SPD property	Aspect	Control	Description
Security	Confidentiality	Confidentiality	Ensures that a processed asset is not known outside the interacting entities.
	Integrity	Integrity	Ensures that the interacting entities know when an asset has been altered.
		Nonrepudiation	Prevents the interacting entities for denying their role in an interaction.
	Availability	Alarm	Notifies that an interaction is occurring or has occurred.
Privacy	Collection	Fairness	Ensures that the PII is collected, used, or disclosed for only the appropriate purposes.
	Access	Challenge compliance (accountability)	Ensures that PII principals can hold PII processors accountable for adhering to all privacy controls.
	Usage	Retention	Ensures that PII that is no longer required is not retained, to minimize unauthorized collection, use, and disclosure.
		Disposal	Provides mechanisms for disposing of or destroying PII.
		Report	Reports that an interaction regarding PII is occurring or has occurred.
		Break or incident response	Manages a PII breach.
Dependability	Reliability	Tolerance	Ensures the required functionality's delivery in the presence of faults.
	Maintainability	Forecasting	Predicts likely faults so that they can be removed or their effects can be circumvented.
	Safety	Prevention	Prevents faults from being incorporated into a system. This is accomplished through good development methodologies and implementation techniques.
		Removal during development	Verifies a system so that faults are detected and removed before the system goes into production.

Limitations

Limitations restrict a control from working properly or prevent the separation of an asset and a threat. There are six types of limitations. The first three directly affect porosity. *Vulnerabilities* increase access pores. They are flaws or errors that prevent access of authorized entities, allow privileged access to unauthorized entities, or allow unauthorized entities to hide assets or themselves. *Disclosure* increases trust pores and represents intentional or unintentional revelation of PII. *Exposure* increases complexity pores. It refers to unjustifiable actions, flaws, or errors that enable direct or indirect visibility of assets.

Two limitation types affect the two control types and their overall effect, influencing all three SPD properties. *Weaknesses* constitute flaws or errors that disrupt, reduce, or nullify interactive controls' positive contribution. Similarly, *concerns* affect the process controls' flow or execution.

The last limitation type is *anomalies*, which include unidentifiable or unknown elements that cannot be observed during normal operation. Because such conditions cannot be controlled, a proper audit should notify developers of any anomalies. An anomaly might obstruct all SPD properties, so we evaluate its effect on each property.

Determining the SPD value

A system's SPD value is a triple vector representing the degree of asset protection in accordance with the controls or the reduction of the threats' impact. Each vector's value ranges from *0* to *100*. However, an SPD value of $\langle 100, 100, 100 \rangle$ does not guarantee perfect protection. It just denotes that all the required protection mechanisms are properly placed for all potential interaction flows and are safe, on the basis of the current set of known limitations – for example, from Common Vulnerabilities and Exposures (CVE) bulletins or US Computer Emergency Readiness Team (US-CERT) advisories. Higher values could denote wasted or redundant protection mechanisms and higher development or operational costs.

Applying many similar controls to protect the same operation type does not provide in-depth protection because defeating one of them usually leads to defeating them all. Increasing an asset's SPD value requires applying different and safe controls. Clear dataflows and business processes are prerequisites for high SPD values.

3.3 SPD Metric Evaluation

The ASM details the methodology for identifying direct or indirect interaction between assets and threats and calculating a pore's DP-E ratio. The porosity value is based on the summation of the TFs, called $Porosity_{Base}$.

3.3.1 Controls

Controls are placed to restrict porosity. Although they are mapped to a specific SPD property, they influence all three of them. So, the final calculation considers the aggregated contribution of each property's controls. For every pore, the applied controls are identified. To achieve perfect coverage of a pore, at least one instance of each control must be mounted. All controls contribute equally to a pore's coverage. The perfect coverage of interactive and process controls is denoted as:

$$PerfectCoverageInteractive = Porosity_{Base} \times 12,$$

$$PerfectCoveragePassive = Porosity_{Base} \times 14.$$

3.3.2 Limitations

The justification of a limitation is a risk decision. You can either control the limitation somewhat or accept the damage it can cause (For example, the cost to fix the problem might not be justified by the potential damage).

Vulnerabilities, disclosure, or exposure

To measure the contribution of vulnerabilities, disclosure, or exposure, a calculation is performed based on the *attack potential* [47]. This function combines the attacker's expertise, motivation, and preference to exploit particular limitations and attack specific assets, and the resources he or she is willing to devote. This function is further extended to include not only malicious attacks but also the general concept of a threat, as it was described earlier.

To analyze a potential threat, five factors are considered:

1. **Elapsed time:** The time required to identify and exploit the limitation, measured in days, weeks, or months.
2. **Specialist expertise:** The technical expertise required (e.g. layperson, proficient, or expert).
3. **Knowledge of the target:** Knowledge of the target's design and operation (e.g. public, restricted, sensitive, or critical information about the target).
4. **Window of opportunity:** A factor related to the elapsed time. An attacker might require considerable access to the target to successfully exploit the threat without detection.
5. **Resources:** The IT hardware or software or other equipment required (e.g. standard, specialized, or bespoke equipment).

On the basis of these factors, a potential threat is characterized as basic, enhanced basic, moderate, high, or beyond high. This approach does not consider every possible circumstance but gives a good indication regarding the protection level in accordance with standard ratings.

The factors' individual ratings could indicate high protection. However, one limitation could lead to another that is easier to exploit, resulting in a lower overall rating. So, for the SPD calculation, the value assigned to each of these three limitations is computed as a weighted summation of the relevant individual ratings: V_V, V_D, and V_E, respectively. These values are divided by the pore type's attackability:

$$L_V = V_V \Big/ \sum Access \times TF_{DP-E} \,,$$

$$L_D = V_D \Big/ \sum Trust \times TF_{DP-E} \,,$$

$$L_E = V_E \Big/ \sum Complexity \times TF_{DP-E} \,.$$

Weaknesses and concerns

The summation of the missing interactive and process controls divided by *PerfectCoverageInteractive* and *PerfectCoverageProcess* determines the impacts of the weaknesses and concerns, called L_w and L_c, respectively. As with controls, these two limitation types influence all three SPD properties.

Anomalies

Anomalies alone do not impact SPD. Their influence is evaluated only in the presence of other limitations, as their summation divided by the summation of *PerfectCoverageInteractive* and *PerfectCoverageProcess*, called L_A.

3.3.3 Vector Calculation

The final SPD vector calculation is the perfect separation (*100*) minus the weighted summation of the limitations:

$$S = 100 - [W_V \times L_V + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

$$P = 100 - [W_D \times L_D + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

$$D = 100 - [W_E \times L_E + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

where W_* is each limitation type's threat bias (similar to RASQ).

3.4 Application of the SPD Multi-Metric Approach

To assist SPD evaluation, we implemented a user-driven scanning tool. It uses two Eclipse plug-ins: CallGraph[2] for parsing C/C++ programs and TACLE (*T*ype *A*nalysis and *C*al*l* Graph Construction for *E*clipse) [53] for parsing Java programs. The plugins create call-graphs of programs presenting the function or method calls, timing, or usage.

We automated most evaluation steps, leaving a limited set of decisions for the user. The tool identifies the high-level entry and exit points and resources (that is, it exploits the Java aspect modifiers – public, protected, default, or private – of an object's methods and parameters), constructing the surface. The user clarifies which of these elements contribute to porosity and indicates the relevant controls. The tool precomputes and then maintains the potential-threat analysis of the limitations and the calculation of the relevant values. Finally, the user maps the limitations to pores and controls, and the tool calculates the SPD value. Designers can easily adjust the system configurations and estimate the best SPD for different settings.

The tool evaluates the protection level of prototype configurable embedded software in the five scenarios of this thesis (subsection 1.4). The final case is given as an example for the

[2] CallGraph: https://marketplace.eclipse.org/content/callgraph-viewer

demonstration of the SPD evaluation because this setting incorporates all the protection mechanisms that are implemented in this study. The same results are used in the rest applications.

3.4.1 SPD Evaluation of the Social Mobility Setting

The social mobility scenario with smart cars of the EU project nSHIELD [151] includes a wireless-sensor-network-equipped smart-city infrastructure that gathers ambient and cybersecurity information, two smart vehicles, and a back-end command and control (C&C) center that aggregates collected information and communicates with all entities. See Section 7 – Section 12 for more details.

nSHIELD is constructing a real-time critical-incident-response system in which the C&C center manages the system automatically to deal with emergencies (e.g. cyberattacks or car crashes). The center employs AI with contextual information regarding the configuration states. The AI calculates the current setting's SPD level and manages the system entities on the basis of scenario events.

Ideal SPD settings might not always be possible. For example, in applications requiring low latency (such as vehicle-to-vehicle communication), smart vehicles might use simpler cryptographic mechanisms, inducing lower security. Also, smart vehicles might denote a larger (inexact) relevant location to obstruct movement tracking and enhance driver privacy. In the case of a crash, a vehicle's dependability will decrease according to the damage (if components critical for dependability break).

The prototype software frameworks are configured at runtime to comply with the specified SPD goals. The functionality includes policy-based access control; trust-based routing; secure communication and storage; GSM (Global System for Mobile Communications) communication and location-based services; and monitoring, alerting, and management services. There are four software frameworks and 14 configurations.

Table 5 details the prototypes, surface sizes, and SPD values. Surface analysis alone was inadequate to determine the protection level. All the framework configurations that provided low protection produced a smaller surface than the safer settings. For example, the secure-storage framework presented the smallest surface when no encryption was deployed. Although the configurations that applied cryptography produced a larger surface owing to the additional functionality, they were not the most unsafe settings. Our SPD methodology effectively derived the protection level and the state of the security, privacy, and dependability.

Table 5 The evaluated software frameworks of the SPD multi-metric under the SPD-Safe framework.

Prototype	Description	Configuration	Attack surface size	SPD value
Secure storage	Storage that locally (at the embedded-device level) maintains protected information regarding user data and log files.	No encryption	2,745.67	<20, 17, 8>
		Lightweight cryptography	3,177.87	<40, 35, 15>
		Mainstream cryptography	3,194.88	<80, 70, 30>
Trust-based ad hoc routing	A secure routing protocol and an intrusion detection	Simple routing	3,808.64	<30, 20, 20>
		Direct trust	4,546.56	<45, 30, 25>

	system for wireless sensor networks that monitor ambient information of the smart city	Direct and indirect trust	4,867.23	<90, 60, 50>
Policy-based access control	A management framework for access control in heterogeneous embedded-system networks, based on policies	Unencrypted communication	3,201.54	<10, 30, 10>
		Authentication	4,137.83	<50, 50, 50>
		Authenticated encryption	4,467.22	<80, 70, 80>
		Web Services Security	4,803.36	<90, 80, 90>
Location-based services	A framework for providing location-based services through GSM (Global System for Mobile Communications) with relevant security and privacy protections	Anonymous services	3,367.55	<20, 80, 20>
		Pseudonymity	3,541.13	<30, 70, 30>
		k-anonymity	3,972.46	<40, 85, 45>
		Authenticated, accurate, and secure communication	4,493.31	<80, 60, 80>

The proposed solution will be offered as a service for secure system management and embedded-system development. Besides smart vehicles, the application of the overall methodology in other domains, such as e-health and the smart grid, can leverage the protection level of existing settings. Moreover, cloud-computing and IoT architectures enable new ways of interaction and impose challenges for software measurement. The SPD methodology could be extended to measure interconnected embedded devices that communicate information to the cloud.

4. SPD METRICS COMPOSITION

This section presents the main methodologies for composing metrics and calculating the overall protection level of a system-of-systems.

4.1 MEDIEVAL CASTLE APPROACH

To quantify the security of a composed system, one has to quantify the security of its sub-components and then measure the overall security. The main method to quantify the security level is the medieval castle approach [36]. The system is modelled as a medieval castle with security doors, which are the target of the attacker. An attack is successful if it breaks through the doors and reaches a treasure room inside the castle – the resources that are protected by the security mechanisms. The difficulty in passing through the security doors and reaching a castle's inner treasure room indicates the castle/system security level. Each door is a security mechanism of the system and its resistance in attacks is measured by relevant metrics.

Figure 4, illustrates the four basic composition settings of the medieval castle approach. The black center of a circle represents the protected assets. The circle denotes a protection mechanism and the dashes act as the relevant doors (attack points).

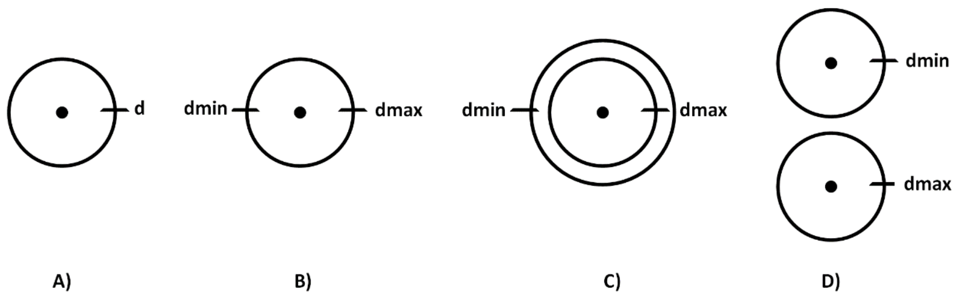


Figure 4 The basic composition settings of the medieval castle approach

Consider d as the overall defence level of the castle. Castle Figure 4.A) is the simplest castle setting, with one door protecting the assets. The defence of the castle is equal with the protection level of this single door. The rest of the castles have two doors (d_{min} and d_{max}). Assume that d_{min} is weaker than d_{max}. The castle in Figure 4.B has two doors at the same layer, enabling simultaneous attacks on both of them. The castle's protection level is equal to or weaker than the protection level of the weaker door (we assume that attacking d_{max} may further weaken the defence at d_{min}). The castle in Figure 4.C has two doors at sequential layers. The attacker must break through both doors to reach the assets. The defence here is stronger than Figure 4.B and is at least as good the security of d_{max}. In the setting Figure 4.D, there are two castles that an attacker may target. An attack is successful if the attacker reaches at least one asset. However, the distance of the two castles is considered too long to allow simultaneous attacks. The overall protection is the interval of d_{min} and d_{max}.

Therefore, three main composition operations are defined:

- **AND-operation:** $0 \leq d \leq d_{min}$

- **OR-operation:** $d_{max} \leq d \leq 1$
- **MEAN-operation:** $d_{min} \leq d \leq d_{max}$

The aforementioned analysis is applied to more complex settings, resembling complex castles of any number of doors. Attack trees are constructing based on the main composition operations, representing the effort to attack the system from the outer layers of defence. More details regarding the medieval approach and the underling formal analysis can be found in the original paper [36].

4.2 The Composed SPD Multi-Metric

In this subsection we sketch the composed SPD multi-metric evaluation that it is proposed in this study. The SPD methodology is utilized as the core metric to evaluate the protection level of the individual system components. Then, the basic security notion of the medieval castle approach is extended with the SPD feature. The SPD multi-metrics are embodied to the medieval castle methodology. They act as a formal and systematic way to calculate the defence of the castle's *SPD doors*. The outcome is the total SPD of the currently composed system.

The composed metric advances the medieval castle composition approach and captures the requirements of two main real-time SPD modelling principles. The medieval castle models only static instances of a system. The proposed methodology evaluates dynamic systems with decomposition/composition events altering the system's structure and SPD properties at run-time. For example, two sub-systems may be secure (in some sense), but their composition need not be secure (in the same sense). The composed SPD multi-metric applies pre- and post-composition validation. At first, it determines if the composition can be succeeded. Then, it derives the composition's outcome and its side-effects. Similarly, for decomposition.

The protection mechanisms at the different layers of a real system may interact with each other. This notion is not properly handled by the castle evaluation. In the proposed methodology, missing controls of the SPD multi-metric can be covered by relevant controls that are placed in the adjacent outer layers, forming the added protection of the interacting sequential defence layers. For example, an insecure data exchange application can be safeguarded by a secure communication service that is imposed at the network layer providing confidentiality, integrity, and authentication. The final SPD represents the overall control coverage that is achieved by the current setting.

The methodology is applied in the ambient intelligence domain of a smart home. In the usual setting, indoor embedded devices monitor ambient parameters and exchange information. The user can have access to the system through a computer or even a smart phone. We model a smart home with embedded devices that assist living conditions. The home deploys a laptop and some smart devices (i.e. TV, surveillance cameras, air-condition, fridge, printer). A wireless router connects the laptop and the devices in a LAN and enables external communications through Internet. Figure 5 illustrates the system composition of the smart home.

At first the SPD multi-metric of each individual system component (laptop, embedded devices and router) is calculated. Then, the components are integrated under the medieval castle composition operations (the composed SPD calculation is presented in the following section).

The devices are composed to a LAN under MEAN-operations, the LAN is composed with the router under an OR-operation and the router is composed to Internet under an OR-operation. This main setting is represented in Figure 5.A). An attacker can attack the home router through Internet and then hit the devices in the LAN. In Figure 5.B) a user connects the LAN with his smart phone (MEAN-operation). The SPD is evaluated similarly with Figure 5.A). In Figure 5.C) the user connects the LAN, as in Figure 5.B) (MEAN-operation), but now the phone has its own mobile Internet connection (AND-operation with Internet). The attacker has now a second option to attack the system through this mobile Internet connection.

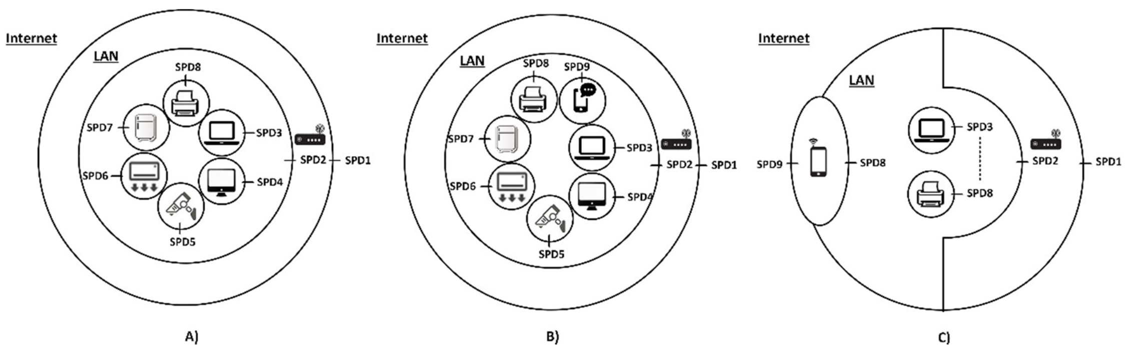


Figure 5 Demonstration of the basic composition operations of the medieval castle approach for a smart home

5. SPD Evaluation and Composition – Smart Home Use Case

This section describes the implementation of the proposed composed SPD multi-metric by the framework component CompoSecReasoner. The application in the main smart home installation is also presented.

5.1 Main Components

CompoSecReasoner is a practical framework which models the dynamic nature of a system where new sub-systems (for which some SPD attributes are known) are integrated into the core pre-existing one. In this work, the secure composition process of systems-of-systems is examined with heterogeneous embedded systems that are composed to form a new system. A formal methodology is applied to prove that the utilized technologies are really composable under a metric-driven composition policy and the composed system fulfils the claimed SPD properties.

CompoSecReasoner is applied for secure system design and provides formal verification of the examined systems. The metric-driven approach is utilized to compare different ongoing designs and figure out the most optimum ones based on security, privacy and dependability analysis.

5.1.1 Attributes

Attributes are the technologies/functionalities that a component provides (e.g. run a specific cryptographic protocol) and are determined when the component is instantiated. The protection level is formally proved by the SPD multi-metric methodology. Attributes are mainly instantiated at the device layer. Each device type (e.g. nano, micro/personal and power node) can feature a constrained set of attributes, depending on its computational capabilities.

5.1.2 Sources

Sources are data or other critical assets that are processed by a component and are the core subject of the SPD evaluation (acting as the treasure room of the medieval castle and the attackers' target). They are created/deleted, sent/received, and processed by components. Each component possesses a set of sources that can manipulate. The component executes operations on sources and changes their SPD level. A source's SPD is determined by the SPD of the last operation that processes it (see the subsection below). When a component sends a source to another component (e.g. send an encrypted file over a network), it actually sends a new instance of the source. The sender retains the original source and the receiver a copy of it – which is now considered a different source (with an indirect association).

5.1.3 Operations

Process Operations

ALGORITHM 1: OperationEvaluation

```
Input: The operation identifier op.
Output: The operation's < SPD >.
for each timepoint $t_i$ in op do
    SequentialAttrs.Add(AttrWithMinSPDAtTi(op)) ;
end
for each attribute $attr_i$ in SequentialAttrs do
    $attr_i$.CoverMissingControlsFromAttr($attri_{i-1}$) ;
end
operationSPD = SequentialAttrs.getFinalAttr().EvaluateSPD();
Return(operationSPD);
```

Operations are a fundamental feature of our methodology. They are series of actions that a component performs to create or process sources. An operation consists of a set of attributes along with their execution order. The attributes can be executed sequentially or in parallel. At each time point, the minimum SPD of the attributes that are executed (in parallel) is found, as it is considered the weakest security link. The operation's SPD is determined by these set of attributes. Missing controls of the SPD multi-metric can be covered by relevant controls that are placed in the adjacent previous time point, forming the added protection of the sequential execution. The final operation's SPD represents the overall pore coverage that is achieved at the last time point. Algorithm 1 describes this evaluation procedure. The outcome of the execution can decrease/increase the SPD level of the source.

Consider an attribute modelling a simple network transmission service. The security feature of the SPD is low, as the relevant controls are missing (authenticity, confidentiality, integrity). Then, a secure encryption service is modelled. The service processes data with an authenticated-encryption primitive, providing authenticity, confidentiality, and integrity. We increase security by executing the two services in sequential order – the data are encrypted by the encryption service and then transmitted by the network transmission service. The security controls are cascaded from the encryption service and cover the missing controls of the network transmission service. The final SPD is the SPD of the network transmission service enhanced in security by the encryption service controls.

Composition Operations

Operations can denote the composition steps that must be met to compose components. Components of the same layer are composed to form a component of the adjacent higher layer, for example a group of nodes that are connected to form a network. A set of operations describe the functionality that a node must be able to perform in order to join the network (e.g. steps of the communication protocol or storage of encrypted data).

A component can perform a composition operation as long as it satisfies the functional and non-functional pre-conditions. The functional pre-conditions are derived from the operation's

ALGORITHM 2: OperationApplicabilityCheck

```
Input: The component C, the set of operations $set_{op}$.
Output: TRUE/FALSE: checks if the operation can be performed or not.
for each operation $op_i$ in op do
    opiAttrs = GetOperationAttrs($op_i$);
    if (!ContainsAllAttrs(C, opiAttrs)) then
        Return(FALSE);
    end
end
Return(TRUE);
```

attribute set. The component must be able to execute all the attributes as denoted by the operation. Algorithm 2 describes this applicability check.

The non-functional pre-conditions are quality factors that must be granted. They are modelled as minimum levels of the SPD multi-metric parameters (e.g. minimum level of confidentiality control coverage). The component must also comply with these SPD constraints to perform the composition operation.

In order to prove that a component can perform an operation, the framework uses Event Calculus (EC) [14]. The context parameters and the validity checks for the functional and non-functional pre-conditions are formally defined in rules, fluents, and events of EC. It can be proved that if the pre-conditions are valid, then the functionality (theorems) and the resulting SPD levels are valid.

In case of component composition, except from the underlying attributes, the definition of the composition operation describes the composition type. There are three different composition types in correspondence with the medieval castle composition operations (AND, OR, MEAN). This parameter is utilized during the SPD calculation of the composed component (see the subsection below).

5.1.4 System Components

Following the proposed methodology, a system is composed of SPD-sensitive components. The components include nodes, networks, middleware, and security agents (the overlay). Each component:

- Has a set of sources – data that are processed
- Has a set of attributes – technologies and protocols
- Performs some operation – series of attributes
- May contain one or more sub-components – components of lower layers
- Achieves specific levels of SPD – based on the evaluation metrics and its sub-components

The SPD level of a component is strongly affected by its sub-components' SPDs (as the weaker inner SPD links) and the component of higher layer with the minimum SPD that contain the investigated component (as it is the weaker outer link). For a single component at the lowest layer, we consider its sources as the equivalent element of the sub-components.

```
ALGORITHM 3: ComponentSPD
Input: The component C.
Output: The component's < SPD >.
structuralSPD = MedievalCastleSPD(C);
constraintSPD = MinRelativeComponentSPD(C);
componentSPD = structuralSPD;
if (structuralSPD > constraintSPD) then
    componentSPD = constraintSPD;
end
Return(componentSPD);
```

Algorithm 3 describes the SPD evaluation process for a component. The structural SPD of the component is calculated by applying the medieval castle evaluation among its sub-components. Each sub-component is considered as a castle door, with the relevant SPD representing the difficulty of passing through that door. The composition operation type determines the architectural significance of each sub-component/door.

A component's SPD cannot exceed the minimum SPD of the relative component of higher layer. For example, a node's SPD is constraint by the networks that it is currently participating in (e.g. the security level of a device is decreased when it is connected to the Internet).

5.1.5 Composition and Decomposition

Let us consider the composition of two nodes into a network (e.g. LAN of two embedded devices) under a secure communication protocol (e.g. IPsec via WiFi). At first, the SPD of each individual node component is evaluated. The communication protocol is modelled as a set of sequential operations. This set represents the functional requirements of the composition. The non-functional pre-conditions are defined by a set of metrics along with specific SPD levels that all the underlying components must provide. A relevant network component is defined; where each participating node must be able to comply with these pre-conditions in order to connect the network (e.g. Figure 5.A). A composition event happens for each node, denoting its attempt to enter the network. When a node does successfully enter the network, the network component's SPD is re-estimated (e.g. Figure 5.B). We also reason, if new properties hold for this composed component (the network – two nodes under the secure communication protocol). For example, if the placed controls for network confidentiality are covered, we reason that the network now provides the confidentiality property as well. Algorithm 4, describes the evaluation of a composition event to compose the sub-component *subC* to the main component *C*.

ALGORITHM 4: Composition

```
Input: The component C, a subcomponent subC.
Output: TRUE/FALSE: checks if the composition is performed or not.
currentSPD = C.SPD;
if (!C.ComplyWithFunctionalPre(subC) or !C.ComplyWithNonFunctionalPre(subC)) then
    Return(FALSE);    // Composition is blocked
end
C.compose(subC);   // Composition is performed
if (ComponentSPD(C) != currentSPD) then
    Return(FALSE);    // Composition is blocked
    for each subcomponent of C do
        ComponentSPD(subcomponent);
    end
end
ComponentSPD(subC);
Return(TRUE);
```

All the components that are directly or indirectly affected by the composition will re-estimate their SPDs and composition relations. A composition event can cascade successive changes in the components' status at different layers and eventually influence the whole system (e.g. Figure 5.C). The system balances after some time points (based on the number of the layers and the components) and the final SPD is formed.

Decomposition events are processed similarly. When a sub-component is decomposed from a component, their SPD features will be re-estimated as described above.

5.2 Formal Description of Composition of SPD Metrics

Formal methods and other validation techniques are used to confirm the provided properties of a system. In computer security it is widely accepted that no computer system can be considered *100%* secure. Thus, we use formal methods to prove that a system can meet some security properties under a pre-defined set of threats, according to our specifications and best practices. For example, if a system applies the security mechanisms for preventing Denial of Service (DoS) attacks as they are derived by the NIST's standards, we can reason that the system is DoS resilient for the present set of DoS attacks.

This subsection frames a formal representation of the composition and the security features of a system. The proposal is based on EC and the theoretical study of [40]. At first there are given some definitions and then the theorems and proofs of the proposed methodology.

5.2.1 General Definitions

Definition 1 [Metrics]: A) Let M be the set of individual metrics. Let Q be the set of all metrics, individual or composed. The composed metrics are combinations of individual metrics. B) Let Γ be the set of threats and attacks.

***Definition 2 [Relations]*:** Let $R\colon M^n \rightarrow Q$ be the set of relations on M.

***Definition 3 [Metrics to threats]*:** Let A: $R \rightarrow \Gamma$ be a function which corresponds a set of SPD metrics to the set of feasible threats and attacks. This set of metrics, called *r*, corresponds in a relation $r \in R$.

***Definition 4 [S, P, D values]*:** A) Let F_S: $R \rightarrow IR$ be a function which corresponds a security level to a real number. This real number represents a security level. B) Let F_P: $R \rightarrow IR$ be a function which corresponds a privacy level to a real number. This real number represents a privacy level. C) Let F_D: $R \rightarrow IR$ be a function which corresponds a dependability level to a real number. This real number represents a dependability level.

In our case, the functions F_S, F_P, F_D, are corresponding to the relevant functions of the SPD multi-metric methodology [54] that calculate the three SPD values.

***Definition 5 [Situations]*:** We define S as situation – a snapshot of the systems in a specific time point.

***Definition 6 [Events]*:** We define a set of events E which contains all events which could change the current situation of the system. If we assume that $e \in E$ and S is the current situation then after $Happens(e, S, t) \rightarrow HoldsAt(S', t + 1)$, S' is the new situation.

***Definition 7 [Security comparison]*:** Situation S_1 is more secure than situation S_2 when $A(S_2) \subseteq A(S_1)$ and $F_S(S_2) < F_S(S_1)$.

This definition shows that a situation S_1 is more secure than another S_2 if and only if the set of possible threats of S_1 is subset of the set of the of possible threats of S_2 and the security level of S_1 is higher than the security level of S_2.

***Definition 8 [Acceptable situations]*:** A) Let AM_S, AM_P, AM_D, be the security, privacy, and dependability levels respectively that are acceptable for the examined system. Let *AA* be the acceptable set of possible threats that can be performed according to the environment and the attacker model [40], [55]. As no system is perfectly secure, *AA* represents the security risk that we are willing to accept. B) A situation S of the system is acceptable if $F_S(S) \geq AM_S \wedge F_P(S) \geq AM_P \wedge F_D(S) \geq AM_D$ and $A(S) \subseteq AA$. C) Let AS be a set with all the acceptable situations.

5.2.2 Composition Verification Definitions

***Definition 9 [Components]*:** Let *Comp* be the set of individual components. Let *QComp* be the set of all components, individual or composed. The composed components are selection of components of the adjacent lower layers and $Comp \subseteq QComp$.

***Definition 10 [Component elements]*:** For each $comp \in QComp$: A) Let *QCompS* be a set of *qcomp*'s sources. B) Let *QCompAttr* be a set of *qcomp*'s attributes. C) Let *QCompSub* be a set of *qcomp*'s sub-components at the adjacent lower layers of *qcomp*. For $qcomp \in Comp$ there holds that Q$CompSub = \emptyset$. D) Let *QCompM* be a set of *qcomp*'s metrics.

***Definition 11 [Component Composition]*:** For each composed component $qcomp \in QComp$ and $qcomp \notin Comp$: A) Let *QCompOp* be a set of composition operations that all *qcomp*'s sub-components must be able to perform. B) Let *QCompCons* be a set of constraints along with a minimum SPD levels that all *qcomp*'s sub-components must comply with. C) A component

$qcomp_2 \in QComp$ at the adjacent lower layers of *qcomp*, can be composed to *qcomp* if it can execute an operation $qcomop \in QCompOp$ and satisfy all the constraints in $QCompCons$. The composition event is defined as $Happens(Comp(qcomp, qcomp_2, qcompop, S, t)) \rightarrow (HoldsAt(S', t+1) \wedge Happens(EvaluateCompSPD(qcomp), S', t+2))$. After a successful composition at *S'*, it holds $qcomp_2 \in (QCompSubofqcomp)$. A component *qcomp₂* can be decomposed from *qcomp* at any time point. The decomposition event is defined as $Happens(DeComp(qcomp, qcomp_2, qcompop, S, t)) \rightarrow (HoldsAt(S', t+1)$. After a successful decomposition at *S'*, it holds $qcomp_2 \notin (QCompSubofqcomp)$.

***Definition 12 [System Composition]*:** Let *Sys* be an examined system. *Sys* is a set that contains the system's composed components along with their metrics, properties and composition relations. The composition of a component $qcomp \in QComp$ to *Sys* by performing the operation $sysop \in QCompOp$ of Sys is defined as $Happens(Comp(Sys, qcomp, sysop), S, t)) \rightarrow (HoldsAt(Sys', t+1)$, where it holds $S' = Sys \cap qcomp$. The composition of two different systems *Sys₁* and *Sys₂* is defined as $Happens(Comp(Sys_1, Sys_2, op), S, t)) \rightarrow (HoldsAt(Sys', t+1)$, where it holds $S' = Sys_1 \cap Sys_2$.

***Definition 13 [Operation execution]*:** To execute an operation, a component must be able to execute all the operation's attributes. Each operation has a set of attributes $attrs_{op}$. Each component has a set of attributes *QCompAttr*. An operation *op* can be executed by a component $qcomp \in QComp$ iff (if and only if) $attrs_{op}$ is a subset of $QCompAttr(Attrs_{op} \subseteq QCompAttr)$.

5.2.3 Security Validation Definitions

Security validation is modeled by the metrics and properties. Metrics present measurable parameters of a system. *Definition 4* defines the functions that corresponds the measured parameters for security, privacy, and dependability of each metric to real numbers. Section 3 describes the basic features and the evaluation process for metrics and the SPD formation. The SPD multi-metric evaluation is formally described by [54]. The Medieval Castle composition approach is formally defined by [36]. The composed SPD multi-metric evaluation of an operation is based in these two approaches and their formal definition. The operation evaluation process is described in subsection 5.1.3. The component's evaluation process is based on similar assumptions and described in the subsection 5.1.4.

Definition 10 determines the relation between metrics and components. *Definition 11* defines the prior-composition validation process for imposing the pre-conditions that must be granted to enable composition. The definitions *Definition 14* and *Definition 15* below, define the post-composition validation process.

***Definition 14 [Constraint validation]*:** For a constraint *cons* of that is imposed by the component $qcomp \in QComp(cons \in QCompCons)$: A) Let *cons-comp-pre* be the constraint's pre-conditions at *qcomp*'s layer. B) For sub-layer *i*, let $cons\text{-}subcomp\text{-}pre_i$ be the constraint's pre-conditions at sub-layer *i*. C) the constraint *cons* holds iff *qcomp* satisfies *cons-comp-pre* and each sub-component $qcomp_i \in (QCompSubofqcomp)$ of sub-layer i satisfies the relevant $cons\text{-}subcomp\text{-}pre_i$. D) if a sub-component $qcomp_i \in (QCompSubofqcomp)$ of sub-layer i violates the relevant $cons\text{-}subcomp\text{-}pre_i$ decompose the sub-component:

$HoldsAt(ViolateSubCompCons(qcomp_i, cons), S, t)) \rightarrow$
$Happens(DeComp(qcomp, qcomp_i, decompop), S, t) \land$
$Happens(EvaluateCompSPD(qcomp, cons), S, t + 1) \land$
$Happens(EvaluateCompSPD(qcomp_i, cons), S, t)).$

***Definition 15 [Metric SPD]*:** For a metric $m \in Q$ of the component $qcomp \in QComp(m \in QCompM)$: A) Let $mParam$ be the system's parameters that affect m. B) If the status of $mParam$ changes, m's SPD value is re-evaluated: $HoldsAt(CompMetric(qcomp, m), S, t)) \land$ $Happens(ChangeParamStatus(mParam), S, t) \rightarrow$ $Happens(EvaluateMetricsSPD(qcomp, m), S, t)).$

***Definition 16 [Component SPD]*:** For a sub-component $qcompsub \in QComp$ of the component $qcomp \in QComp(qcompsub \in QCompSub)$: A) If the status of $qcompsub$ changes, $qcomp$'s SPD value is re-evaluated: $HoldsAt(CompSub(qcomp, qcompsub), S, t)) \land$ $Happens(ChangeCompStatus(qcompsub), S, t) \rightarrow$ $Happens(EvaluateCompSPD(qcomp), S, t))$. B) If the status of $qcomp$ changes, $qcompsub$'s SPD value is re-evaluated: $HoldsAt(CompSub(qcomp, qcompsub), S, t)) \land$ $Happens(ChangeCompStatus(qcomp), S, t) \rightarrow$ $Happens(EvaluateCompSPD(qcompsub), S, t)).$

5.2.4 Formal Example

The functions A and F are defined in dynamic situations. In the first example, the composition verification process is demonstrated in a simple scenario, where two smart home devices (node components) are composed to a LAN (network component). Each device can securely transmit data with three variants of IPsec for 128-, 192-, and 256-bit cryptographic keys (node component attributes). In order to communicate, they must agree on a common set of attributes that will be used (length of the cryptographic key). First, the SPD metrics for each attribute and individual nodes are calculated. If their composition is successful, the relevant metrics for the composed network are calculated. The second example presents the SPD validation process and how the different potential states of the system (for 128-, 192-, and 256-bit keys) can be compared in terms of security.

<u>Composition Verification Example</u>

This example formally presents the composition verification process for composing two nodes to a network. There are modelled the two nodes $n_1, n_2 \in Comp$ and the network $net \in QComp$. There are also modelled the source *s₁* and *s₂*, where $s_1 \in (QCompSofn_1)$ and $s_2 \in (QCompSofn_2)$. Each node implements the attribute $attr_{IPsec}$ (with the three aforementioned configurations) and $attr_{IPsec} \in ((QCompAttrofn_1) \land (QCompAttrofn_2))$.

In order to be composed to the *net*, all nodes must be able to perform the composition operation $op \in QCompOp$ of *net*, which implies the functional requirement of using a specific configuration of the IPsec attribute. Non-functional requirements can also be enforced (e.g. the security property of a node component must be higher than *20*). The composition type is an

AND-operation. To enter *net*, each node must execute *op*, which is modelled by the rule r_1 in EC.

r_1: $Happens(AttemptComp(net, n_i, op), t) \land HoldsAt(CompAttr(n_i, IPsec_i), t) \rightarrow Happens(Comp(net, n_i, op), t) \land Happens(ChangeCompStatus(net), t)$

After a successful composition it holds that $n_i \in QCompSub$ of *net*. The event $Happens(Comp(net, n_i, op), t)$ triggers the evaluation of the SPD of *net* (*Definition 11*). The event $Happens(ChangeCompStatus(net), t)$ denotes that the status of *net* has change and can potentially trigger the evaluation of the SPD of *net*'s its sub-components (*Definition 16*).

Security Validation Example

Assume the following configuration $r_{128}, r_{191}, r_{256} \subset R$ for the three instances of the system that uses IPsec with 128-, 192-, and 256-bit cryptographic keys respectively. In each case, if *net* has security level l_{net}, n_1 has security level l_{n1}, and n_2 has security level l_{n2}, then the level of security is equal with the minimum of the l_{net}, l_{n1}, and l_{n2}. The security level of each component depends on the composition operation *op* that is performed, which is the security level of the IPsec configuration.

For $i \in \{128,192,256\}$, it is formally defined $r_i(net,n_1,n_2)$. An attacker has to compromise at least one the network nodes (AND-operation composition). Thus, the number of possible attacks $A(r_i)$ is the union of the attacks targeting the components *net*, d_1 and d_2 – $A(r_i) = \{a | a \in A(net) \lor a \in A(n_1) \lor a \in A(n_2)\}$. The potential attacks to an IPsec key size is a subset of the relevant attacks of the smaller IPsec key sizes, so it holds that $A(r_{128}) \subseteq A(r_{192}) \subseteq. A(r_{256})$. The overall security level $F(r_i)$ is the minimum security level between *net*, n_1 and n_2 – $F(r_i) = \min(F(net), F(n_1), F(n_2)) = \min(n = l_{net}, l_{n1}, l_{n2})$. Based on the SPD multi-metric evaluation, for $i=128$ the security level is $F(r_{128})=30$, for $i=192$ the security level is $F(r_{192})=45$ and for $i=256$ the security level is $F(r_{256})=60$. Thus, according to *Definition 7*, r_{192} is safer than r_{128} and r_{256} is safer than r_{128} and r_{192}.

5.3 Smart Home Composition Scenario

CompoSecReasoner acts as the reasoning behavior of SPD-Safe agent that implements the aforementioned SPD metrics and the composition validation process. As an environment setting, we consider the smart-home for assisting living and as an attacker model, we deal with individual attackers with moderate computer security knowledge and low attack and computational capabilities (e.g. *1* to *5* PCs).

5.3.1 Technical Details

The agent reasoning engine that is presented in the following section is utilized for the modelling of the CompoSecReasoner behavior. The smart home system is implemented that is described in the subsections 4.2 and 5.3. We mainly model the cryptographic service with an IPsec implementation and the PBAC framework [62], which consists of four entities:

- **Policy Enforcement Point (PEP):** performs access control at the node layer
- **Policy administration Point (PAP):** creates the policies at the middleware layer
- **Policy Decision Point (PDP):** evaluates applicable policies and renders authorization decisions at the middleware layer
- **Policy Information Point (PIP):** acts as a repository for the provided features that are evaluated by the PBAC at the middleware layer

PEP communicates information with the PDP (between a node and the middleware) under four settings for different levels of SPD: plaintext-only, authentication-only, encryption and authentication, and WS-Security. PEP exchanges information with a device's user under three different setting: plaintext-only, authentication-only, encryption and authentication. For more details refer to the original paper [62].

For the node layer, we model the nodes n_1 and n_2 as the two BeagleBone/BeagleBoard devices and n_3 as the laptop. n_1 and n_2 are identical and provide the attributes ULCL, WiFi, IPsec, and PEP (with the three aforementioned SPD levels). As sources, they maintain the data of the local instances of PEP. n_3 provides the attributes ULCL, WiFi, IPsec, PBAC (with the four aforementioned SPD levels of PDP), and CompoSecReasoner. As sources, it stores the data of PDP, PAP, and PIP respectively.

For the network layer, we model the network *net*. The network provides the attributes WiFi and IPsec. In order to enter to the network *net*, a node must be able to perform a composition operation (MEAN-operation), which requires the sequential execution of PEP, IPsec, and WiFi (op_5). The nodes n_1 and n_2 perform this operation to enter *net*.

For the middleware layer, we model *midl*. It provides the attribute PBAC and the platform security mechanisms of OSGi middleware. There are two different composition operations for connecting this middleware for components of the node and the network layer respectively. Nodes must be able to perform the attributes PBAC and IPsec op_2) while networks must perform only IPsec (op_6). The middleware runs on the laptop. Thus, a composition event must be performed by n_3 to *midl*. The network *net* performs the second operation to connect *midl*.

For the demo system, we describe only one SPD agent, the *SA*. It provides the attributes SPD-Safe and the platform security mechanisms of JADE-S add-on. It offers two different composition operations for composing node and middleware components respectively. A node must perform the attribute SPD-Safe (op_1) and a middleware must perform the attribute PBAC (op_3). SA runs on the laptop along with the PBAC middleware. Thus, a composition event must be performed by n_3 to *SA*. Then, *midl* is composed by performing the second operation.

Finally, for the overlay layer we model the overlay over which contains two identical *SAs*. An agent must be able to run SPD-Safe in order to be composed (op_4).

5.3.2 SPD Levels

We perform the SPD multi-metric to calculate the SPD level of each examined attribute. The ULCL attribute provides a single SPD level of *ULCL=<80,50,30>* for secure storage of private information and influences the node's confidentiality property. Similarly, the WiFi attribute offers *WiFi=<20,20,50>* for allowing devices to exchange data wirelessly and influences the network property data integrity. The IPsec attribute provides three levels of SPD for the three cryptographic key sizes *($IPsec_{128}$ = <68,20,70>*, *$IPsec_{192}$ = <75,20,70>*, *$IPsec_{256}$ = <85,20,70>*). It provides authentication and encryption. When all the nodes of a network perform IPsec to communicate, the SPD of the data confidentiality property is increased. The node attribute PEP has three SPD levels for the three levels of security (*PEP_{Plain} = <10,30,10>*, *PEP_{Auth} = <50,50,50>*, and *$PEP_{Enc\text{-}Auth}$ = <80,70,80>*). Similarly, the PBAC attribute has four SPD levels for the communication between the entities PEP and PDP (*$PBAC_{Plain}$ = <10,30,10>*, *$PBAC_{Auth}$ = <50,50,50>*, *$PBAC_{Enc\text{-}Auth}$ = <80,70,80>*, and *$PBAC_{WS\text{-}Sec}$ = <90,80,90>*). The SPD of the SPD-Safe attribute is evaluated at *<70,50,70>*.

All metrics are evaluated dynamically at runtime. The security factor is enhanced by several security mechanisms, like JADE-S add-on and IPsec. The overall privacy is mainly retained by the PBAC framework policies. All mechanisms contribute to the provided dependability.

5.3.3 Composition and SPD Evaluation

At first the system components (*n_1*, *n_2*, *n_3*, *router*, *net*, *over*) are instantiated for low power consumption and moderate security. The instances of the PEP, PBAC, and IPsec attributes are configured at the SPD level of *PEP_{Auth}*, *$PBAC_{Auth}$*, and *$IPsec_{128}$* respectively. CompoSecReasoner evaluates the SPD of each individual component – no component contains sub-components at this time point.

The *n_3* performs three operations to encrypt the underlying sources (PDP, PAP, and PIP) with the *ULCL*: $Happens(PerformOp(n_3, ULCL, Source_i), SourceSPD(Source_i, S, P, D), t)) \rightarrow HoldsAt(SourceSPD(Source_i, S', P', D'), t + 1)$

The SPD of each source is *<80,50,30>* (the SPD of *ULCL*). These changes trigger the SPD evaluation for *n_3 (<80,50,30> – Definition 16*).

Similarly, *n_1* and *n_2* encrypt their own local instances of PEP. The SPD of the nodes and the sources is evaluated to *<80,50,30>*.

A node must perform the network and middleware technologies in order to connect the LAN. Operation *op_1* imposes these composition requirements, denoting that a node must sequentially perform the attributes WiFi, IPsec, and PEP. The composition type for the composed sub-components is a MEAN-operation (as discussed in subsection 4.3).

The router provides local networking capabilities. The LAN is a sub-component of the router. Operation *op_2* defines the composition requirements of our network as the sequential execution of the attributes WiFi and IPsec (OR-operation).

Except from local networking capabilities, the router provides Internet connection, defined by *op_3*. The composition type of the router and the Internet is an OR-operation.

Then, *n_3* is composed to the *net* (*op_1*): $Happens(Comp(net, n_3, op_1), CompSub(net), t)) \rightarrow (HoldsAt(CompSub(net), n_3, t + 1) \wedge Happens(EvaluateCompSPD(net), CompSPD(net, S, P, D), t + 2))$. The SPD of *net* is re-evaluated as a result of the composition (*<70,50,70> – Definition 11*) and then the SPD of *n_3* is revisited (*<70,50,30> – Definition 16*). *n_1* and *n_2* are also composed to the *net* (*op_1*). The SPD of *net* is *<68,30,70>*.

net is composed to the *router* (*op_2*). The *router* SPD constrains the SPDs of *router*'s sub-components (*net* and consequently n_1, n_2, n_3). Finally, the *router* connects to the Internet (*op_2*). After the composition of the system we can reason about the metrics that are eventually offered. The total SPD of the system is <*70,50,30*> which is strongly affected by the SPD of n_3.

The smart building can connect with other smart entities through Internet (*op_3*). Figure 6 illustrates the final composed system of the demo with two identical smart buildings along their main features.

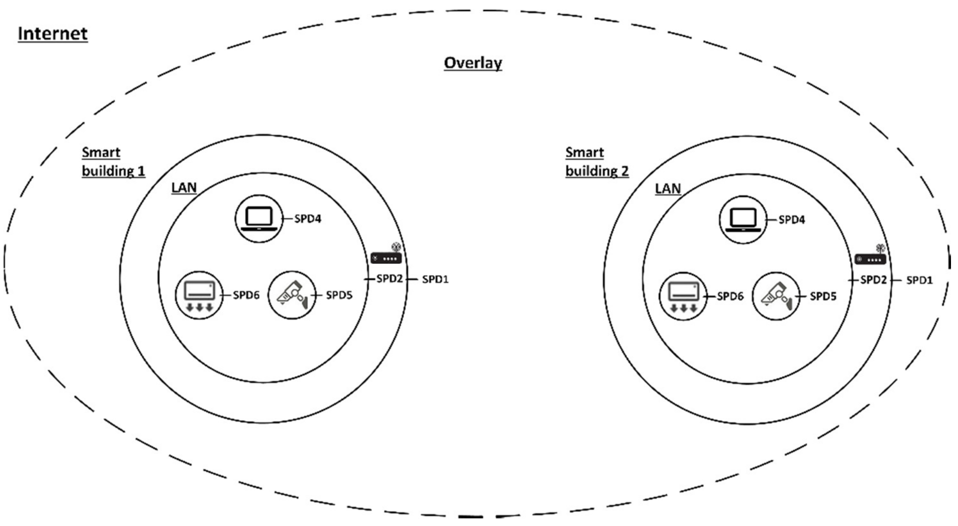


Figure 6 The application of the composed SPD multi-metric for two cooperating smart buildings

The metric-driven runtime management capabilities of the tool can be utilized in order to dynamically configure the 36 settings of the system (combination of 3 IPsec, 3 PEP, and 4 PDP settings) according to our high-level SPD strategy and the AI reactive plan, which is triggered by real-time monitoring events.

Consider the case where the SPD-Safe agent of a neighboring smart building detects a cyber-attack in its region. The agent sends a relevant alarm-message to the overlay in order to inform the other agents. The smart home SPD-Safe agent decides to change the configuration strategy from low power and moderate security to high security. The reactive plan imposes the system components to use the attributes $PEP_{Enc\text{-}Auth}$, $PBAC_{WS\text{-}Sec}$, and $IPsec_{256}$. Several operations are performed, triggered by the plan's events to configure the system. The overall protection is now increased (*Definition 7*) and the new SPD of the building is <*90,80,90*>. Figure 7 depicts the final composition results and the SPD evaluation of CompoSecReasoner.

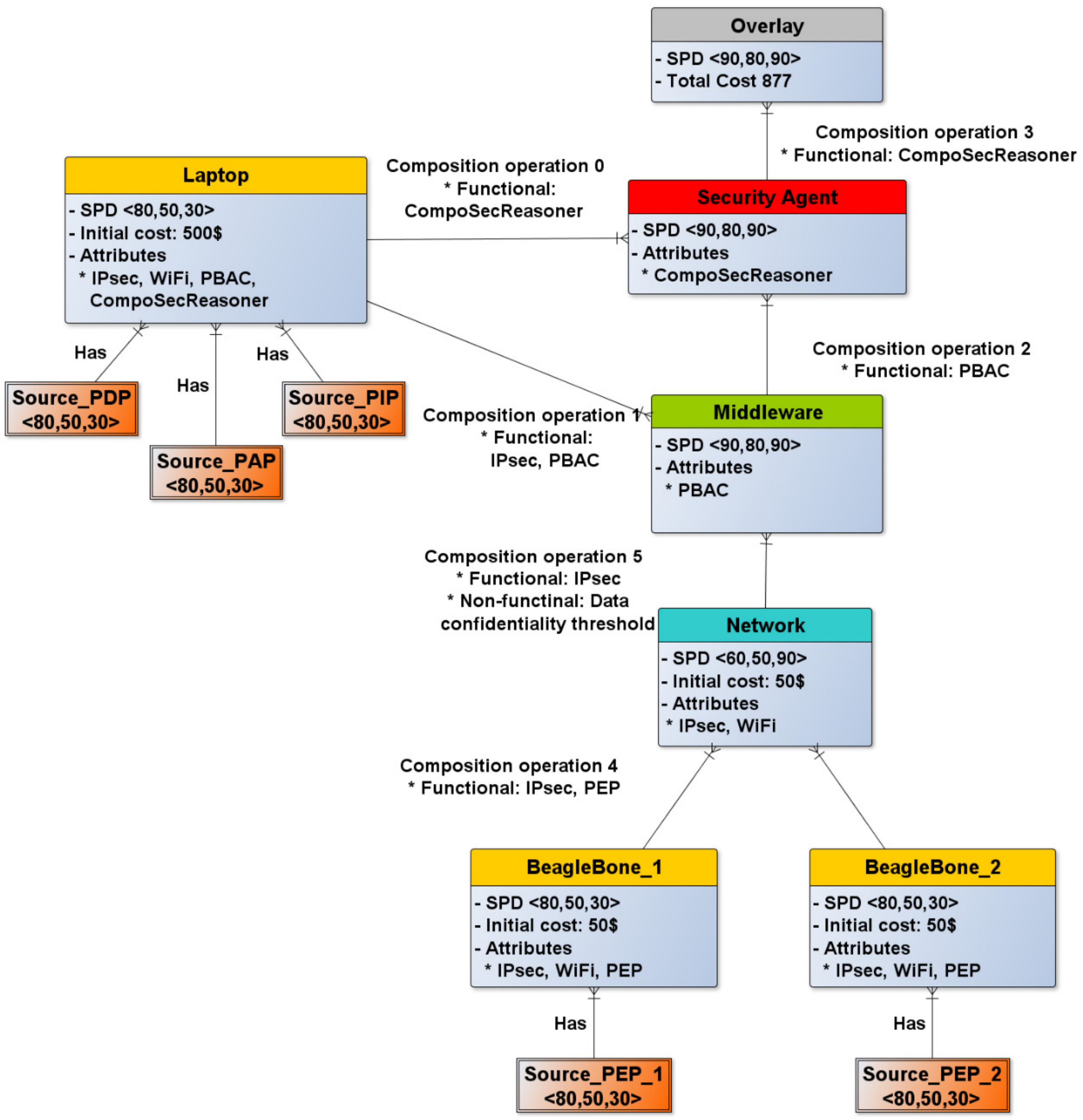


Figure 7 The CompoSecReasoner's outcome

6. Core Technologies & Entities

The need to manage embedded systems, brought forward by the wider adoption of pervasive computing, is particularly vital in the context of secure and safety-critical applications. Technology infiltrates in ordinary things, hitching intelligence and materializing smart systems. Each of these individual entities monitors a specific set of parameters and deduces a constrained local view of the surrounding environment. Many distributed devices exchange information in order to infer the real system state and achieve a consistent global view.

Considering the aforementioned aspects of these applications, context-aware technologies are required in order to specify the ambient environment conditions [14], [15], [16], [17]. Semantic ontologies model the ambient domain, presenting it in machine-readable format. Then, logic languages can process this knowledge to reason about and react to the current state of the ambient environment. Software agents are a form of intelligent entity that incorporates these features, providing reasoning and management capabilities on the underlying system (e.g. [7], [16], [17], [18]).

6.1 Reasoning Engine

This subsection describes the core reasoning process of each individual agent. The EC models the knowledge base and the reasoning theory. Then, it is processed by a real-time model-checking method. The overall approach enables several reasoning capabilities, like reasoning with missing pieces of knowledge, real-time events, and rule priorities.

6.1.1 Knowledge Representation and Reasoning

Knowledge Representation and Reasoning (KR&R) is the research field of Artificial Intelligence (AI) that deals with the representation of information about the world in a manner that it is understandable by a machine and can be utilized in order to solve complex tasks. KR&R is strongly related to the semantic technologies which encode knowledge to express ontologies, and logic languages which include reasoning rules for processing that knowledge.

For real-time processing, event-driven model-based methods have been proposed to describe the behavior of a dynamic system in a formal manner [16]. Moreover, such methods are applied in ambient intelligence applications for reasoning about the ambient environment state as new events occur.

An early version of the reasoning process is presented in [56]. SPD-Safe extents the reasoning engine and adopts the latest advancements in the field. The whole operation is enriched with the SPD metric feature, implementing metric-driven reasoning and management. Moreover, the system is deployed on embedded devices and applied on a real use-case.

The following subsections describe the core calculus and the logic language that are utilized by SPD-Safe for KR&R.

6.1.2 Event Calculus

Event Calculus (EC) [14] is a logic language for representing and reasoning about actions and their effects as time progresses. It is a formalism for reasoning about action and change. The EC has: *actions* – which are called events and indicate changes in the environment; *fluents* – which are time-varying properties (predicates/functions); and a *timepoint* sort – which implements a linear time structure on which actual events occur. It is based on first-order predicate calculus and is capable for simulating a variety of phenomena such as actions with indirect effects, actions with non-deterministic effects, compound actions, concurrent actions, and continuous change. The EC defines predicates for expressing, among others, which fluents hold when (*HoldsAt*), what events happen (*Happens*) and which their effects are (*Initiates*, *Terminates*). It adopts a straightforward solution to the frame problem which is robust and works in the presence of each of these phenomena.

The EC supports context-sensitive effects of events, indirect effects, action preconditions, and the commonsense law of inertia. Certain phenomena are addressed more naturally in the event calculus, including concurrent events, continuous time, continuous change, events with duration, nondeterministic effects, partially ordered events, and triggered events. Examples of such phenomena could be:

- **The commonsense law of inertia:** when moving a glass does not cause a glass in another room to move.
- **Release from the commonsense law of inertia:** if a person is holding a PDA, then the location of the PDA is released from the commonsense law of inertia so that the location of the PDA is permitted to vary.
- **Event ramifications or indirect effects of events:** the PDA moves along with the person holding it (state constraint) or instantaneous propagation of interacting indirect effects, as in idealized electrical circuits (casual constraints).
- **Conditional effects of events:** the results of turning on a television depend on whether it is plugged in or not.
- **Events with nondeterministic effects:** when flipping a coin results in the coin landing either heads or tails.
- **Gradual change:** the changing height of a falling object or volume of a balloon in the process of inflation.

The EC contains a set of fluents, a set of events, and a partially ordered set of time points. In the EC, the description of the worlds (possible scenarios) is based on the following axiom (assume *e* is an event, *f* is a fluent, and *t*, t_1, and t_2 are time points):

1. **Initiates(*e*, *f*, *t*):** *f* holds after event *e* at time *t*
2. **Terminates(*e*, *f*, *t*):** *f* does not hold after event *e* at time *t*
3. **InitiallyP(*f*):** *f* holds from time *0*
4. **InitiallyN(*f*):** *f* does not hold from time *0*
5. **Happens(*e*, t_1, t_2):** event *e* start at time t_1 and ends at t_2
6. **HoldsAt(*f*, *t*):** *f* is holds at time *t*
7. **Clipped(t_1, *f*, t_2):** *f* is terminated between t_1 and t_2
8. **Declipped(t_1, *f*, t_2):** *f* is initiated between t_1 and t_2

In EC specific values of the fluents describe a situation. An event which changes the value of one or more fluents has as a consequence the change of the situation. An evolution of the world is a sequence of actions and situations.

Events and fluents are formally defined. Events occur sequentially or in parallel on denoted time points. Events can change the state of fluents and trigger new events. These transactions are modelled by rules. The rules have preconditions. If the preconditions are satisfied (left-hand statements), the rule is executed and may change the state of a fluent. At this point, statements are accepted that have been proved by the predicate calculus. According to these statements, rules are modelled that implement the management logic. A fire alarm can be modelled by an event. The event triggers a set of rules which check if there are any actions that must be taken (functional and non-functional properties of reaction strategy). When the counteractions are completed, another set of rules can be triggered to determine which SPD and safety properties are satisfied. The events and the changes they cause, produce a trace in time. The final state of the trace determines the final outcome.

Definition A: FL is the set of fluents

Definition B: AC is the set of events

Definition C: Each event $e \in AC$ could take place if a logical proposal is true. This proposal is the preconditions of the event *e*.

For example, assume the event $e = SetKeyLength(node_1, 256)$, which tries to set the cryptographic key length for the node *node$_1$* at 256-bits. The execution of this action has as a consequence the change of the length ($f_1 = KeyLength(node_1, 256)$). The action is possible only if the key length is supported by *node$_1$*. The precondition *SupportKeyLength(node,length)* means that a node supports the specific cryptographic key length. If the precondition is false in the time point *t*, the event *SetKeyLength* cannot be executed.

Rule *r$_1$*:

$$[Happens(SetKeyLength(N, L), t) \ \land \ HoldsAt(SupportKeyLength(N, L), t)]$$

$$\rightarrow \ Initiates(SetKeyLength(N, L), UseKeyLength(N, L), t)$$

r$_1$ means that in each time point that the event *SetKeyLength* occurs and the relevant key length is supported, the node will use this key length from now on.

Rule *r$_2$*:

$$Initiates(SetKeyLength(N, L), UseKeyLength(N, L), t)$$

$$\rightarrow \ [Terminates(SetKeyLength(N, L), UseKeyLength(N, L'), t) \ \land \ L' \neq L]$$

r$_2$ means that only one key length can be used at each time point. When a new key length is set, the previous selection is terminated.

Definition D: We call situation a first order term which contains all members of set *FL* with a specific value for each one.

For example, assume that at time point *1*, holds the situation

$S_1 = \{ f_1 = UseKeyLength(node_1, 128),\ f_2 = SupportKeyLength(node_1, 128),\ f_3 = SupportKeyLength(node_1, 256), \cdots \}$. The *node₁* currently uses 128-bit keys and also supports 256-bit length. At time *3*, the event *Happens(e₂, 3)* takes place. This results in:

- r_1: *Initiates(e₂, UseKeyLength(node₁,256), 3)* – The event e_2 sets the fluent *UseKeyLength(node₁,256)*, at time point *3* (*1st* EC axiom).
- r_2: *Terminates(e₂,f₁,3)* – The event e_2 terminates the fluent f_1, at time point *7* (2nd EC axiom)

The consequence of the execution for event e_1 is the change of situation S_1 into $S_2 = \{ f_1' = UseKeyLength\ (node_1, 256), f_2, f_3, \cdots \}$. The *node₁* now increases the security level and uses 256-bit keys.

Figure 8 illustrates this process diagrammatically.

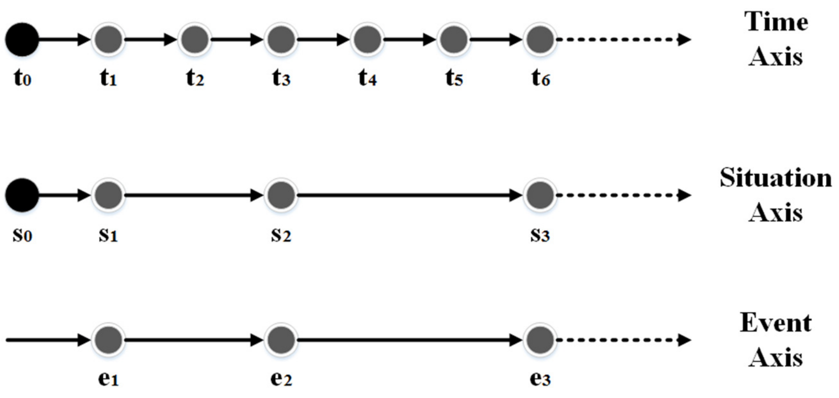


Figure 8 The correspondence among Time-Actions / Events-Situations in the Event Calculus

The subsection below describes the specific implementation of EC that is used in this study along with the agent platform where the SPD-Safe agents are deployed.

6.1.3 Discrete Event Calculus Knowledge Theory

EC supports context-sensitive effects of events, indirect effects, action preconditions, and the commonsense law of inertia. The Discrete Event Calculus Knowledge Theory (DECKT) [15] is an extension of EC in Java and the rule engine Jess, which additionally supports automated epistemic, temporal, and casual reasoning for dynamic, uncertain, and partially-known domains.

In this work, DECKT is enriched with real-time events, preferences, and priorities. The whole reasoning model is formed as an agent's reasoning behavior and is deployed in JADE, implementing a multi-agent reasoner. The agents utilize the Agent Communication Language (ACL)[3], a FIPA[4] standard, to exchange information.

Common Alerting Protocol (CAP)[5], an OASIS standardized XML-based data format for exchanging public warnings and emergencies, models semantic information that is exchanged

[3] FIPA, ACL: http://en.wikipedia.org/wiki/Agent_Communication_Language
[4] FIPA: http://www.fipa.org/
[5] OASIS, CAP: http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf

between the entities. The standard increases warning effectiveness and simplifies the task of activating a warning for responsible officials. Real use-cases of CAP include wireless sensor networks for automatic tsunami warning, aggregation and correlation on real-time map of current warning activity in a state's operations center, repudiating of false alarms by responsible officials, and integrated public alerting through multiple channels. In this work, CAP alerts are transformed into Jess-DECKT events and trigger the rule-based reasoning of SPD-Safe.

The reasoning process of SPD-Safe is further enriched with the SPD metrics feature that were described in the previous sections. The overall system is applied in ambient intelligence applications for assisting living, implementing the metric-driven SPD and safety management. The reasoning process configures the system at runtime in order to achieve specific SPD goals.

The next subsection summarizes the overall reasoning capabilities of the proposed deployment.

6.1.4 Reasoning Capabilities

SPD-Safe uses the Discrete Event Calculus Knowledge Theory (DECKT) – an EC reasoner implemented in the rule engine Jess. DECKT extends the basic reasoning process with hidden casual dependencies (HCD). It enables reasoning with missing pieces of knowledge which may become known in a future time-point, like sensory measurements.

DECKT performs reasoning over virtual discrete time-points. This process is extended to express real time events by integrating the Java Timer object. For every time-point where an event with duration must occur, a timer is created that fires when the specific time elapses. Then, the reasoning process is automatically activated, deducing the current state of the system. A hash table of active Timer objects is maintained with the different time-points representing the hash keys. Each slot contains the Timer and the list of rules that cause the objects' creation. A rule's effect can be cancelled in a future time-point before the Timer fires. Thus, the rule is erased from the Timer's list. When all rules are removed from a list before the Timer expiration, the object is deleted. We deploy several variations of these process, implementing different functionality for the time duration events, like create, pause, stop, kill, start, replay, and reset.

Each rule in Jess assigns a value, called salience, which determines the execution order of rules that fire simultaneously (the higher the salience, the most recent the rule is processed). The mechanism is exploited by SPD-Safe in order to model rules' priorities. Thus, high priority to specific rules is expressed by defining high salience values. When the first rule among a group of competitive ones fires, it assigns a relevant fact in the Jess engine internal memory to denote that the rest rules are blocked. For example, consider the case where the system must automatically inform the fire brigade in case of a safety-related incident, either via email or SMS. A relevant event fires that can trigger two rules for the two aforementioned types of transmission respectively. If both types are currently active, SPD-Safe can select the communication form based on the user's preferences.

The information exchange capabilities and the SPD-Safe MAS are detailed in the following subsections. A deconfliction mechanism [56] is also included to retain coherency in cases where conflicts occur.

6.2 Multi-Agent Conflict Resolution

This subsection presents the MAS and the multi-agent reasoning process. The agents form a peer-to-peer network and exchange information. The agents aggregate these pieces of knowledge and derive the global state of the system. When conflicts occur, a resolution mechanism will retain coherency.

6.2.1 Multi-Agent System Architecture

In SPD-Safe, the agents are set as nodes in a peer-to-peer network, following the super-node approach [57]. Each agent acts as an individual entity: it performs its own actions (e.g. sensing and reasoning) and maintains a local knowledge base. The knowledge is formed by facts, rules, and constraints. Agents communicate these pieces of knowledge over the network to perform collaborative tasks or achieve commonsense knowledge and determine the global state of the system.

Two main agent types are considered: Simple Agents (SAs) and Master Agents (MAs). The high majority are SAs with constrained computational and storage capabilities. They sense local events and distribute new pieces of knowledge to the agent system. To retain system security and user privacy, SAs may maintain locally protected knowledge that is not disclosed to other entities (e.g. internal security configurations or usage preferences). Each SA is directly connected to a MA. In the smart building application, every smart device runs a SA which is connected to the building's MA.

MAs constitute the backbone of the system, adopting the super-node role. They carry out the full communication among agents and are responsible for the proper knowledge distribution. Each MA can communicate with its underlying SAs and the MAs that are directly connected with. They receive the local knowledge from the attached SAs and publish it to the interested entities. The MA can also perform the individual operation of a SA. Nonetheless, they possess sufficient computational and storage resources.

Performance depends on the analogy among master and simple agents, and the average number of SAs that are managed by the MAs (average MA effort). In an open environment, agents enter and leave the network dynamically. SAs can be dynamically transformed into MAs, and vice-versa, according to the system's needs and the devices' capabilities. Thus, the system becomes more flexible and efficient as it enhances load-balancing.

All agents share common context representation and data definitions for the pieces of knowledge that are distributed and jointly used. In the examined setting, all agents are considered honest, perform absolute cooperation and no agent possesses secret goals and aspirations, except from the aforementioned locally protected knowledge. Security and privacy are retained as agents only reveal the required information. No game theory issues are examined.

The subsection below details the exchange of knowledge between the agents and how the multi-agent reasoning process works. The proposed conflict resolution mechanism retains the proper operation of the system in case of affairs.

6.2.2 Multi-Agent Reasoning Behavior

Sensing, reasoning, and knowledge distribution are the main actions that are performed by agents. At start-up, the agent performs a reasoning task to determine its initial state. When a message that contains new pieces of knowledge is received (e.g. periodic sensing of system parameters or distributed information from other parties), it is stored in the local knowledge base and a reasoning task is performed to calculate the current state. The sensed or exchanged data are well-formed by the CAP scheme and are encapsulated in ACL messages before transmission. The reasoning phase transforms the content of the CAP events into Jess-DECKT events that trigger the rules of the reasoning theory.

Except from parameters that are monitored by a single agent, there are others that are observed or processed by many agents. In this case, agents express their interest on these specific variables to the supervising MA. The MA retrieves the missing information from its SAs or other collaborative MAs and automatically propagates changes that occur. After processing new knowledge, an agent informs its MA about possible changes in the local environment. The MA updates its knowledge base and publishes the new state, informing the interested agents. When the agent is no longer interested about these changes, it informs the MA to stop the information updating process.

However, as agents distribute knowledge, local conflicting conclusions may occur. For example, two agents in different rooms may report the presence of the same person in their monitored region. As the person cannot be in two places at the same time, the agents must communicate to resolve the affair. The agents have to pass through a proof of evidence mechanism and justify their point of view. Each agent argues about its own view and tries to convince the rest involving entities. The subset of the facts and the reasoning theory that participate in the conflict are utilized. The proposed conflict resolution mechanism of SPD-Safe is constituted from two main novel components: the **share theory** and the **certainty degree**.

An agent will reason about the state of an examined variable, if it includes rules that contain the variable and fire. These rules are considered for the share theory construction. A MA undertakes the clarification of an arising conflict. At first, it collects the aforementioned involving agent sub-theories and creates a share theory structure. The share theory integrates these theory rules and duplicates are removed. Except from the rules, each agent sends a small history of relevant events that influence the examined variables. The MA sorts the events according to the real-time points that they were happened. Then, it performs a reasoning process for the share theory and these events, determining the final model.

Additionally, the certainty degree can be utilized as a recommender's grading mechanism. The rules that support a specific value for an examined variable contribute positively in the certainty degree calculation, while the opposing rules contribute negatively. The summation of these degrees constitute the theory contribution (TC) of this agent to the variable's state estimation. Policies are also supported, where the combination of rules can produce different results. For example, if only one of the rules r_1 or r_2 fires, add one (*+1*), but if they fire simultaneously, subtract one (*-1*). The TC value range is limited between *0-10*. Values of *0-5* indicates normal certainty degree, while a value of *6-10* represents strong evidence.

As mentioned in previous sections, an agent can maintain locally protected knowledge that is not distributed. In case of conflict, a similar TC value is calculated for this subset of the agent's theory, called locally protected knowledge contribution (LPKC). The LPKC takes values from *1-5* if the agent uses its protected knowledge within the conflicting reasoning process, otherwise it is *0*.

The final certainty degree is the weighted summation of TC and LPKC. The usage of private information is considered more important, thus LPKC gains higher weight. Each agent calculates its own certainty degree for the conflicting variables, which is sent to the responsible MA that resolves the affair along with variables' states. The MA determines the final state according to the information that is contributed by the agent with the higher certainty degree. In case of tie, the MA can take into consideration the agents' role or hierarchy. The rest agents adopt the outcome and adjust their local point of view.

If, nonetheless, every resolution phase fails, the set of the conflicting variables that cause the uncertainty are excluded. The current global state is then computed for the remaining variables, keeping no knowledge for the excluded ones.

The Section 10 describes the real deployment of SPD-Safe and the management of embedded platforms. The MAS control the underlying devices while build-in security mechanisms protect the overall operation.

7. Implementation – The Smart Home Use Case

SPD-Safe consists of two components: CompoSecReasoner and AmbISPDM. The abovementioned technologies are integrated in order to implement the composed SPD multi-metric approach of CompoSecReasoner and the management strategies of AmbISPDM.

7.1 The SPD-Safe Platform

The underlying embedded devices and platforms constitute a main building block for a real MAS. This subsection outlines the software implementation of a service-oriented architecture for managing the devices and their services. This functionality is then integrated with the multi-agent platform and enables the dynamic and real-time administration of the system by the intelligent agents. Security is enhanced with build-in defence mechanisms.

7.1.1 Device Service-Oriented Architecture

As SoAs are widely used, there is now an effort to apply them in embedded systems as well (e.g. [58], [59]). To enable these services in such systems, several standards and platforms have been proposed. The Devices Profile for Web Services (DPWS)[6] OASIS standard targets resource-constrained embedded devices and enables secure web service messaging, discovery, description, and eventing.

The Open Service Gateway initiative (OSGi)[7] is a standard module system and service platform in Java and implements a complete and dynamic component model. Components are modelled as bundles for deployment, which can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot.

In the proposed framework, DPWS is integrated into OSGi by implementing operators which are controlled by the relevant system components via an OSGi interface, as it will be detailed in following subsections. Embedded devices specify their type and the provided services in DPWS. Each system component implements a component operator bundle that handles its underlying DPWS devices and their services.

The next subsection defines the system architecture layers and how they are deployed by the proposed setting.

7.1.2 Embedded System Architecture

In order to effectively model embedded systems and their components, the architecture is segregated into layers. The EU funded project nSHIELD examined novel multi-layer architectures for heterogeneous embedded systems. It defined 4 layers for modelling such settings.

From bottom-up, the node layer represents all the embedded devices themselves (e.g BeagleBone, MemSic Iris). The network layer consists of nodes connected in networks. The

[6] OASIS, DPWS: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf
[7] OSGi Alliance: http://www.osgi.org/

middleware layer is the management-software of the networks. OSGi operates as the middleware. The overlay layer is formed by agents, who control distinct sub-systems and exchange high-level SPD- and safety-related information. The agents and the overlay are implemented in the JADE platform. Figure 9 illustrates the proposed system architecture.

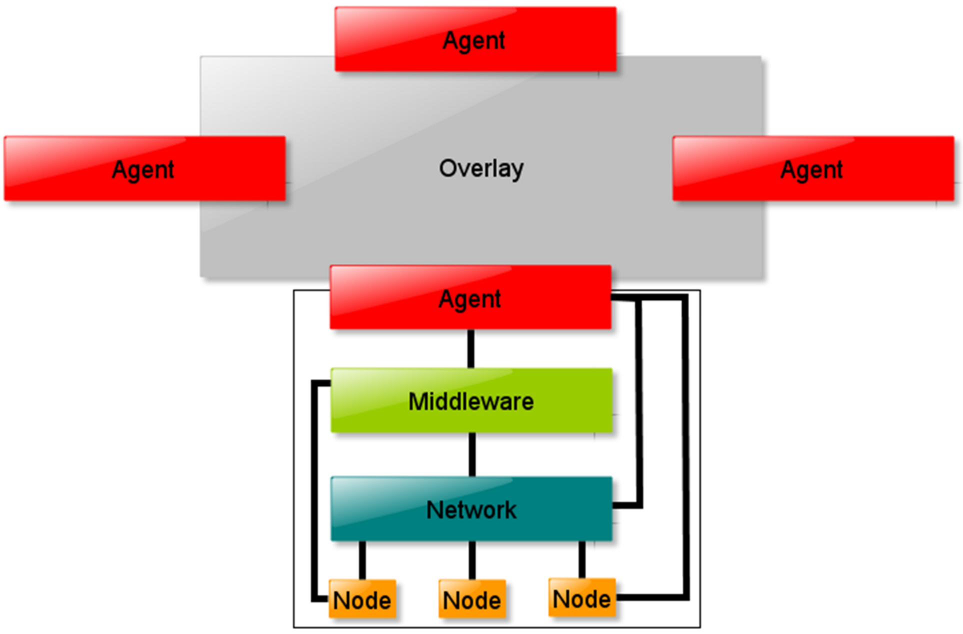


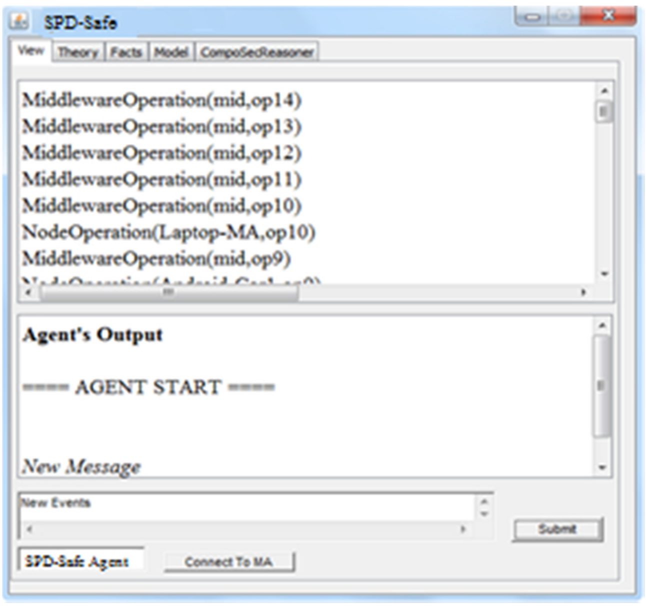
Figure 9 The layers of a typical embedded system architecture

The subsection below presents the integration of the multi-agent reasoning process, which was presented in the Section 7, and the embedded system management platform.

7.1.3 Integration of MAS with the Embedded System Platform

The proposed framework is a MAS, implemented in the JADE platform, which, via the OSGi middleware is able monitor and manage DPWS devices. The AI reasoning process is based on EC. The context theory is aware of the pertinent SPD metrics of the underlying embedded equipment and the safety rules of the system (ambient information, reactive plans and strategies, safety goals).

The core of the reasoning process is an event-based model checker [15] which extends the EC. This component is implemented in Java, using the Jess rule engine. Moreover, WSDL defines the SPD and safety related ontologies, and CAP models the system messages. The whole reasoning process is transformed into a JADE agent's reasoning behavior and is implemented as a multi-agent reasoner. Figure 10 illustrates the SPD-Safe JADE agent API.



```
MiddlewareOperation(mid,op14)
MiddlewareOperation(mid,op13)
MiddlewareOperation(mid,op12)
MiddlewareOperation(mid,op11)
MiddlewareOperation(mid,op10)
NodeOperation(Laptop-MA,op10)
MiddlewareOperation(mid,op9)
```

Agent's Output

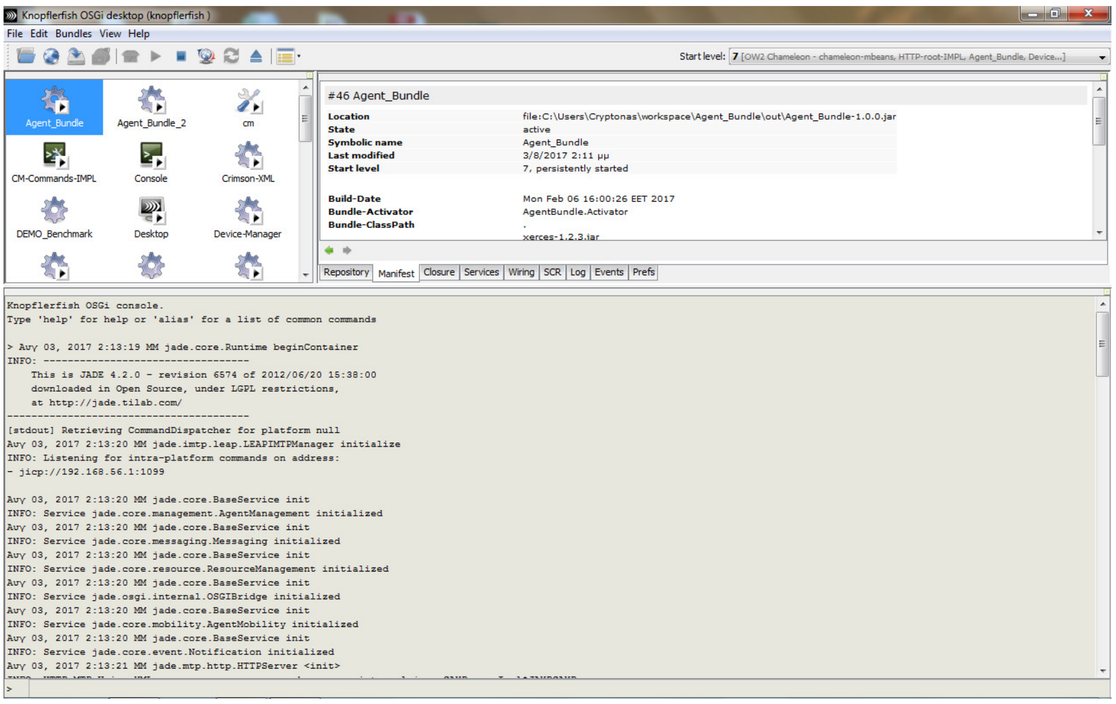
```
==== AGENT START ====

New Message
```

Figure 10 The SPD-Safe JADE agent GUI

The agents are then encapsulated into OSGi bundles and are deployed on Knopflerfish[8], an open-source implementation of OSGi. Each agent controls a distinct system via the OSGi middleware. Figure 11 depicts the Knopflerfish OSGi middleware and the JADE-agent bundles. The various agents communicate with each other via JADE using the standardized ACL language, with messages containing CAP events.

[8] Makewave, Knopflerfish: http://www.knopflerfish.org/


Knopflerfish OSGi desktop (knopflerfish)
File Edit Bundles View Help
Start level: 7 [OW2 Chameleon - chameleon-mbeans, HTTP-root-IMPL, Agent_Bundle, Device...]
Agent_Bundle
Agent_Bundle_2
cm
CM-Commands-IMPL
Console
Crimson-XML
DEMO_Benchmark
Desktop
Device-Manager
Repository
Manifest
Closure
Services
Wiring
SCR
Log
Events
Prefs

#46 Agent_Bundle

```
Location           file:C:\Users\Cryptonas\workspace\Agent_Bundle\out\Agent_Bundle-1.0.0.jar
State              active
Symbolic name      Agent_Bundle
Last modified      3/8/2017 2:11 μμ
Start level        7, persistently started

Build-Date         Mon Feb 06 16:00:26 EET 2017
Bundle-Activator   AgentBundle.Activator
Bundle-ClassPath   .
                   xerces-1.2.3.jar
```

```
Knopflerfish OSGi console.
Type 'help' for help or 'alias' for a list of common commands

> Αυγ 03, 2017 2:13:19 ΜΜ jade.core.Runtime beginContainer
INFO: ----------------------------------
    This is JADE 4.2.0 - revision 6574 of 2012/06/20 15:38:00
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
----------------------------------------
[stdout] Retrieving CommandDispatcher for platform null
Αυγ 03, 2017 2:13:20 ΜΜ jade.imtp.leap.LEAPIMTPManager initialize
INFO: Listening for intra-platform commands on address:
- jicp://192.168.56.1:1099

Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.core.resource.ResourceManagement initialized
Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.osgi.internal.OSGIBridge initialized
Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
Αυγ 03, 2017 2:13:20 ΜΜ jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
Αυγ 03, 2017 2:13:21 ΜΜ jade.mtp.http.HTTPServer <init>
>
```

Figure 11 Knopflerfish OSGi middleware and JADE-agent bundles GUI

The embedded devices specify their type and provided services using DPWS, which is also used to facilitate the exchange of CAP messages between managed devices and agents. The DPWS-related components are developed using the WS4D-JMEDS API[9] and are also integrated into OSGi bundles. WS4D-JMEDS produces a WSDL file for each deployed DPWS device.

Each underlying system component (node, network, middleware) that communicates directly with the agent, creates a component-operator that controls the component's services. The agent and the component operators exchange well-formed information determined by the CAP scheme.

The next subsection introduces the build-in security mechanisms that are deployed by this setting and protect the system.

7.1.4 Platform Security Mechanisms

From a security perspective, the framework can protect the communication between the managed embedded devices and their respective security agents through the use of the WS-Security standard [60], typically used alongside DPWS. Said standard can provide end-to-end security, non-repudiation, alternative transport bindings, and reverse proxy/common security token in the application layer.

[9] WS4D.org, WS4D-JMEDS DPWS Stack: http://sourceforge.net/projects/ws4d-javame/

With regard to the protection of the communication between agents, the JADE-S add-on can be used to safeguard the exchanged ACL messages, by providing user authentication, agent actions authorization against agent permissions, as well as message signature and encryption.

The OSGi security features can additionally be used to provide inner-platform security for both agents and component operator bundles (i.e. managed devices) by limiting bundle functionality (e.g. which bundles can be started/stopped, when, by whom etc.) to pre-defined capabilities.

Figure 12, illustrates the software layers of the proposed framework, along with the respective security mechanisms.

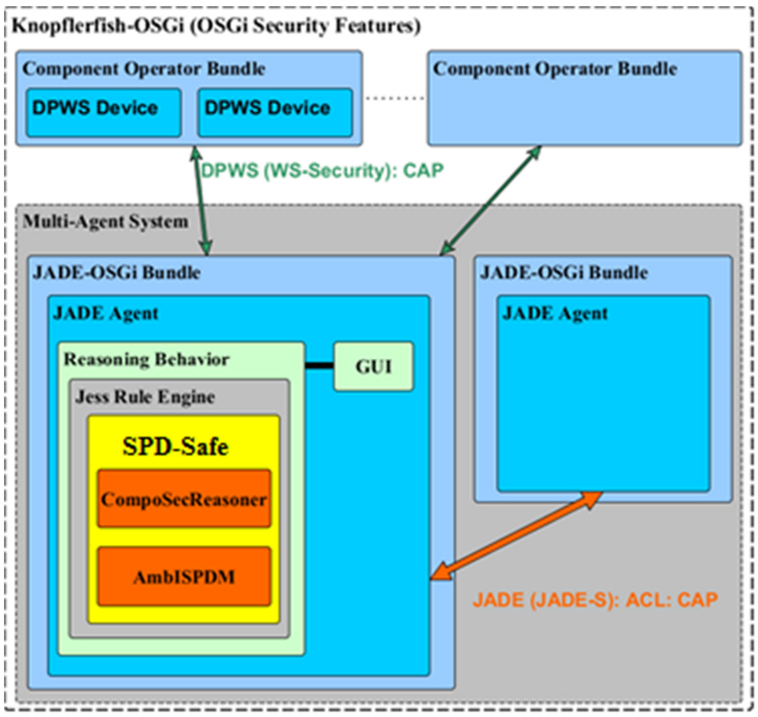


Figure 12 Software layers of SPD-Safe

7.1.5 Additional Security Mechanisms

The current solutions of smart home devices do not provide any secure access control mechanism – the devices respond to every request they are made. This fact enables cyber-attacks as such devices are either connected directly to the Internet or indirectly through the users mobile phone [61]. To enhance security, we apply the secure policy-based access control management framework (PBAC) for heterogeneous embedded system networks that is proposed by [62]. At the device level, the framework imposes access control based on pre-defined security policies for every interaction activity. The policies are securely managed by the back-end framework that runs on a laptop.

Moreover, we utilize the lightweight cryptographic library ULCL [63] to implement a secure storage service. The service encrypts data at device level to prevent information disclosure in case of compromise. The data are encrypted with the cryptographic cipher AES. The lightweight cryptographic solutions are described in more detail in Section 8.

Trust-based schemes are adopted in wireless networking and accomplish secure routing functionality. Reputation is formed by a node's past behavior, revealing its cooperativeness. In secure routing, reputation mainly evaluates the routing and forwarding, the use of encryption and authentication mechanisms, and the proper transmission of acknowledgements per transmitted packet. Trust is the aggregation of the reputation values that the node holds for another entity. It reveals the level of confidence that the node holds about other participants. Entities with high reputation values are considered trustworthy. Legitimate entities depend mostly on trustworthy ones to perform communication activities. On the other hand, low reputation is used for intrusion detection as it reveals selfish or malicious nodes. Legitimate entities avoid disreputable nodes and do not serve their traffic. Motivated by the above, SPD-Safe utilizes a novel trust-base secure routing and authorization system in WSNs. For the network layer protection, the proposed system utilizes the abovementioned security mechanisms are utilized and implements three main components:

- A cryptographic service that enforces authentication, message integrity, and confidentiality
- An efficient trust-based routing scheme that protects the communication against ad-hoc routing attacks
- A policy-based access control framework that provides authorization.

The trust-based routing is detailed in Section 9.

7.2 Embedded Hardware Devices' Classification

The node layer of the nSHIELD architecture includes the embedded devices where the SPD-Safe components are meant to be deployed. In the IoT setting there are various types of devices which exhibit their own intrinsic characteristics in terms of computational capabilities, networking, power consumption, and size. Based on these limitations, the different SPD-Safe technologies are deployed on specific type of hardware.

Four main categories are considered, ranging from powerful computers to lightweight motes:

- **Power** devices
- **Mobile** devices
- **Micro/Embedded** devices
- **Nano/Sensor** devices

The four categories cover main hardware demands of the market areas for several application settings, like home solutions and assist living, smart city and transportation, public and critical infrastructure, and outdoor precision agriculture. The categories are described in the following subsections.

Power devices exhibit high to medium performance in terms of computational capabilities, with no particular resource constraints. The mostly act as gateways to other networks. Nonetheless, the can acquire their own sensory equipment and collect environmental data. Typical examples of power devices include laptops or powerful embedded platforms like, BeagleBoard and BeagleBoard-xM (Figure 13). In SPD-Safe, these devices deploy the overlay and middleware technologies and software platforms, like the JADE multi-agent framework, the Knopflerfish OSGi middleware, the DPWS peers, and the access policies repository.

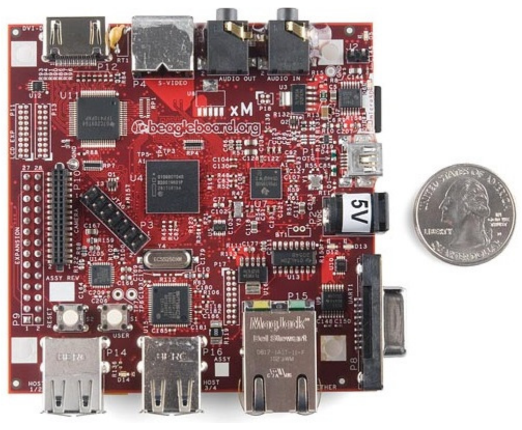


Figure 13 BeagleBoard-xM embedded device

Personal and portable devices. This type includes power devices with energy or other constraints, like a smart phone, a tablet, or a smart watch). In SPD-Safe, they mainly install DPWS clients or servers and the access policy decision and enforcement units.

The embedded/micro devices are smaller and more compact equipment, typically with lower computational and networking capabilities, that are integrated into other entities, like smart vehicles and appliances. BeagleBone (Figure 14) is the ordinary selection in the proposed framework. For SPD-Safe, embedded devices are responsible for DPWS functionality, policy decisions and enforcement, and bridging between 802.15.4/6LoWPAN and IPv4/IPv6 networks.

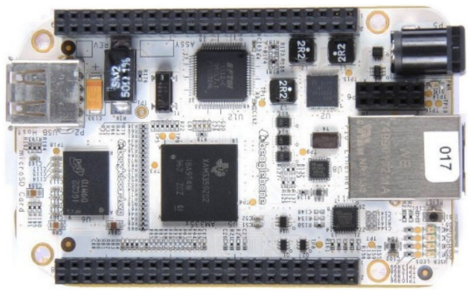


Figure 14 BeagleBone embedded device

Nano devices include small battery-powered sensors and motes with ultra-lightweight computational and networking resources, such as MemSic Iris and Zolertia Z1 (Figure 15). SPD-Safe utilizes nano nodes for the main sensory operation.

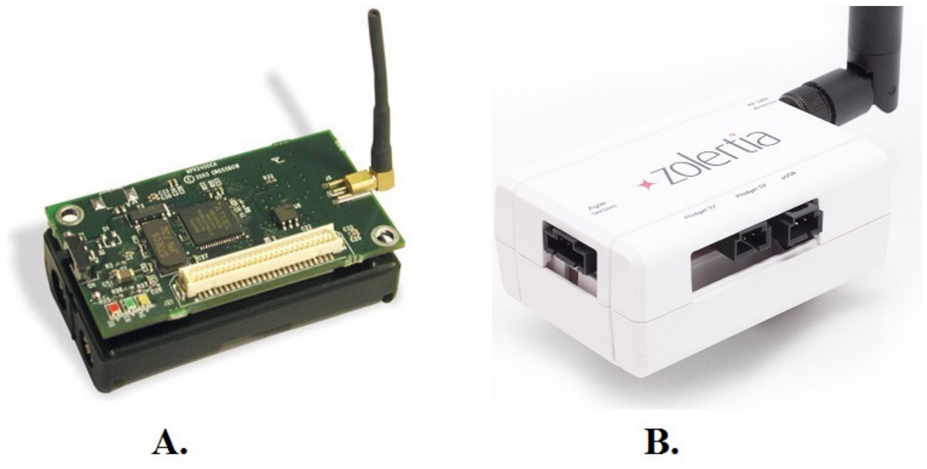


Figure 15 A. MemSic Iris mote device, B. Zolertia Z1 mote device

Table 6 summarizes the technical features of each device type.

Table 6 Technical features of the equipped embedded devices

Device type	Power			Mobile		Micro	Nano	
Device Model	**Laptop**	**BeagleBoard-xM**	**BeagleBoard**	**Tablet**	**Smart phone**	**BeagleBone**	**Zolertia Z1**	**MemSic Iris**
Operating system	Linux / Windows	Linux / WinCE	Linux / WinCE	Android	Android	Linux	Contiki/ Tiny OS	Contiki / Tiny OS
Processor	Quad-core i5 at 2.6 GHz	DSP at 1 GHz (DM3730)	MPU at 720 MHz (OMAP)	Quad-core at 1.9 GHz	Single-core Cortex-A5 at 600 MHz	MPU at 720 MHz (OMAP)	MSP430 16 MHz	Atmel ATMega 1281 at 16 MHz
Storage	512 GB, SD / MMC / SDIO card slot	SD / MMC / SDIO card slot	SD / MMC / SDIO card slot	32 GB, SD / MMC / SDIO card slot	16 GB SD / MMC / SDIO card slot	SD / MMC / SDIO card slot	2 MB flash memory	128 KB program flash memory, 512 KB measurement (serial) flash
Memory	8 GB RAM	512 MB RAM	256 MB RAM	2 GB RAM	512 MB RAM	256 MB RAM	8KB RAM, 92 KB flash memory	8 KB RAM, 4 KB EEPROM
Networking	Wired Ethernet, WiFi, Bluetooth	Wired Ethernet, No built-in wireless module (but has interface)	Wired Ethernet, No built-in wireless module (but has interface)	WiFi, Bluetooth	WiFi, Bluetooth	Wired Ethernet, No built-in wireless module (but has interface)	2.4 GHz radio	2.4 GHz radio
Power	Built-in battery / DC	USB / DC	USB / DC	Built-in battery / DC	Built-in battery / DC	USB / DC	Batteries (2×AA)	Batteries (2×AA)

The main embedded system for SPD-Safe consists of BeagleBone and BeagleBoard devices[10] (nodes with computational constrained capabilities) and a laptop (power node which controls the system). The BeagleBone/BeagleBoard devices monitor environment parameters (e.g. temperature and lightness) and control electrical devices (e.g. air-condition, room lights, fridge) and WSNs of nano devices (i.e. Iris or Z1). The BeagleBones/BeagleBoard control their underlying electrical devices based on polices that are imposed by PBAC. Alternative the user

[10] BeagleBone.org, BeagleBone device manual: http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE SRM.pdf

can access this functionality through a personal mobile device. The laptop runs the management framework for controlling the system, collects all pieces of information and presents them to the user. The functionality that is provided by each device is deployed as a service. All devices communicate wirelessly with WiFi and IPsec. The data are encrypted and different levels of security are supported for low-energy consumption or high security.

SPD-Safe runs on the laptop along with the management framework of PBAC. AmbISPDM acts as the main reasoning behavior of the system's SPD-Safe agent and communicates with other agents of other smart buildings in the overlay layer. The management framework is the middleware layer that controls the local ambient environment. The devices are interconnected and form networks. The BeagleBone/BeagleBoard, personal mobile device, nano sensory devices, and the laptop are the nodes.

7.3 Performance Evaluation

An experimental test-bed was used to evaluate SPD-Safe. An Android application was developed where the user could manage the SPD level of the whole setting. It was deployed on a smart phone (Android 5.0 KNOX OS, 2.5GHz quad-core CPU, 16GB RAM, Wi-Fi, 4G) connecting the smart home LAN via WiFi. PEP and the electric device control software was deployed on BeagleBone devices (Ubuntu Linux OS, 720MHz ARM Cortex-A8 processor, 256MB RAM, USD-WiFi). Two Beagle are used which are connected to the LAN via USB-WiFi. The Beaglebone is equipped with the weather cape for monitoring environmental parameters (i.e. temperature, barometric pressure, humidity and ambient light) and is power-supplied by a battery cape. The BeagleBoard controls a surveillance USB-camera and is plugged to the smart home power supply. The PBAC framework and SPD-Safe was deployed on a laptop (Windows 8.1 Pro OS, 2.1GHz Inter Core i-7 CPU, 8GB RAM, WiFi). Figure 16, depicts the demo and test-bed settings.

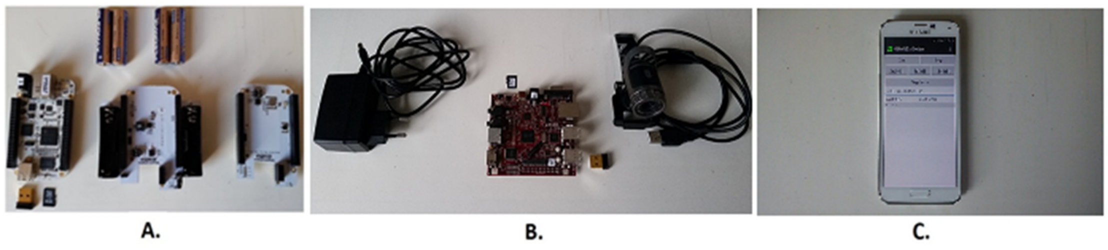


Figure 16 Set of embedded devices for the smart home use case

We utilize the smart phone application to trigger the benchmark where the user changes the SPD state of the devices (e.g. IPsec and PBAC SPD levels). We evaluate the impact of these changes on the computational complexity of SPD-Safe and the responsiveness of the system.

In terms of performance, JESS uses the RETE algorithm for optimized speed and is a very efficient pattern matching tool. The reasoning theory of SPD-Safe on JESS consists of around *30* rules and *400* facts. It requires *45MB* RAM and *1.87MB* code size. A reasoning process takes *1.6* seconds on average.

The most significant performance parameter is the delay that is experienced by the user when he attempts to retrieve information or SPD changes occur. Figure 17, illustrates the response time for the concurrent requests that are performed by the management application. The spikes represent the SPD changes (i.e. incoming SA requests, configure system and inform the SA once the changes take place). These changes introduce significant delay, but they should be rare during normal operation. The system exhibits an acceptable response time, even for real-time operation.

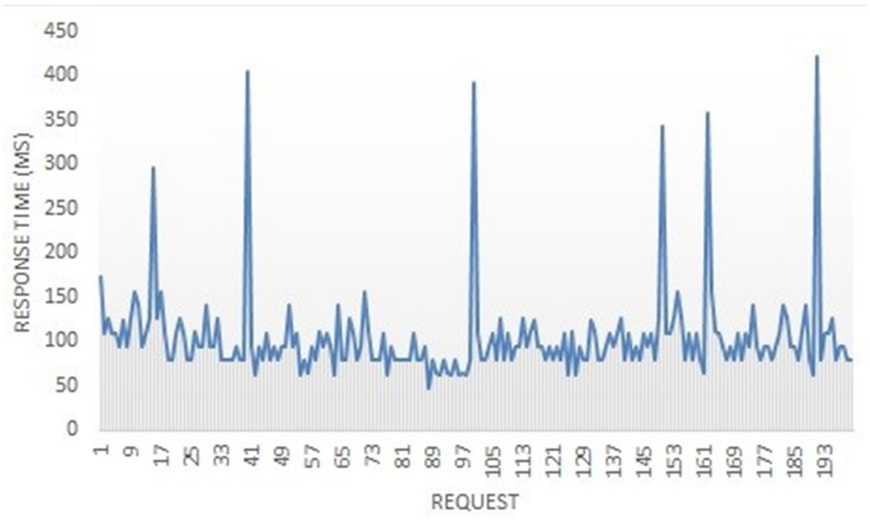


Figure 17 Performance evaluation of SPD-Safe – Response time (in ms) per request on embedded devices

In terms of scalability, the computational complexity is determined by the average number of patterns per rule left-hand-side (*P*), the number of rules (*R*) and the number of facts (*F*) on the working memory. Complexity is linearly affected by the working memory size and is of the order of *O(PRF)*. The pattern matching ratio for SPD-Safe is low (in the order of one to three) – each entity is assigned a unique identifier and the events affect specifically declared parameters. The number of the theory rules is also kept low (around *30* rules). Only the number of facts affects scalability. In the smart home scenario, every new system component requires about *5-10* facts to be modelled, resulting in less than *40* extra bytes in RAM for the SPD agent and low computational overhead (in the nanosecond range).

7.4 Comparison with Existing Systems

This subsection compares the SPD-Safe with relevant approaches in terms of the underlying technologies and the reasoning capabilities.

7.4.1 Multi-Agent Systems

Agent technologies are the ordinary choice to model ambient intelligence entities that interact with the user and the environment [64]. Agents monitor the environment and take action during normal operation as events occur.

Applications for inner spaces like smart home settings are common, especially in European and Japanese research communities, with applications focusing in e-health, assisting living, and independent elder living systems. Context-aware agents process data originated from embedded devices which are deployed in the surrounding environment. The belief-desire-intention (BDI) software model [65] is applied for the agent design, like in the systems AmIciTy [17], Confidence [58], and AmIlab [66]. The in-house activity is continuously monitored and analyzed by the agents to recognize the user's health state (e.g. faint or fall injury) and raise relevant alarms.

Systems for outer spaces are also proposed [59], [67], [68], [69]. Applications include precision agriculture, airport or flight support, and space exploration. These systems are open and dynamic, while mobility is also important. The agents controls cyber-physical systems (CPS) that may include UAVs or telescopes. The use of network simulators is common for the evaluation and the improvement of a candidate solution, like the systems AGENTFLY [69] and the spraying pesticides aircraft [59].

In the industrial domain, only a few multi-agent systems (MAS) are devised, deployed and adopted [70]. The case study of Italy reveals that several industries follow this approach [71]. Magenta monitors boats in marine reserves and helps shipping companies to improve oil distribution shipping networks. Telecom Italia MAS guides technicians performing maintenance operations in the telecommunication network. Living Systems Adaptive Transportation Networks applies a logistics management system for heat and sequence optimization in the supply chain of steel production. Skoda Auto control system management deploys a MAS in mass-production planning of car engines. Similar results are reported for the industries in USA [72]. Despite the strong industrial involvement, the full potential of the agent technology is not fully utilized and not all the developed agent concepts and techniques are completely utilized in practice.

The aforementioned systems utilize semantic representation and reasoning. The service-oriented designs comply with specific application particularities. Nonetheless, in all these deployments, each agent is uniquely responsible for specific tasks and activities. In most cases, no conflicting patterns are considered among the communicating entities, restricting the applicability of the relevant designs.

A review of intelligent techniques for resolving conflicts of knowledge in MAS is presented in [19]. Data fusion and trust mappings are popular solutions for relational reasoning while negotiation techniques are applied in epistemic reasoning.

Markov Logic Networks (MLN) are utilized for rule-based conflict resolution in data integration [73]. For weak affairs, where the variation of the conflicting values is low, the resolution is determined by voting. For high degree affairs, a MLN is trained to determine the result. However, the training process is time consuming and not applicable in real-time applications.

AmIlab is a research smart home installation [66]. The reasoning is modelled in the rule language Event Condition Action (ECA). The entities' types, properties, and relations are reported in a BlackBoard structure. Open affairs are resolved based on trust, the natural hierarchies of the system entities, and the user preferences.

The Air Force Research Laboratory in New York deploys the AGENTFLY – an UAV deconfliction MAS [69]. Each UAV implements a controlling agent that inspects the fight plan. When another flying object is going to penetrate the flying route, the agent starts a negotiation process to avoid any threat or accident, by safely re-routing the flights. Each agent proposes alternative routes based on its preferences and flying capabilities (e.g. fuel level). The system is built on top of the AGLOBE agent platform. The simulation analysis is based on a trace of real air-traffic from the Los Angeles International Airport for modelling non-cooperative flying objects.

In all these cases, the conflict resolution mechanisms are based either on relational or on epistemic reasoning. Relational solutions are more common, benefiting from the fast resolution process and the ease to implement them. The epistemic solutions produce more robust results and long term stability, and are preferred in mission-critical applications. SPD-Safe, combines the two approaches in an attempt to optimize the conflict resolution process and enhance the accuracy of the reasoning outcome.

7.4.2 Agent Technologies and Middleware Platforms

A comparative analysis for deductive rule engines is held in [74]. Jess and the engines ILOG, OPSJ, Microsoft, Business Rule Engine, and Drools are tested over common benchmarks on Windows XP. It is concluded that Jess is efficient and handles dynamic facts and dynamism in rules and variables. The engine is suitable for the mission critical technologies of NASA [67] and is applied in several research applications.

Communication protocols for IoT (DPWS, CoAP, MQTT) are examined in [75]. DPWS is the benchmark in terms of the ease in designing. It is robust with flexible discovery, subscription, and eventing. A minimal setup phase is required, and the entities then discover each other and interact seamlessly.

JADE and 23 other agent platforms, like Jason and AGLOBE [69], are compared in [76]. JADE is user-friendly with a useful GUI and is easy to learn. It fully complies with the FIPA and Semantic Web standards, like asynchronous ACL. The platform is portable and compatible with any JVM. The developer releases are open-source, stable, and robust, and support many programming languages (e.g. Java, Jess, RuleML, and Prolog). Moreover, JADE is efficient with high scalability. It enables very quick and fast agent communication.

In terms of security, the platform provides strong user authentication. It deploys the aforementioned encryption and signature mechanisms along with HTTPS support.

JADE has been applied among others in general purpose distributed simulations, mobile computing, reasoning in multiple domains, artificial life and behavioral observation, economics and e-commerce.

The analysis concludes that JADE still remains the most popular platform since it is purely designed in Java and supports different kinds of systems operating in the web. Moreover, the platform is supervised by five reputable organizations: Motorola, France Telecom, TILAB, Whitestein TEchnologies AG, and Profactor.

7.4.3 Reasoning Capabilities and Conflict Resolution

According to our knowledge, no system for AmI offers the combination of the aforementioned reasoning capabilities of SPD-Safe. The core process of each individual agent supports automated epistemic, temporal, and casual reasoning with context-sensitive effects of events, indirect effects, action preconditions, the common law of inertia, real-time events, and rule priorities in dynamic, uncertain, or partially-known domains.

The share theories are introduced as the main mechanism of conflict resolution. Besides the complexity of constructing a share theory in a dynamically evolving environment, errors can occur. Each local theory is considered consistent. However, their integration does not always outcomes a globally consistent theory.

For example, consider the scenario where two agents raise a conflict for the state of the variable a. The agents SA_1 and SA_2 contribute with the following theories:

SA_1	SA_2	$Share\ Theory$
$r_1 : b \rightarrow a$	$r_2 : b \rightarrow \neg a$	$r_1 : b \rightarrow a$
$HoldsAt: b$	$HoldsAt: b$	$r_2 : b \rightarrow \neg a$

The resulting share theory is inconsistent. For SPD-Safe, it is assumed that the theory developers are responsible for keeping both local and share theories consistent [77], but such matters remain possible.

Moreover, when locally protected knowledge is interfered, the agents cannot argue about their point of view. In these cases, grading mechanisms can resolve the conflict coherently. The certainty degree is proposed in such cases or for fast resolution of affairs without creating the relevant share theories. Other approaches that consider grading mechanisms [19], [78], [79] apply genetic algorithms, fuzzy systems, the Bayesian law or the Kappa calculus. These mechanisms try to predict how likely or unlikely a possible world state may be. However, as with the certainty degree, errors can occur. The features that in each case give points to a possible state are usually subjective. So if agent theories cannot be rated well, a share theory approach must be used.

A distributed reasoning algorithm with conflicts for P2P systems is proposed in [80], where such problems are resolved by using trust information for the involving nodes. Rules originated by trusted nodes are taken into account, restoring the system's inconsistency. SPD-Safe could support these trust-based policies in the agent domain by utilizing the abovementioned rules' priorities reasoning mechanisms. In the AMI domain, agents' roles or hierarchy are also common in different multi-agent settings. For the scenarios that are examined in this study, the approach is applied with the agents' hierarchy feature. Thus, when the conflict cannot be resolved by a share theory or the certainty degree, the agent's theory that is higher in the hierarchy is considered. Although this mechanism can resolve inconsistency issues, false conclusions can still be deduced. Nonetheless, the system will remain at least globally coherent in all cases.

The proposed SPD-Safe uses the certainty degree when reasoning with locally protected knowledge is needed and the share theories in all other cases. When the two mechanisms fail two resolve the conflict, the agents' hierarchy can be taken into account to retain coherency.

Table 7 summarizes the evaluation results of SPD-Safe and 5 MAS for AmI applications (AmIciTy [17], Confidence [58], Spraying Pesticides [59], AmIlab [66], AGENTFLY [69]).

Table 7 Ambient Intelligence systems' features

Aspects	SPD-Safe	AmIciTy	Confidence	Spraying Pesticides	AmIlab	AGENTFLY
Context-aware	YES	YES	YES	NO	YES	NO
Self-organizing	YES	YES	NO	NO	YES	YES
Agent platform	JADE	NO	NO	NO	NO	AGLOBE
Cooperation	Message exchange	Message exchange	NO	NO	Black Board	NO
Embedded devices	BeagleBone, BeagleBoard, MemSic Iris, Zolertia Z1, Smart phone	Embedded devices	Xsens sensors	UAV, WSN	NO	UAV
Devices management platform	OSGi, DPWS	NO	NO	NO	NO	NO
Semantic technologies	EC, CAP, XACML, WSDL	XML	OWL	NO	ECA	YES
Reasoning engine	Jess	Java	Java (ANN)	C++	NO	NO
Conflict resolution	Epistemic and relational	NO	NO	NO	Relational	Epistemic
Build-in security	JADE-S, OSGi security, WS-Security, PBAC	NO	NO	NO	NO	NO
SPD	YES	NO	NO	NO	NO	NO

In all five cases no embedded devices management platform is utilized. This issue is restrictive in the era of IoT, where many and heterogeneous devices must be deployed and cooperate. Popular agent platforms that support standards and offer adequate and efficient agent-related functionality are also ignored by most systems. The reasoning process is implemented by general purpose programming languages neglecting the benefits of logic processing engines. The conflict resolution, if any, is based either on epistemic or relational reasoning. No build-in security features are used and the systems do not protect security, privacy, or dependability.

SPD-Safe is a novel system regarding SPD management in AmI applications. It is the first attempt to safeguard these properties in a configurable and dynamic domain. The SPD formation is based on well-structured metrics that evaluate the different system settings. The AI process adjusts the system at runtime to counter related threats and attacks. As aforementioned, SPD-Safe utilizes the best choices regarding platforms and technologies for implementing reasoning, and agent and embedded system management. The conflict resolution mechanism incorporates both epistemic and relational reasoning. Several semantic technologies and standards are also supported.

8. Lightweight Cryptographic Modules Implementation

This section details the additional cryptographic operations that are implemented for SPD-Safe. The cryptographic solutions include a software crypto-library, a lightweight password-hashing primitive for WSN, and an authenticated encryption primitive in hardware.

8.1 Crypto-Library

The evolution of embedded systems and their applications in every daily activity, derive the development of lightweight cryptography. Widely used crypto-libraries are too large to fit on constrained devices, like sensor nodes. Also, such libraries provide redundant functionality as each lightweight and ultra-lightweight application utilizes a limited and specific set of crypto-primitives and protocols. This subsection presents the ULCL crypto-library for embedded systems. It is a compact software cryptographic library, optimized for space and performance. The library is a collection of open source ciphers (27 overall primitives).

8.1.1 ULCL

The library provides *'built-in'* cryptographic functionalities for embedded systems that make use of a specific set of cryptographic primitives and protocols. It utilizes open source ciphers' implementations, two lightweight APIs and a configurable compilation process.

Only block/stream ciphers and hash functions are included. The library provides basic cryptographic functionality for constrained and ultra-constrained devices. It targets on application environments where asymmetric cryptography cannot be applied. As asymmetric cryptography is much more resource demanding than symmetric one, these applications depend on dependable authentic key distribution mechanisms [81]. Such mechanisms are lightweight key management solutions that utilize only symmetric cryptography.

ULCL utilizes open source implementations of known ciphers. It is a collection of lightweight or compact implementations of standard block/stream ciphers and message authentication code (MAC) primitives. In [82], all these primitives are evaluated and the best of them are proposed for different types of embedded devices.

For every crypto-primitive type (block cipher, stream cipher, MAC), we implement a common API for utilizing all the different primitives with their parameters. The API was designed with developers in mind and is easy-to-use. There are common functions in each category for initialization and processing. All size values are measured in bytes.

Moreover, we implement a second API for developers that are not familiar with cryptography. The developers do not deal with selecting crypto-primitives and their parameters. When this simple API is compiled, ULCL performs a test program to figure out the system's capabilities in memory and processing capabilities. According to this test, ULCL invokes internally the most appropriate primitives for this device. The API is suitable for cryptographic applications in homogeneous systems. Also, it is appropriate for educational purposes in computer security. All the function arguments are strings and contain hexadecimal numbers in string form.

The main novelty is the configurable compilation process. A user can define an exact set of crypto-primitives that are compiled without compiling the whole library. Thus, the executable code that runs on the embedded device is small. For example a user can compile only a compact AES implementation and use it through the common API for block ciphers. As embedded devices run a specific set of protocols and crypto-primitives, our configurable implementation of lightweight primitives is a good candidate library.

The block ciphers API occupies 5.22-23.5 KB of ROM memory. The overhead is high as we implement the modes of operation and the padding schemes except from the API for encryption/decryption. The stream ciphers API occupies 1.74-7.4 KB and the hash functions API occupies 1.75-7.4 KB. The total API overhead is about 1.74-38.3 KB and the library occupies 4-516.7 KB.

Figure 18, illustrates the segmented compilation process. Each box represents a different compilation option. For example, a user can compile the whole library, the block ciphers or specific crypto-primitives.

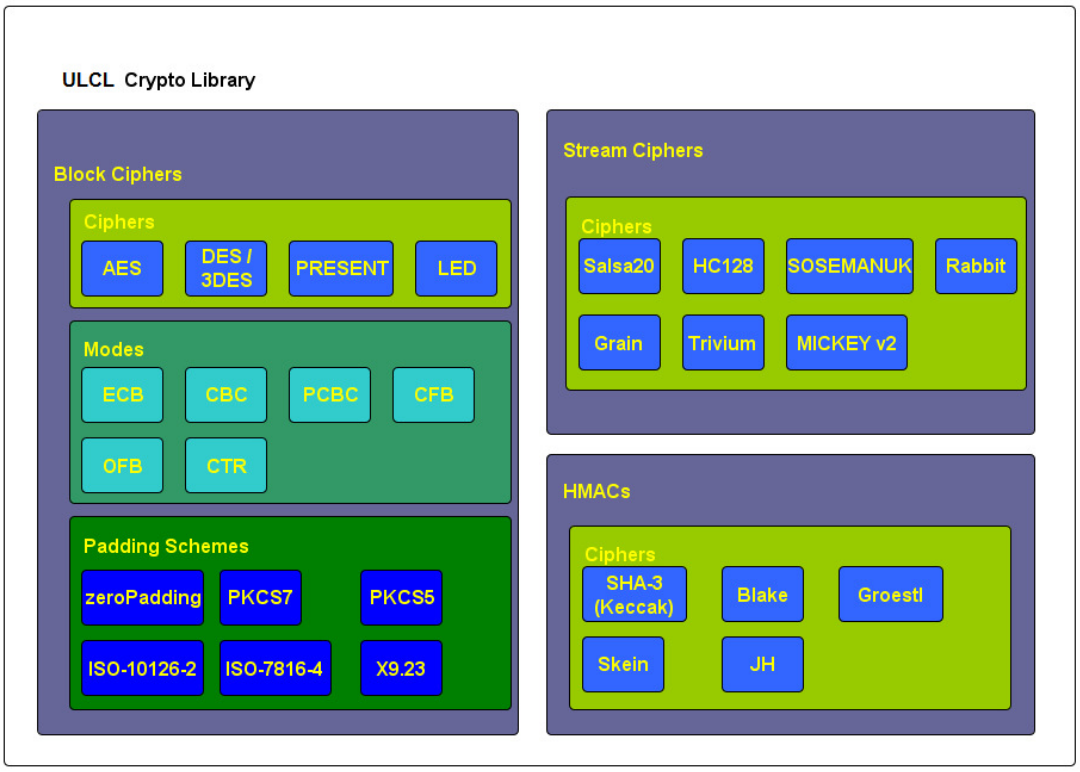


Figure 18 Segmented compilation of the ULCL crypto library

ULCL is open source and well-documented. The library contains a series of examples and test files. The examples demonstrate the compilation process capabilities and the utilization of the APIs. A user can execute command line tests for each compiled primitive. Furthermore, a benchmark suite is provided for measuring the library's features in the application setting. The APIs performs extensive error checking and reports relative error codes. The correct functionality of the whole library is validated. Each crypto-primitive is verified through the manufacturer's test vectors and common tests with well-known libraries, like OpenSSL.

8.1.2 Evaluation

The measurements that are reported in following subsections were performed on a PC with Intel core 2 duo e8400 (3GHz), 2GB of RAM and Linux operating system.

AES and Camellia are designed for mainstream applications with high level of security and throughput. Their space requirements are high for ultra-constrained devices and are included only for more powerful embedded devices. DES, 3DES, and CLEFIA are designed for lightweight cryptography on embedded systems and support high and moderate level of security, lower throughput and consume less computational resources. They are appropriate for constrained devices. PRESENT, XTEA, and XXTEA are designed for ultra-constrained devices with even lower level of security and throughput and are efficient in power-energy-memory. LED, KATAN, and KTANTAN are mainly implemented in ultra-constrained hardware and their performance in software is low. We include them for higher compatibility in heterogeneous systems where nodes may use hardware-accelerated cryptography.

All block ciphers operate in ECB, CBC, and CTR modes of operation [83]. Furthermore, the library supports all known padding schemes: zeroPadding, PKCS5, PKCS7, ISO_10126-2, ISO_7816-4, and X9.23.

Table 8, summarizes the block ciphers' features. The ciphers are executed in ECB node of operation with zero padding.

Table 8 Software implementations of block ciphers in ECM mode of operation with zero padding in ULCL

Cipher	Key (bits)	Block (bits)	Code (KB)	RAM (KB)	Throughput (MBps)
AES	128	128	25	10.25	56.35
AES	192	128	25	10.33	63.03
AES	256	128	25	10.41	69.82
3DES	64	64	12	10.02	20.15
3DES	128	64	12	10.07	20.25
3DES	192	64	12	10.11	20.21
Camellia	128	128	33	10.08	68.31
Camellia	192	128	33	10.13	55.57
Camellia	256	128	33	10.18	55.17
CLEFIA	128	128	6.9	10.08	4.65
CLEFIA	192	128	6.9	10.13	3.83
CLEFIA	256	128	6.9	10.17	3.29
XTEA	128	64	2.7	10.06	26.07
PRESENT	80	64	2.6	14.46	0.44
PRESENT	128	64	2.6	14.50	0.44

RC4 is the most widely used stream cipher. It achieves high throughput but it is considered insecure for new applications. The finalists Salsa20, Rabbit, HC128, and SOSEMANUK are designed for software. They achieve higher throughput and are considered secure against all attacks faster than the exhaustive search. Salsa20 and Rabbit are the most attractive for constrained devices. HC128 utilizes two large tables to perform encryption/decryption. Due to its table-driven approach it is very fast but requires much memory. The finalists Grain, Trivium, and MICKEY v2 are designed for hardware but perform reasonable in software. They were also cryptanalyzed and found secure. Table 9 summarizes the stream ciphers' features.

Table 9 Software implementations of stream ciphers in ULCL

Cipher	Key / IV (bits)	Code (KB)	RAM (KB)	Throughput (MBps)
HC128	128/ 128	7.9	16.58	517.60
Rabbit	128 / 64	3.1	8.41	264.29
Salsa20	128 / 64	3.1	8.27	155.10
Salsa20	256 / 64	3.1	8.35	154.97
SOSEMANUK	128 / 128	15	9.07	300.50
SOSEMANUK	256 / 128	15	9.14	299.03
Grain	80 / 64	2.7	16.26	64.24
Grain-128	128 / 96	3	24.4	91.39
MICKEY v2	80 / 80	3.2	8.22	2.85
Trivium	80 / 80	6.9	8.25	180.43
ARC4	40 / 0	1.22	8.55	165.69
ARC4	2048 / 0	1.22	9.78	164.50

MD5 and SHA-2 are the most known MACs for mainstream applications. MD5 is not collision resistant, thus less secure, but is very fast. SHA-2 is still secure and the SHA-3 contest targeted to establish an alternative standard. The SHA-3 functions are newer progressive MACs that adopt different design features than SHA-2. Table 10 summarizes the MACs' features.

Table 10 Software implementations of MACs in ULCL

Hash	Digest (bits)	Code (KB)	RAM (KB)	Throughput (MBps)
Blake	224	17	2.47	94.86
Blake	256	17	2.48	95.66
Blake	384	17	2.53	36.63
Blake	512	17	2.57	36.91
Groestl	224	29	2.46	29.62
Groestl	256	29	2.47	29.82
Groestl	384	29	2.83	20.08
Groestl	512	29	2.87	20.13
JH	224	7.7	2.32	17.26
JH	256	7.7	2.33	17.26
JH	384	7.7	2.38	17.27
JH	512	7.7	2.42	17.25
Keccak	224	68.1	2.54	62.12
Keccak	256	68.1	2.55	62.71
Keccak	384	68.1	2.6	52.57
Keccak	512	68.1	2.65	36.80
Skein	256	52.4	2.39	61.51
Skein	512	52.4	2.48	61.35
Skein	1024	52.4	2.67	46.37
MD5	128	3.3	2.15	308.64
SHA-1	160	6	2.17	167.85

SHA256	256	3.7	2.22	95.44
SHA512	512	17	2.41	42.82

ULCL is appropriate for lightweight and ultra-lightweight applications where specific cryptographic primitives are required. It provides basic cryptographic functionality and supports progressive ciphers. The APIs achieve low overhead and comparable overall performance while remain easy-to-use even by developers that are not familiar with cryptography. The size of the executable code is the smallest possible as the compilation is adjusted to the application scenario. The library is open source.

8.2 Lightweight Password-Hashing

Passwords constitute the main mean for authentication in computer systems. In order to maintain the user-related information at the service provider end, password-hashing schemes (PHS) are utilized. The limited and old-fashioned solutions led the international cryptographic community to conduct the Password-Hashing Competition (PHC). The competition proposed a small portfolio of schemes suitable for widespread usage in 2015 [84]. Embedded systems form a special application domain, utilizing devices with inherent computational limitations.

Regarding passwords, embedded systems maintain a small amount of authentication-related data. Device-to-device and short-term communication forms the most common interaction (e.g. in wireless sensor networks) [85], making session key deviation from passwords a desirable goal to enhance security. The garbage-collector attacks [86] can be countered by build-in memory safety techniques, specifically designed for embedded applications [87].

Lightweight cryptography focuses in designing schemes for such devices and targets moderate levels of security. In this subsection, a lightweight poly PHS suitable for lightweight cryptography is presented. At first, two lightweight versions are designed for the PHC schemes Catena and PolyPassHash. Then, they are integrated and implement the proposed scheme – called LightPolyPHS. The schemes are applied on a MANET with BeagleBone embedded devices and a fair comparison with similar proposals on mainstream computer is presented.

8.2.1 Poly Password-Hashing

Strong PHS can protect the password data that are maintained at the service-end. However, attackers have proven themselves adept at cracking large amounts of passwords once the stored data is compromised.

To further fortify security and harden attackers' cracking capabilities, poly (many) password-hashing (PPH) schemes have been recently proposed. They leverage cryptographic hashing and threshold cryptography by combining strong PHS with shares.

Cryptographic hashing and PHSs are described in the previous subsection. A cryptographic *(k,n)*-threshold scheme protects secret information, by deriving n different shares from this information. The threshold determines how any k shares out of total n can recover the secret information. If fewer than k shares are known, no secret information is disclosed.

The Shamir Secret Sharing (SSS) [88] is a fundamental threshold scheme in this domain. It computes *k-1* random coefficients for a *k-1* degree polynomial *f(x)* in a finite field (e.g. GF-256 or GF-65536). The k^{th} term comprises the secret (usually the constant term of the polynomial). The share is identified by a share value *x*, taking values between 1 and the order of the field. The share *x* is the polynomial value of *f(x)*. The secret can be reconstructed by interpolating the values of *k* shares to find the constant term of the polynomial (i.e., the secret). Interpolation is computationally optimized and only the constant term is revealed.

In the PPH domain, there is one share for each account. The share is XORed with the relevant PHS result and is maintained by the server (instead of the pure PHS result). The shares are derived from a master key. This key is only known to the service provider and is not stored on disk in order to prevent attacks that would disclose the key along with the stolen password data. When the server starts, *k* clients must login and be correctly verified in order to reconstruct the shares. Implementations of SSS provide integrity check mechanisms to detect if incorrect shares are parsed. After this startup phase, the server operates in the ordinary manner. Figure 19, illustrates the generic PPH scheme and the account verification of a login request after initialization.

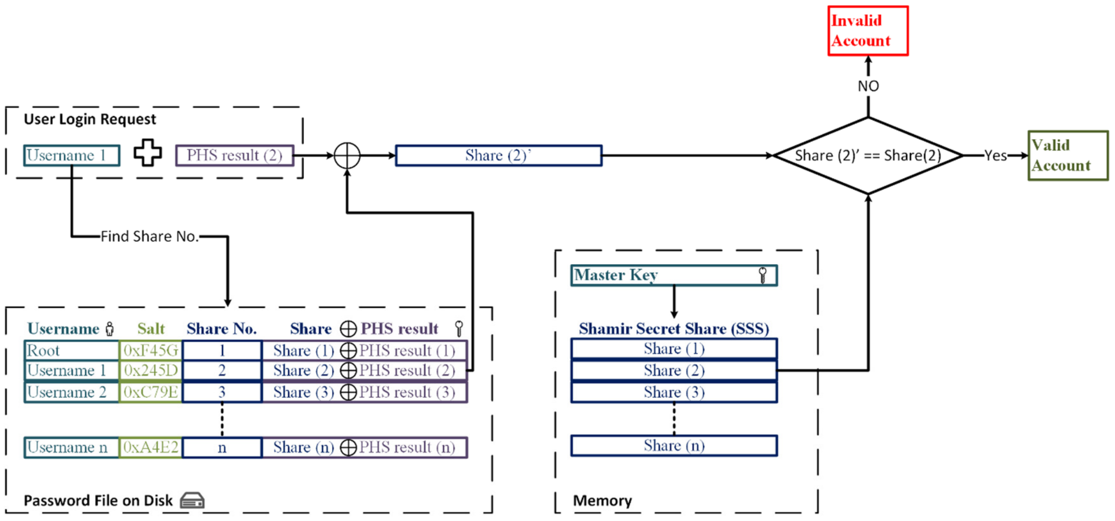


Figure 19 Poly password-hashing scheme

The attacker has to crack a threshold of password hashes before being able to recover passwords. At a small additional cost by the server, security increases by many orders of magnitude.

Poly password-hashing is easily implemented and deployed on a server without any changes to clients and can be integrated to current forms of authentication (e.g. two factor authentication, hardware tokens and fingerprint authentication). It is also efficient in terms of storage, memory and computational demands.

8.2.2 Lightweight PHS/PPH and Security Analysis

LightPolyPHS is a lightweight PHS and PPH, designed for embedded systems and constrained devices. The overall system complies with the principles of LWC. First, we replace the inner cryptographic primitives that are utilized by the PHS Catena and the PPH PolyPassHash and implement two relevant lightweight schemes. Then, we integrate them by using the lightweight Catena as the PHS for the lightweight PolyPassHash and implement the proposed LightPolyPHS.

Lightweight Catena

Catena utilizes a cryptographic hash function to instantiate the graph-based structure. The reference implementation propose the functions SHA512 and BLAKE2b. Both hashes result in a 512-bit output and offer high level of security. SHA512 is selected as a widely-used and implemented standard while BLAKE2b is proposed due to its high efficiency and security against massively parallel attacks.

The lightweight Catena utilizes PHOTON-256 as the cryptographic hash function, which outputs a 256-bit digest. This results in a smaller datapath and implementation size than the original scheme as well as lower computational and memory requirements. The output size complies with the relevant primitive in PolyPassHash and provides moderate level of security.

The security level of the Catena is determined by the underlying hash function. Consider that Catena-sha512, Catena-blake2b and Catena-photon256 offer 2^{512}, 2^{481} and 2^{244} bits security respectively.

Lightweight PolyPassHash

PolyPassHash processes the passwords with SHA256. The hash function outputs 256-bits and AES with 128-bit key encrypts the SSS shares.

The lightweight PolyPassHash replaces SHA256 with the lightweight PHOTON256. AES is substituted by CLEFIA with the same key size. The two schemes exhibit the same datapath size and the resource saving is low. The security level of the lightweight version is similar to the original one.

In both schemes, the disk space requires 1 additional byte for each account to store the share value, in contrast to PHS-only solutions. The server must also store the polynomial coefficients for the SSS in memory. The total size is small: the XORed share and hash (256 bits long) multiplied by the threshold value (usually 2-5). In real systems, this value would result in a few hundred bytes.

PolyPHS

In PolyPassHash, passwords are simply parsed by SHA256. To further increase security, we replace the hash function with the Catena PHS. The PHS enhances resistance against attacks but is more resource demanding than SHA256. Also, Catena exhibits larger output size (512-bits) and the integrated implementation size is higher.

LightPolyPHS

The original Catena offers high level of security but it cannot be applied on constrained devices. The lightweight Catena offers moderate level of security and is appropriate for the targeted systems. To fill the gap, the lightweight PolyPassHash is applied to increase security. The simple hash function is replaced by the PHS. Lightweight Catena uses the same datapath size as the SHA256 of PolyPassHash and provides higher password protection. With 3 shares as the threshold, an attacker must guess 3 lightweight-Catena passwords simultaneously to recover the password file. The security level is increased by 23 magnitudes on GPUs, resulting in $2^{244} \times 10^{23} \approx 2^{320}$ bits security.

8.2.3 Evaluation

The examined PHSs, PPHs, and the core cryptographic primitives are evaluated under an Intel Core i7 at 2.10GHz CPU with 8GB RAM, running 64-bit operating systems. Reference C or C++ implementations are utilized in order to provide a fair comparison with the unoptimized versions of PHC [89]. All implementations are installed on Windows 8.1 Pro and are executed on cygwin. The different primitives are assessed under common assumptions. We measure the code size, memory consumption, execution time and throughput of each scheme.

Cryptographic primitives

The software details of the block ciphers and hash functions are based on the ULCL benchmark. All lightweight primitives consume lower resources than the mainstream ones but are slower. For block ciphers, CLEFIA produces about 3.6 times smaller implementation than AES for slightly lower memory consumption and 1/12 of the speed. PRESENT has even lower code requirements but consumes more memory than AES and is slower than CLEFIA. For the hash functions, PHOTON consumes the least memory and exhibits the largest code size. Compared to SPONGENT it is about 8 times faster. Both CLEFIA and PHOTON can fit in constraint devices. The implementation size of both primitives is 21.9KB and the maximum RAM consumption cannot exceed the 11.28KB.

PHSs and PPHs

Table 11 summarizes the software evaluation of the examined PHSs and PPHs based on the default sizes for output, password and salt, and the indicative *t_cost* and *m_cost* parameters as reported by each scheme.

Table 11 Software implementations of password-hashing and poly password-hashing designs on PC

PHS	Password (bytes)	Salt (bytes)	Output (bytes)	t_cost	m_cost	ROM (KB)	RAM (KB)	CPU (secs)
(PHSs)								
PBKDF2	24	8	64	1000	0	30	0	0.002024
	24	8	64	2048	0	30	0	0.004150
	24	8	64	4096	0	30	0	0.008141
	24	8	64	10000	0	30	0	0.019386
bcrypt	12	16	54	12	0	27	492	2.668653
scrypt	8	32	64	5	0	182	450656	2.837654
Catena-blake2b	8	16	64	3	18	25	16384	0.353742
	8	16	64	3	20	25	65596	2.619238
	8	16	64	3	21	25	128484	5.461030
Catena-sha512	8	16	64	3	18	25	16496	0.783590
	8	16	64	3	20	25	65720	5.389355
	8	16	64	3	21	25	131240	11.66496
Catena-photon256	8	16	64	3	18	26	8188	1.749200
	8	16	64	3	20	26	32760	13.06562
	8	16	64	3	21	26	65532	27.30194
(PPHs)								
PolyPassHash	16	16	32	1	0	78	3412	0.000054
	16	16	32	2	0	78	3412	0.000055
	16	16	32	4	0	78	3412	0.000055
Light-PolyPassHash	16	16	32	1	0	66	3410	0.000060
	16	16	32	2	0	66	3410	0.000068
	16	16	32	4	0	66	3410	0.000080
PolyPHS	16	16	64	1	0	89	19794	0.353695
	16	16	64	2	0	89	19794	0.707538
	16	16	64	4	0	89	19794	1.415020
LightPolyPHS	16	16	32	1	0	77	11579	1.749253
	16	16	32	2	0	77	11579	3.498454
	16	16	32	4	0	77	11579	6.996854

The standardized PBKDF2 is not memory-hard and consumes neglected memory. bcrypt has low memory requirements and achieves similar performance as scrypt. scrypt is the first widely-used memory-hard PHS and exhibits the higher memory consumption and larger implementation size.

Catena is a novel PHS that applies memory hardness to enhance security. Three versions are evaluated based on the underlying hash function. Catena-blake2d is the fastest and consumes similar memory as Catena-sha512. Catena-photon256 reduces memory demandings around 50% in exchange of lower performance. All three versions produce similar code size.

PolyPassHash is a novel PPH that utilizes the hash function SHA256 and the block cipher AES. It is quite efficient and has low and constant memory requirements. The Light-PolyPassHash version uses the hash function PHOTON and the block cipher CLEFIA. It decreases the code size and accomplishes slightly lower memory consumption and worsen speed.

The security of the initial scheme is fortified by replacing the hash function with a PHS. The PHS constitutes the most resource demanding component. The *t_cost* parameter determines the *k* shares of the SSS component and linearly affects the execution time. As *t_cost* increases, the number of password-hashing operations, which are performed by the PHS, also increases. PolyPHS uses the Catena-blake2b (*t_cost=3*, *m_cost=18*) as the PHS of PolyPassHash. LightPolyPHS uses the Catena-photon256 (*t_cost=3*, *m_cost=18*) as the PHS of Light-PolyPassHash.

Figure 20, illustrates the evaluation results of the 10 PHS and PPH schemes. For *k=2*, LightPolyPHS has slightly worsen performance than bcrypt and scrypt. The memory-hard Catena-photon256 component enhanced with the SSS provide adequate security for around 39 times lower memory consumption and 2.3 smaller implementation size than scrypt.

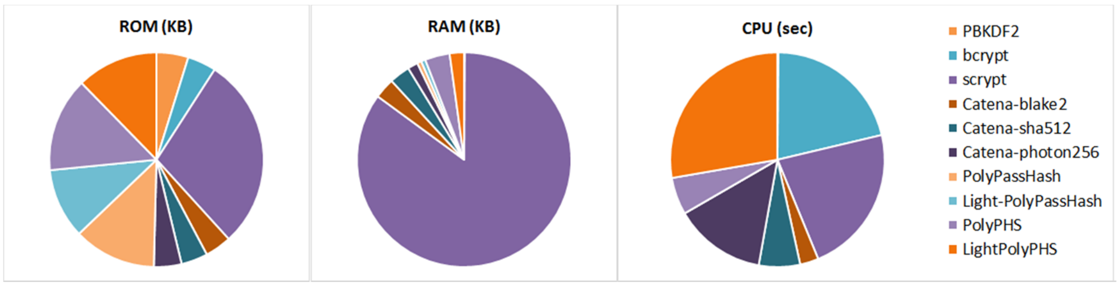


Figure 20 Software comparison of the examined password-hashing designs

MANET Application

As a proof of concept, we apply LightPolyPHS on BeagleBone embedded devices. BeagleBone is a credit-card-sized Linux computer with Internet connection that runs Android and Ubuntu OSs. The processor is an AM335x 720MHz ARM Cortex-A8 with 256MB DDR2 RAM and 4GB microSD.

We create a mobile ad hoc network (MANET) with 20 BeagleBone client devices and 1 BeagleBone acting as the server. The devices sense environmental parameters (e.g. temperature and moisture) and periodically upload this information to the server. They communicate wirelessly through USB-WiFi equipment in order to login the service and exchange data.

For 3 shares as the threshold, the server must authenticate three clients at initialization. Then, the account verification requires a single lightweight-Catena operation. The BeagleBone server takes on average 5 sec to startup and 1.8 sec to execute the PHS verification.

8.3 Authenticated Encryption in Hardware

This subsection presents a lightweight authenticated encryption scheme based on the integrated hardware implementation of the lightweight block cipher PRESENT and the lightweight hash function SPONGENT. The presented combination of a cipher and a hash function is appropriate for implementing authenticated encryption schemes which are commonly utilized in one-way and mutual authentication protocols. Their inner structure is exploited to discover hardware elements usable by both primitives, thus reducing the circuit's size. The integrated versions demonstrate a 27% reduction in hardware area compared to the simple combination of the two primitives. The resulting solution is ported on an FPGA and a complete security application with input/output from a UART gate is created. In comparison with similar implementations in hardware and software, the proposed scheme represents a better overall status.

8.3.1 PRESENT-SPONGENT Authenticated Encryption

Symmetric cryptography performs well in ESs. It provides confidentiality and data integrity and is utilized in authentication protocols. For LWC, block ciphers are the most common choice for confidentiality. Hash functions are another type of cryptographic primitive, taking an input message and producing a fixed-length tag. They form message authentication code (MAC) mechanisms and provide data integrity and authenticity.

Except from the basic crypto-primitive types there are efforts to offer more advanced functionality. Authenticated encryption (AE) integrates ciphers and integrity mechanisms to provide simultaneously confidentiality and integrity with authentication. AE schemes are becoming popular due to the ongoing Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [90], organized by the international cryptographic research community. CAESAR will select a portfolio of algorithms offering advantages over AES-GCM (the current AE standard [91]) in 2017.

PRESENT is a block cipher with 64-bit blocks and 80/128-bit keys. It is one of the first ciphers that achieve a compact hardware implementation of about 1000 GE. Its substitution-permutation network requires 31 rounds for de/encryption. The cipher is extremely efficient in hardware and uses a fully wired diffusion layer without any algebraic unit. Its main feature is the replacement of the ordinary eight S-Boxes with a carefully selected one.

In this prototype, the encryption-only implementation of PRESENT with 80-bit key, available in OpenCores [92], is extended (e.g. decryption functionality added). The presented work is implemented using in Verilog (under an LGPL license). The cipher does not require an initialization phase.

SPONGENT applies a sponge construction which is instantiated with a wide PRESENT-type permutation, following the hermetic sponge strategy. The SPONGENT-88 variant is designed for extremely restricted applications with low pre-image security requirements. In this paper the pipelined version of SPONGENT-88 is implemented in Verilog. For 88-bit hashes, the produced implementation requires 1127 GE.

When it is executed simultaneously with PRESENT over the same data, the I/O of the hash function is slightly changed in order to keep the same interface for the different integrated versions. The input data are processed in 64-bit blocks, to comply with the I/O interface of PRESENT. For SPONGENT, we pad 24 bits in each block, in order to construct an 88-bit block (the block size of SPONGENT) and also retain a small common datapath with PRESENT (the main target of an integrated design).

8.3.2 Design

PRESENT and SPONGENT are the hardware-oriented standardized primitives for LWC. They are extensively analyzed and are found secure against the current set of attacks. Moreover, the two primitives exhibit several structural similarities that can be exploited by an integrated implementation in hardware. SPONGENT has a sponge structure that is initiated by a PRESENT-type permutation. Also, both primitives use 4-bit S-boxes. The authors’ goal is to exploit these similarities and merge the two primitives in a concrete hardware implementation. This implementation utilizes the same memory elements for storing the input and output data of both primitives. Moreover, SPONGENT uses the key-update module of PRESENT, which requires 481 GE, in order to produce a MAC. The implementations of PRESENT and SPONGENT are combined, achieving 27% hardware reduction.

Every primitive can also function independently. Each part of the circuit is being implemented with its own pipeline, achieving the highest throughput for encrypted data and the corresponding MAC computation. Three pipelines are used for the cipher production, the key update and the digest generation.

PRESENT pipeline: An encryption of a 64-bit block is performed after 31 rounds through the 64-bit Substitution-Permutation Network (SPN) of PRESENT and the key-update module. The pipeline consists of the register *State Reg*, which stores the data among the iterations, and the 64-bit SPN. The key-update procedure consists of the state register *Key Reg* which stores the updated key and one 4-bit S-box which updates the key in every round. The *data_i* (data input) serves as input for both the encryption key and the message data. This input is 80-bit wide to fit the key. Only the 64 least significant bits are used to load the message data. In the first round the key is loaded in the key register (*Key Reg*) and in the next round the message data is loaded in the cipher state (*State Reg*).

SPONGENT pipeline: The hash of an 88-bit block is produced after 45 iterations through a separate 88-bit SPN for SPONGENT. The pipeline consists of the state register *Hash Reg*, which stores the current digest, and the 88-bit SPN.

Dependent & independent data: The two primitives are executed in parallel over dependent or independent data and all three AE approaches are supported (Figure 21). It requires 4324 logic gates to implement.

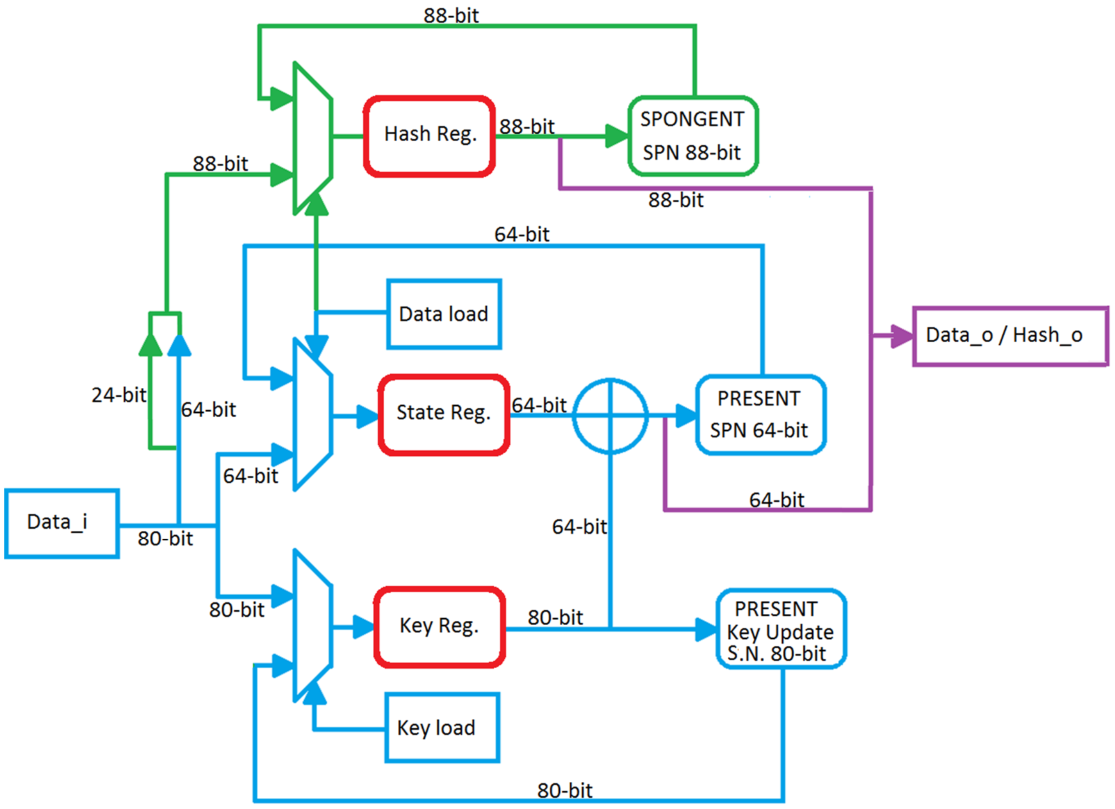


Figure 21 Parallel integration of PRESENT and SPONGENT in hardware

In parallel-independent mode (PIm), the cipher and the MAC process independent data simultaneously. SPONGENT loads the first 80-bit block of data. As the hash function processes the data, PRESENT loads its 64-bit block of plaintext data. After 45 rounds, one block of ciphertext data and the hash are produced. This version processes 144-bits of data every 45 rounds and achieves a throughput of 320 Kbps (at 100 KHz frequency).

In Encrypt-and-MAC (E&M) the cipher and the MAC process the data simultaneously in 64-bit blocks. While in parallel execution, the two primitives are synchronized; the plaintext block is encrypted in 31 rounds and the hash is produced after 14 additional rounds. The cipher is in standby mode and waits for the MAC during these extra rounds. After 45 rounds one block of ciphertext data and the relevant hash are produced. This version processes 64-bits of data every 45 rounds and achieves a throughput of 142.22 Kbps.

In Encrypt-then-MAC (EtM) the cipher encrypts the plaintext in 64-bit blocks. Once a ciphertext block is produced it can be processed by the MAC while the cipher proceeds to the next plaintext block. Thus, it takes 31 initial rounds to encrypt the first block and 45 rounds multiple the remaining blocks to encrypt the plaintext, with 45 additional rounds to produce the final MAC. It takes 121 rounds to encrypt 1Kbit data, achieving 82.64 Kbps throughput.

In MAC-then-Encrypt (MtE) the data are processed as in E&M. The 88-bit MAC outcome is then encrypted by the cipher as two 64-bit blocks, requiring 62 more rounds than E&M (the second block is padded with zeros). Totally, it takes 152 rounds to encrypt 1Kbit of data with 65.78 Kbps throughput.

8.3.3 Network Application

The complete network application was implemented and tested, with an FPGA platform connected to a PC via UART. The FPGA communicates through the network with other entities. The application supports the following basic operations:

- send plaintext (unencrypted communication)
- send plaintext with MAC (for integrity checks)
- send encrypted message (for confidentiality)
- send encrypted message with MAC (for confidentiality and integrity – the PIm)
- send a message where each transmitted packet is encrypted with a MAC for both data and packet's headers (for AE schemes – the three AE approaches)
- and send encrypted message with MAC for the encrypted message and the packet headers, with encryption only for data and MAC both for encrypted data and packet's headers (for Authenticated Encryption with Associated Data (AEAD) schemes – the three AE approaches).

The integrated implementation is not used for unencrypted messages. Processing a plaintext to produce its MAC requires 45 rounds per 88-bits. Encrypting a message requires 31 rounds per 64-bits. The encryption of a message and the calculation of its MAC can be done simultaneously and takes 45 rounds.

Ordinarily, in AE and AEAD modes each message must be processed twice. However, our integrated implementations can encrypt and produce the MAC simultaneously, processing the message once and thus reducing the overall processing time (as described in the previous subsection). When decryption is supported, we can also apply this strategy to reduce the execution time.

The encryption-only versions of the integrated implementations are appropriate for one-way authentication protocols. When both encryption and decryption are supported, they can be applied in mutual authentication protocols.

8.3.4 Security Analysis

The security analysis is based on the relevant work of the original papers for PRESENT and SPONGENT. Moreover, the general security properties of the three AE approaches are detailed in [93].

Differential and linear cryptanalysis constitute the most important techniques for the analyst. PRESENT is resilient to these attacks and provides a lower bound to the number of active S-boxes that affect the differential or linear characteristics. It is also resistant to structural attacks, like integral and bottleneck, due to the exclusively bitwise design and the regular permutation operation. Algebraic attacks resolve the cipher's equations that imply diffusion. Analysis based on simulations suggests that it is unlikely to perform these attacks on PRESENT. Related-key and slide attacks identify relationships among different sets of sub-keys and impose the most effective key scheduling attacks. PRESENT counters them by the appliance of a round-dependent counter and a non-linear operation to mix the contents of the key register.

After the cipher's release in 2007 several independent cryptanalysis results have been announced. Side-channel [94] and related-key attacks [95] have been reported on reduced versions of PRESENT. The full round cipher is safe and no practical attacks have been reported.

The design and structural similarities of PRESENT and SPONGENT allow the reuse of the aforementioned analysis results. Cryptanalysis on PRESENT was taken into consideration and SPONGENT is more resistant to linear attacks. The hash function is resistant against differential analysis. Collision attacks, like the rebound attack, are mitigated by the bit-oriented permutation. Full preimage and second-preimage security can be provided. As most of the embedded applications do not necessarily provide full second-preimage security, the most compact variants provide lower protection to enhance performance in the most constrained environments. Linear distinguisher attacks on reduced versions of SPONGENT are reported in [96]. In this paper, we use the most constrained variant of SPONGENT-80-8-88. It provides 2^{80} bits preimage security and 2^{40} bits second-preimage and collision security.

The encryption part of the AE approaches intends to provide four main properties of indistinguishability (IND) and non-malleability (NM) under chosen-plaintext (CPA) or chosen-ciphertext attacks (CCA) – abbreviated as IND-CPA, IND-CCA, NM-CPA, NM-CCA. The authentication tag implies two properties for integrity:

- of plaintexts (INT-PTXT) – it is computational infeasible to produce a ciphertext decrypting to a message that the sender has never encrypted
- and integrity of ciphertexts (INT-CTXT) – it is computational infeasible to produce a ciphertext that has not been previously produced by the sender.

The study in [93] analyses the provided security of the three AE approaches in terms of these six properties under the assumptions that the encryption part is IND-CPA and the authentication tag is either weakly or strongly unforgeable. Table 12 summarizes these results. NM-CCA is omitted since it is equivalent to IND-CCA. The integrate solution that is proposed in this paper provides strong unforgeability.

Table 12 The security properties of the authenticated encryption approaches

AE	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Weakly unforgeable authentication					
E&M	Insecure	Insecure	Insecure	Secure	Insecure
EtM	Secure	Insecure	Insecure	Secure	Insecure
MtE	Secure	Insecure	Insecure	Secure	Insecure
Strongly unforgeable authentication					
E&M	Insecure	Insecure	Insecure	Secure	Insecure
EtM	Secure	Secure	Secure	Secure	Secure
MtE	Secure	Insecure	Insecure	Secure	Insecure

8.3.5 Evaluation

Hardware implementations

The Xilinx Virtex-5 FPGA platform is utilized to implement and test the proposed integrated solution, using the tools provided by the Xilinx ISE Design Suite 12.1. A UART is used to check the bandwidth of each version due to its simple design and operation. UART is platform independent and can be used to transmit data to other FPGAs and embedded systems. Table 13 summarizes the implementation details of PRESENT, SPONGENT and their integrated versions in FPGA. As is evident from the table, the PIm version achieves a throughput of 370 Kbps and is the most efficient.

Table 13 Hardware implementations of authenticated encryption primitives in FPGA

Cipher	Key / Tag (bits)	Rounds	FFs	Total slices	Throughput (Kbps)	Efficiency (Kbps / slices)
PRESENT	80/-	31	-	162	206.5	1.28
SPONGENT	-/88	45	-	95	195.5	2.06
PIm	80/88	45	149	174	320	1.83
E&M	80/88	45	149	174	142.22	0.81
EtM	80/88	121	149	174	82.64	0.47
MtE	80/88	152	149	174	65.78	0.37

The relevant features are also estimated in ASIC, based on the implementation details of PRESENT in [97] (Virtual Silicon, VST, standard cell library, 0.18μm, 1P6M Logic process). Table 14 presents the implementation details of PRESENT, SPONGENT and the four integrated variants in ASIC. The reported metrics are referred to encryption-only implementations. The integrated implementations are compared with the relevant implementations of Hummingbird-2 [98], Grain-128a [99], ALE [100], FIDES [101], and AES-CCM [101]. Hummingbird-2, Grain-128a, and ALE are vulnerable to attacks. They are included in this comparison analysis as a proof of concept for the qualification of the proposed solutions.

Table 14 Hardware implementations of authenticated encryption primitives in ASIC

Cipher	Key / Tag (bits)	Throughput (Kbps)	GE	FOM	Tech (nm)	Power (μW)
PRESENT	80/-	206.5	1569	838	180	2.7
SPONGENT	-/88	195.5	967	209	180	1.5
PIm	80/88	320	2508	508	180	3.9
E&M	80/88	142.22	2508	226	180	3.9
EtM	80/88	82.64	2508	131	180	3.9
MtE	80/88	65.78	2508	104	180	3.9
FIDES-80-S	80/80	10.64	1153	80	130	1.9
FIDES-96-4S	96/96	26/09	1870	74	130	3.1
HB2-ee4c	128/64	222.2	3220	214	180	5.1
HB2-ee16c	128/64	83.3	2332	153	180	4.7
HB2-ee20c	128/64	68.9	2159	147	180	4.3

Grain-128a	128/32	80	2867	97	-	-
ALE	128/128	0.6	2579	0.9	65	94
AES-CCM	128	0.14	3472	0.1	65	128

In ASIC the individual implementation of PRESENT is more compact than Hummingbird-2, but for encryption the latter is slightly faster. FOM reveals that PRESENT is the best choice, as it achieves a better overall status. Moreover, all the integrated solutions that encrypt and produce a MAC are compact enough to fit in embedded devices (less than 3000 GE) and perform well. The PIm and the AE versions achieve better overall status than Hummingbird-2 and consume less power. The Hummingbird-2 protocol (encryption and MAC) performs worse than the encryption-only version that is reported in Table 3, as the MAC production is almost 7 times slower than the encryption. Thus, the FOM gap is even higher when comparing Hummingbird-2 with the integrated AE versions of PRESENT and SPONGENT.

Grain-128a produces the shortest authentication tag (32-bit). In contrast to the core lightweight cipher, it deploys different non-linear functions to enhance security, adding processing overhead and hardware space. The shift registers are regularly clocked and 1-bit output is produced every second clock (Grain produces 1-bit per cycle). Grain-128a is slower and requires more chip area than the EtM version of the integrated solution.

ALE uses 128-bit keys. Among the lightweight schemes, it provides the longest authentication tag. Moreover, it implements AES and requires initialization due to the stream cipher components. As a result, it exhibits the worst overall status, with high area demands and low throughput. ALE also consumes the most power.

FIDES achieves the smallest chip area and power consumption. However, it presents low throughput. FIDES has a short initialization phase, and the encryption and MAC components process the data sequentially. The presented integrated solutions are larger but quite faster due to the parallel operation by PRESENT and SPONGENT, resulting in a better overall status.

AES-CCM is designed for mainstream applications. It provides higher levels of security in return of more area. AES-CCM operates as E&M and, in general, it is faster than AES-GCM that works as EtM. In 100 KHz frequency the scheme is quite slow. We have to mention that the referred implementation is not optimized. Both primitives require IV.

<u>Hardware versus software implementations</u>

PRESENT and SPONGENT were designed with lightweight hardware implementations in mind. Still, PRESENT also produces a compact software implementation, suitable for embedded devices. The authors' comparative analysis of LWC [82], revealed that the most efficient implementation of PRESENT-80 in embedded software requires 936 bytes of code for 23.8 Kbps throughput at 4MHz frequency. This is 8.6 times slower than the hardware implementation. SPONGENT has not been studied in the context of embedded software but it is estimated that it would perform worse than in hardware. Thus, the individual hardware implementations that are reported in this paper are more efficient than the relevant implementations in embedded software; their throughput ranges from 65.78 Kbps to 206.5 Kbps. The same conclusion stands for the integrated solutions in hardware and software.

Hummingbird-2 performs well in embedded software. Its most efficient implementation achieves 200 Kbps throughput for encryption [82]. However, it remains slower than the hardware implementations of the same algorithm (222.2 Kbps) and PRESENT (206.5 Kbps). Thus, the Hummingbird-2 protocol performs worse in embedded software than in hardware.

Grain-128a, ALE, and FIDES are hardware-oriented schemes. Their performance is worse in embedded software. ALE, AES-GCM, or –CCM utilize the AES-NI instruction set on Intel CPUs for increased performance, but it is out of bounds for embedded devices.

9. Trust-Based Routing – Precision Agriculture Use Case

Trust-based schemes are used in wireless networking to provide secure routing functionality. Reputation [102] is formed by a node's past behavior and reveals its cooperativeness. In secure routing, reputation mainly evaluates the routing and forwarding, the use of encryption and authentication mechanisms, and the proper transmission of acknowledgements per transmitted packet. Trust [102] is the level of confidence that an entity holds about others. It is the aggregation of all reputation values that the entity holds for another participant. A node with high reputation values is considered trustworthy. Legitimate nodes depend mostly on trustworthy entities to accomplish communication tasks. On the other hand, low reputation can reveal selfish or malicious entities and is used for intrusion detection. Legitimate nodes try to avoid disreputable entities and do not serve their traffic.

Many trust-based systems have been utilized to achieve secure routing [103]. Each one of them is evaluated under specific ad hoc applications and tackles a constraint set of security threats. Six of the most representative state-of-the-art systems are described below.

The Semi-Distributed Reputation-based Intrusion Detection System for mobile ad-hoc networks (S-D RepIDS) [104] implements many novel reputation metrics for secure routing and is tolerant to failures due to traffic congestion. AODV Reputation EXtension (AODV-REX) [105] adopts a multi-layer model. The privacy of the recommender is protected by aggregating the direct and indirect reputation values. The recommendations are encapsulated in the underlying protocol's routing messages to preserve performance. AODV-REX proposes the virtual lengthen of a path to punish misbehavior. In contrast to the rest secure routing schemes, the Reputation based Framework for Sensor Networks (RFSN) [106] evaluates both the node's routing cooperativeness and the reported sensed variables (e.g. temperature). In the Trusted based Routing using Dominating Set Approach (TRDSA) [107] only a set of trusted nodes with sufficient remaining energy need to operate in promiscuous mode and capture malicious activity, reducing the overall energy consumption. Expected Forwarded Counter (EFW) [108] combines cross network-layer observations. The network-layer module observes the routing protocol's functionality, as in most relevant schemes, and estimate the forwarding or dropping probability of a node. Additionally, MAC-layer measurements about the wireless link quality are considered to select reliable and high performance paths. The Secure Resilient Reputation-based Routing (SR3) [109] combines a reinforced random walk algorithm with reputation.

These systems target limited sets of security vulnerabilities and attacks [110]. While the basic malfunctioning (e.g. packet drop or avoid routing participation) is efficiently detected by trust systems, there still remain several network aspects that are not properly handled, like overburden or isolate distant nodes [111], [112], [113], [114].

This work introduces the Self-Channel Observation Trust and REputation System (SCOTRES) – a novel system for secure routing in wireless ad-hoc networks and CPS [115]. Instead of adopting features from heavy reputation-based schemes, the proposed system exploits the knowledge that each node already possesses about the network. The protocol aims to maximize the knowledge that can be inferred about the network from the data that a node already processes. For this purpose, SCOTRES utilizes efficient mechanisms that retain networks security and performance. The authors suggest that each node should invest more resources in achieving a robust individual reasoning process in order to minimize wrong decisions and the amount of data that has to be exchanged among the nodes, like the remaining node energy. This work provides evidence that a core portion of the required knowledge can be efficiently mined from the data that each node already possesses.

SCOTRES is a novel trust management scheme and counters the abovementioned routing attacks. It includes three innovative metrics:

- An efficient energy metric that protects low-energy or overloaded nodes from power dissemination, increasing the network's lifetime
- A novel topology metric that protects topology-significant nodes and enhances load-balancing
- A channel-health metric that estimates the channel state between the nodes and avoid jamming areas

and two core reputation and trust metrics that integrate the state-of-the-art features in the secure routing domain with a few new ideas:

- A core reputation metrics that evaluates specific network operations and acknowledges reputable nodes
- A trust metrics that integrates the reputation values of all examined operations, identifying the malicious nodes and preserves the legitimate nodes private ratings.

9.1 Threats and Attacks

Routing protocols fail to protect the network against selfish and malicious activity. Surveys of routing attacks in ad-hoc networks are presented in [110], [111], [116], [117], [118].

For forwarding, a flooding attacker exhausts the resources of the network and its underlying nodes (e.g. DoS, inject arbitrary packets). Blackhole and grayhole attacks discard all or selective parts of the forwarding packets respectively. In sleep deprivation, the malicious node interacts with other nodes in a manner that appears to be legitimate, while keeping them out of the power-conserving sleep mode. Other threats on packets include modification, interruption and replay.

For routing, link spoofing attackers advertise fake routing information (e.g. optimal paths) in order to avoid or impose their participation and then perform the attacks on forwarding. In wormhole attacks a pair of attackers, which communicate through a private high speed network, collude to record packets at one location and replay them at another location of the network. In colluding misrelay, a pair of neighboring attackers conspire to avoid participating in a route. In routing table poisoning, the attacker advertises false routing information (i.e. nonexistent paths, loops, false link break, and HELLO flooding) to harm the nodes' routing capability.

Trust schemes are integrated with routing protocols as a defense mechanism. They prevent attacks on forwarding and link spoofing as they detect and negatively rank the misuse and discarding of a packet. Trust systems do not deal with wormhole attacks. However, countermeasures (e.g. [119], [120]) could be integrated, providing a more robust intrusion detection mechanism. Moreover, trust schemes do not deal with colluding misrelay attacks directly. Still, as the two nodes do not participate in routes as intermediate nodes, they cannot gain positive ratings from traffic forwarding. Thus, they are not able to make requests either. Routing poisoning is countered by evaluating the positive contribution on routing.

Nevertheless, these schemes can then become the new target of more sophisticated attacks. Survey of attacks and defense techniques for trust systems are conducted in [121], [122]. They are categorized in terms of identity-, ballot-, social- and topology-based threats and attacks.

To counter identity-based attacks (i.e. impersonation or Sybil attacks), in most relevant systems, it is assumed that a secure underlying mechanism is applied (e.g. [12], [123]), which provides authentication and confidentiality network-wide. Ballot-based attacks are countered by robust ranking and recommendation mechanisms. In order to reduce the effect of social-based attacks, a robust reasoning process and a ranking policy based mostly on direct knowledge and restrictive use of indirect knowledge are imposed. Topology-based attacks have not been extensively examined and the current systems provide no specific security treatment.

Moreover, due to the open medium, wireless communications are vulnerable to jamming attacks, which significantly degrade the network's performance. A survey of jamming attacks and countermeasures is reported in [112]. The accurate detection of the jammer is challenging, while precision is also significant [124]. Some of these techniques only detect some types of jamming attacks while others produce high false positives. After detection, recovery countermeasures are performed, like channel hopping and spatial retreat, based on the type of the jamming attack. However, such techniques are not always applicable, especially in wireless sensor networks (WSNs) where the nodes may have constrained communication capabilities.

9.2 Secure Routing Protocols

In this study, we concentrate on six representative secure routing protocols: TRDSA, EFW, SR3, S-D RepIDS, AODV-REX, and RFSN. TRDSA, EFW, and SR3 utilize basic reputation mechanisms for their core deductive components. S-D RepIDS, AODV-REX, RFSN, and SCOTRES apply more robust reasoning processes for evaluating direct knowledge and making recommendations.

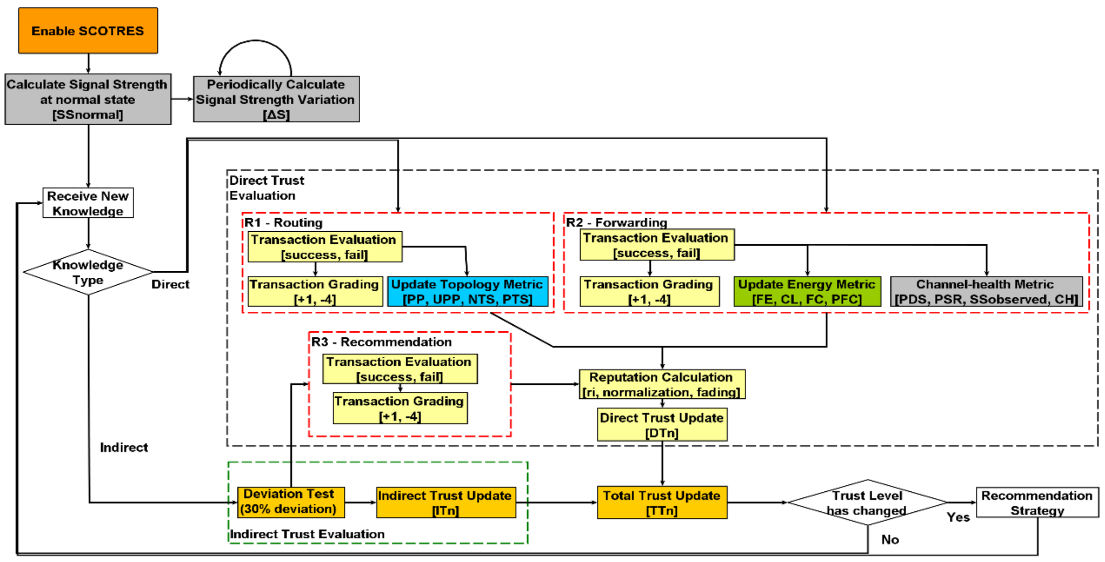
Information regarding the remaining energy and traffic congestion is also processed. TRDSA's routing operation takes into account the energy consumption. S-D RepIDS provides protection in congested periods. SR3 reduces congestion traffic due to the random nature of the random walk algorithm. The MAC-layer measurements of EFW assigns low communication reliability to the overloaded links, routing the traffic through alternative paths. SCOTRES protects the network in all these cases. Moreover, TRDSA, SR3 and SCOTRES perform load-balancing mechanisms to enhance performance and increase the longevity of the nodes. EFW and SCOTRES are the only systems that implement fault tolerance mechanisms to mitigate the effect of jamming attacks.

SCOTRES is efficient in terms of energy consumption and performance, and protects the network against most threats. It achieves sufficient load-balancing and retains the nodes' energy dissipation. The overall security of SCOTRES surpasses the protection that is provided by current solutions.

9.3 SCOTRES

Dynamic Source Routing (DSR) performs well in static and low-mobility environments with the routing overhead being proportional to the path length. For secure routing functionality in the network layer, the proposed SCOTRES scheme is integrated with the DSR.

Categorizing a node as trusted, legitimate, selfish or malicious is performed after evaluating a new transaction for this specific node. The evaluation of a transaction's result is the main function of the protocol that assesses direct and indirect knowledge based on the SCOTRES's metrics. Indirect recommendations can then be sent to trusted and legitimate 1-hop neighbors, whenever the status of an inspected node changes. The paths that contain malicious nodes are excluded, thus, the malicious activity is addressed and several attacks are countered. Figure 22 depicts the evaluation process and the underlying parameters of the five metrics that are described in the subsections below.



Enable SCOTRES
Calculate Signal Strength at normal state [SSnormal]
Periodically Calculate Signal Strength Variation [ΔS]
Receive New Knowledge
Knowledge Type
Direct
Indirect
Direct Trust Evaluation
R1 - Routing
Transaction Evaluation [success, fail]
Transaction Grading [+1, -4]
Update Topology Metric [PP, UPP, NTS, PTS]
R2 - Forwarding
Transaction Evaluation [success, fail]
Transaction Grading [+1, -4]
Update Energy Metric [FE, CL, FC, PFC]
Channel-health Metric [PDS, PSR, SSobserved, CH]
R3 - Recommendation
Transaction Evaluation [success, fail]
Transaction Grading [+1, -4]
Reputation Calculation [ri, normalization, fading]
Direct Trust Update [DTn]
Deviation Test (30% deviation)
Indirect Trust Update [ITn]
Indirect Trust Evaluation
Total Trust Update [TTn]
Trust Level has changed
Yes
No
Recommendation Strategy

Figure 22 Evaluation process scheme of SCOTRES

9.3.1 Network Assumptions

This thesis concentrates on wireless sensor networks and wireless ad-hoc networks with no or low mobility. Consider an ad hoc network with nodes $N = \{1, 2, \cdots, i, \cdots, k\}$. We assume that all links are bidirectional. If node i can receive packets that are directly transmitted by node j, then node j can receive packets that are directly transmitted by node i. The wireless network is modeled by a directed graph $G = (N \cup \{d\}, L)$, where d is the destination or sink, $L \subset \{(k_1, k_2): k_1, k_2 \in N \cup \{d\}, k_1 \neq k_2\}$ represents the set of communication links.

Attacks on routing and forwarding are mainly studied. We assume that there exists a secure underlying mechanism that is performed by all nodes and accomplishes authentication and confidentiality network-wide. Studies that apply broadcast authentication protocols [102], [123] or lightweight authenticated encryption [9], [109] can be embodied. They provide required security properties, like authentication, integrity check, and resistance to replay attacks, thus, safeguarding forwarding packet misuse. These mechanisms also counter the identity-based attacks (i.e. impersonation, clone ID, Sybil or newcomer attacks, injecting arbitrary packets, and HELLO flooding).

9.3.2 Cryptographic Service

As the network nodes start communicating, the need to authenticate the sender of an incoming message becomes imperative. Broadcast authentication can overcome the computational (i.e. asymmetric cryptography) and operational (e.g. key distribution) issues and the Timed Efficient Stream Loss Tolerant Authentication (TESLA) broadcast authentication protocol [125] is a common choice. It is based on loose time synchronization between the sender and the receiver. TESLA achieves the security properties of asymmetric cryptography by using keyed Message Authentication Code (MAC) functions. The sender attaches a MAC to each packet where the key is known only to itself. The receiver stores the packet without being able to authenticate it at that moment; shortly after, the sender discloses the key and the receiver can then authenticate the packet. TESLA is efficient, with low communication and computation overhead, while it scales to large networks with many nodes and tolerates packet loss, as demonstrated in the original paper.

For the purposes of this work, the Ultra-Lightweight Cryptographic Library (ULCL) [63] was used to implement TESLA, providing node authentication and message integrity. In the used testbed (presented in following sections), packet data sizes vary between 28 to 364 bytes. SHA-256 is applied for TELSA's MAC computations, taking 0.29μs to 3.8μs, respectively, to verify a message on a BeagleBone embedded device (ARM Cortex-A8 500-700MHz CPU, 256MB RAM, Linux OS). If a key distribution method is deployed, SCOTRES can also achieve message confidentiality by encrypting the data segment of a packet; for this, AES-256 is used, taking 0.4μs to 5.2μs to encrypt the data.

The authentication for RFSN and Ariadne is similar. CSRAN uses certificates. RSA is utilized for the asymmetric encryption. On a Mobile Pentium III (Intel PIII 750/600 MHz CPU, 128MB RAM, Linux OS), the additional overhead for the digital signature generation at the sender required 8.5ms and the verification operation at the receiver took 0.5ms. A transmitted message of SR3 contains the hash of a random nonce encrypted with the message. If we utilize the same assumptions and primitives as for SCOTRES, the cryptographic processing of a message only takes from 0.79μs to 5.5μs [126], depending on data size.

The broadcast authentication operation of RFSN, Ariadne and SCOTRES is efficient with low added overhead. CSRAN and SR3 require a key distribution mechanism to provide the cryptographic service that further increases the computational resources. All three approaches provide security proofs.

9.3.3 Topology Metric

Topology-based attacks are critical in WSNs and ad-hoc networks. Moreover, they are more effective in settings with trusted components. Malicious entities take into consideration the network's topology in order to manipulate, disclose or prevent access to legitimate components of high importance and cause more damage.

Nevertheless, relevant countermeasures are not well studied. SCOTRES analyzes the routing table data to discover the topological features of the network. It specifies the topological significance of each evaluating node based on the information that can be mined by the routing information that each node already possesses. Thus, nodes that are significant for the network topology are considered more important for the system's durability. For these nodes, the rating system becomes more tolerant in cases of failure and, thus, it is harder to classify them as malicious, countering a high variety of topology-based attacks.

The topology metric is calculated by every node. It determines the importance of each related node for the own routing operation. The first parameter of the topology metric is the ratio of paths that a node participates in, called Path Participation (PP). Nodes with high participation are important, as they serve the packets of many paths. The rating component becomes tolerant in cases of failure and the path-selection component balances the communication effort through alternative paths (contributing to the overall load-balancing). The parameter reveals the paths that must be re-established in case that the evaluating node is falsely recognized as malicious (which decreases performance).

However, a malicious node could also participate in many paths to gain high topological significance. Still, the reputation component will force it to serve the high communication effort from all these paths in order to avoid being expelled. Tolerance thresholds are estimated in order to retain the reputation of legitimate nodes in cases of topology-based attacks (preventing the attack), while punishing or forcing to cooperate nodes that try to exploit the metric.

The second parameter of the metric is the ratio of destinations that are uniquely reachable through an examined node, called Unique Path Participation (UPP). Expelling a node as malicious will also derive these destinations unreachable. The rating component is tolerant to such nodes in order to prevent false accusations in case of attacks and punishes a malicious node that exploits the metric after passing a relevant tolerance threshold.

Based on PP and UPP, the topological significance parameters are estimated and utilized in the reasoning process of SCOTRES. The Node Topological Significance (NTS) is the weighted summation of the node's PP and UPP parameters.

In the routing operation, a path from a source to a destination needs to be selected. The Path Topological Significance (PTS) is calculated as the average NTS value for all path's nodes. The topology metric is a lightweight feature with low computation overhead, as the four parameters are only calculated whenever routing changes occur.

9.3.4 Energy Metric

Energy consumption is crucial for the examined applications, like green networks. The routing operation should balance the communication effort among the nodes in order to retain their longevity.

Many routing protocols base the selection of the more suitable route on the shortest path, without considering the remaining energy of the selected nodes. Similarly, many trust and reputation schemes select the short paths with well-reputed nodes (e.g. [105], [106]). However, as legitimate nodes consume their resources to serve others, they will continue to be selected as the most appropriate forwarding nodes. If they try to avoid the additional effort and save resources, their reputation is harmed.

To address the above issue, routing protocols are designed that integrate the energy consumption data in the path selection process [107]. The nodes exchange their remaining energy level. Such systems are considered efficient when applied on legitimate networks as they achieve good load-balancing, but they are still vulnerable to attacks [113], [114]. Selfish nodes report low levels of remaining energy to avoid selection, while malicious nodes report high levels of energy to inforce their participation. Moreover, attackers identify nodes with low energy (which may be more vulnerable to attacks) or exploit the energy information exchange mechanism to attack the network.

In a typical wireless routing protocol, the nodes operate in promiscuous mode in order to overhear the communication channel and update their routing data. For the energy metric, this mechanism is also considered, in order to enrich the information that is mined by the overheard communication and make decisions about the energy consumption of the evaluating nodes based on self-observation and not on exchanging knowledge. Although the exact energy level is not specified, the extracted information is adequate for achieving efficient energy and load-balancing.

This backpressure algorithm chooses the least busy nodes to forward packets. Combined with the rest metrics, it enhances both performance and security. Every node that overhears the communication channel, records the participation of the rest nodes. Specifically, they register the number of packets that have been forwarded in a time-window. A node that participates as intermediate in many transactions in the current time-window, has consumed a corresponding amount of energy. This is the first parameter of the energy metric, called the Forwarding Effort (FE). The nodes that have consumed excessive energy (an amount beyond a threshold), are given the *excess-energy-consumption* bonus.

The second and the third parameters of the metric act as a defense mechanism against selfish and malicious attacks that aim to overload the network. Except from measuring the participation of the intermediate nodes (FE parameter) the energy metric measures the transactions that are initiated by the source and destination nodes. Particularly, the number of packets that the two nodes exchange are noticed down in order to calculate the communication load that they add to the network, forming the Cooperation Load (CL) parameter. If the added load prevents the legitimate node from turning on sleep mode, the relevant CL contribution will be doubled (to limit sleep deprivation). The Fair Cooperation (FC) parameter is the ratio of FE and CL, and reveals the fair usage of the network or its exploitation. A value lower than 1 for the FC, indicates that the node has overburdened the network. After this point, the node is charged regressively. Every additional routing packet through nodes with the *excess-energy-consumption* bonus counts as two CL ranks. When the FC exceeds a threshold, it discloses a selfish behavior. If a node is categorized as selfish, it is punished and the rest nodes stop forwarding its traffic. Thus, the selfish node is forced to cooperate in order to restore its reputation and later transmit own traffic. When evaluating the energy metric for a path, the average FC values from all path's nodes are calculated by the Path Fair Cooperation (PFC) parameter. This type of information is utilized by the routing operation.

SCOTRES exploits this information in order to implement an effective energy and load-balancing mechanism, and retain the longevity of the legitimate nodes. For selecting a path from a source to a destination, the system calculates the average node energy consumption of each candidate path. The paths with the lower participation are foregone for serving the new transaction. Moreover, for the nodes that have earned the *excess-energy-consumption* bonus in the current time-window, the rating component is tolerant to failures and beneficial to continued successful cooperation.

The energy metric maximizes the performance and durability of the network. The legitimate nodes are not excessively burdened and are better safeguarded against attacks that exploit energy consumption. Moreover, the system performs well and is more robust against relevant attacks in congested periods. The computational overhead of the metric is low and, in any case, lower than in the cases of exchanging energy information.

9.3.5 Channel-Health Metric

The state of the wireless channel can be affected by several factors; e.g. bad weather conditions and jamming attacks can disrupt the wireless communications. In the case of trust routing schemes, bad channel state causes the failure of several transactions. The trust of legitimate nodes is decreased until they are eventually expelled. The current systems deal indirectly with this occasional malfunctioning by performing automatic reintegration strategies, like periodic re-entrance and redemption [104]. However, malicious nodes are also rejoining to the network.

In [127], an efficient model is proposed to detect and classify jamming attacks in wireless networks. In contract to relevant models [112], it utilizes features that are general and independent from the communication protocol and can be easily implemented by a high variety of WSNs. Specifically, the features of packet delivery ratio, signal strength, and pulse width are deployed to detect constant, random, deceptive, reactive, and short noise-based intelligent jammers. Moreover, the reported results and thresholds for detecting jamming are experimentally validated under real use-cases and attacks.

SCOTRES's channel-health metric integrates these features. We adopt the aforementioned detection model and implement a jamming-aware routing functionality. In the examined setting, it is only needed to discriminate the normal state of the communication channel from the bad, thus, the whole model is not integrated. Still, it could be fully deployed to provide a more robust intrusion detection mechanism.

For the channel-health metric, SCOTRES utilizes three parameters, which are detailed below. The Packet Delivery Ratio (PDR) is the average successfully received packets and measures the performance of the communication link between two nodes. The receiver measures the packets that have been successfully received (e.g. by checking the Cyclic Redundancy Check, CRC). According to the detection model, PDR is around 78% under normal network operation.

Similarly, at the transmitter side, the Packet Sent Ratio (PSR) measures the average volume of acknowledgements that have been received to the total number of packets that was sent. We consider that the parameter should behave like PDR in normal network operation.

The Signal Strength (SS) measures the power of the signal in dB that is observed at the receiver end. The SS feature is the average signal strength in a time window. When the SS is high, the PDR and PSR are also high, while the converse is not true. At initialization, each node calculates the SS value for every 1-hop neighbor. This value reflects to the ordinary signal strength between two nodes at the normal state. The Signal Strength Variation (ΔS) is the change of SS among the ordinary SS value and the SS that it is currently observed.

When the PDR and the PSR go below the thresholds for normal network state (78%), they are collated to the ΔS. If the ΔS is in the boundaries of normal network state, then the glitch is caused by the network. Otherwise, the malfunctioning is caused by the bad channel state (detailed information about the detection of the jamming attack are available in [127]). The Channel-Health (CH) parameter of the metric describes the channel state. *'1'* is assigned for normal state and *'0'* for bad channel state after detecting an attack.

In contrast to the original study that only uses PDR, this work considers both PDR and PSR in order to trigger the health metric either as receivers or transmitters respectively. Thus, selfish or malicious nodes cannot exploit the metric by acting like they are under an attack while successfully transmitting their own traffic at the same time.

When a node detects a bad channel state with one of its 1-hop neighbors in a time-window, it starts becoming tolerant to failures. Moreover, it tries to forward the communication through alternative paths. The path that contains such a 1-hop neighbor gains the *bad-channel-health* bonus which is subtracted by the path's overall score in the routing operation. This makes the selection of the path for new forwarding transactions difficult. Thus, the reputation of the legitimate nodes that are under an attack is protected and the performance of the network is retained (both by forwarding the traffic though non-affected routes and by not falsely accusing legitimate nodes as malicious). Thereafter, the system periodically checks the ΔS parameter in order to detect when the attack is over and restore the affected node's status.

After detecting a jamming attack, the current solutions, like spread spectrum and frequency hopping aren't always applicable. Our metric implements a build-in routing protection mechanism that retains the reputation of the legitimate nodes and routes the communication through non-affected paths. We choose this generic strategy instead of the well-known solutions, as it is generic and technology independent, covering a high variety of WSN settings.

The metric requires more resources than an automatic re-entrance strategy. However, the overall robust reasoning process of SCOTRES makes the false accusations of legitimate nodes harder even in bad channel states, making such attack strategies inoperative. Moreover, re-entrance can be more beneficial for attackers than for legitimate entities.

The channel-health metric requires a training session at the initialization phase to determine the normal network state. It then performs simple and lightweight operations with low computational overhead. PDR and PSR are already observed by the reputation schemes. The overall performance of the metric can be considered efficient, as it is an additional security mechanism for a series of problematic cases that are entirely countered by this component.

9.3.6 The Core Reputation and Trust Scheme

SCOTRES applies the state-of-the-art features for the core reputation and trust metrics. The selection is based on a comparative analysis that was carried out in [13]. In that study, several trust schemes for secure routing were compared under identical attack scenarios. The internal components and the main reputation and trust features were further analyzed in terms of security and performance.

Reputation evaluates the performance of a node on a specific operation. A reputation structure is implemented for every evaluated operation. This structure maintains all the available information for implementing the reputation features of the specific operation. SCOTRES evaluates the three main operations of routing, forwarding, and making suggestions about other participants – referred to as R_1, R_2 and R_3 respectively. The three reputation structures are identical and implement the reputation features that are described below.

When an operation is performed, a relevant evaluation process is executed to derive the transaction's result. The outcome is a numeric value that is aggregated in the operation's reputation estimation for the affected nodes. Initially, it is assigned *+1* for success and *-4* for failure (based on the previous analysis in [13]). Then, the three proposed metrics of topology and energy are utilized in order to refine this value and calculate the final outcome for the transaction. The initial rating is multiplied with the NTS, FE and CH parameters. Each of the three parameters is multiplied with a weight, indicating its effect in the final value. The resulting value is aggregated to the relevant reputation structure (R_1, R_2 and R_3) for the examined operation. For routing, SCOTRES evaluates the nodes participation in a route request (i.e. accept or deny) and the validity of the routing information that it broadcasts (e.g. inactive links or loops, and false route breaks) as all messages are authenticated by the underlying cryptographic mechanisms. For forwarding, the system evaluates the forwarding effort that a node serves or evokes and the packet misuse (based on the proper operation of the cryptographic mechanisms). For recommendation, the node is ranked based on the suggestions that is makes for the rest participants, as described in the following paragraphs.

Reputation fading is a core protection mechanism in reputation systems for countering several types of attacks, where a malicious entity gains high reputation in order to be categorized as legitimate and later perform more effective attacks (e.g. Sybil attacks). Fading based on beta or Bayesian distributions is considered more suitable for reputation systems (e.g. [104], [106]). SCOTRES implements a reputation fading mechanism based on the beta distribution. It maintains a history-record of the latest transactions (1000 results, in specific). The operation's reputation is the weighted summation of these values, with the most recent ones having higher weigh. Thus, a reputation fading mechanism is implemented where the latest outcomes are considered more important in estimating the current behavior for a node.

Periodic malfunctioning is a common occurrence in WSNs. SCOTRES enhances the reputation evaluation process by adopting a statistical normalization procedure prior to beta distribution calculation. The historical values are normalized, based on the statistical normalization approach proposed in [13], and then the reputation value is calculated based on beta distribution. This normalized beta distribution system enhances the individual reasoning process of a node that applies SCOTRES; a novel approach in secure routing.

Trust aggregates reputation values from all the evaluated operations and is calculated as their weighted summation. A trust structure for a node contains its three reputation structures. SCOTRES assigns a weight of 20% for the routing, 60% for the forwarding (as forwarding is consider more important for the network durability e.g. [104], [107], [108]), and 20% for the recommending operation (based on a previous security and performance analysis on trust systems for secure routing, conducted by the authors [13]).

SCOTRES is mostly based on direct observation but can also utilize indirect knowledge. There are two evaluation processes for evaluating direct transactions with other participants and for aggregating recommendations made by 1-hop neighbors respectively. Thus, there are two relevant trust structures for each evaluated node. The Direct Trust (DT) structure contains the reputation structures R_1, R_2 and R_3 that are maintained from direct interaction with the evaluated node.

The Indirect Trust (IT) structure contains the DT values that are received from other nodes acting as recommenders. A node only receives indirect knowledge by its legitimate and trusted 1-hop neighbors. For evaluating indirect trust, the deviation test and the weighted summation approaches are combined for ranking the recommenders and aggregating the recommending trust values respectively, as described below.

A deviation test is performed prior aggregating a new recommendation. If the recommendation deviates significantly (namely, more than 30%) from the direct opinion that it is possessed about the evaluated node, the recommender is ranked negatively (as a failed transaction for its direct R_3) and its recommendation is discarded.

The IT value of a node is the average weighted summation of the recommending DTs. The weight (W_{ni}) of each DT is determined by the trust value of the relevant recommender. Thus, the recommendations that are made by trusted nodes gain higher weight.

As in most of the pertinent systems and a previous security analysis (e.g. [13], [104]) a node is categorized as trusted, legitimate, selfish or malicious based mostly on direct observation. Thus, direct knowledge gains higher weight (i.e. 80%) in the total calculation of trust. Based on the three distinct reputations and this aggregated total trust value, thresholds are set for categorizing a node in one of the four aforementioned types (trusted, legitimate, selfish and malicious). Then, the node can make recommendations. As in AODV-REX, the recommender's privacy is protected by sending this aggregated direct and indirect knowledge.

9.3.7 Policy-Based Access Control

After securing the routing operation and enhancing the correct transmission of requests and responses, there is the need to provide authorization functionality. As nodes receive requests they must decide if the requested actions can be made.

A Policy-Based Access Control (PBAC) framework [62] manages direct access requests to the resources of an embedded device, based on a predefined set of rules and policies. The solution is based on the eXtensible Access control Markup Language (XACML) policies [128] and the DPWS standard for device discovery and message exchange. The framework consists of the following four components. The *Policy Enforcement Point (PEP)* performs access control by making decision requests and enforcing authorization decisions. Typically, this component runs on any node with accessible resources and services. The *Policy Decision Point (PDP)* evaluates requests against applicable policies and renders an authorization decision. The component lays on a trusted node with sufficient resources to parse applicable policies and make the decisions. The *Policy Information Point (PIP)* and *Policy Administrator Point (PAP)* act as a source of attribute values and are used for creating and managing policies or policy sets.

A node might include one or more of these functional components, depending on its computational capabilities. The component interconnection is illustrated in Figure 23. By combining the above technologies, PBAC enables the fine-grained, policy-based control of resources from remote locations (e.g. cameras, sensors, and control stations). Based on the active policy, the framework allows for accessing the provided resources (e.g. video stream or sensor data), updating settings or even receiving alerts (e.g. in case of emergency, like fire).

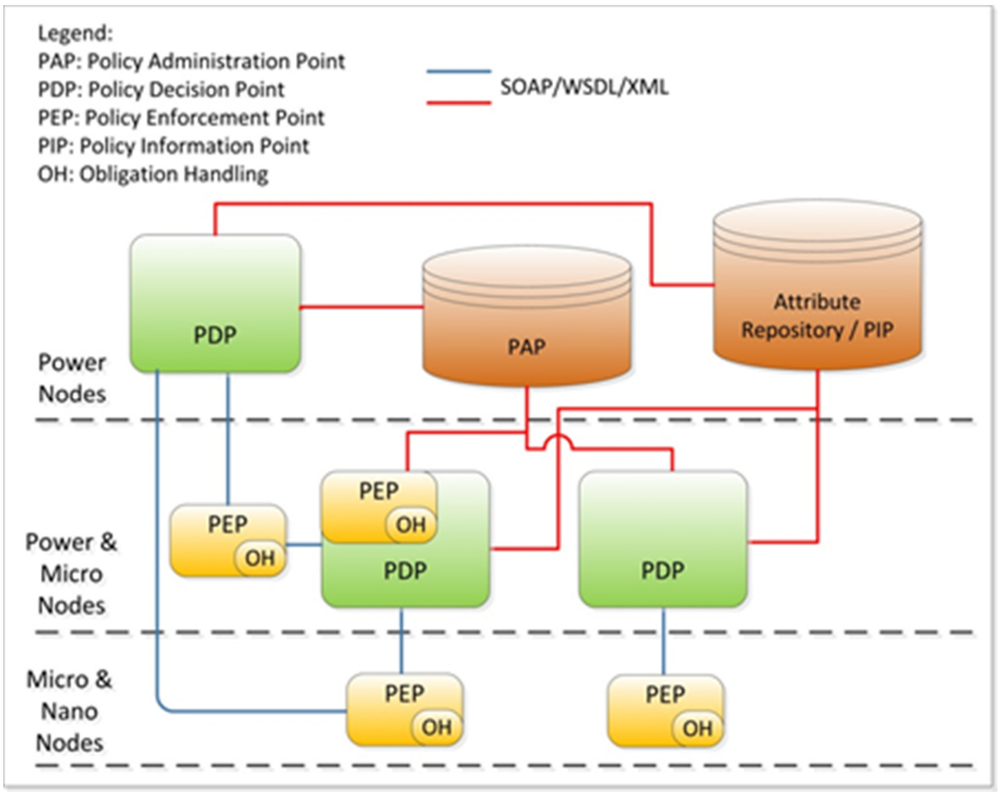

Legend:
PAP: Policy Administration Point
PDP: Policy Decision Point
PEP: Policy Enforcement Point
PIP: Policy Information Point
OH: Obligation Handling
SOAP/WSDL/XML
PDP
PAP
Attribute Repository / PIP
Power Nodes
Power & Micro Nodes
PEP
OH
PDP
PEP
OH
PDP
Micro & Nano Nodes
PEP
OH
PEP
OH

Figure 23 The policy-based access control architecture.

9.4 Security Analysis

In this subsection, we provide the theoretical analysis of our proposal and its effectiveness in countering the attacker models that are detailed in the simulation study (based on the relevant analyses in [118] and [123] respectively).

9.4.1 Theoretical Analysis

Theorem C1 is the main building block of the theoretical security analysis and describes the bounded number of packet loss. At first, we give some definitions and then the theorem and the proof.

Definition C1: Let pkt^{+} be the total number of successfully transmitted packets.

Definition C2: Let pkt^{-} be the total number of lost packets.

Definition C3: Let $Tpkt$ be the total number of transmitted packets, determined as the summation of pkt^{+} and pkt^{-}.

Definition C4: Let ρ be the transmission success rate of the totally transmitted packets $Tpkt$.

Theorem C1: The ideal network exhibits $pkt^{-} - \rho \cdot pkt^{+} \leq 0$. For up to an additive constant, ignoring a bounded number φ of packets lost, it holds that the number of lost packets is a ρ-fraction of the number of transmitted packets. Specifically, there exists an upper bound φ, as described in Equation (C1).

$$pkt^{-} - \rho \cdot pkt^{+} \leq \varphi \qquad (C1)$$

Proof: Assume that there are *N* nodes, *m* of which are malicious and $m < N$. Let *ML* be the set of links that are controlled by the malicious nodes. The maximum value of *ML* is *mN*.

Consider a faulty link *e*, which is convicted c_e times and rehabilitated r_e times. The links weight w_e is at most *mlen*, where *mlen* is the upper bound of the length of a non-faulty path in the network. If the link's weight reaches *mlen* it is considered less efficient that any possible non-faulty path. Therefore, w_e is given by Equation (C2).

$$w_e = 2^{c_e - r_e} \qquad (C2)$$

Let β be the number of lost packets that exposes a path as faulty. The number of convictions is at least pkt^{-}/β. Thus,

$$\frac{pkt^{-}}{\beta} - \sum_{e \epsilon ML} c_e < 0 \qquad (C3)$$

Similarly, the number of rehabilitation operations is at most $\frac{pkt^{+}}{\beta/\rho}$. Thus,

$$\sum_{e \epsilon ML} r_e - \frac{pkt^{+}}{\beta/\rho} < 0 \qquad (C4)$$

Therefore,

$$\frac{pkt^{-}}{\beta} - \frac{pkt^{+}}{\beta/\rho} \leq \sum_{e \epsilon ML} (c_e - r_e) \qquad (C5)$$

From Equation (C3), it holds that $c_e - f_e \leq \log w_e$. Thus,

$$\sum_{e \epsilon ML} (c_e - r_e) = \sum_{e \epsilon ML} \log w_e \qquad (C6)$$

By combining Equation (C5) and Equation (C6), we derive:

$$pkt^{-} - \rho \cdot pkt^{+} \leq \beta \sum_{e \epsilon ML} \log w_e \leq \beta \cdot mN \cdot \log mlen \qquad (C7)$$

Since $\beta = b \log mlen$, where b is the number of lost packets per window, Equation (C5) becomes:

$$pkt^{-} - \rho \cdot pkt^{+} \leq b \cdot mN \cdot log^{2}\, mlen \qquad (C8)$$

Therefore, the amount of disruption a dynamic attacker can cause to the network is bounded. If there are no malicious nodes Equation (C8) describes the ideal case, where $pkt^{-} - \rho \cdot pkt^{+} \leq 0$. ■

In contrast with the simple grading that is applied by the most relevant trust schemes and rates failure with *-1*, gradual grading that is proposed by SCOTRES rates failed transactions with *-4*.

Lemma C1: The gradual grading of SCOTRES decreases the attack rate to at least *¼*.

Proof: Based on Equation (C8), it is derived that $4 \cdot b_s \geq b_g$, where b_s and b_g are the number of lost packets per window with simple and gradual grading respectively. ■

Statistical normalization erase the effects of occasional malfunctioning. Then, reputation fading assigns higher weight to more recent behavior. Thus, b is further decreased in the current time window due to failures.

The overall load balancing mechanisms of the topology and energy metrics decrease failures in congested periods (pkt^{-}) while enhancing the successful transactions (pkt^{+}).

In case of a jamming attack, gradual grading and reputation fading, constrain the number of packet loss (pkt^{-}) until the channel-health metric detects and re-routes traffic, enhancing pkt^{+}.

Selecting the shortest of the most reputable paths instead of selecting the shortest ones, may increase $mlen$, bounding the attacker's disruption.

When evaluating the forwarding operation, these mechanisms protect the network against ballot-staffing attacks, like blackhole, grayhole, selective forwarding, sleep deprivation and flood rushing. For the routing operation, SCOTRES can detect Hello flooding, routing table poisoning, and false route error. The indirect trust evaluation restricts badmouthing and ballot-stuffing attacks to at most *20%*, preventing the establishment of wormholes and sinkholes.

As attacking nodes exceed the malicious threshold they are detected and excluded ($pkt^{-} \geq mal_{thr} \rightarrow (m = m - 1)$). The attack rate is further decreased as *mN* is decreased. If all attackers are detected *mN* becomes *0*, resulting the ideal case.

9.4.2 Attacker Model

We consider active attackers that perform any type of the aforementioned attacks on routing protocols and trust systems. We define the attacker model (Definitions C5-C9) and the attack effort (Lemmas C2 and C3), based on the formal analysis of the Ariadne secure routing scheme [123].

Definition C5: *Attacker-m-N* denotes an attacker who owns *m* malicious nodes out of the total *N* nodes of the WSN.

Definition C6: *Legitimate-n-N* are the remaining *n* legitimate nodes out of *N* (where *m*+*n*=*N*).

As the volume of *m* increases, there may be malicious nodes that do not require to actively participate in the attack.

Definition C7: *Active-x-m* reports that *x* malicious nodes out of m were active during the attack.

Definition C8: *Counter-y-x* refers that *y* active malicious nodes out of *x* where successfully countered by the underlying system.

Definition C9: *FalselyAccused-z-n* reveals that *z* legitimate nodes out of *n* where falsely countered as malicious nodes.

Lemma C2 and Lemma C3 describe the situation where an attack is considered successful and the attacker's effort to launch the attack respectively. No proofs are required.

Lemma C2: An attack is successful if either there is one or more active malicious node that have not been countered *([Active-x-m – Counter-y-x]* > *0*) or there is at least one falsely accused legitimate node (*FalselyAccused-z-n* > *0*).

Lemma C3: The effort to accomplish the attack is the percentage of malicious nodes that are required to exploit the system (*[m*×*100]/N*).

Definition C10 determines the attacker's effectiveness that is utilized in the simulation analysis to assess the SCOTRES's resilience to different types of attacks.

Definition C10: The effectiveness of a successful attack is designated by the analogy of the malicious nodes that are deployed to the non-countered and falsely accused nodes. Equation (C9), defines the attack's effectiveness (*AE*).

$$AE = \left(\frac{[Active-x-m]-[Counter-y-x]}{Attacker-m-N} + \frac{FalselyAccused-z-n}{Legitimate-n-N}\right)/2 \qquad \text{(C9)}$$

We regard that a system counters an attack if *AE* is lower than 0.2 and partially counters it if *AE* is from 0.2 to 0.3. We consider settings where the attacker owns up to 50% of the total nodes. WSNs with higher malicious ratio should be discarded.

9.5 Simulation Analysis

NS2 implements the DSR protocol in C++. This implementation is extended to deploy the integrated SCOTRES_DSR. The assignment of all constant coefficients is based on a previous security and performance analysis, conducted by the authors [13] under the same platform. The simulation study revealed the effectiveness of the core components for different reputation and trust settings. Results from relevant studies regarding the thresholds for detecting active attackers [129], [130] are also considered. All these results are adopted by SCOTRES (e.g. the positive and negative reputation rating values). The systems S-D RepIDS, AODV-REX, RFSN, TRDSA, EFW, and SR3 are also implemented on the same platform, in order to provide a fair comparative analysis under the same attack scenarios.

In order to evaluate the various routing mechanisms, a WSN was modeled, consisted of 50 nodes. The two-ray ground reflection model is used for propagation and the IEEE 802.11 Distributed Coordination Function (DCF) is used for the MAC layer. Every node has a raw bandwidth of 2Mbps and a physical radio range of 100m. The simulation area occupies 350m × 350m. Each experiment includes two phases. At initialization, nodes start with the default trust values as defined by each scheme. At evaluation, the normal operation is monitored, as well as four attack scenarios, measuring performance and security. In both phases, 10 source nodes on one end of the WSN send 1KB data with Constant Bit Rate (CBR) to 10 destination nodes at the other end of the WSN. Each phase lasts 1 min.

The first scenario examined was the one of normal operation, i.e. with no attacks taking place. Thus, we compared the normal behavior of the pure DSR and the seven secure routing schemes. One experiment per system is performed, as the result is deterministic for each setup.

Then, malicious nodes are introduced in the WSN in order to perform four types of attacks, determining the security level of the examined systems. For each attack, 5 experiments were performed per system, using 10, 20, 30, 40, and 50 percent of malicious nodes respectively. Each experiment was executed 10 times and the average metrics were calculated. In total, there were 50 iterations for each system per attack. At each iteration, the malicious nodes were assigned randomly. For the first evaluated system, the malicious nodes of each iteration were recorded – a necessary step in order to use the exact same setting in all systems, ensuring they are compared under identical situations.

9.5.1 Normal WSN State

The systems are initialized in the first phase and the following four metrics for load-balancing and power consumption are evaluated.

$\mathbf{M_{1\text{-}1}}$ is the percentage of inactive intermediate nodes that do not participate in the forwarding operation. A high percentage of inactive intermediates reveals poor load-balancing as a few intermediate nodes serve all the communication effort. $\mathbf{M_{1\text{-}2}}$ is the average amount of data (in KB) that is forwarded by the active intermediate nodes. The lower the value, the better. $\mathbf{M_{1\text{-}3}}$ estimates the power consumption by calculating the operations that are performed by each node. One operation unit is added for each reception or sending operation. Based on $M_{1\text{-}3}$, $\mathbf{M_{1\text{-}4}}$ measures the variation of node consumed power (COV). Equation (C10), describes the COV metric.

$$COV = \sigma(M_{1-3})/\mu(M_{1-3}) \qquad \text{(C10)}$$

Where σ is the standard deviation and μ is the mean deviation. Energy-balancing is achieved for low COV values. The metrics M_{1-2} and M_{1-4} are utilized to identify the systems that achieve good load- and energy-balancing. The lower the values of the two metrics, the better. Figure 24, illustrates the evaluation results.

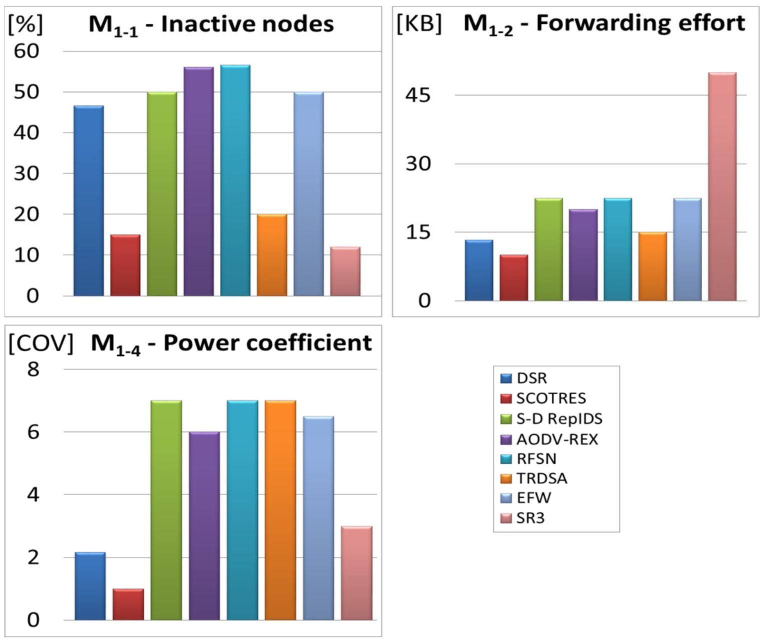


Figure 24 Evaluation results of the secure routing schemes for the simulation metrics M_{1-1}, M_{1-2}, and M_{1-4}

In four secure routing systems (S-D RepIDS, AODV-REX, RFSN, EFW), more than half of the intermediate nodes remained inactive during the experiment and did not forward any packets. This is the result of the typical reasoning process used in such systems. All nodes start with the default reputation values. As the first nodes start forwarding traffic, their reputation is increased and they are continually selected (as no malfunctioning occurs). Thus, a few nodes serve all the traffic. Pure DSR achieve a slightly better load-balancing as it selects the shortest path from a source to a destination. Thus, nodes that are located at the boundaries of the WSN were also selected for some transactions, while in the former case the intermediate nodes at the center of the WSN gain higher reputation and were preferred. TRDSA takes into account the energy consumption. It achieves low inactive-node ratio and forwarding effort. SR3 produce the best dispersion as it entails longer and random routes. This results in higher forwarding effort, as the intermediate nodes participate in more routes than in the rest systems. SCOTRES achieved the best load-balancing performance with the lower average forwarding activity and only 10% of the intermediate nodes remaining inactive.

The load-balancing capability proportionately affects the power consumption and the longevity of the nodes. Thus, SCOTRES consumed the least mean power among the evaluated systems. For the rest of the systems, high values of the COV metric (M_{1-4}) reveal that a few active intermediate nodes were overburdened.

9.5.2 Attack Scenarios

Four attack scenarios are modelled to evaluate the security of each setting. The attacks target both the trust schemes and the routing protocol vulnerabilities. In the first case, the malicious nodes perform a social-based on-off attack at initialization and a blackhole attack at the evaluation phase. In the second scenario, the attackers perform ballot-based attacks (ballot-stuffing and badmouthing) at initialization, enabling link-spoofing attacks during the evaluation phase. In the third attack, the malicious nodes take advantage of the congested periods and network's topology in order to make legitimate nodes unavailable through flooding attacks (in forwarding or routing) at the evaluation phase. The goal is to overburden these legitimate nodes and make them misbehave, harming their reputation. In the last attack, the malicious nodes perform constant jamming attacks at the evaluation phase. The affected nodes start misbehaving as they cannot properly send and receive data and their reputation is decreased.

Under a blackhole and jamming attacks, the performance of a system is measured as the delivery ratio – the percentage of the packets that were successfully transmitted from the source to the destination to the total packets that were sent, defined as $\mathbf{M_2}$ and $\mathbf{M_5}$ respectively. The higher the value, the better. The link-spoofing effect, metric $\mathbf{M_3}$, is calculated as the percentage of the total transactions that were routed through paths that contain at least one malicious node due to the effect of the ballot-based attacks. The lower the value, the better. The metric $\mathbf{M_4}$ calculates the number of legitimate nodes that were rendered unreachable by the energy- and topology-based attacks. The lower the value, the better. Figure 25, illustrates the resulting metrics.

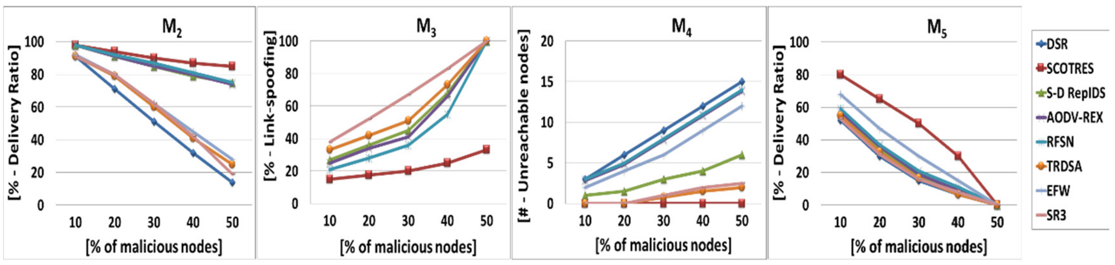


Figure 25 Evaluation results of the secure routing schemes for the simulation metrics M_2, M_3, M_4, and M_5

The evaluation of DSR reveals the effectiveness of the attack on an unprotected system. The systems' performance worsens as the malicious nodes increase. In general, systems that can quickly discover the malicious nodes achieve high detection ratio and retain network's performance by eliminating the negative effects. As the malicious nodes increase, the delivery ratio is decreased, while the average affected transactions per source node are increased. The detection ratio follows a conflicting behavior. As the volume of the malicious nodes increases, many malicious nodes participate in the same path. Thus, the first malicious nodes perform the attack until they are detected and punished, while the rest of the malicious nodes could remain inactive during the evaluation phase.

TRDSA and EFW perform poorly under blackhole attacks due to their simple transaction evaluation processes and slow adaptability. SR3 detects malfunctioning faster, but the lack of recommendations and the high dispersion lead each one of source nodes to come across almost every malicious node. AODV-REX, RFSN, S-D RepIDS and SCOTRES discover the attackers faster, due to the reputation fading feature. SCOTRES achieves the best results due to the more robust reasoning process for evaluating direct knowledge and making recommendations.

Badmouthing attacks lower the trust level of legitimate nodes and harden their selection as forwarding nodes. As a result, the malicious nodes with legitimate trust are preferred and their trust is increased. Similarly, ballot-stuffing increases the trust values of malicious nodes. Then, malicious nodes are preferred for forwarding while legitimate nodes participate in less transactions and their trust level is decreased. In all cases, as the malicious nodes increase, their average trust value is also increased, while the average trust value of the legitimate nodes is decreased. Thus, the link-spoofing succeeds in both cases.

The ballot-based attacks that are performed are determined by the type of recommendations that are imposed by the evaluated system. S-D RepIDS and TRDSA use negative recommendations (badmouthing), RFSN and SCOTRES utilize both negative and positive recommendations (both badmouthing and ballot-stuffing). SR3 are based on acknowledgement messages to verify a successful transmission. In that cases, the colluding attackers drop the acknowledgement in paths where the legitimate nodes are more than 50%. Thus, a high rate of legitimate nodes can be falsely accused, while paths with many malicious nodes are seem reliable. The pure DSR and EFW are not evaluated in this scenario, as no recommendations are used.

SR3 is vulnerable to acknowledgement misuse and produces the highest false accusations and link-spoofing effect. TRDSA also achieves high false accusation ratio due to badmouthing as it does not properly handle negative recommendations; the system accepts these when they originate from trusted nodes or from malicious leaders with higher trust respectively. Thus, malicious nodes who cooperate in the initialization phase, exploit the recommendation operation. S-D RepIDS is less vulnerable to badmouthing as it gives higher weight in direct knowledge (a node is recognized as malicious only by direct interaction). AODV-REX, RFSN and SCOTRES deal with both attacks as the recommendations are also collated with direct knowledge.

With regard to link-spoofing, SR3 provides little protection. TRDSA uses simple recommendation operations that cannot effectively counter such an attack. AODV-REX weights direct and indirect knowledge the same; making it vulnerable to high ratio of attackers. S-D RepIDS uses more advanced mechanisms for mitigating these effects. RFSN is the best among the examined systems of the related work. SCOTRES implements an even more robust recommendation evaluation mechanism and rates the recommenders against direct interaction. It is the only evaluated system that can detect the ballot-stuffing attacks and successfully counter the link-spoofing effect. For M_3, SCOTRES achieves the best result as it stops the link-spoofing attack, once the bad recommenders are detected and punished.

The nodes that were found vulnerable due to congested periods were mostly located at the center of the WSN, where the bulk of the data was forwarded to. The nodes that were vulnerable to topology-based attacks were located at the boundaries of the network, connecting source or destination nodes with the rest network. From the evaluated systems, DSR, AODV-REX, RFSN and EFW provide no special treatment for congested periods or topology-based attacks. S-D RepIDS provides protection in congested periods. SR3 reduces congestion traffic due to the random nature of the random walk algorithm. TRDSA makes energy-aware decisions about routing and mitigates the energy-based attacks. SCOTRES protects the network in both cases.

The most flooding attacks were performed under DSR, AODV-REX, RFSN and EFW. For DSR, there were some paths that were rendered unavailable and falsely considered as broken. AODV-REX and RFSN perform the same and exhibit the most false accusations and unreachable nodes. S-D RepIDS exploits its fault-tolerance mechanism in congested periods, mitigating the false accusations. The MAC-layer measurements of EFW assigns low communication reliability to the overloaded links. It routes the traffic through alternative links mitigating the false accusations. In the cases of TRDSA, SR3 and SCOTRES, mostly topology-based attacks were performed, as the load-balancing mechanisms mitigated the overloaded nodes and the relevant flooding attacks. SCOTRES detects the flooding as malicious activity when it is performed in congested periods through overload nodes or topological-significant nodes. It demonstrated the best behavior in terms of countering the attack, as there was only one false accusation in the scenario involving 50% of malicious nodes, and all source and destination nodes remained reachable.

For a high volume of jamming attackers (50%) no communication could be accomplished, as they disrupt all the paths from the source nodes to the destinations. DSR and the five secure routing schemes (S-D RepIDS, AODV-REX, RFSN, TRDSA, SR3) perform poorly, even under a small number of attackers. EFW and SCOTRES are the only system to implement a fault tolerance mechanism to mitigate the effect of a jamming attack, achieving a good delivery ratio, up to a certain amount of jammers (10% - 30%). SCOTRES is more advanced as the health-metric is responsible for detecting jamming attacks. Moreover, the reputation of the legitimate nodes is protected in any case, as there were no false accusations even with 50% of malicious nodes.

9.5.3 SCOTRES's Measured Resilience to Attacks

We model as an *Attacker-m-50* the nodes that perform the aforementioned attacks in the four scenarios. In the first two cases, SCOTRES successfully countered the attacks for all *Active-x-m* that were evaluated without any false accusation (*Active-x-m = Countered-x-x, and FalselyAccused-0-n*). The AE was 0 for all models. In the third case, the active malicious nodes were always detected while there was a small number of false accusations for high ratio of malicious nodes. The best attack achieved 1 false accusation (*FalselyAccused-1-25*) for *Attacker-25-50*, with *AE=0.02*. SCOTRES successfully countered all three types of attacks. In the fourth case, all malicious nodes were active (*Active-m-m*). *FalselyAccused-0-n* holds in all cases. SCOTRES successfully countered the attack for low ratio of malicious nodes (*Attacker-5-50* achieved *AE=0.1*, *Attacker-10-50* achieved *AE=0.15*). For medium volume of attackers, it partially countered the attack (*Attacker-15-50* achieved *AE=0.23*). Little to no protection was provided for high number of attackers (*Attacker-20-50* achieved *AE=0.35*, *Attacker-25-50* achieved *AE=0.5*).

9.6 Precision Agriculture Issues

The evolution of the Internet of Things (IoT) motivates the deployment of intelligent cyber-physical systems (CPS) that utilize wireless networking. The market value of industrial settings was \$157.05 million in 2016 and it is expected to grow at a compound annual growth rate (CAGR) of 33.3% from 2016 to 2021[11]. According to the United Nations Environment Programme and the International Fund for Agricultural Development[12] around 2.5 billion people live from agricultural production systems. Research and governmental initiatives try to increase productivity and consumer's safety by applying IoT technologies to monitor weather and crop growth [131] and CPS to manage production systems, like watering [132] and spraying pesticides with UAVs [133].

In such wireless ad-hoc networks, each entity relies on its neighbors to carry its messages and successfully communicate with all participants. Due to the open medium and the dynamic entry of new nodes to these networks, routing protocols must establish trust relationships to avoid malicious nodes. The trust-based schemes, like SCOTRES, are utilized in wireless networking to provide secure routing functionality.

In addition to the NS2 simulations, SCOTRES is also deployed on real embedded platforms and mote devices, implementing an agricultural application. The complete system consists of three main components: (a) A cryptographic service that enforces authentication, message integrity, and confidentiality; (b) An efficient trust-based routing scheme that protects the communication against ad-hoc routing attacks, as mentioned above; (c) A policy-based access control framework that provides authorization. SCOTRES adopts efficient and state-of-the-art features for trust. It also enforces authentication and integrity checks as the rest schemes. Nonetheless, after successfully authenticating a node and validating the fair use of the network, authorization is required in order to perform the requested activity. The combination of the three services is a novel approach to provide an effective overall protection.

9.6.1 Precision agricultural CPS Scenario

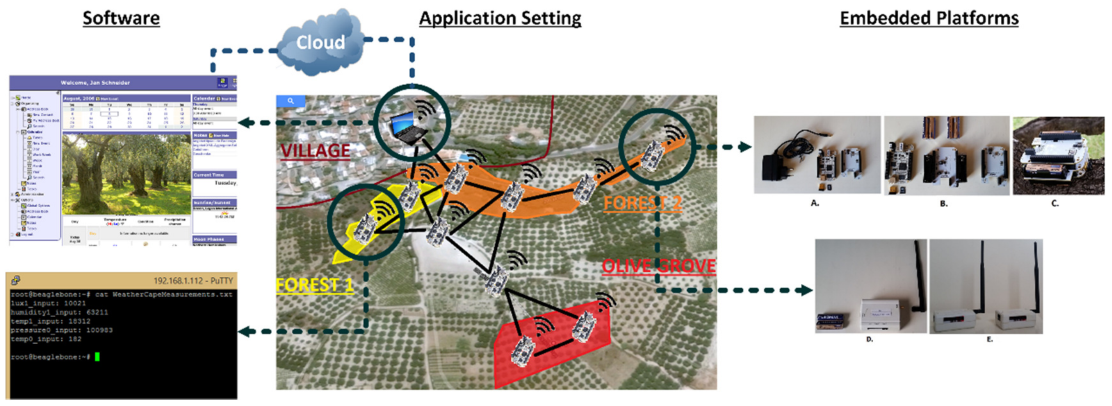
A precision agricultural CPS is considered, where the devices collect environmental parameters (e.g. temperature and humidity) and transmit them to a processing center (laptop with WiFi capability). The processing center publishes the measurements to a cloud management system (e.g. [134]), where the user can gain access via Internet connection. Then, the farmers make decisions regarding fertilizer appliance, watering, pruning, digging or other rural activities.

The user can access the aforementioned functionality through cloud. The application is implemented on the Greek Research and Academic Community cloud service, named okeanos [135].

The system is tested in a rural settlement of a small village of the Crete Island in Greece. The WSN monitors two small forests and one olive grove, nearby the village. The devices transmit the measured parameters every one hour, total 24 transmissions for each device per day. The processing center is located at the community's public building and the users have access through the community's free WiFi connection that covers the whole village. Figure 26 illustrates the application setting. The picture is captured by Google Maps at location (35.277297, 25.028250). The laptop runs the web service for managing the system. Figure 26.A. BeagleBone with USB-WiFi, SD memory card, weather cape and power supply. Figure

[11] Markets And Markets: http://www.marketsandmarkets.com/Market-Reports/internet-of-things-market-573.html
[12] UNEP: http://unep.org/pdf/smallholderreport_web.pdf

26.B. BeagleBone with USB-WiFi, SD memory card and the weather and battery capes. Figure 26.C. The composed equipment of Figure 26.B. applied in the olive grove. Figure 26.D. Z1 with 2×AA batteries and wire antenna. Figure 26.E. Two Z1 motes that communicate the sensed temperature with IEEE 801.15.4 and 6LowPAN.



```
root@beaglebone:~# cat WeatherCapeMeasurements.txt
lux1_input: 10021
humidity1_input: 63211
temp1_input: 18312
pressure0_input: 100983
temp0_input: 182

root@beaglebone:~#
```

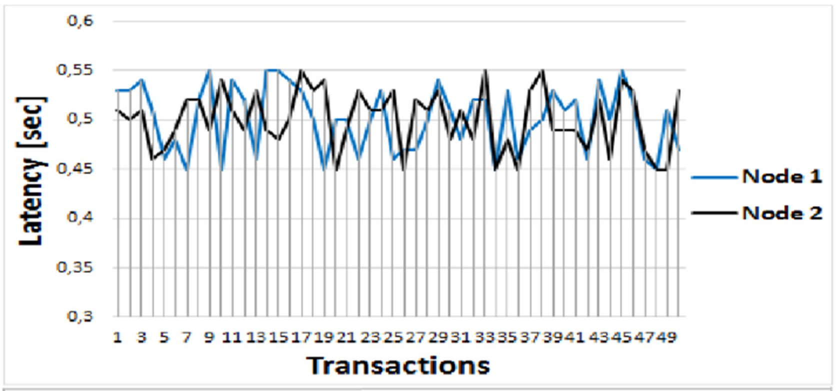
Figure 26 The precision agriculture use case architecture and demonstration setting

9.6.2 Demonstration on Embedded Platform

SCOTRES is implemented on real embedded devices and its overhead compared to the pure DSR protocol was measured. The DSR-UU implementation of the routing protocol is extended and deployed on BeagleBone devices (low-cost credit-card-sized embedded device that runs a compact Linux OS). A network of ten BeagleBone devices is created and connected wirelessly via a USB Wi-Fi module.

We measure the overhead that is added to the DSR-UU protocol by the SCOTRES scheme during normal network conditions (i.e. without any attacks) in terms of executable code size (KBs in ROM), and average memory requirements (RAM consumption in bytes) and processing delay (ms of CPU time). Measurements were taken for a one day operation of each system. The three proposed SCOTRES metrics for topology, energy and channel health consume little resources. The reputation calculation is the most resource-consuming component, due to the history of past values that is maintained for each evaluated operation. This feature proportionally affects the resource consumption of the trust metric. The added overhead for routing and forwarding is low. Thus, the average network latency (end-to-end communication) is also low, around 0.5 sec. Figure 27.A depicts the latency of two randomly selected nodes. The integrated SCOTRES_DSR-UU requires around 50% more memory and 70% more computational resources than pure DSR-UU. The proposed system consumes more resources, however, the additional overhead can be considered acceptable for the combination of good security and load-balancing behavior that it achieves. Figure 27.B, summarizes the overall resource consumption.

A.



B.

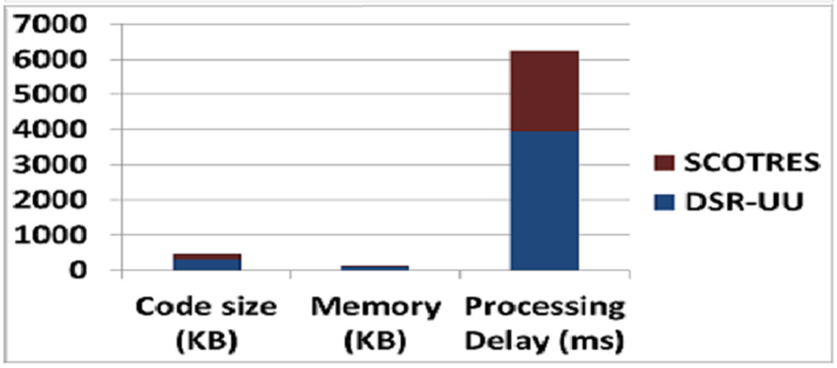


Figure 27 Performance evaluation of SCOTRES – A. SCOTRES's latency. B. Resource consumption of SCOTRES_DSR-UU

9.6.3 Demonstration on Mote Devices

As a proof of concept, SCOTRES is also applied on Zolertia Z1 motes. Z1 is a low-power WSN module that runs the Contiki OS. It uses a single core 16-bit RISC CPU at 16 MHz (MSP430F2617 microcontroller) with 8 KB RAM and 92 KB flash memory. Z1 is equipped with build-in temperature (TMP102) and accelerometer (ADXL345) digital sensors. The networking features include the CC2420 transceiver at 2.4GHz, enabling communication with IEEE 802.15.4 and 6LowPAN. A wire antenna is utilized with radio range of around 25 meters. It is powered by two AA batteries (3.3V) or through a u-USB port (5V).

SCOTRES is applied on Z1 motes that are powered by batteries and communicated wirelessly the sensed temperature via IEEE 801.15.4 with 6LowPAN. In order to reduce the resource consumption, the history elements of each reputation structure is reduced from 1000 to 100. Moreover, all data structures and variables are becoming static with fix size to minimize the RAM usage. The power consumption is measured in mW, the energy dissipation (the time instance since the network deployment until the first node exhausts its energy below the minimum energy required for transmission under any channel condition) and the network life time in months of operation (the time duration from the instant of the sensor network deployment to the instant that the signal field cannot be reconstructed with a given QoS requirement from the current live sensors) under a lab setting and the aforementioned rural application.

Under the lab setting, the motes continuously transmit data. On average a mote running SCOTRES consumes *5.87 mW*. It takes around *15 days* until a mote runs out of energy. Under the rural application setting, the motes run SCOTRES to transmit data and evaluate the result every one hour (not continuously). The energy dissipation is around *40 months* and the network life time is *50 months*.

10. AI Reasoning and Management – Smart Campus Use Case

AmbISPDM is applied on a smart campus setting, providing protection against cyber-attacks and supporting the evacuation plan in case of fire. A real installation is presented and the performance during the two scenarios is evaluated.

10.1 Smart Campus Issues

AmbISPDM focuses on two real-world problems that trouble smart building applications: the protection of the system's SPD properties and the safeguard of peoples' safety.

As mentioned in the related work section, the security and privacy concepts of smart buildings are not properly handled by current solutions. The early smart-home installations are vulnerable to cyber-attacks and the management of the SPD properties remains in an unacceptable level. AmbISPDM tends to counter these issues and provide an efficient and effective automatic response to attacks by reconfiguring the system at runtime and protecting the SPD-sensitive assets.

In USA, 3870 structures in campuses and dorms caught fire on an annual basis in 2009-2013[13]. In EU, Sweden exhibits the most fires in schools with an associated average annual cost of up to 120 million dollars in 1998-2004[14]. Except from the economic cost, serious injuries or even several deaths occurred. In Greek housing or off-campus housing 89 fires killed 126 people during the period 2000-2015.

Our system continuously monitors the campus's indoor and outdoor regions in order to quickly detect fire, inform people and assist the successful execution of the evacuation plan. The beginning of fire is detected through smoke detectors. The reasoning activity of a fire warden is modelled to assist the emergency management based on the emergency evacuation guidelines of the USA National Fire Protection Association[15]. The goal is to achieve the targeted evacuation time of *2.5 minutes*[16] after the fire alarm indication.

A set of rules implements the reactive evacuation plan strategy once the fire gets started. In brief, cyber-warden's actions include:

1. As the first responder, the agent that detects the fire informs the fire brigade, the incident controller (caretaking staff), and the rest campus agents.
2. On raising the alarm, each agent performs the fire warden plan in its region
3. Except from the main areas and rooms, the alarm must be propagated in every section (e.g. inner offices and WCs).
4. People are directed to safe designated areas.

[13] National Fire Protection Association (NFPA): http://www.nfpa.org/public-education/by-topic/property-type-and-vehicles/campus-and-dorm-fires
[14] Kaggle Datasets: https://www.kaggle.com/mikaelhuss/swedish-school-fires
[15] http://www.nfpa.org/
[16] http://www.it-tallaght.ie/contentfiles//Documents/Estates%20Office/Full_Evacucation_Procedures_Nov%2013.pdf

5. Once reaching the safe area, people are continuously advised to stay there until the *"all-clear"* is given.
6. The agents check that all persons have left the building.
7. Agents should be aware of any person with disability (e.g. wheelchair user, impaired hearing) and assist them to a safe place.
8. In case of injury or entrainment the incident controller is informed.
9. Conventional electric equipment is closed (e.g. air-conditions, computer systems that do not participate in the crisis management) and all lights are switched on.
10. Once an indoor space has been evacuated, electronically controlled doors and windows are closed (in order to decrease the fire spreading time)
11. Data from the whole operation are recorded for later analysis and discussion.

The following subsection details the implementation of AmbISPDM on real embedded devices and its demonstration on two emulated scenarios.

10.2 Smart Campus Scenario

As a proof-of-concept example, AmbISPDM assesses and manages a system which involves an ambient environment, deployed for protecting the residents of the smart buildings on a university campus.

The hardware platforms consist of ARM-based devices (BeagleBones) that communicate wirelessly, monitor environment parameters, and control *"smart"* equipment, like electronic doors. Moreover, the PBAC mechanism [62] is implemented for managing the residents' access based on their access rights, defined by XACML policies; said mechanism exploits and extends the DPWS functionality already present on the framework's devices, to realize the necessary mechanisms (i.e. Policy Enforcement Points on the embedded devices and Policy Decision and Information Points on the control nodes). Each agent manages a smart building and runs on a local PC.

The agents evaluate the current security level of their underlying sub-systems and the system as a whole. They also monitor their respective domains, managing them based on the SPD and safety artificial intelligence reasoning processes. The agents can change the configurations of a system in order to increase the SPD level when an attack is detected (e.g. increase the size of the encryption keys) and then return to the previous state when the attack is over (to conserve resources). In a safety-related incident, the agents can control the electronic equipment to inform or help the residents in the case of an emergency.

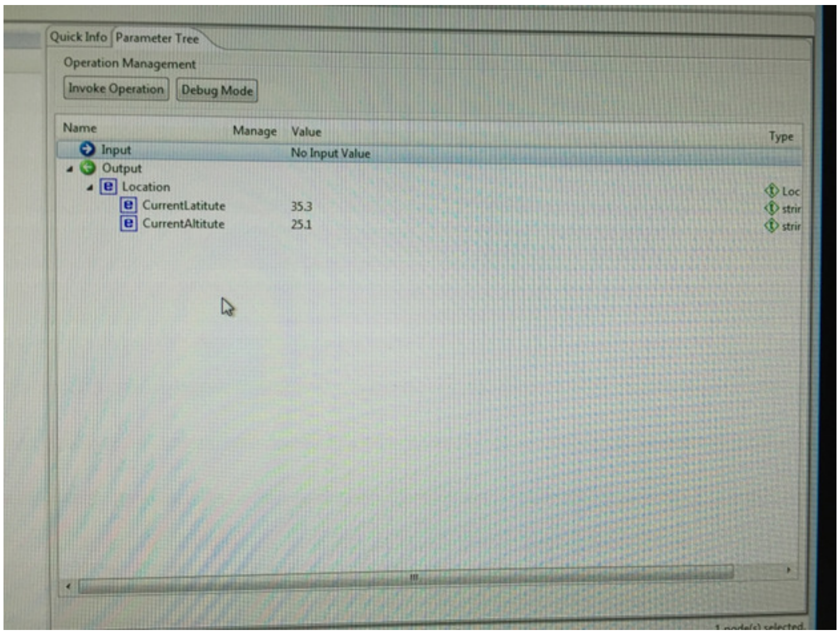
For the first scenario, the defense strategy against a cyber-attack is planned. An attacker performs a Denial-of-Service (DoS) attack to one of the servers that are located in the universities computer center. The responsible agent that detects the malicious activity, configures the system in order to counter the attack and increases the SPD level. It also informs the rest agents regarding the attack status. They also increase their SPD as a precautionary measure. When the attack is successfully encountered, the whole system is returned to the state of normal operation and SPD.

The second demonstration scenario involves modelling the response plan in case of fire alarm where decisions about both SPD and safety are made. Normally, the residents can only open the doors that they are allowed to by their access rights (based on access control rules). When the sensor nodes detect fire, they raise an alarm informing the monitoring agent. The following piece of code presents the relevant CAP alert for the fire detection.

CAP Alert for Fire Detection

```
<soap:Envelope xmlns:soap=
"http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ns3:Notify xmlns="http://docs.oasis-open.org/wsrf/bf-2"
xmlns:ns2="http://www.w3.org/2005/08/addressing"
xmlns:ns3="http://docs.oasis-open.org/wsn/b-2"
xmlns:ns4="http://protectrail.eu/model/events/resource"
xmlns:ns5="urn:oasis:names:tc:emergency:cap:1.2"
xmlns:ns6="http://docs.oasis-open.org/wsn/t-1">
<ns3:NotificationMessage>
<ns3:Topic Dialect="http://docs.oasis-open.org/wsn/
t-1/TopicExpression/Full">FireDetection</ns3:Topic>
<ns3:Message>
        <ns5:alert>
        <ns5:identifier>urn:rixf:com.tuc.AmbISPDM:id/
        FireAlert_01</ns5:identifier>
        <ns5:sender>WSN</ns5:sender>
        <ns5:sent>2016-10-19T10:43:09.000+02:00
        </ns5:sent>
        <ns5:status>Actual</ns5:status>
        <ns5:msgType>Alert</ns5:msgType>
        <ns5:source>urn:rixf:com.tuc.fireprotection/
                        devices/WSN</ns5:source>
        <ns5:scope>Public</ns5:scope>
        <ns5:info>
                <ns5:category>Fire</ns5:category>
                <ns5:event>FireDetection</ns5:event>
                <ns5:responseType>Evacuate
                </ns5:responseType>
                <ns5:urgency>Immediate</ns5:urgency>
                <ns5:severity>Extreme</ns5:severity>
                <ns5:certainty>Observed</ns5:certainty>
                <ns5:parameter>
                        <ns5:valueName>Area
                        </ns5:valueName>
                        <ns5:value> 35.317469,25.102844
                        0.01 </ns5:value>
                </ns5:parameter>
        </ns5:info>
        </ns5:alert>
</ns3:Message>
</ns3:NotificationMessage>
</ns3:Notify>
</soap:Body>
</soap:Envelope>
```

Figure 28 illustrates the DPWS request operator interface of the device that detected the fire. The requested latitude and altitude parameters are included in the CAP message.



Name	Manage	Value	Type
Input		No Input Value	
Output			
Location			Loc
CurrentLatitute		35.3	strir
CurrentAltitute		25.1	strir

1 node(s) selected

Figure 28 DPWS request operator for the device that detected the fire in the smart campus use case.

The piece of code below presents the rule in Jess of the relevant SA that handles the fire detection alert. *ReportAlert* is a Java method that implements the aforementioned functionality of informing the rest participants about the fire.

Handle *FireDetection* Alert in DECKT-Jess

```
                    ; Handle FireDetection Alert
(defrule MAIN::HandleFireDetectionAlert
(declare (salience 570))
(Time (tpoint ?t))
?event <- (event (name FireDetection) (arg ?n))
(EC (predicate Happens) (event ?event) (time ?t) )
?fluent <- (fluent (name CaughtFire) (arg ?n))
(EC (predicate HoldsAt) (posLtrs ?fluent) (time ?t) )
(not (EC (predicate Terminates) (event ?event)
        (posLtrs ?fluent) (time ?t)))
=>
(assert (EC (predicate Terminates) (event ?event)
        (posLtrs ?fluent) (time ?t)))
(ReportAlert "Fire Detection")
(printout t "Handle Fire Detection Alert" ?t crlf) )
```

The agent decides to lower the security level, unlocking all doors and permitting the unhindered exit from the building. Privacy is also decreased in order to permit the continuous monitoring of the residents through smart cameras and ensure that everyone has left the building. All lights are switched on, based on the emergency management plan, and the conventional electronic equipment is turned down once a room is evacuated. As fire damages the system's equipment and the communication is lost, dependability is lowered, revealing the subsystem's inability of working properly and provide the full functionality.

The fire brigade and the incident controller are automatically informed by email or SMS. The agent also informs its peers (i.e. agents of nearby buildings) to support the evacuation plan. The plan includes the routes that the residents are meant to follow in order to reach some predefined safe areas around the campus.

As the plan is executed, the fire is extended. Another agent detects the danger in its region that threats the evacuating personnel and raises a conflict. The responsible MA collects the local information from the two involving agents and perform the conflict resolution mechanism. After considering the most recent evidence from the second agent, the MA terminates the current plan, re-runs the reasoning theory with the new facts and announces the next evacuation activities. As the fire is extinguished, the system is returning to the normal state. When the damage equipment will be replaced, dependability will also be restored.

10.3 Demonstration

The scenario steps appear in more detail in Table 15 (red illustrates SPD values of 0-50, yellow illustrates SPD values of 51-70, and green illustrates SPD values of 71-100).

Table 15 Scenario steps of the smart campus use case

STEP	Events	Effect	Overall (S,P,D) Value	State Visualization
1	Power-on of all systems and discovery/registration	Initial State	**80,70,65**	S 🟢 P 🟡 D 🟡
2	A DoS attack is detected at the server of SA_1. *MA* is informed that an attack occurs and it sends a command to the rest building agents to increase security.	Security level decreases	**60, 70, 65**	S 🟡 P 🟡 D 🟡
3	Security level is increased on all SAs. MA is informed.	Security level increases	**85,70,65**	S 🟢 P 🟡 D 🟡
4	The server has counteracted the DoS attack and SA_1 reports the change to the *MA*. The *MA* asks the *SAs* to return to normal state (to conserve resources).	Security level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡
5	The central building catches fire. The fire is detected by the buildings WSN and the relevant SA_2 is informed. SA_2 transmits SMS to the fire brigade and the incident controller. SA_2 Dependability decreases. SA_2 Privacy decreases (incident controller now has access to exact resident location). SA_2 Security decreases (encryption is disabled to facilitate emergency response and emergency services). The *MA* is informed of the changes and sends email to operators/administrators and other stakeholders.	S, P, D levels decrease	**50, 40, 50**	S 🔴 P 🔴 D 🔴

6	Fire expands. *SA_1* detects the fire in each region. *SA_1* performs similar actions as *SA_2*. S, P, D levels decrease. *SA_1* raises a conflict regarding the evacuation route. *MA* collects information from *SA_1* and *SA_2* and resolves the conflict.	S, P, D levels decrease	**45, 45, 30**	S 🔴 P 🔴 D 🔴
7	Fire is extinguished. Security and privacy levels are restored. *SA_1* and *SA_2* report new state to *MA*.	S, P levels increase	**80, 70, 30**	S 🟢 P 🟡 D 🔴
8	The damaged equipment is replaced. Dependability level increases. *SA_1* and *SA_2* report new state to *MA*.	Dependability level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡

Figure 29 illustrates the proof of concept AmbISPDM implementation including the test-bed used for demonstration and performance evaluation. The yellow circles illustrate the building agents. The red arrows show the alarm messages (i.e. cyber-attack or fire alarm) among the campus agents in the local network and the orange arrow shows the email or SMS to the fire brigade in case of a safety-related incident. The deployed embedded platforms consist of: Figure 29.A BeagleBone with SD memory card, USB-WiFi, weather and battery capes, Figure 29.B BeagleBoard with SD memory card, USB-WiFi, USB-Camera, and power supply, and Figure 29.C Smart phone device with Android OS and the smart building management application.

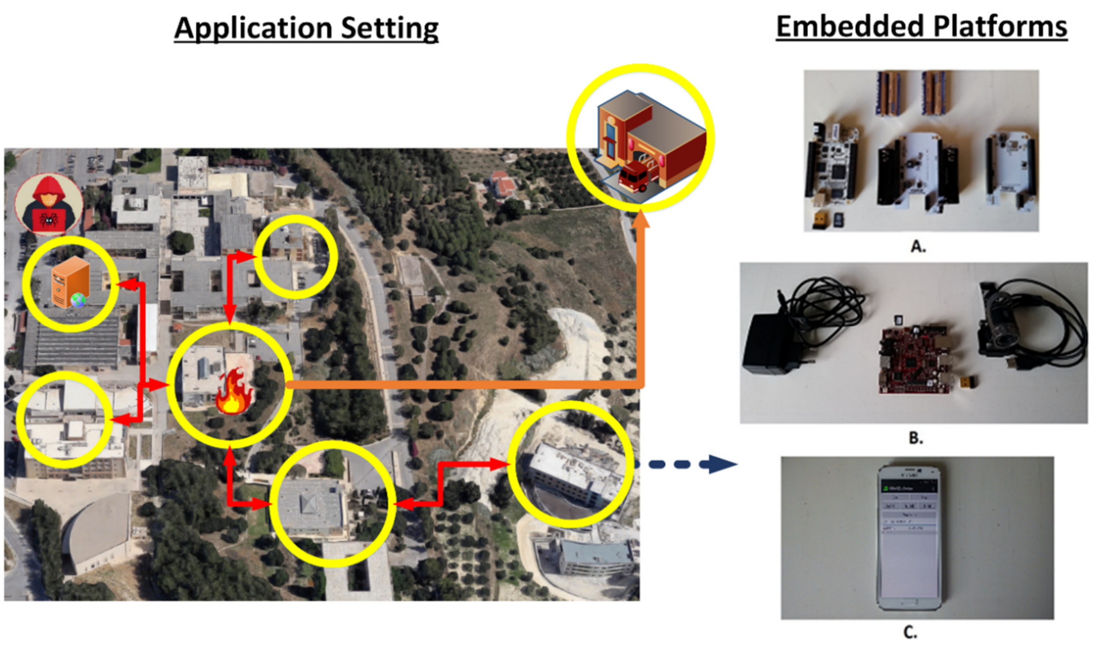


Figure 29 The application setting of AmbISPDM for the smart campus use case.

The system is evaluated in the campus of Technological Educational Institute of Crete. A cyber-attack and a fire alarm are emulated to trigger the two scenarios. Each agent runs on a laptop or PC and controls a set of embedded devices. BeagleBones equipped with the weather cape sense environmental parameters (e.g. temperature and humidity) or control electronic doors. BeagleBoards equipped with a USB-Camera perform a surveillance service. The operator (e.g. the incident controller) manages the system through a smart phone device.

10.4 Comparison with Existing Systems

This subsection compares the SPD-Safe with relevant approaches in terms of the effectiveness in disaster mitigation planning.

10.4.1 Disaster Mitigation Planning with Multi-Agent Systems

Research efforts use information technology to help mitigating and preventing natural disasters or accidents. MAS have been successfully used in disaster mitigation planning applications, exploiting their ambient intelligence capabilities.

DrillSim [136] is an augmented reality simulator that plays out the activities of a crisis response for testing related IT solutions. Agents model the involving human roles, like the first responders and crisis managers. Events occur in a hybrid world that is composed of the simulated world generated by the MAS and a real world captured by a smart space. The user interacts with the virtual world via portable devices, like cellphones and PDAs. Interfaces are implemented in Java and Java3D and the agents are deployed in the Java Agent Development framework (JADE). The reasoning process is modelled as a stochastic neural network (SNN). This fact results in scalability issues, making the reasoning engine the most computationally intensive module.

Demonstrating Effective Flexible Agent Coordination of Teams through Omnipresence (DEFACTO) [137] is a high fidelity system that incorporates 3D visualization omni-viewer and human-interaction reasoning. The 3D rendering is implemented in the JME game engine, with emergency scenarios being evaluated in the Robocup Rescue simulator[17]. DEFACTO can be used as a training tool for simulated disaster scenarios. The user interacts with the coordinating agent team in a complex environment. Valuable lessons are drawn that are applicable in real world cases. The system architecture is semi-distributed. Simple agents are connected to a proxy coordinator. The inter-proxy communication is based on a blackboard design. The coordinator chooses a variety of team-level interaction strategies, which are performed by the connected agents. Epistemic reasoning resolves conflicts based on negotiation algorithms. The coordinator can also communicate with other proxies if necessary.

The Applications, Decentralised System Architectures, Individual Agents, and Multiple Agents (ALADDIN) project [138] models, designs, and builds decentralized MAS. The project integrates information from various heterogeneous sources in order to take informed actions. It can function in uncertain and dynamic environments. ALADDIN considers different aspects such as data fusion, decision making, machine learning, and system architecture. Evaluating scenarios are executed in the Robocup Rescue simulator. The individual agents integrate

[17] ROBO science group: www.robocuprescue.org

information from other sources based on trustworthiness. Gaussian processes and Bayesian inference can detect erroneous sensed data during emergency.

The Distributed Online Multi-Agent Planning System (DOMAPS) [139] tackles the disaster planning for the floods domain in Brazil. Flood events are common in the region, especially during heavy-rain. They are often caused by intense hydro-meteorological hazards, leading to severe economic losses and even deaths. The Centre for Disaster Management assigns plans to a set of robots based on the ongoing emergency situation. The water paths are traversed by naval units and the ground paths are traversed by ground units. Unmanned surface vehicles (USV) and unmanned ground vehicles (UGV) move through water and ground paths respectively, collecting water samples, taking pictures of flood events and transporting first-aid kids to the first responders that are close to victims. The MAS is implemented in JaCaMo (Jason, CArtAgO, and Moise). The underlying Jason platform offers moderate scalability and security, with partial compatibility with the FIPA standards. The SHOP2 planner implements the individual reasoning behavior of each agent. Each agent controls a relevant robot. No embedded system platform is utilized. The formalism of decentralized domains and problems is based on a Hierarchical Task Network (MA-HTN). DOMAPS utilizes relational reasoning to avoid or resolve conflicts. Domain-dependent social laws (e.g. network hierarchy) are supplied by the designer off-line.

In contrast to the general AmI MAS that operate continuously during normal operation, the disaster mitigation ones start their operation only in emergency. The equipment remains inactive during the normal period. Most of these systems focus on supporting response operations, with small interest on improving effectiveness. Also, the response activities are stopped when the system components' are damaged. No system deals with adaptation to environment changes or tracking actors' actions.

10.4.2 Comparison of Disaster Mitigation Systems

Table 16 summarizes the evaluation results of SPD-Safe and the MAS for disaster mitigation planning (DOMAPS [139], DrillSim [136], DEFACTO [137], and ALADDIN [138]).

Table 16 Disaster mitigation systems' features

Aspects	**SPD-Safe**	**DOMAPS**	**DrillSim**	**DEFACTO**	**ALADDIN**
Response Effectiveness	YES	YES		NO	YES
Standards	YES	PARTIAL	NO	PARTIAL	NO
Control	YES	YES	NO	YES	YES
Communication	YES	NO	NO	NO	YES
Resources	YES	YES	NO	NO	YES
Information	YES	YES	YES	YES	YES
Actors Action	YES	YES	NO	NO	NO
Adaptation	YES	NO	NO	NO	NO
Planning	YES	YES	NO	NO	YES
Integration	YES	YES	YES	NO	NO
Learning	NO	NO	YES	YES	NO
Security	YES	PARTIAL	NO	NO	NO

Privacy	YES	NO	NO	NO	NO
Destroyed Equipment Substitution	YES	NO	NO	NO	NO

Regarding crisis response operations, DrillSim, DEFACTO and ALADDIN constitute the main MAS solutions. The evaluation study in [140] compares the 3 systems and indicates open research issues in the domain. As is evidence the current solutions have failed in managing either overloaded communication or dynamic resources over time. Also, most systems focus on supporting response activities, with small interest on improving the effectiveness of response operations. No system deals with adaptation of system components to environment changes or tracking actors' actions. The response operations are stopped or their effectiveness is degraded in case where the system components get damaged. In all cases, no standards were followed.

SPD-Safe covers the gaps of current solutions. The deployed platforms for the MAS and the embedded devices management are efficient, robust, and scalable. The reasoning process tackles dynamic resources over time by design. Both the individual entities and the system as a whole, adapt to environment changes. Moreover, the residents' actions are monitored in order to change the evacuation plan on the fly. As the system components are damaged, the dependability factor is decreased. When the dependability of a device or an agent reaches beyond a threshold, the component is considered unreliable and the rest components take control. The overall implementation follows the relevant standards for the deployment and management of embedded devices and their services, agent modelling and communication, and alerting formation.

DOMAPS is a state-of-the-art proposal that also tackles many of these issues. In contrast to DOMAPS, SPD-Safe considers the private information of each agent, which is also taken into account during multi-agent reasoning. DOMAPS does not examine privacy issues due to the inherited properties of the MA-HTN component. Moreover, SPD-Safe automatically resolves affairs while DOMAPS annotates the MAS developer when conflicts occur. The SPD-Safe agents are placed in the monitoring area and react to events instantly. In case of emergency, they inform the responsible authorities. The DOMAPS agents bid to plans that are published by a central Centre for Disaster Management.

11. SMART TRANSPORTATION USE CASE

Railways constitute a main mean of mass transportation. Public, private, and military settings traverse long distances everyday. The railway infrastructure is vulnerable to several types of criminal activity, ranging from vandalism to terrorism. Physical security of the infrastructure assets, like train, platform, public areas and surrounding spaces, tunnel, bridge, and the command & control center (C&C), is imperative. Modern surveillance systems can protect the railway region. The goal is to forecast physical threats, such as thefts, vandalism, sabotage, and terrorism.

Railway controlling software must collect spatial information and effectively manage these systems. Wireless sensor networks (WSNs) are the indicated solution to cover the area along-side the railway routes. In-carriage WSNs are also studied in cases of dangerous cargo transportation. The secure communication of all these devices becomes important as successful attacks can harm the railway's business operation or cause serious injuries and deaths.

Currently security issues include the cyber-attacks on the information system and the C&C. Usually, the buildings along the line are located in isolated areas, where monitoring or physical protection by the personnel is difficult. In particular some buildings, called shelters, are vulnerable as they maintain some electronic/electrical equipment for the railway management, like power supply, switches, and so on. This buildings are usually located far from the main station and the monitoring of physical threats and cyber-attacks is significant.

The buildings are ordinarily monitored by cameras and sensors, measuring environmental parameters (e.g. temperature and humidity), to avoid physical intrusions. The collection of environmental parameters can prevent a fire or a possible overheating of equipment. However, the communication among the devices is wireless and, thus, it is subjected to different types of attacks.

Current railway systems deploy a limited set of countermeasures that protect against the most common threats, lacking in cyber-attack detection. This kind of assets are vulnerable to several attacks at the network level, like blank hole, bad mounting, and jamming attacks.

The evolution of the Internet of Things (IoT) motivates the deployment of intelligent cyber-physical systems (CPS) that utilize wireless networking. Railway settings are such intelligent systems that include urban, industrial and military railways. Wireless sensor networks (WSNs) cover the large operational area, collecting and processing ambient information. A gateway is the link to send the information to a control center.

Except from security, safety is also important as high volumes of passengers and cargo are being transported everyday, traversing long distances. In the past years, several serious accidents caused many deaths and high economic losses for the involving organizations. Nowadays, the electronic control systems are decreasing the frequency of these situations. However, accidents can still occur due to equipment malfunctioning (e.g. signal loss) or uncovered areas across the long railway network (e.g. unprotected car crossing).

11.1 Smart Railway Issues

Intelligent and reliable management systems must be deployed for the protection of critical infrastructure. Cyber-security is also important in the era of IoT as successful attacks lead to potential damage or personal injury.

Communications-based train control (CBTC) is a railway signaling system that enables communication between the train and the track equipment for traffic management and infrastructure control. The international wireless communications standard for railway communication and applications in the European Union includes the Global System for Mobile Communications – Railway (GSM-R). GSM-R is further combined with the General Packet Radio Service (GPRS) and form the basis for an intelligent transport system.

Wireless sensor networks (WSNs) cover the large operational area, collecting ambient information. Embedded devices are used in order to provide enhanced intelligence services in such critical infrastructures and other smart city settings. The heterogeneity of the devices used and the variety of the application domains make the management of the deployed embedded systems a very challenging task.

The SPD-Safe can act as a CBTC for railway CPS. A proof-of-concept example is detailed, focusing on the railway scenario of managing in-carriage and on-route equipment. Two scenarios are modelled, where the system is configured at runtime to counter cyber-attacks and manage safety-related events respectively. The system is implemented and demonstrated under the EU-funded project nSHIELD [143] with the contribution of industry partners, like Ansaldo STS[18] and HAI[19].

11.2 Smart Railway Scenario

As a proof-of-concept example, the proposed framework is used to assess and manage a system which involves an ambient environment deployed with the purpose of protecting a railway's routes and the train's carriages. The hardware platforms consist of ARM-based devices (BeagleBones) that communicate wirelessly, monitor environment parameters, and control *"smart"* equipment, like electronic doors and cameras. Moreover, the PBAC mechanism [62] is deployed for managing access of the personnel based on their access rights, defined by XACML policies. The implementation exploits the DPWS functionality, already present on the framework's devices. Each agent manages a smart subsystem (e.g. train or WSN) and runs on a local PC. Figure 30 illustrates the railway system architecture.

[18] Ansaldo STS: http://www.ansaldo-sts.com
[19] HAI: http://www.haicorp.com/en/

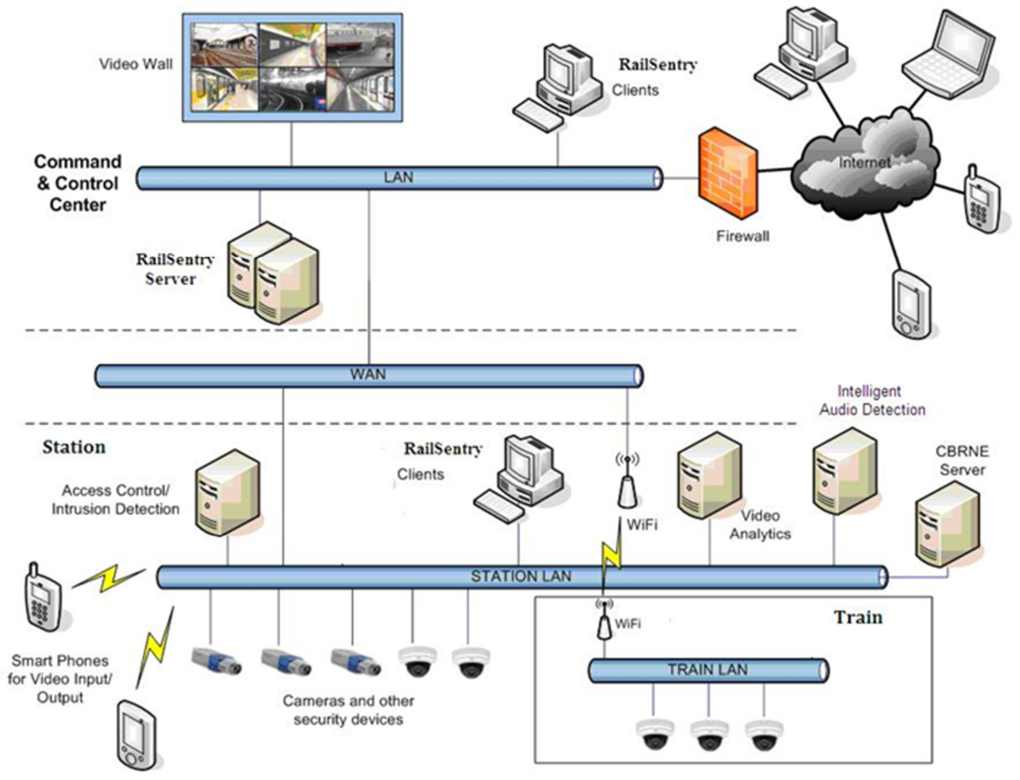


Figure 30 The smart railway use case architecture

Two WSN with MemSic Iris devices are deployed. Both networks run a trust-based secure routing scheme [12], [13] to detect and counter ad hoc routing attacks. A cryptographic service implemented in ULCL [9], [10], [63] safeguards the communication and provides authentication.

For in-carriage communication, a WSN of nine Iris devices is created and connected wirelessly via USB Wi-Fi modules. Each device monitors temperature and light. The devices use batteries for power supply. Moreover, the device at the carriage's entrance is equipped with a smart camera. A second WSN is also deployed for redundancy, consisting of Zolertia Z1 motes that collect temperature data. All devices run the PBAC's PEP component. Through two relevant base stations, the devices transmit information to a gateway, which maintains the PBAC access policies and runs the agent that controls the train network. Figure 31 illustrates the in-carriage WSN [143].

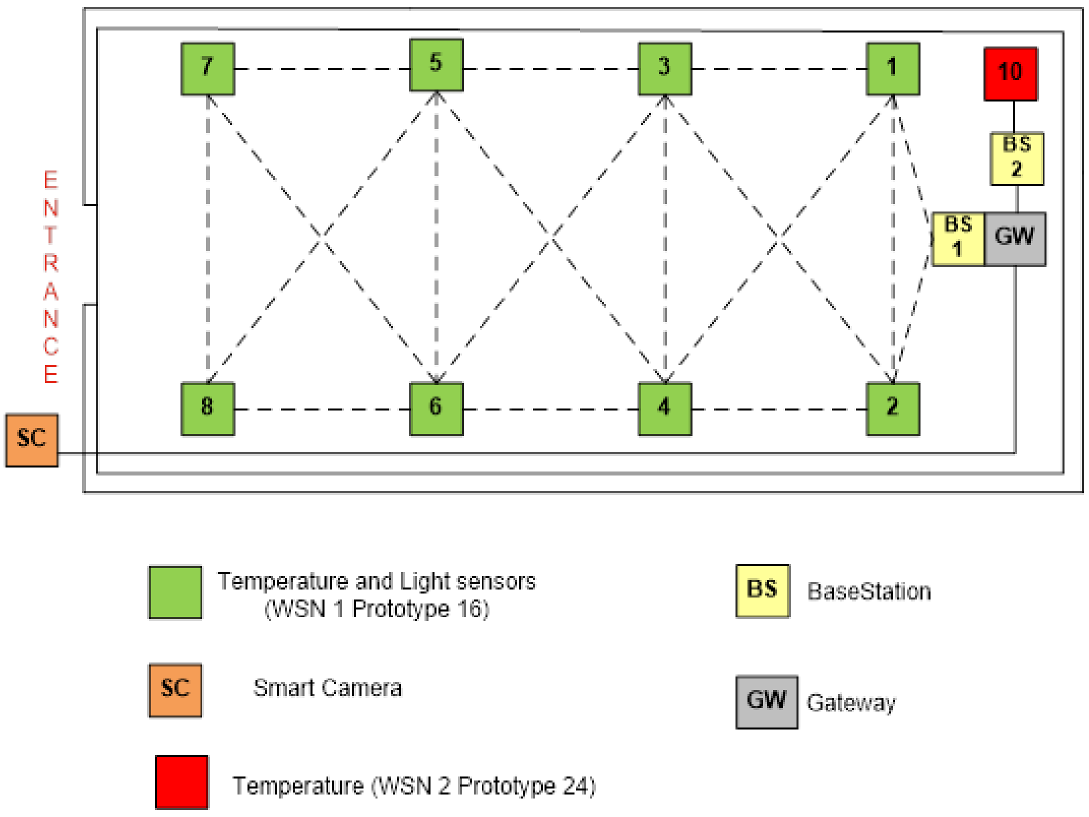

ENTRANCE
7
5
3
1
10
BS 2
BS 1
GW
8
6
4
2
SC
Temperature and Light sensors (WSN 1 Prototype 16)
SC Smart Camera
Temperature (WSN 2 Prototype 24)
BS BaseStation
GW Gateway

Figure 31 The carriage WSN setting

For outdoor on-route protection, a similar WSN setting of four nodes is deployed. All devices are connected to power supply and are equipped with weather sensors and a smart camera. In the emulated scenario, the devices are placed on the carriage departure, the line, the passenger's station and tunnels or bridges along the route. The devices transmit real-time data to a security control center with the relevant monitoring agent. Figure 32 illustrates the on-route WSN [143].

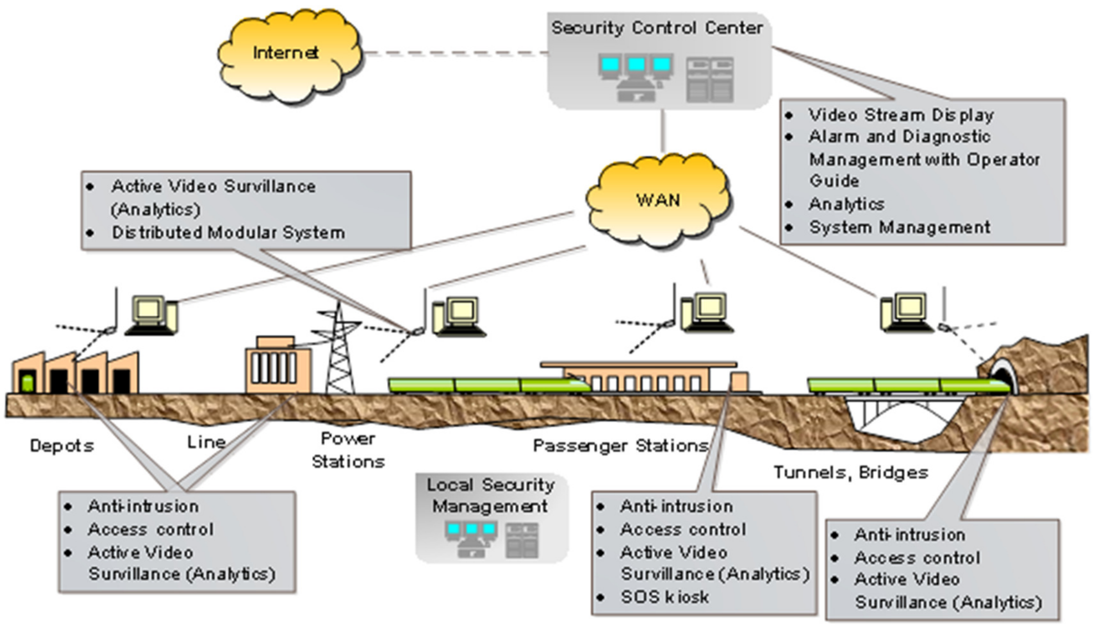
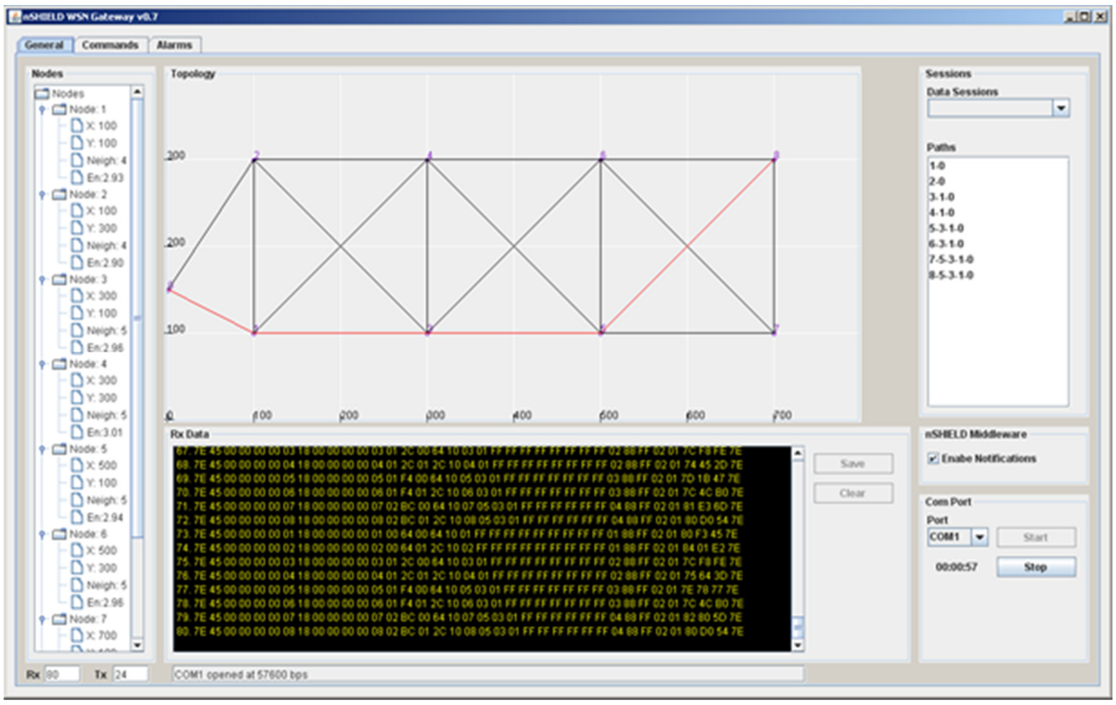

Internet
Security Control Center
Video Stream Display
Alarm and Diagnostic Management with Operator Guide
Analytics
System Management
WAN
Active Video Survillance (Analytics)
Distributed Modular System
Depots
Line
Power Stations
Passenger Stations
Tunnels, Bridges
Local Security Management
Anti-intrusion
Access control
Active Video Survillance (Analytics)
Anti-intrusion
Access control
Active Video Survillance (Analytics)
SOS kiosk
Anti-intrusion
Access control
Active Video Survillance (Analytics)

Figure 32 The route WSN setting

In both cases, the devices collect environmental parameters and transmit them to a processing center or base station (laptop with WiFi capability). The processing center deploys an application where the user can gain access and manage the system.

The agents evaluate the current SPD level of their underlying components and the system as a whole. They also monitor their respective domains, managing them based on the SPD and safety artificial intelligence reasoning processes. The system can also be configured at runtime to adjust to performance and security goals that are described in the active policy. The agents can change the configurations of a system in order to increase the SPD level when an attack is detected and then return to the previous state when the attack is over (to conserve resources). The cryptographic service for PBAC and routing supports three communication settings: plaintext, authenticated, and authenticated encryption. The trust scheme additionally offers two trust evaluation settings: direct trust only, and direct and indirect trust.

To retain resources at normal operation, the system uses authenticated communication and direct trust. Then, the cyber-attacks, like blackhole or link-spoofing, are performed. When the system detects an attack, it informs the network nodes to increase their security level. A security policy orders a specific set of actions (applicable to this type of devices), like using authenticated communication with both direct and indirect knowledge. The WSNs adopt the new policy, become stricter with misbehaving nodes and isolate the malicious ones. Similarly, when the emergency situation is over, the system returns back to the normal (initial) configuration. Figure 33 illustrates the GUI of the in-carriage WSN, developed by HAI and TUC.



Nodes
- Nodes
  - Node: 1
    - X: 100
    - Y: 100
    - Neigh: 4
    - En:2.93
  - Node: 2
    - X: 100
    - Y: 300
    - Neigh: 4
    - En:2.90
  - Node: 3
    - X: 300
    - Y: 100
    - Neigh: 5
    - En:2.96
  - Node: 4
    - X: 300
    - Y: 300
    - Neigh: 5
    - En:3.01
  - Node: 5
    - X: 500
    - Y: 100
    - Neigh: 5
    - En:2.94
  - Node: 6
    - X: 500
    - Y: 300
    - Neigh: 5
    - En:2.96
  - Node: 7
    - X: 700

Rx: 80 Tx: 24

Rx Data

```
67. 7E 45 00 00 00 00 03 18 00 00 00 00 03 01 2C 00 64 10 03 01 FF FF FF FF FF FF FF FF 02 88 FF 02 01 7C F8 FE 7E
68. 7E 45 00 00 00 00 04 18 00 00 00 00 04 01 2C 01 2C 10 04 01 FF FF FF FF FF FF FF FF 02 88 FF 02 01 74 45 2D 7E
69. 7E 45 00 00 00 00 05 18 00 00 00 00 05 01 F4 00 64 10 05 03 01 FF FF FF FF FF FF FF 03 88 FF 02 01 7D 18 47 7E
70. 7E 45 00 00 00 00 06 18 00 00 00 00 06 01 F4 01 2C 10 06 03 01 FF FF FF FF FF FF FF 03 88 FF 02 01 7C 4C B0 7E
71. 7E 45 00 00 00 00 07 18 00 00 00 00 07 02 BC 00 64 10 07 05 03 01 FF FF FF FF FF FF 04 88 FF 02 01 81 E3 6D 7E
72. 7E 45 00 00 00 00 08 18 00 00 00 00 08 02 BC 01 2C 10 08 05 03 01 FF FF FF FF FF FF 04 88 FF 02 01 80 D0 54 7E
73. 7E 45 00 00 00 00 01 18 00 00 00 00 01 00 64 00 64 10 01 FF FF FF FF FF FF FF FF FF 01 88 FF 02 01 80 F3 45 7E
74. 7E 45 00 00 00 00 02 18 00 00 00 00 02 00 64 01 2C 10 02 FF FF FF FF FF FF FF FF FF 01 88 FF 02 01 84 01 E2 7E
75. 7E 45 00 00 00 00 03 18 00 00 00 00 03 01 2C 00 64 10 03 01 FF FF FF FF FF FF FF FF 02 88 FF 02 01 7C F8 FE 7E
76. 7E 45 00 00 00 00 04 18 00 00 00 00 04 01 2C 01 2C 10 04 01 FF FF FF FF FF FF FF FF 02 88 FF 02 01 75 64 3D 7E
77. 7E 45 00 00 00 00 05 18 00 00 00 00 05 01 F4 00 64 10 05 03 01 FF FF FF FF FF FF FF 03 88 FF 02 01 7E 78 77 7E
78. 7E 45 00 00 00 00 06 18 00 00 00 00 06 01 F4 01 2C 10 06 03 01 FF FF FF FF FF FF FF 03 88 FF 02 01 7C 4C B0 7E
79. 7E 45 00 00 00 00 07 18 00 00 00 00 07 02 BC 00 64 10 07 05 03 01 FF FF FF FF FF FF 04 88 FF 02 01 82 80 5D 7E
80. 7E 45 00 00 00 00 08 18 00 00 00 00 08 02 BC 01 2C 10 08 05 03 01 FF FF FF FF FF FF 04 88 FF 02 01 80 D0 54 7E
```

COM1 opened at 57600 bps

Sessions

Data Sessions

Paths
1-0
2-0
3-1-0
4-1-0
5-3-1-0
6-3-1-0
7-5-3-1-0
8-5-3-1-0

nSHIELD Middleware

[✓] Enabe Notifications

Com Port

Port: COM1

00:00:57

Figure 33 The in-carriage WSN GUI

In a safety-related incident, the agents can control the electronic equipment to inform the personnel or help the passengers in the case of an emergency. The demonstration scenario involves modelling the response plan in case of fire alarm where decisions about both security and safety are made. Normally, the personnel and passengers can only open the doors that they are allowed to by their access rights (based on security and safety rules). When the sensor nodes detect fire, they raise an alarm to the monitoring agent. The agent decides to lower the security level, unlocking all doors, thus permitting unhindered exit from the train. The agent also sends automatically an SMS via GSM to the responsibly authorities regarding the event (e.g. location through GPS, situation's severity) and informs its peers (i.e. agents of nearby trains) to be aware. When the fire is extinguished and the damaged equipment is restored, the system returns to the normal state. Figure 34 depicts the demonstrator software of the on-route equipment, developed by Ansaldo STS.

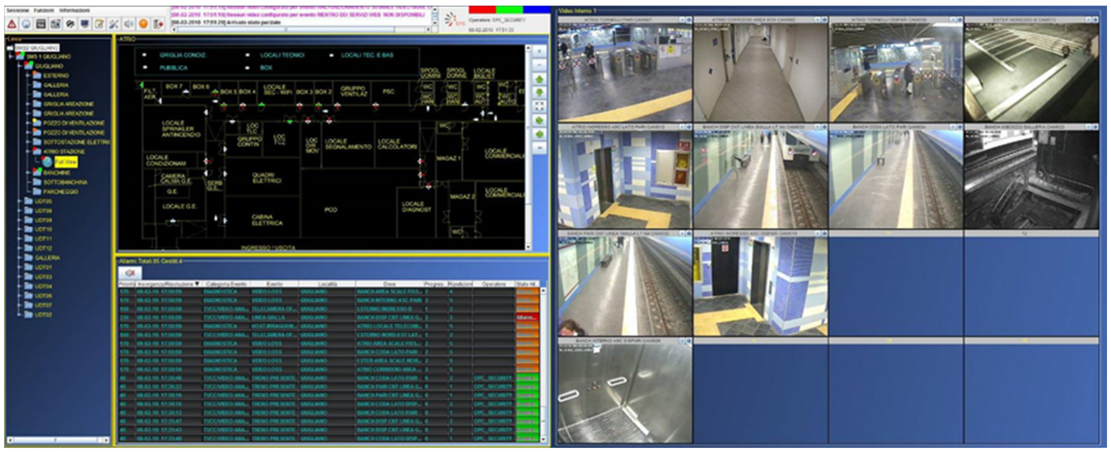


Figure 34 The railway on-route WSN GUI

11.3 DEMONSTRATION

The demonstration scenario consists of a shelter monitoring application featuring the following devices:

- Smart surveillance cameras
- WSNs

These assets are vulnerable to network layer attacks (e.g. black hole and jamming attacks). The network topology is depicted in Figure 31 in a shelter where the following devices are installed:

- A camera is installed at the entrance to detect a physical intrusion detection,
- Two WSNs are installed inside the shelter. WSN_1 (green color) measures temperature and light, and WSN_2 (red color) measures temperature. WSN_1 and WSN_2 are different in hardware in order to guarantee redundancy and diversity in some measurement parameters.
- The gateway links the aforementioned equipment with the C&C.

The system starts from a moderate SPD configuration to save resources. The state and SPD of each prototype is then changed in response to attacks, in order to accomplish an adequate protection level during the system's lifecycle. The main protection mechanism against cyber-attacks is provided by the trust-based secure routing protocol, while physical protection is enhanced through the smart camera. Figure 35 illustrates the WSN_1, the gateway, and the feed from the smart camera.

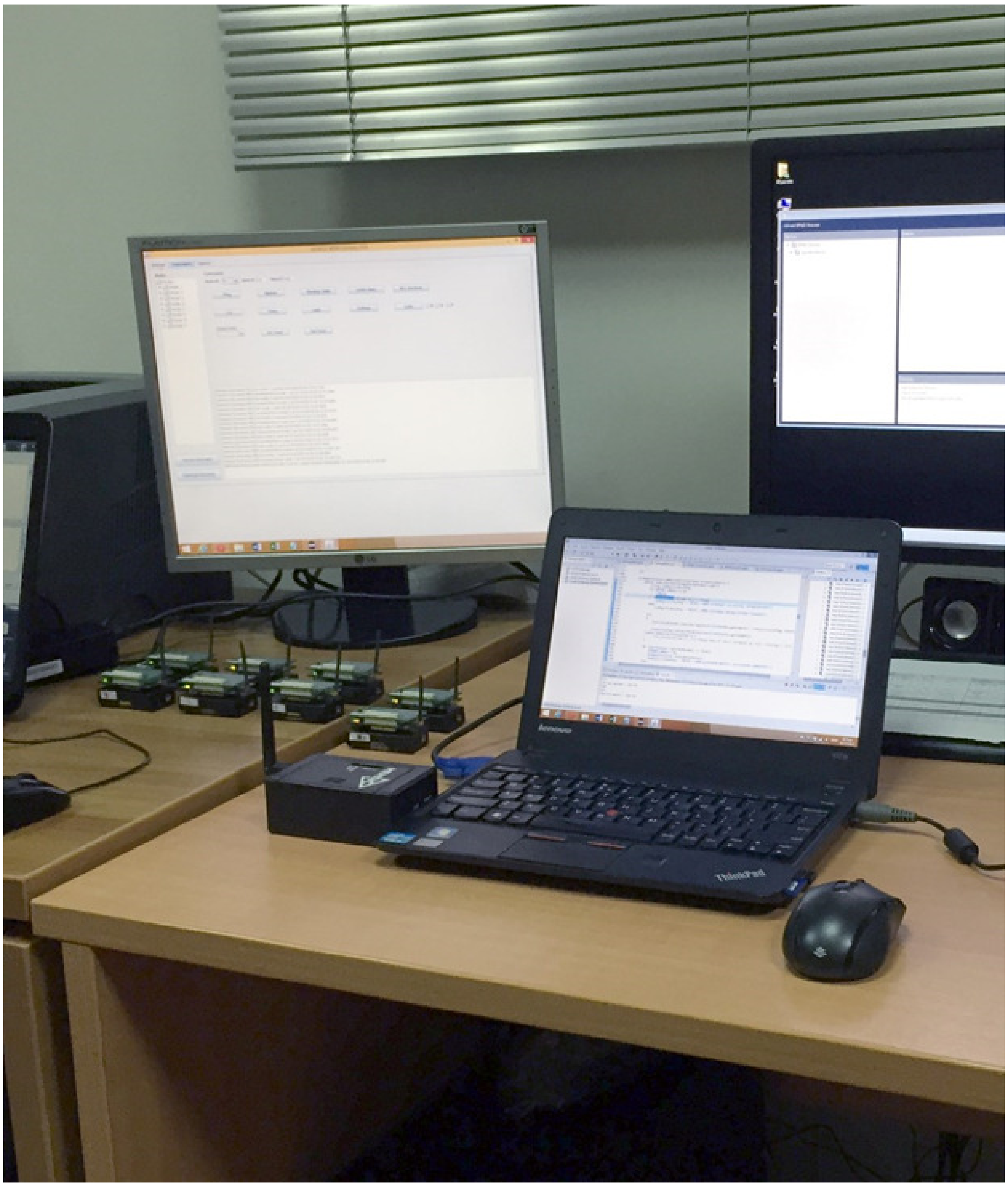


Figure 35 The railway shelter demonstration setting

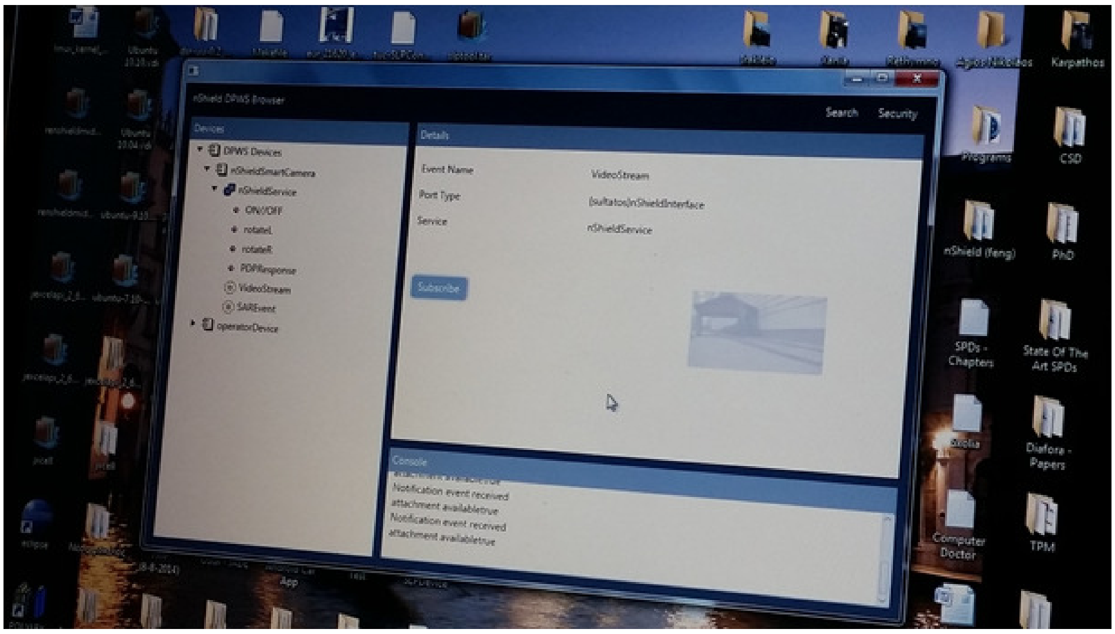
For WSN$_1$, we emulate the cases where a node is malfunctioning due to low battery and a malicious node that performs a bad-mouth attack. The first node is protected when the low energy level is detected and avoided from routing. The administrator is informed respectively. When the issue is resolved the node trust level is restored. The malicious node is detected once the attack rate passes a threshold and the node is excluded from routing. For WSN$_2$, we emulate a blackhole and a jamming attack against congested or topology significant nodes. The secure routing scheme successfully detects both attacks and counters the attackers. The scenario steps are summarized in Table 17 (red illustrates SPD values of *0-50*, yellow illustrates SPD values of *51-70*, and green illustrates SPD values of *71-100*).

Table 17 Scenario steps of smart transportation use case

STEP	Events	Effect	Overall (S,P,D) Value	State Visualization
1	Power-on of all systems and discovery/registration	Initial State	**80,70,65**	S 🟢 P 🟡 D 🟡
2	Bad-mouthing attack to WSN_1. *MA* is informed that an attack occurs and it sends a command to the rest agents to increase security.	Security level decreases	**60, 70, 65**	S 🟡 P 🟡 D 🟡
3	Security level is increased on all SAs. *MA* is informed.	Security level increases	**85,70,65**	S 🟢 P 🟡 D 🟡
4	WSN_1 has counteracted the bad-mouthing attack and SA_1 reports the change to the *MA*. The *MA* asks the *SAs* to return to normal state (to conserve resources).	Security level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡
5	Blackhole attack to WSN_2. *MA* is informed that an attack occurs and it sends a command to the rest agents to increase security.	Security level decreases	**50, 70, 65**	S 🔴 P 🟡 D 🟡
6	Security level is increased on all SAs. *MA* is informed.	Security level increases	**85,70,65**	S 🟢 P 🟡 D 🟡
7	WSN_2 has counteracted the blackhole attack and SA_2 reports the change to the *MA*. The *MA* asks the *SAs* to return to normal state (to conserve resources).	Security level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡
8	A node is died in WSN_1. *MA* is informed.	Dependability level decreases	**80, 70, 30**	S 🟢 P 🟡 D 🔴
9	The dead node is replaced. Dependability level increases. SA_1 reports new state to *MA*.	Dependability level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡

10	Simulated jamming attack against the network layer of *WSN*$_2$. *MA* is informed.	S & D levels decrease	**40, 70, 40**	S 🔴 P 🟡 D 🔴
11	The trust-based routing component counters the attack. *SA*$_2$ reports new state to *MA*.	S & D levels return to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡

A safety-related incident is also emulated on the on-route WSN, where fire is detected via the surveillance cameras. A fire-alarm is signed, similar with the smart campus case. Here, the train agents that traverse the area are informed to take relevant actions (i.e. change route or stop to the nearest station. Also, each smart camera transmits captured frames at low rate during normal operation. As the alarm is signed, the configuration is changed at runtime, enabling continuous monitoring of the examined area. When the fire is over, the system returns to the normal state to reserve resources. Figure 36 illustrate the smart camera's feed and the accessing controls (ON/OFF, Rotate, etc.)



Event Name VideoStream
Port Type {sultatos}nShieldInterface
Service nShieldService

```
Notification event received
attachment availabletrue
Notification event received
attachment availabletrue
```

Figure 36 Smart camera's feed and controlling GUI

12. Secure Real-Time Vehicle Management with Critical Incident Response

The intelligence being built into various vehicle types, will not only improve their safety and comfort but also enable new modes of transportation and new types of services, creating the corresponding markets. In all cases, it will be important to be able to monitor, preferably in real-time, various parameters of the smart vehicles' condition. By monitoring various operational parameters, which are assessed using SPD metrics, SPD-Safe enables real-time monitoring and interaction with a smart vehicle or a smart vehicle fleet. A proof-of-concept implementation is developed and deployed on real vehicles, demonstrating the successful integration of all the technological components into an actual working framework and its use on a real-world use case. Moreover, its performance overhead is assessed, validating the feasibility of the proposed approach.

12.1 Smart Cars

A typical car may currently utilize over 80 built-in microprocessors, providing advanced safety systems, emission monitoring and in-car commodities [144], [145] which aim to enhance the comfort and safety of their passengers, also protecting the vehicle's sub-systems by providing early warning of failures and/or adjusting their operation accordingly. Typically, electronic control units (ECU) manage and interconnect the distinct systems [146], [147] and the infotainment infrastructure provides various enhanced facilities, like entertainment and navigation, to passengers [148].

The new generation of smart vehicles will be a mobile intelligent system within a larger smart city infrastructure, supporting communication with other vehicles, the city infrastructure and backend systems. Prototype deployments of cars and road infrastructure exchanging information regarding lane state and traffic are already under construction by large manufacturers in the European Union (EU) and the United States (USA). Public entities and private transportation organizations and businesses will also take advantage of the individual computational and communication capabilities of smart vehicles to achieve effective fleet management through real-time monitoring of the vehicle's state and the driver's driving behavior. These features will allow organizations to minimize the risks associated with vehicle investment and promote strategies for increasing productivity and safety while reducing transportation and staff costs.

Government regulations are decisive to the direction of pertinent research efforts. The United Kingdom tries to minimize road deaths in business-owned vehicles; starting in 2008, road death is treated as an unlawful killing, enabling seizing of the company's records and bringing prosecutions against directors who fail to enforce safe driving policies for the drivers they employ. Therefore, fleet management is now imperative for the functional operation of an organization which owns a significant amount of vehicles.

The European Commission also defines new regulations for vehicle safety. One such initiative is the European Union-based eCall system, which is expected to become mandatory for every vehicle moving in the European Union by 2018. This emergency service, described in regulation EN 15722:2011 [149], dictates that, when an accident occurs, the vehicle should be able to automatically relay essential information (the vehicle's location, its direction and speed before the crash, number of passengers etc.) to appropriate Public Safety Answering Points (PSAP), informing that an accident has occurred. By providing early notification and allowing efficient coordination of the emergency services, this will enable faster response to such incidents (expected to decrease 50% in the countryside and 60% in built-up areas), drastically reducing the number of deaths and the severity of injuries for the thousands of people involved in road accidents every year. Moreover, the EU and the USA are collaborating to define a common subset of rules and standards related to smart vehicles and smart cities infrastructure.

However, security-related incidents have already been reported, where vulnerabilities in the infotainment infrastructure are exploited by attackers to remotely operate vehicle components [5]. An attacker can control the breaking system, turn off the lights or even lock the car while on the move. As more and more vehicles provide seamless connection to Internet, security becomes an important aspect that must be considered at the design phase of any relevant proposal. To this end, systematic methodologies for developing secure and efficient vehicular embedded systems are needed.

Researchers of the Ford motor company have presented a methodology for modeling automotive systems in terms of security, privacy, usability and reliability (SPUR) [150]. Every evaluated counterpart is analyzed based on the offered SPUR functionality and a qualitative value is assigned to each parameter (low, medium or high). The method is applied on real system attributes, such as the valet key and the anti-lock braking system.

The EU-funded project nSHIELD [151] proposed two quantitative methodologies for building secure embedded systems in terms of security, privacy, and dependability (SPD), namely the attack surface metric and the multi-metric methodologies. The multi-metric methodology, which was demonstrated in social mobility applications, is a key SPD-Safe component, as it is specifically aligned with its design goals and intended use cases, and was detailed in the previous sections.

ScudWare is a semantic and adaptive middleware platform for smart vehicle spaces, presented in [152], with similar design goals as SPD-Safe. It constitutes a multi-agent system that synchronizes context-aware and adaptive components. The agent's reasoning process is based on first-order predicate logic and implemented in DL with OWL ontology rules. Nevertheless, the authors do not provide clear implementation details nor a performance evaluation, focusing on the theoretical background of ScudWare. SPD-Safe provides a more robust reasoning process, and is based on modern standards, adopting a SOA approach. Moreover, it offers higher degree of context representation and reasoning which are further combined with quantifiable metrics for security, privacy and dependability.

In all cases, mechanisms should be included to allow the vehicle owner (e.g. a logistics company or a father lending his car to his son) to specify driving rules for the drivers, setting, for example, the maximum travel speed and/or the operating area (through geo-fencing). Furthermore, sensors could be used to monitor the vehicles health, informing the owner about engine malfunctions or other incidents in real-time.

Finally, all mechanisms should ideally be developed with backward compatibility in mind, maintaining the ability to retrofit the necessary modules into existing vehicles. It is estimated that there are over 500 million vehicles already roaming US and EU roads alone; a huge market for anyone involved in retrofitting such modules to make existing vehicles *“smarter”* and let their drivers enjoy some of these new enhanced services.

SPD-Safe aims to act as an enabler for addressing the above issues. The most important concern is the enhancement of passenger safety and the framework can help achieve this in various ways: it can assist in keeping vehicle fleets in good condition (by constant monitoring of vehicle health), it can monitor the drivers’ compliance to good-driving practices, and it can enable faster response in cases of emergency. The adoption of SPD-Safe would, thus, allow a smart city or any other public or private organization with vehicle fleets to reduce the risks to their personnel and their vehicle investment, advancing productivity while reducing transportation and staff costs. Moreover, it can help achieve compliance with upcoming regulations of EU and other government organizations both for vehicle safety and green infrastructure management.

12.2 Smart Cars Scenario

Smart vehicles are expected to constitute an important segment of the upcoming IoT – enabled world, where computing devices will not only permeate our lives but will also be easy to design and create at home [153]. Modern vehicles already feature a number of embedded electronics that monitor and control the various automotive sub-systems, aiming to enhance passenger comfort and safety, achieve energy-efficient operation and maximize the vehicle’s lifetime. Many types of accidents can be reduced through superior safety features, like early breaking and road lane departure warnings; this is also the focus of various governmental initiatives worldwide, which define stricter regulations for vehicle safety. Automotive legislation initiatives also necessitate the production of more eco-friendly vehicles, a target partly achieved by sub-systems which monitor the vehicles’ operation in real-time and trigger adjustments to engine parameters. Based on the above stimuli, the electronics integrated into vehicles increase with every vehicle generation and are expected to rise steeply with the introduction of smart and, eventually, self-driving vehicles. This *“intelligence”* built-into vehicles will also enable the introduction of a variety enhanced services that everyone will enjoy, from end-users (e.g. parents lending their vehicle to their child), to private entities (e.g. logistics or car-rental) and public entities (e.g. governments, smart cities, emergency services) operating vehicle fleets.

To facilitate the deployment and operation of these services in a secure and interoperable manner, SPD-Safe is utilized for the implementation of a smart vehicle management framework based on the integration of novel primitives with standardized communication technologies and mechanisms. The proposed approach can provide real-time monitoring of a vehicle or a fleet of vehicles, collecting various operating parameters (e.g. regarding engine health or passenger safety), quantified using a set of security, privacy and dependability –related metrics, and also allowing interaction with the vehicles (to adjust some parameters) in real-time. The owner and other stakeholders can also set policies of fair use (maximum speed, allowed detours etc.), tracking the driver’s behavior based on these features. Thus, it allows individual users, companies relying on vehicle fleets for day-to-day operations and even public entities (e.g. a smart city) to minimize the risks associated with passenger safety, protect vehicle investments, enhance productivity, and reduce transportation and staff costs.

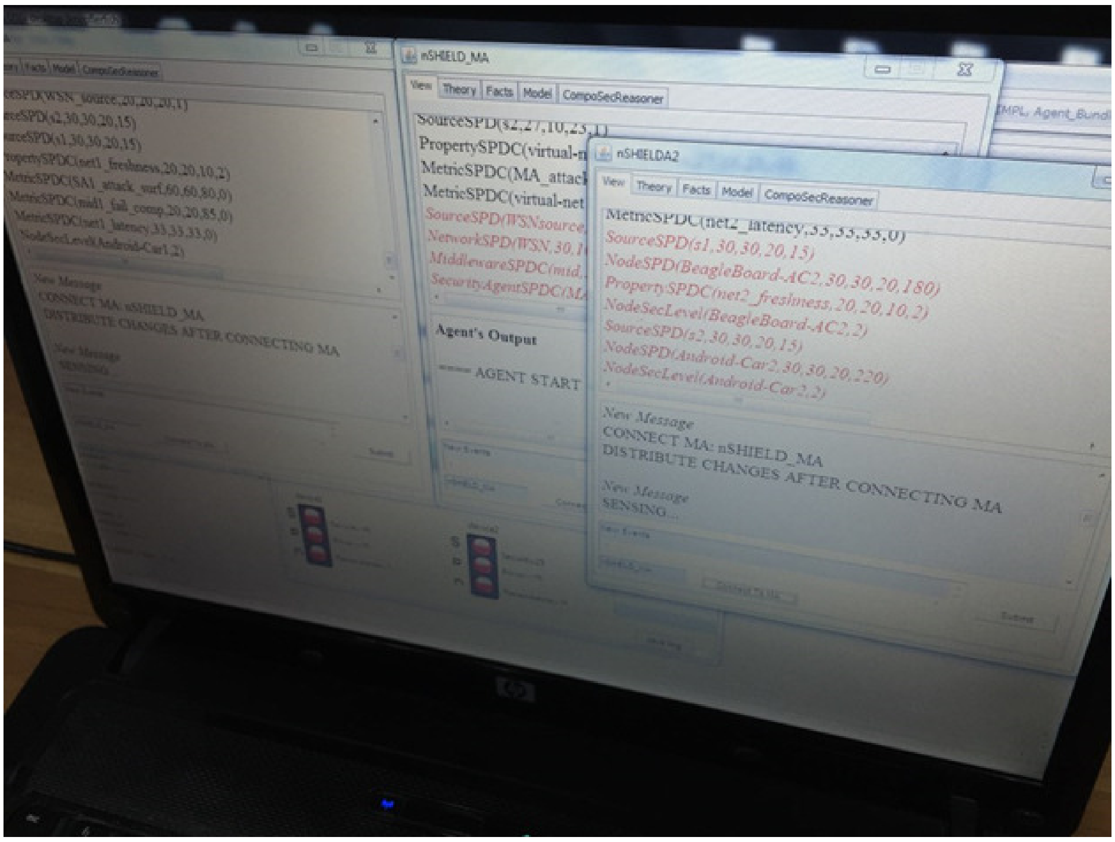
SPD-Safe consists of various core components, comprising a low-cost multi-agent system for smart vehicles, also allowing the integration of smart infrastructures (e.g. smart road and/or smart city sub-systems) into the monitoring and decision-making process.

The main technological building blocks of SPD-Safe were detailed in the previous sections, while for underlying communications, a Service Oriented Architecture (SOA) is adopted, in order to provide seamless interaction between the framework's entities. Thus, all of the framework's entities specify semantic information regarding their type and provided services using the DPWS OASIS standard, supporting device and service description, discovery, messaging and eventing. This allows the deployment of devices aligned with the Web Services technologies, thus facilitating interoperability among services provided by resource-constrained device and providing seamless Machine-to-Machine (M2M) discovery and interactions.

The ambient context is modelled in Jess-EC and develop reasoning services based on a formal theory that reasons about the composability and integration of the underling devices and technologies and verifies the current SPD level of the system, and a management theory of the ambient environment, also responsible for administrating the system in real time. The OASIS standard CAP is used to model the semantic information that is exchanged between the in-vehicle equipment with its agent and the agent with other agents outside the vehicle. The external communication is achieved via GSM.

At the smart vehicle end, an SPD-Safe agent monitors the system and takes simple decisions regarding SPD and safety. The agent lies in the infotainment infrastructure communicating internally with the ECU and externally with other agents. The agent acts as an intermediate layer between the ECU and external systems, providing an abstract level of communication and protecting the internal system from attacks.

The Command & Control (C&C) center is responsible for smart vehicle management. A master SPD-Safe agent is deployed at the backend, collecting information from the smart vehicle and infrastructure agents. The master agent has global knowledge of the whole system and carries out the fleet management strategy. It can trivially be extended to use databases to store vehicle and driver history records, maintaining extensive log files and execution reports. Moreover, it can be used to display the positioning information of the smart vehicles into a geographic information system (GIS) in order to provide location-based services (e.g. geo-fencing). To implement this functionality at the backend, DPWS is integrated into the OSGi middleware, as described in previous sections. The JADE-agents are also integrated into the same platform and manage the vehicles' devices in real-time through OSGi interfaces. ECU and supplementary devices model the provided functionality in DPWS and exchange information with the vehicle's agent through OSGi. Figure 37 depicts the aforementioned car agents and the master agent of the C&C.



```
MetricSPDC(net2_latency,33,33,33,0)
SourceSPD(s1,30,30,20,15)
NodeSPD(BeagleBoard-AC2,30,30,20,180)
PropertySPDC(net2_freshness,20,20,10,2)
NodeSecLevel(BeagleBoard-AC2,2)
SourceSPD(s2,30,30,20,15)
NodeSPD(Android-Car2,30,30,20,220)
NodeSecLevel(Android-Car2,2)
```

```
New Message
CONNECT MA: nSHIELD_MA
DISTRIBUTE CHANGES AFTER CONNECTING MA

New Message
SENSING...
```

Figure 37 The car agents and the master agent of the backend GUIs

With regard to the dynamic SPD levels, the smart vehicle can transmit subsets of the monitored information. Ideal security, privacy, and dependability settings may not always be possible, as operation may be hampered by congestion, missing mobile network coverage or other variables. So, for example, lack of mobile connectivity can be represented in the SPD levels reported on the Backend System, as lower a lower Security level if security mechanisms are omitted because of network issues or to preserve power.

Accordingly, in another case, the service user may decide not to report his exact location, but a larger area (i.e. one including more service users), in order to obstruct the tracking of his exact whereabouts and, thus, protect his privacy. This will have a direct effect on the Privacy level visible at the backend monitoring system.

Variations in the Dependability level will be more common. The vehicle's health is monitored in real time (e.g. tire pressure, mileage, warning messages from the ECU etc.), and the dependability value should be adjusted accordingly. For example, when the car exceeds the planned travel mileage between services, the Dependability value reported will be lower.

12.3 Demonstration

As a proof of concept, SPD-Safe is retrofitted onto an existing vehicle, using off-the-shelf components. This setup relies on the infotainment system - or the user's smart phone, if a smart infotainment system is not available, as was the case with the vehicle we used – for communication with the backend system. An Android-based application is deployed on the smart device, which performs the SPD and safety related reasoning process, also communicating and receiving commands from the command & control. The application collects information regarding the car's sensors (e.g. fuel consumption, engine status) from the ECU via a Bluetooth-enabled OBD scan tool. These OBD tools are widely available nowadays for a very reasonable cost; the one used in the setup cost ~*10* Euros. The application collects additional data from sensors already integrated into infotainment or smart phone, like acceleration and GPS position. The insert in Figure 38 depicts the above car setup.

A smart city's infrastructure can also be integrated into SPD-Safe. As proof of this, a WSN is included in the test setup. The sensors communicate via a trust-based secure routing protocol (Appendix C), allowing them to detect several types of sensor malfunctions and malicious attacks. The backend collects and processes this information from the WSN's gateway and can, in turn, inform other agents.

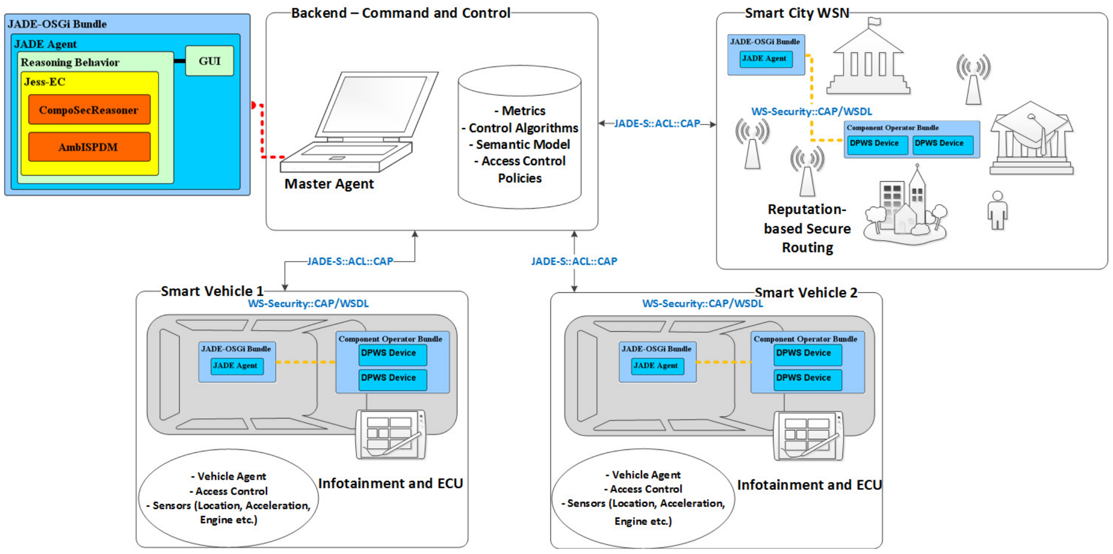


Figure 38 Social mobility use case architecture

The use case scenario is designed to demonstrate the SPD variations of the involved systems in the cases of: attacks on the smart city infrastructure, engine malfunction on one of the vehicles and, finally, a car crash. The scenario steps appear in more detail in Table 18.

Table 18 Scenario steps of Social Mobility use case

STEP	Events	Effect	Overall (S,P,D) Value	State Visualization

1	Power-on of all systems and discovery/registration	Initial State	**80,70,65**	S 🟢 P 🟡 D 🟡
2	A Black Hole attack is detected at the WSN. *MA* is informed through WSN that an attack occurs and it sends a command to the vehicles to increase security.	Security level decreases	**60, 70, 65**	S 🟡 P 🟡 D 🟡
3	Security level is increased on both *Car₁* & *Car₂*. *MA* is informed.	Security level increases	**85,70,65**	S 🟢 P 🟡 D 🟡
4	The WSN has counteracted the Black Hole attack, reporting the change to the *MA*. The *MA* asks *Car₁* & *Car₂* to return to normal state (to conserve resources)	Security level returns to initial state	**80, 70, 65**	S 🟢 P 🟡 D 🟡
5	ECU of *Car₁* informs of increased temperature. MA is informed.	Dependability decreases	**80, 70, 50**	S 🟢 P 🟡 D 🔴
6	*Car₁* crashes. Transmits eCall SMS. *SA₁* Dependability decreases. *SA₁* Privacy decreases (operator at C&C now has access to exact vehicle location). *SA₁* Security decreases (encryption is disabled to facilitate emergency response and emergency services). The *MA* is informed of the changes and sends email to C&C operators/administrators and other stakeholders.	S & P & D levels decrease	**50, 50, 40**	S 🔴 P 🔴 D 🔴
7	*Car₁* is repaired. *SA₁* reports new state to *MA*.	S & P & D levels increase	**80,70,65**	S 🟢 P 🟡 D 🟡

As a full scale deployment on an actual smart city was not feasible, an experimental test-bed was used to evaluate SPD-Safe's performance under this scenario. The Linux-based device was deployed on a BeagleBone embedded device (720MHz ARM Cortex-A8 processor, 256MB RAM, Linux OS), emulating the ECU. The Android-based version of the agent was deployed on a tablet (1.9GHz quad-core processor, 2GB RAM, 16GB, Android OS), emulating the vehicle's infotainment. The backend system run on a laptop PC, while a separate laptop was used as a gateway for the *"smart city infrastructure"* WSN which was based on IRIS motes (Atmel ATmega1281 16MHz CPU, 8kB RAM, 128Kb program flash memory) running the trust-based secure routing algorithm. Finally, a desktop PC was used as a client to the services provided by the vehicle (e.g. to access its location) for benchmarking purposes. This setup, which was able to successfully carry out all of the demonstration steps detailed above, appears in Figure 39.

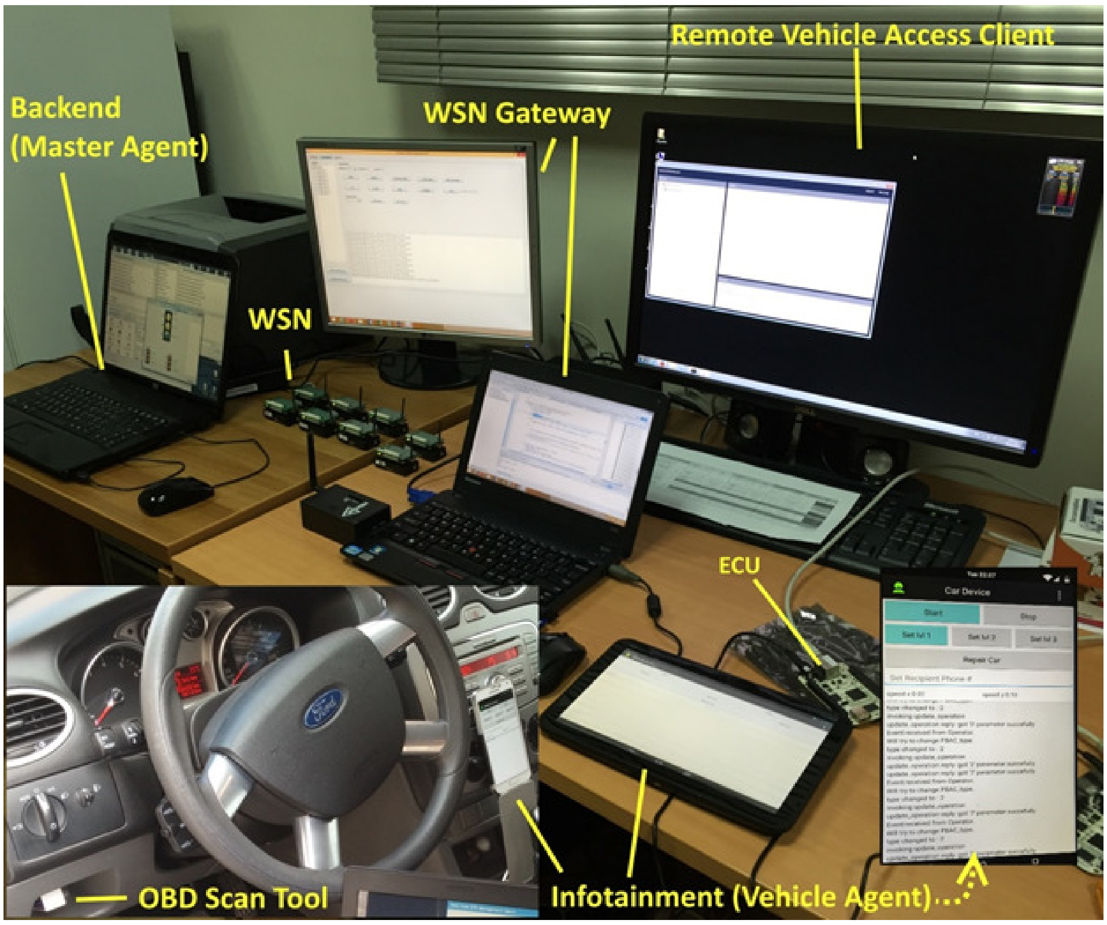


Figure 39 The social mobility and smart car demonstration setting.

The performance of SPD-Safe was evaluated focusing mostly on the resource-constrained devices, as documenting all performance aspects of all entities involved in a complex framework like this would not be practical. To investigate the framework's behavior, a benchmark client was developed that issued consecutive requests to access the smart vehicle's location (from the infotainment device) and its engine temperature (from the ECU). Both these operations were protected by the access control mechanism detailed above, thus before replying the devices verified (by communicating with the backend) that the client was authorized to access this data, based on the active policy set. Furthermore, at random intervals, the benchmark client would communicate with the master agent to trigger changes in the SPD state of the prototypes (e.g. to increase the key length used for encryption), thus also evaluating the impact such changes can have on the responsiveness of the devices and the system as a whole.

The master agent is the most computationally demanding entity of SPD-Safe; its code size is *1.87MB*, it occupies *45MB* RAM and needs *1.6 seconds*, on average, to perform a reasoning process. Moving to the rest of the entities, we focused on the infotainment device (i.e. the android application) and the corresponding Linux-based application developed for the embedded platform (emulating the ECU). Their CPU load during tests was not that significant, with an average of *4.8%* recorded on the Beaglebone and an average of *4.1%* on the Android platform. Memory consumption was also acceptable, averaging *33.46MB* on the Beaglebone and *26MB* on the Android device. Perhaps the most interesting performance parameter is the delay experienced by the user attempting to access the smart vehicle's services and the latency of the system when changing between SPD states. These results appear in Figure 40, which shows the response time (recorded client-side) for each of the requests issued concurrently to both the vehicle's infotainment (tablet) and its ECU (Beaglebone). As is evident from the graph, the Linux-based embedded device is more responsive, with an average response time of *93.16ms* compared to *198,7ms* for the infotainment. This is expected, as there are more processes running at the background on the Android tablet and also features a GUI which further impacts its responsiveness. Nevertheless, both devices demonstrated acceptable response times. Also evident in the graph are the spikes recorded when the devices had to change SPD states (and, thus, had to process the incoming master agent request, change their operating parameters accordingly and inform the master agent once the changes were in place). These changes in SPD states do introduce significant delays, but they are not a concern, as such changes are expected to happen rarely during normal operation (when the framework needs to react to some attack, when a vehicle crashes etc.).

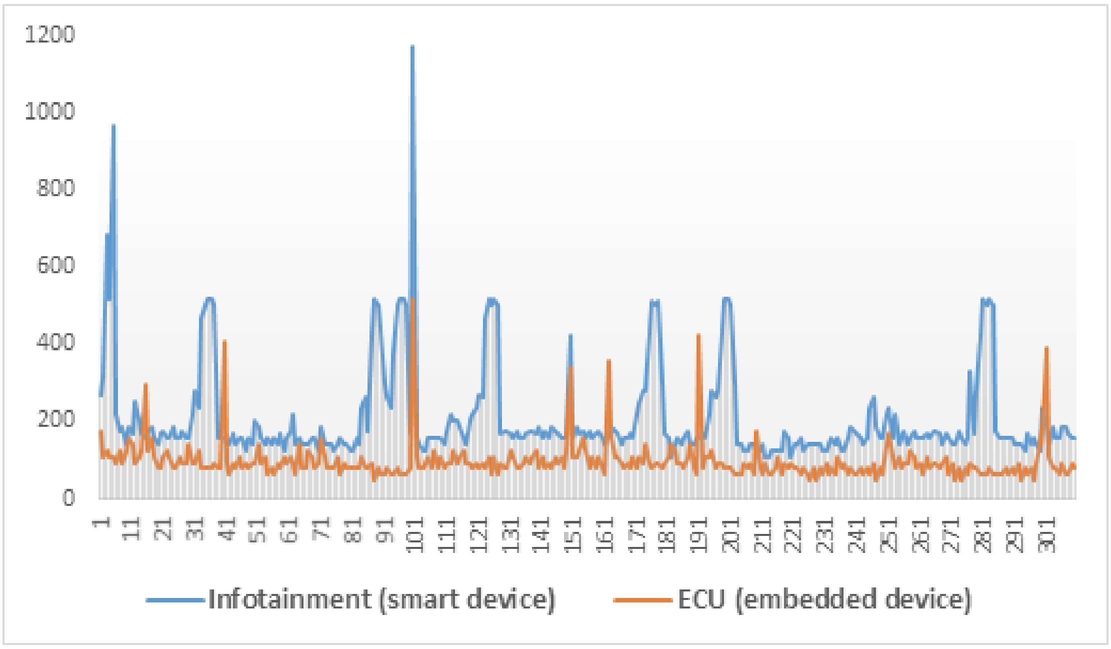


Figure 40 Performance evaluation of SPD-Safe – Response time (in ms) per request for embedded and mobile devices

13. CONCLUSIONS & FUTURE WORK

This thesis presented SPD-Safe – a management framework for embedded systems deployed in ambient secure and safety-critical applications. The multi-agent system is implemented in the JADE platform and deployed on OSGi middleware for controlling DPWS devices, also exploiting the security mechanisms already specified in the abovementioned technologies. The core of the reasoning process is an event-based model checker which extends the Event Calculus. Security, privacy, and dependability (SPD) related theory is modelled and implemented as a formal framework for system composition verification, security validation and metric-driven management. Moreover, modelling of safety-related theory and the implementation of AI ambient plans and strategies are supported. These two features are combined to manage the underlying embedded systems, considering both the SPD and the safety perspectives. The proposed solution is applied in five scenarios: smart home, the smart-buildings of a university campus, smart vehicle fleet, railway infrastructure, and precision agricultural. SPD-Safe, in this context, manages and configures the underlying systems at runtime to perform AI reactive plans that retain the safety of the residents in cases of emergency or counter cyber-security attacks.

The SPD evaluation is useful in reasoning if a system can meet some SPD requirements and counter attacks under a specific threat model as well as comparing different system settings and determining which is the safest.

However, from a business and engineering perspective, it is also desirable to determine which of the systems that meet our requirements is the most profitable/cost-effective. Thus, SPD-Safe could be extended to also evaluate the economic cost of implementing each system state. The above-mentioned cost and SPD evaluations can be integrated in business frameworks for assessing investments and performing cost-benefit analysis on embedded and information systems in general.

Technology ageing is another important factor for long-term product usage. Smart devices in the smart home domain, like fridge or surveillance cameras, are expecting to function for many years. This long device lifetime affects the protection status and can derive several of the current safe solutions, inadequate after some decades. For example, in the cryptography field, the cipher AES substituted the outdated cipher DES and elliptic curve crypto-systems are considered as a future replacement of RSA ones. The proper metrics and measurement techniques should take into consideration system ageing issues. The composed SPD multi-metric can resolve some of these problems. The analysis that estimates the protection level of the applied defence mechanisms based on a known set of limitations can be periodically updated to map the current status. Thus, a product vendor can keep up to date the SPD of his products and the system designer can re-estimate the composed SPD of his deployment, when changes occur.

14. REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Commun. Surv. Tutorials, vol. 17, no. 4, pp. 2347–2376, Jan. 2015.

[2] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance modelling and analysis of software defined networking under bursty multimedia traffic," ACM Trans. Multimedia Comput. Commun. Appl., vol. 12, issue 5s, article no. 77, Dec. 2016.

[3] S.L. Toral, F. Barrero, F. Cortes, and D. Gregor, "Analysis of embedded CORBA middleware performance on urban distributed transportation equipments," Computer Standards & Interfaces, vol. 35, issue 1, pp. 150-157, Jan. 2013.

[4] Y. Yan, R. Q. Hu, S. K Das, H. Sharif, and Y. Qian, "A security protocol for advanced metering infrastructure in smart grid," IEEE Network, vol. 27, issue 4, pp. 64-71, Jul 2013.

[5] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, "Experimental security analysis of a modern automobile," IEEE Symposium on Security and Privacy, pp. 447-462, 2010.

[6] D. Albright, P. Brannan, and C. Walrond, "Did Stuxnet take out 1,000 centrifuges at the Natanz enrichment plant?," Technical report, Institute for Science and International Security, 22 December 2010.

[7] A. Sharpanskykh and R. Haest, "An agent-based model to study compliance with safety regulations at an airline ground service organization," Applied Intelligence, Springer, vol. 45, issue 3, pp. 881-903, October 2016.

[8] C. Levy-Bencheton. E. Darra, G. Tetu, G. Dufay, and M. Alattar, "Security and resilience of smart home environments – Good practices and recommendations," ENISA, December 2015.

[9] G. Hatzivasilis, G. Floros, I. Papaefstathiou, and C. Manifavas, "Lightweight authenticated encryption for embedded on-chip systems," Information Security Journal: A Global Perspective, Taylor & Francis, vol. 25, issue 4-6, pp. 151-161, August 2016.

[10] G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, and I. Askoxylakis, "Lightweight password hashing scheme for embedded systems," 9th WG 11.2 International Conference on Information Security Theory and Practice (WISTP), IFIP, Heraklion, Crete, Greece, Springer, LNCS, vol. 9311, pp. 249-259, 2015.

[11] I. Papaefstathiou, A. Bilanakos, K. Fysarakis, G. Hatzivasilis, and C. Manifavas, "An efficient anti-malware intrusion detection system implementation exploiting GPUs," International Conference on Advanced Technology & Sciences (ICAT 2014), Antalya, Turkey, pp. 1-9, 2014.

[12] G. Hatzivasilis, I. Papaefstathiou, I. Askoxylakis, and K. Fysarakis, "SecRoute: End-to-End Secure Communications for Wireless Ad-hoc Networks," 22nd IEEE Symposium on Computers and Communications (ISCC 2017), IEEE, Heraklion, Crete, Greece, pp. 1-6, 2017.

[13] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "ModConTR: a Modular and Configurable Trust and Reputation-based system for secure routing," 11th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), IEEE, Doha, Qatar, pp. 56-63, 2014.

[14] E. T. Muller, "Commonsense reasoning: an Event Calculus based approach," M. Kaufmann, edition 2, 2015.

[15] T. Patkos and D. Plexousakis, “DECKT: epistemic reasoning for ambient intelligence,” ERCIM News magazine – Special Theme: Intelligent and Cognitive Systems, vol. 2011, issue 84, pp. 1-30, January 2011.

[16] G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, and N. Papadakis, “A reasoning system for composition verification and security validation,” 6th International Conference on New Technologies, Mobility and Security (NTMS), IEEE, Dubai, UAE, pp. 1-4, 2014.

[17] A. Olaru and C. Gratie, “Agent-based, context-aware information sharing for ambient intelligence,” International Journal of Artificial Intelligence Tools, World Scientific, vol. 20, no. 06, pp. 985-1000, December 2011.

[18] C. Manifavas, K. Fysarakis, K. Rantos, and G. Hatzivasilis, “DSAPE–Dynamic Security Awareness Program Evaluation,” 2nd International Conference on Human Aspects of Information Security, Privacy, and Trust (HCI International), Heraklion, Greece, Springer, LNCS, vol. 8533, pp. 258-269, 2014.

[19] K. M. Khali, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, “Intelligent techniques for resolving conflicts of knowledge in multi-agent decision support systems,” International Conference on Intelligent Computing & Information Systems (ICICIS), Cairo, Egypt, pp. 235-239, 2013.

[20] M. Garagnani, M. Fox, and D. P. Long, “Belief systems for conflict resolution,” 13th European Conference on AI (ECAI) – Workshop on conflicts Among Agents, Brighton, UK, pp. 55-60, August 1998.

[21] C. Darnon, C. Buchs, and F. Butera, “Epistemic and relational conflicts in sharing identical vs. complementary information during cooperative learning,” Swiss journal of Psychology, Hogrefe AG, vol. 61, no. 3, pp. 139-151, September 2002.

[22] J. Sztipanovits and G. Karsai, “Model-integrated computing, Computer Volume,” vol. 30, issue 4, pp. 110-111, 1997.

[23] Architecture analysis & design language, “SAE Standard AS-5506,” 2004.

[24] T. Szemethy and G. Karsai, “Platform modeling and model transformation for analysis,” Journal of Universal Computer Science, vol. 10, issue 10, pp. 1383-1406, 2004.

[25] M. Eby, “Integrating security modeling into embedded system design,” Master Thesis, Vanderbilt University, 2007.

[26] G. Madl and S. Abdelwahed, “Model-based analysis of distributed real-time embedded system composition,” EMSOFT’05, New Jersey, USA, pp. 371-374, 19-22 September, 2005.

[27] R. Blanco and P. Alencar, “Towards modularization and composition in distributed event based systems,” Technical report 2009, Cheriton School of Computer Science, Waterloo, 2009.

[28] E. Kushilevitz, Y. Lindell, and T. Rabin, “Information-theoretical secure protocols and security under composition,” SLAM Journal on Computing, vol. 39, issue 4, pp. 2090-2112, 2010.

[29] D. Kidron and Y. Lindell, “Impossibility results for universal composability in public-key models and with fixed inputs,” Journal of Cryptology, vol. 24, issue 3, pp. 517-544, 2011.

[30] K. M. Khan and J. Han, “Deriving Systems Level Security Properties of Component Based Composite Systems,” ASWEC 2005, pp. 334–343, 29 March-1 April 2005.

[31] N. Bielova, “A theory of constructive and predictable runtime enforcement mechanisms,” PhD Dissertation, University of Trento, November 2011.

[32] R. M. Savola and P. Heinonen, “A visualization and modeling tool for security metrics and measurements management,” ISSA, Johannesburg, South Africa, pp. 1-8, 15-17 August 2011.

[33] O. Xinming, S. Govindavajhala, and A. W. Appel, "MulVAL: a logic-based network security analyzer," 14th USENIX Security Symposium, 2005.

[34] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic, "A derivation system and compositional logic for security protocols," Journal of computer Security, IOS Press, vol. 3, pp. 423-482, 2005.

[35] J. P. de Albuquerque, H. Krumm, and P. L. de Geus, "Formal validation of automated policy refinement in the management of network security systems," International Journal of Information Security, vol. 9, issue 2, pp. 99-125, 2010.

[36] M. Walter and C. Trinitis, "Quantifying the security of composed systems," PPAM'05, Springer, pp. 1026-1033, 2006.

[37] P. K. Manadhata and J. M. Wing, "An attack surface metric," IEEE Transactions on Software Engineering, vol. 37, issue 3, pp. 371-386, 2010.

[38] I. Eguia and J. D. Ser, "A meta-heuristically optimized fuzzy approach towards multi-metric security risk assessment in heterogeneous system of systems," MeSeCCS 2014, Lisbon, Portugal, 7-9 January 2014.

[39] R. M. Savola and M. Sihvonen, "Metrics Driven Security Management Framework for E-Health Ecosystem Focusing on Chronic Diseases," International Conference on Management of Emergent Digital EcoSystems, pp. 75-79, 2012.

[40] L. Krautsevich, F. Martinelli, and A. Yautsiukhin, "Formal approach to security metrics. What does "more secure" mean to you?," ECSA'10, pp. 162-169, 2010.

[41] J. Sztipanovits and G. Karsai, "Model-integrated computing, Computer Volume," vol. 30, issue 4, pp. 110-111, 1997.

[42] N. Fenton, P. Krause, and M. Neil, "Software Measurement: Uncertainty and Causal Modeling," IEEE Software, vol. 19, no. 4, pp. 116-122, 2002.

[43] F. Shull et al., "Fully Employing Software Inspections Data," Innovations in Systems and Software Eng., vol. 8, no. 4, 2012, pp. 243-254.

[44] L.S. Azevedo et al., "Assisted Assignment of Automotive Safety Requirements," IEEE Software, vol. 31, no. 1, 2014, pp. 62-68.

[45] K.V. Prasad, T.J. Giuli, and D. Watson, "The Case for Modeling Security, Privacy, Usability and Reliability (SPUR) in Automotive Software," Model-Driven Development of Reliable Automotive Services, LNCS 4922, Springer, 2008, pp. 1-14.

[46] M. Howard and Microsoft Corp., "Determining Relative Attack Surface," US patent 7299497 B2, Patent and Trademark Office, 2007.

[47] Common Criteria for Information Technology Security Evaluation, "ISO/IEC 15408," 1996–2015: www.commoncriteriaportal.org.

[48] Open Source Security Testing Methodology Manual, "ISECOM," 1988–2015: www.isecom.org/research/osstmm.html.

[49] Privacy Framework, "ISO/IEC 29100," 2011: www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en.

[50] Code of Practice for Protection of Personally Identifiable Information (PII) in Public Clouds Acting as PII Processors, "ISO/IEC 27018," 2014: www.iso.org/iso/catalogue_detail?csnumber=61498.

[51] International Standards on Dependability, "IEC 60300," 2006: www.iec.ch/about/brochures/pdf/technology/dependablility.pdf.

[52] T. Zimmermann et al., "An Empirical Study on the Relation between Dependency Neighborhoods and Failures," Proc. IEEE 4th Int'l Conf. Software Testing, Validation and Verification (ICST 11), pp. 347-356, 2011.

[53] J. Sawin and A. Rountev, "Estimating the Run-Time Progress of a Call Graph Construction Algorithm," Proc. 6th IEEE Int'l Workshop Source Code Analysis and Manipulation (SCAM 06), pp. 53-62, 2006.

[54] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "Software Security, Privacy and Dependability: Metrics and Measurement," IEEE Software, IEEE, vol. 33, issue 4, pp. 46-54, 2016.

[55] D. Basin and C. Cremers, "Know Your Enemy: Compromising Adversaries in Protocol Analysis," ACM Trans. Info. Syst. Sec. (TISSEC), vol. 17, issue 2, article 7, 2014.

[56] G. Hatzivasilis, "Multi-agent distributed epistemic reasoning in ambient intelligence environments," Master thesis, University of Crete, Department of Computer Science, FORTH Institute of Computer Science, Heraklion, Crete, Greece, code 000370769, pp. 1-88, 2011.

[57] S. Guha, N. Daswani, and R. Jain, "An experimental study of the Skype peer-to-peer VoIP system," 5th International Workshop on Peer-to-Peer Systems (IPTPS 2006), California, USA, pp. 1-6, February 2006.

[58] B. Kaluza, B. Cvetkovic, E. Dovgan, H. Gjoreski, M. Gams, and M. Lustrek, "A multi-agent care system to support independent living," International Journal of Artificial Intelligence Tools, World Scientific, vol. 23, no. 01, article 1440001, pages 30, February 2014.

[59] B. S. Faical, G. Pessin, G. P. R. Filho, A. C. P. L F. Carvalho, P. H Gomes, and J. Ueyama, "Fine-tuning of UAV control rules for spraying pesticides on crop fields: an approach for dynamic environments," International Journal of Artificial Intelligence Tools, World Scientific, vol. 25, no. 01, article 1660003, pages 19, February 2016.

[60] K. Lawrence, C. Kaler, A. Nadalin, R. Monzilo, and P. Hallam-Baker, "Web services security: SOAP message security 1.1," OASIS standard specification, pp. 1-76, 2006: available at: https://www.oasisopen.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[61] G. Mantas, D. Lymberopoulos, and N. Komninos, "Security in smart home environment," Wireless Technologies for ambient assisting living and healthcare: systems and applications, IGI Global, chapter 10, pp. 170-191, 2010.

[62] K. Rantos, K. Fysarakis, C. Manifavas, and I. Askoxylakis, "Policy-controlled authenticated access to LLN-connected healthcare resources," IEEE Systems Journal, vol. PP, no. 99, pp. 1-11, 2015

[63] G. Hatzivasilis, E. Gasparis, A. Theodoridis, and C. Manifavas, "ULCL: an Ultra-Lightweight Cryptographic Library for embedded systems," MeSeCCS 2014, Lisbon, Portugal, pp. 11-18, 7-9 January 2014.

[64] M. Fahad, O. Boissier, P. Maret, N. Moalla, and C. Gravier, "Smart places: multi-agent based smart mobile virtual community management system," Applied Intelligence, Springer, vol. 41, issue 4, pp. 1024-1042, December 2014.

[65] A. S. Rao and M. P. Georgeff, "BDI agents: from theory to practice," 1st International Conference on Multi-Agent Systems (ICMAS), California, USA, The MIT Press, pp. 312-319, 1995.

[66] M. Garcia-Herranz, X. Alaman, and P. A. Haya, "Easing the smart home: a rule-based language and multi-agent structure for end user development in intelligent environments," Journal of Ambient Intelligence and Smart Environments, IOS Press, vol. 2, no. 4, pp. 437-438, October 2010.

[67] G. Semmel, S. Davis, K. Leucht, D. Rowe, A. Kelly, and L. Boloni, “Launch commit criteria monitoring agent,” 4th International joint conference on Autonomous Agents and MultiAgent Systems (AAMAS), ACM Press, Utrecht, Netherlands, pp. 3-10, July 2005.
[68] P. Aschwanden, V. Baskaran, S. Bernardini, C. Fry, M.D. R-Moreno, N. Muscettola, C. Plaunt, D. Rijsman, and P. Tompkins, “Model-Unified Planning and Execution for Distributed Autonomous System Control,” American Association for Artificial Intelligence (AAAI) 2006 Fall Symposia, Washington DC, USA, October 2006
[69] D. Sislak, M. Pechoucek, P. Volf, D. Pavlicek, J. Samek, V. Marik, and P. Losiewicz, “AGENTFLY: towards multi-agent technology in free flight air traffic control,” Defence Industry Application of Autonomous Agents and Multi-Agent Systems, Springer, pp. 73-96, 2008.
[70] J. McKean, H. Shorter, M. Luck, P. McBurney, and S. Willmott, “Technology diffusion: analysing the diffusion of agent technologies,” Autonomous Agents and Multi-Agent Systems, Springer, vol. 17, issue 3, pp. 372-396, December 2008.
[71] F. Bergenti and E. Vargiu, “Multi-agent systems in the industry. Three notable cases in Italy,” 11th Workshop from Objects to Agents (WOA), CEUR Workshop Proceedings, Rimini, Italy, vol. 621, pp. 1-6, September 2010.
[72] M. Pechoucek and V. Marik, “Industrial deployment of multi-agent technologies: review and selected case studies,” Autonomous Agents and Multi-Agent Systems, Springer, vol. 17, issue 3, pp. 397-431, December 2008.
[73] Z. Yong-Xin, L. Qing-Zhong, and P. Zhao-Hui, “A novel method for data conflict resolution using multiple rules,” Computer Science and Information Systems, ComSIS Consortium, vol. 10, issue 1, pp. 215-235, January 2013.
[74] R. Thirumalainambi, “Pitfalls of JESS for dynamic systems,” International Conference on Artificial Intelligence and Pattern Recognition (AIPR), Florida, USA, ISRST, vol. 1, pp, 491-494, July 2007.
[75] K. Fysarakis, O. Soultatos, I. Askoxylakis, H. Manifavas. I. Papaefstathiou, and V. Katos, “Which IoT protocol? Comparing standardized approaches over a common M2M application,” IEEE Global Communications Conference (GLOBECOM), IEEE, Washington DC, USA, pp. 1-6, December 2016.
[76] K. Kravari and N. Bassiliades, “A survey of agent platforms,” Journal of Artificial Societies and Social Simulation (JASSS), vol. 18, issue 1, pp. 11-29, January 2015.
[77] M. Jarrar and S. Heymans, “Towards pattern-based reasoning for friendly ontology debugging,” International Journal of Artificial Intelligence Tools, World Scientific, no. 17, issue 04, pp. 607-634, August 2008.
[78] S. E. Shimony and E. Nissan, “Kappa calculus and evidential strength: A note on Aqvist’s logical theory of legal evidence,” Artificial Intelligence and Law, Springer, vol. 9, issue 2, pp. 153-163, September 2001.
[79] Y. Deng, “Generalized evidence theory,” Applied Intelligence, Springer, vol. 43, issue 3, pp. 530-543, October 2015.
[80] A. Bikakis and G. Antoniou, “Distributed reasoning with conflicts in an ambient peer-to-peer setting,” Constructing Ambient Intelligence: AmI 2007 Workshops, Darmstadt, Germany, Springer, CCIS, vol. 11, pp. 24-33, 2008.
[81] C.-Y. Chen and H.-C. Chao, “A survey of key distribution in wireless sensor networks,” Security and Communication Networks, Wiley 2011, vol. 7, issue 12, pp. 2495-2508, 2014.
[82] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, “Lightweight Cryptography for Embedded Systems – a Comparative Analysis,” 6th International Workshop on

Autonomous and Spontaneous Security – SETOP 2012, Springer, LNCS, 8247, pp. 333-349, 2012.

[83] M. Dworkin, "Recommendation for block cipher modes of operation," NIST special publication 800-38A, Edition 2001, 2001, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf .

[84] G. Hatzivasilis, "Password-Hashing Status," Cryptography, MDPI Open Access Journal, vol. 1, issue 2, number 10, 2017.

[85] J.-S. Leu and W.-B. Hsieh, "Efficient and secure dynamic ID-based remote user authentication scheme for distributed systems using smart cards," IET Information Security, vol. 8, issue 2, pp. 104-113, 2014.

[86] C. Forler, S. Lucks, J. Wenzel, "The Catena Password Scrambler, PHC submission," May 15, 2014. https://password-hashing.net/submissions/specs/Catena-v3.pdf .

[87] D. Dhurjati, S. Kowshik, V. Adve, C. Lattner, "Memory safety without garbage collection for embedded applications," ACM Transactions on Embedded Computing Systems (TECS), vol. 4, issue 1, pp. 73-111, 2005.

[88] A. Shamir "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[89] Password Hashing Competition (PHC): Candidates, March 31, 2014. https://password-hashing.net/candidates.html .

[90] CAESAR, "CAESAR: Competition for Authentication Encryption: Security, Applicability, and Robustness," CAESAR, 2013, http://competitions.cr.yp.to/caesar.html .

[91] NIST, "NIST Special Publication 800-38D Recommendation for block cipher modes of operation Calois/Counter Modes (GCM) and GMAC," NIST, 2007.

[92] OpenCores, "PRESENT-80 Verilog implementation," 2011 http://opencores.com/project,present_encryptor .

[93] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm," ASIACRYPT, Springer, LNCS, vol. 1976, 2000, pp. 531-545, 2000.

[94] L. Yang, M. Wang, S. Qiao, "Side channel cube attack on PRESENT," CANS, Springer, LNCS, vol. 5888, pp.379-391, 2009.

[95] O. Ozen, K. Varici, C. Tezcan, C. Kocair, "Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT," Information Security and Privacy, Springer, LNCS, vol. 5594, 2009, pp. 90-107, 2009.

[96] M. A. Abdelraheem, "Estimating the probabilities of low-weight differential and linear approximations on PRESENT-like ciphers," ICISC 2012, Springer, LNCS, vol. 7839, pp. 368-382, 2013.

[97] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, and A. Poschmann, "PRESENT: An Ultra-Lightweight Block Cipher," Cryptographic Hardware and Embedded Systems, CHES 2007, Springer, LNCS, 4727, pp. 450-466, 2007.

[98] D. Engels, M.O. Saarinen, P. Schweitzer, and E.M. Smith, "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm," RFID Security and Privacy, Springer, LNCS, 7055, pp. 19-31, 2011.

[99] M. Agren, M. Hell, T. Johanson, W. Meier, "A New Version of Grain-128 with Optional Authentication," International Journal of Wireless and Mobile Computing, vol. 5, issue 1, pp. 48-59, 2011.

[100] A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, E. Tischhauser, “ALE: AES-Based Lightweight Authenticated Encryption,” FSE 2013, Springer, LNCS, 8424, pp. 447-466, 2014.

[101] B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, O. Wang, “FIDES: Lightweight authenticated cipher with side-channel resistance for constrained hardware,” CHES 2013, Springer, LNCS, vol. 8086, pp. 142-158, 2013.

[102] G. Hatzivasilis and C. Manifavas, “Building trust in ad hoc distributed resource-sharing networks using reputation-based systems,” 16th Panhellenic Conference on Informatics (PCI 2012), IEEE, Piraeus, Greece, pp. 416-421, 2012.

[103] R. Dalal, M. Khari, Y. Singh, “Survey of trust schemes on ad-hoc network,” Advances in Computer Science and Information Technology Networks and Communications, Springer, vol. 84, pp. 170-180, 2012.

[104] A. Kr Trivedi, R. Arora, R. Kapoor, S. Sanyal, S. Sanyal, A Semi-distributed Reputation Based Intrusion Detection System for Mobile Adhoc Networks, Journal of Information Assurance and Security, vol. 1, pp. 265-274, 2006.

[105] F. Oliviero and S. P. Romano, “A reputation-based metric for secure routing in wireless mesh networks,” IEEE GLOBECOM 2006, pp. 1-5, 2006.

[106] S. Ganeriwal, L. K. Balzano, M. B. Srivastava, “Reputation-based framework for high integrity sensor networks,” ACM TOSN, vol. 4, no 3, pp. 1-37, 2008.

[107] D. Kukreja, U. Singh, B. V. R. Reddy, “Analytical models for trust based routing protocols in wireless ad hoc networks,” ACM SIGSOFT Software Engineering Notes, vol. 37, pp. 1-16, 2012.

[108] S. Paris, C. Nita-Rotaru, F. Martignon, A. Capone, “Cross-layer metrics for reliable routing in wireless mesh networks,” IEEE/ACM TON, vol. 21, no 3, pp. 1003-1016, 2013.

[109] K. Altisen, S. Devismes, R. Jamet, P. Lafourcade, “SR3: secure resilient reputation-based routing,” IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 258-265, 2013.

[110] M. A. Ngadi, R. H. Khokhar, S. Mandala, “A review of current routing attacks in mobile ad-hoc networks,” International Journal of Computer Science and Security, Vol 2, issue 3, pp. 18-29, 2008.

[111] D. Airehrour, J. Gutierrez, S. K. Ray, “Secure routing for internet of things: A survey.” Journal of Network and Computer Applications, Elsevier, vol. 66, pp. 198-213, 2016.

[112] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, G. Pantziou, “A survey on jamming attacks and countermeasures in WSNs,” IEEE Communications Surveys & Tutorials, vol. 11, issue 4, pp. 42-56, 2009.

[113] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, J. Chen, “Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks,” IEEE Internet of Things Journal, vol. 3, issue 5, pp. 816-829, 2016.

[114] J. Lin, W. Yu, X. Yang, G. Xu, W. Zhao, “On false data injection attacks against distributed energy routing in smart grid,” IEEE/ACM 3rd International Conference on Cyber-Physical Systems (ICCPS), pp. 183-192, 2012.

[115] G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, “SCOTRES: Secure Routing for IoT and CPS,” IEEE Internet of Things Journal (IoT), IEEE, 2017.

[116] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, “A survey of routing attacks in mobile ad hoc networks,” IEEE Wireless Communications, vol. 14, issue 5, pp. 85-91, 2007.

[117] L. Ertaul and B. Ibrahim, “Evaluation of secure routing protocols in mobile ad hoc networks (MANETs),” International Conference on Security & Management (SAM 2009), vol. 2, 2009.

[118] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, H. Rubens, “ODSBR: An On-Demand Secure Byzantine Resilient Routing protocol for wireless ad hoc networks,” ACM Transactions on Information and System Security (TISSEC), vol. 10 issue 4, article no. 6, pp. 1-35, 2008.

[119] E. Karapistoli, P. Sarigiannidis, A. A. Economides, “Visual-Assisted Wormhole Attack Detection for Wireless Sensor Networks,” International Conference on Security and Privacy in Communication Networks, vol. 152, pp. 222-238, 2015.

[120] Y.-C. Hu, A. Perrig, D. B. Johnson, “Wormhole attacks in wireless networks,” IEEE J-SAC, vol. 24, pp. 370-380, 2006.

[121] K. Hoffman, D. Zage, C. Nita-Rotaru, “A survey of attack and defense techniques for reputation systems,” ACM CSUR, vol. 42, issue 1, article 1, pp. 1-31, 2009.

[122] E. Carrara and G. Hogben, “Reputation-based systems: a security analysis,” ENISA Position Paper 424, 2007.

[123] Y.-C. Hu, A. Perrig, D. B. Johnson, “Ariadne: A secure on-demand routing protocol for ad hoc networks,” Wireless networks, Springer, vol. 11, issue 1-2, pp. 21-38, 2005.

[124] E. Karapistoli and A. A. Economides, “Defending jamming attacks in wireless sensor networks using stackelberg monitoring strategies,” IEEE/CIC International Conference on Communications in China (ICCC), 2014.

[125] A. Perrig et al., “The TESLA Broadcast Authentication Protocol,” CryptoBytes, vol. 5, issue 2, pp. 2-13, 2002.

[126] K. Sanzgiri et al., “A secure routing protocol for ad hoc networks,” 10th IEEE International Conference on Network Protocols, IEEE, pp. 78-87, 2002.

[127] N. Sufyan, N. A. Saqib, M. Zia, “Detection of jamming attacks in 802.11 b wireless networks,” EURASIP Journal on Wireless Communications and Networking, Springer, vol. 2013, article 208, pp. 1-18, 2013.

[128] B. Parducci and H. Lockhart, “eXtensible Access Control Markup Language (XACML) version 3.0,” OASIS Standard, pp. 1-154, 2013.

[129] N. Zhang, W. Yu, X. Fu, S. K. Das, “Maintaining defender’s reputation in anomaly detection against insider attacks,” IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 40, no. 3, pp. 597-611, 2010.

[130] T. H. Hai, E.-N. Huh, M. Jo, “A lightweight intrusion detection framework for wireless sensor networks,” Wireless. Communication Mobile Computing, vol. 10, pp. 559-572, 2010.

[131] E. Karapistoli, I. Mampentzidou, A. A. Economides, “Environmental Monitoring Based on the Wireless Sensor Networking Technology: A Survey of Real-World Applications,” International Journal of Agricultural and Environmental Information Systems (IJAEIS), IGI Global, vol. 5, no. 04, 39, 2014.

[132] Z. Wang, H. Song, D. W. Watkins, K. G. Ong, P. Xue, Q. Yang, X. Shi “Cyber-physical systems for water sustainability: challenges and opportunities,” IEEE Communications Magazine, vol. 53, issue 5, pp. 216-222, 2015.

[133] B. S. Faical, G. Pessin, G. P. R. Filho, A. C. P. L F. Carvalho, P. H Gomes, J. Ueyama, “Fine-tuning of UAV control rules for spraying pesticides on crop fields: an approach for dynamic environments,” International Journal of Artificial Intelligence Tools, vol. 25, issue 01, 2016.

[134] G. Xu, W. Yu, Z. Chen, H. Zhang, P. Moulema, X. Fu, C. Lu, “A cloud computing based system for cyber security management,” International Journal of Parallel, Emergent and Distributed Systems, vol. 30, no. 1, pp. 29-45, 2015.
[135] Greek Research and Academic Community, okeanos cloud services, https://okeanos.grnet.gr/home/
[136] V. Balasubramanian, D. Massaguer, S. Mehrotra, and N. Venkatasubramanian, “DrillSim: a simulation framework for emergency response drills,” 4th IEEE International Conference on Intelligence and Security Informatics (ISI), California, USA, Springer, LNCS, vol. 3975, pp. 237-248, 2006.
[137] N. Schurr, J. Marecki, J. P. Lewis, M. Tambe, and P. Scerri, “The Defacto system: coordinating human-agent teams for the future of disaster response,” Multi-Agent Programming, Springer, series of Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15, part III, pp. 197-215, 2005.
[138] N. M. Adams, M. Field, E. Gelenbe, D. J. Hand, N. R. Jennings, D. S. Leslie, D. Nicholson, S. D. Ramchurn, S. J. Roberts, and A. Rogers, “The Aladdin project: intelligent agents for disaster management,” IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance (RISE), Benicassim, Spain, January 2008.
[139] R. C. Cardoso and R. H. Bordini, “A distributed online multi-agent planning system,” 26th International Conference on Automated Planning and Scheduling(ICAPS) proceedings of the 4th Workshop on Distributed and Multi-Agent Planning (DMAP), London, UK, pp. 15-23, June 2016.
[140] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, “Bridging the gap between crisis response operations and systems,” Annals of the University of Craiova, pp. 1-6, August 2009.
[141] K. Fysarakis, G. Hatzivasilis, I. G. Askoxylakis, and C. Manifavas “RT-SPDM: Real-time Security, Privacy and Dependability Management of Heterogeneous Systems,” HCI International 2015, Springer, LNCS, vol. 9190, pp. 619-630, 2015.
[142] K. Fysarakis, G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas “RtVMF – a secure real-time vehicle management framework with critical incident response,” IEEE Pervasive Computing Magazine – Special issue on Smart Vehicle Spaces, vol. 15, issue 1, pp. 22-30, 2016.
[143] nSHIELD, “D7.9: Railway security system demonstrator validation and verification report,” Railway scenario, public deliverable, 2014.
[144] J. Waldo, “Embedded computing and formula one racing,” IEEE Pervasive Computing, vol. 4, no. 3. pp. 18-21, 2005.
[145] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson, and K. V. Prasad, “Control, computing and communications: Technologies for the twenty-first century model T,” Proc. IEEE, vol. 95, no. 2, pp. 334-355, 2007.
[146] A. Doshi, B. T. Morris, and M. M. Trivedi, “On-road prediction of driver’s intent with multimodal sensory cues,” IEEE Pervasive Computing, vol. 10, no. 3, pp. 22-34, 2011.
[147] J. F. Coughlin, B. Reimer, and B. Mehler, “Monitoring, managing, and motivating driver safety and well-being,” IEEE Pervasive Computing, vol. 10, no. 3, pp. 14-21, 2011.
[148] K. Lee, J. Flinn, T. J. Giuli, B. Noble, and C. Peplin, “AMC: verifying user interface properties for vehicular applications,” 11th annual international conference on Mobile systems, applications, and services (MobiSys ‘13), pp. 1-12, 2013.
[149] Economic commission for Europe, “Telematic applications: eCall HGV/GV, additional data concept specification,” September 2011.

http://www.unece.org/fileadmin/DAM/trans/doc/2011/dgwp15ac1/INF.30e.pdf. [Accessed: 02-Aug-2017]

[150] T. J. Giuli, D. Watson, and K. V. Prasad, "The last inch at 70 miles per hour," IEEE Pervasive Computing, vol. 5, no. 4, pp. 20-27, 2006.

[151] nSHIELD, "Build secure embedded systems with nSHIELD," EU-funded project – new embedded Systems arcHItecturE for multi-Layer Dependable solutions (nSHIELD), Project no: 269317, 2011-2014. http://www.newshield.eu/wp-content/uploads/2015/01/NSHIELD-D8.7_Build_Secure_Systems_with_SHIELD_v3.pdf. [Accessed: 02-Aug-2017]

[152] Z. Wu, Q. Wu, H. Cheng, G. Pan, M. Zhao, and J. Sun, "ScudWare: A semantic and adaptive middleware platform for smart vehicle space," IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 1, pp. 121-132, 2007.

[153] S. Hodges, N. Villar, J. Scott, and A. Schmidt, "A new era for ubicomp development," IEEE Pervasive Comput., vol. 11, no. 1, pp. 5-9, 2012.

[154] Standard, NIST FIPS, "Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication, 197, 2001.

[155] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations," IEEE Design & Test of Computers, 24(6), pp. 522-533, 2007.

[156] R. Roman, C. Alcaraz, and J. Lopez, "A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes," Mobile Networks and Applications, 12(4), pp. 231-244, 2007.

[157] C. Paar, A. Poschmann, and M. J. B. Robshaw, "New designs in lightweight symmetric encryption," RFID Security, vol. 3, pp. 349-371, 2009.

[158] P. Kitsos, N. Sklavos, M. Parousi, and A.N. Skodras, "A comparative study of hardware architectures for lightweight block ciphers," Computers & Electrical Engineering, 38(1), pp. 148-160, 2012.

[159] M., Cazorla, K. Marquet, and M. Minier, "Survey and benchmark of lightweight block ciphers for wireless sensor networks," IACR Cryptology ePrint Archive, 295, 2013.

[160] T. Eisenbarth et al., "Compact implementation and performance evaluation of block ciphers in ATtiny devices," Progress in Cryptology – AFRICACRYPT 2012, Springer, LNCS, 7374, pp. 172-187, 2012.

[161] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Grobshadl, and A. Biryukov, "Triathlon of Lightweight Block Ciphers for the Internet of Things," IACR Cryptology ePrint Archive, 209, 2015.

[162] A. Anjali, Priyanka and S.K. Pal, "A Survey of Cryptanalytic Attacks on Lightweight Block Ciphers," International Journal of Computer Science and Information & Security (IJCSITS), 2(2), 2012.

[163] K. Fysarakis, G. Hatzivasilis, K. Rantos, A. Papanikolaou, and C. Manifavas, "Embedded systems security challenges," Measurable security for Embedded Computing and Communication Systems – MeSeCCs 2014, Lisbon, Portugal, pp. 1-10, 2014.

[164] C. Manifavas, G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, "A survey of lightweight stream ciphers for embedded systems," Security and Communication Networks, Wiley, issue 9, pp. 1226-1246, 2015.

[165] S.E. Sarma, "Towards the five-cent tag," MIT-AUTOID-WH-006, 2001.

[166] S. Weis, "Security and privacy in radio-frequency identification devices," Faculty of the Massachusetts Institute of Technology (M.I.T.), 2003.

[167] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, "Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents," Smart Card Research and Advanced Applications, Springer, LNCS, 5189, pp. 89-103, 2008.

[168] D. Gligoroski, "Edon-library of Reconfigurable Cryptographic Primitives Suitable for Embedded Systems," Workshop on Cryptographic Hardware and Embedded Systems, 2003.

[169] T. Akishita and H. Hiwatari, "Very compact hardware implementations of the blockcipher CLEFIA," Selected Areas in Cryptography (SAC'12), Springer, LNCS, 7118, pp. 278-292, 2012.

[170] Z. Gong, S. Nikova, and Y.W. Law, "KLEIN: A New Family of Lightweight Block Ciphers," RFID Security and Privacy, Springer, LNCS, 7055, pp. 1-18, 2012.

[171] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED Block Cipher, Cryptographic Hardware and Embedded Systems," CHES 2011, Springer, LNCS, 6917, pp. 326-341, 2011.

[172] A. Poschmann, S. Ling, and H. Wang, "256 Bit Standardized Crypto for 650 GE – GOST Revisited," Cryptographic Hardware and Embedded Systems, CHES 2010, Springer, LNCS, 6225, pp. 219-233, 2010.

[173] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An Ultra-Lightweight Blockcipher," Cryptographic Hardware and Embedded Systems (CHES 2011), Springer, LNCS, 6917, pp. 342-357, 2011.

[174] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "Twine: A lightweight, versatile block cipher," ECRYPT Workshop on Lightweight Cryptography (LC11, pp. 146-169), 2011.

[175] W. Wu and L. Zhang, "LBlock: a lightweight block cipher," Applied Cryptography and Network Security, Springer, LNCS, 6715, pp. 327-344, 2011.

[176] H. Yap, K. Khoo, A. Poschmann, and M. Henricksen, "EPCBC - A Block Cipher Suitable for Electronic Product Code Encryption," Cryptology and Network Security (CANS), Springer, LNCS, 7092, pp. 76-97, 2011.

[177] M. Çakiroglu, "Software implementation and performance comparison of popular block ciphers on 8-bit low-cost microcontroller," International Journal of the Physical Sciences, 5(9), pp. 1338-1343, 2010.

[178] H. Cheng and H.M. Heys, "Compact ASIC implementation of the ICEBERG block cipher with concurrent error detection," IEEE International Symposium on Circuits and Systems - ISCAS 2008, Seattle, Wash, pp. 2921-2924, 2008.

[179] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: a very compact and a threshold implementation of AES," Advances in Cryptology – EUROCRYPT 2011, Springer, LNCS, 6632, pp. 69-88, 2011.

[180] S.K. Ojha, N. Kumar, K. Jain, and Sangeeta, "TWIS – A Lightweight Block Cipher," Information Systems Security, Springer, LNCS, 5905, pp. 280-291, 2009.

[181] A. Poschmann, "Lightweight Cryptography: Cryptographic Engineering for a Pervasive World," Ruhr-University Bochum, Germany, 2009.

[182] TOSHIBA, "Toshiba CMOS Technology Roadmap for ASIC," 2015. http://www.toshiba-components.com/ASIC/Technology.html .

[183] B. Gerard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert, "Block Ciphers that are Easier to Mask: How Far Can we Go?," IACR Cryptology ePrint Archive, 2013.

[184] P. Hamalainen et al., "Design and implementation of low-area and low-power AES encryption hardware core," Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006, 9th IEEE EUROMICRO Conference, pp. 577-583, 2006.

[185] S. Nikova, V. Rijmen, and M. Schlaffer, “Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches,” Journal of Cryptology, 24(2), pp. 292-321, 2011.

[186] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” Advances in Cryptology, CRYPTO’99, Springer, pp. 388-397, 1999.

[187] L. Knudsen, G. Leander, A. Poschmann, and M. J. B. Robshaw, “PRINTcipher: a block cipher for IC-printing,” Cryptographic Hardware and Embedded Systems, CHES 2010, Springer, LNCS, 6225, pp. 16-32, 2010.

[188] X. Guo and P. Schaumont, “The technology dependence of lightweight hash implementation cost,” ECRYPT Workshop on Lightweight Cryptography (LC ‘11), 2011.

[189] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen, “The NOEKEON Block Cipher,” pp. 1-30, 2000, http://gro.noekeon.org/ .

[190] F.-X. Standaert, G. Piret, G. Rouvroy, J. Quisquater, and J.-D. Legat, “ICEBERG: An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware,” Fast Software Encryption (FSE 2004), Springer, LNCS, 3017, pp. 279-298, 2004.

[191] C.H. Lim and T. Korkishko, “mCrypton - A lightweight block cipher for security of low-cost RFID tags and Sensors,” Information Security Applications, Springer, LNCS, 3786, pp. 243-258, 2006.

[192] C. Wang and H.M. Heys, “An ultra compact block cipher for serialized architecture implementations,” Canadian Conference on Electrical and Computer Engineering (CCECE '09), St. John's, Newfoundland, IEEE, pp. 1085-1090, 2009.

[193] G. Piret, T. Roche, and C. Carlet, “PICARO – A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance,” Applied Cryptography and Network Security, Springer, LNCS, 7341, pp. 311-328, 2012.

[194] J. Borghoff et al., “PRINCE – A Low-latency Block Cipher for Pervasive Computing Applications,” Advances in Cryptology – ASIACRYPT 2012, Springer, LNCS, 7658, pp. 208-225, 2012.

[195] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, “RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms,” IACR Cryptology ePrint Archive, 2014.

[196] M. R. Z'aba, N. Jamil, M. E. Rusli, M. Z. Jamaludinm, and A. A. M. Yasir, “I-PRESENT^{TM}: An Involutive Lightweight Block Cipher,” Journal of Information Security, Scientific Research, vol. 5, pp. 114-122, 2014.

[197] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalcin, “Block Ciphers – Focus On The Linear Layer (feat. PRIDE),” Advances in Cryptology - CRYPTO, Springer, LNCS, 8616, pp. 57-76, 2014.

[198] G. Leander, C. Paar, A. Poschmann, and K. Schramm, “New Lightweight DES Variants, Fast Software Encryption,” FSE 2007, Springer, LNCS, 4593, pp. 196-210, 2007.

[199] D. Wheeler and R. Needham, “TEA, a Tiny Encryption Algorithm,” Fast Software Encryption (FSE 1994), Springer, LNCS, 1008, pp. 363-366, 1994.

[200] R. Needham and D. Wheeler, “TEA extensions,” Technical report, Computer Laboratory, University of Cambridge, October, 1997.

[201] D. Wheeler and R. Needham, “Correction to XTEA,” Technical report, Computer Laboratory, University of Cambridge, October, 1998.

[202] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, “Camellia: a 128-bit block cipher suitable for multiple platforms – design and analysis,” Selected Areas in Cryptography (SAC'01), Springer, LNCS, pp. 39-56, 2001.

[203] F.-X. Standaert, G. Piret, N. Gershenfeld, and J. Quisquater, “SEA: A Scalable Encryption Algorithm for small embedded applications,” Smart Card Research and Advanced Applications, Springer, LNCS, 3928, pp. 222-236, 2006.

[204] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, “The 128-bit blockcipher CLEFIA (extended abstract),” Fast Software Encryption (FSE 2007), Springer, LNCS, 4593, pp. 181-195, 2007.

[205] ETSI’S Security Algorithms Group Of Experts (SAGE), Specification of the 3GPP confidentiality and integrity algorithms, Document 2: Kasumi specification, 2007.

[206] M. Izadi, B. Sadeghiyan, S.S. Sadeghian, and H.A. Khanooki, “MIBS: a new lightweight block cipher,” Cryptology and Network Security (CANS), Springer, LNCS, 5888, pp. 334-348, 2009.

[207] R. Beaulieu, S. Douglas, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK Families of Lightweight Block Ciphers,” IACR Cryptology ePrint Archive, 404, 2013.

[208] F. Karakoc, H. Demirci, and A.E. Harmanci, “ITUbee: A Software Oriented Lightweight Block Cipher,” Lightweight Cryptography for Security and Privacy, Springer, LNCS, 8162, pp. 16-27, 2013.

[209] M. Kumar, S. K. Pal, and A. Panigrahi, “FeW: a lightweight block cipher,” IACR Cryptology ePrint Archive, 2014.

[210] V. Grosso, G. Laurent, F.-X. Standaert, and K. Varici, “LS-Designs: Bitslice Encryption for efficient Masked Software Implementations,” Fast Software Encryption, FSE 2014, Springer, LNCS, 8540, 2014.

[211] S. S. M. Aldabbagh, I. F. T. A. Shaikhli, and M. A. Alahmad, “HISEC: A New Lightweight Block Cipher Algorithm,” International Conference on Security of Information and Networks (SIN’14), Glasgow, Scotland, UK, pp. 151-157, 2014.

[212] X. Lai and J. L. Massey, “A proposal for a new block encryption standard,” Advances in Cryptology – EUROCRYPT '90, Springer, LNCS, 473, pp. 389-404, 1991.

[213] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, “LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors,” International Workshop on Information Security Applications (WISA 2013), Springer, LNCS, 8267, pp. 3-27, 2014.

[214] J. Jacob, “BEST-1: A Light Weight Block Cipher,” IOSR Journal of Computer Engineering (IOSR-JCE), vol. 16, issue 2, ver. XII, pp. 91-95, 2014.

[215] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel, “A practical attack on Keeloq,” Advances in Cryptology - EUROCRYPT 2008, Springer, LNCS, 4965, pp. 1-18, 2008.

[216] C. DE Canniere, O. Dunkelman, and M. Knezevic, “KATAN & KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers,” Cryptographic Hardware and Embedded Systems, CHES 2009, Springer, LNCS, 5747, pp. 272-288, 2009.

[217] S. Das, “Halka: a lightweight, software friendly block cipher using ultra-lightweight 8-bit S-box,” IACR Cryptology ePrint Archive, 2014.

[218] D. Engels, X. Fan, G. Gong, H. Hu, and E.M. Smith, “Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices,” Financial Cryptography and Data Security - FC 2010, Springer, LNCS, 6054, pp. 3-18, 2010.

[219] G. Bansod, N. Raval, and N. Pisharoty, “Implementation of a New Lightweight Encryption Design for Embedded Security,” IEEE Transactions on Information Forensics and Security, IEEE, vol. 10, issue 1, pp. 142-151, 2014.

[220] D. Canright, “A very compact S-Box for AES,” Cryptographic Hardware and Embedded Systems, CHES 2005, Springer, LNCS, 3659, pp. 441-455, 2005.

[221] A. Bogdanov, D. Khovratovich, and C. Rechbergerm, “Biclique Cryptanalysis of the full AES,” ASIACRYPT 2011, Springer, LNCS, 7073, pp. 344-371, 2011.

[222] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen, “The NOEKEON Block Cipher,” pp.1-30, 2000, http://gro.noekeon.org/ .

[223] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen, “On Noekeon, no!,” 2001, http://gro.noekeon.org/ .

[224] T. Plos, C. Dobraunig, M. Hofinger, A. Oprisnik, C. Wiesmeier, and J. Wiesmeier, “Compact hardware implementation of the block ciphers mCrypton, NOEKEON, and SEA,” Progress in Cryptology – INDOCRYPT 2012, Springer, LNCS, 7668, pp. 358-377, 2012.

[225] Y. Sun, M. Wang, S. Jiang, and Q. Sun, “Differential Cryptanalysis of Reduced-Round ICEBERG,” AFRICACRYPT 2012, Springer, LNCS, 7374, pp. 155-171, 2012.

[226] P. Junod, “On the Complexity of Matsui’s Attack,” Selected Areas in Cryptography (SAC’01), Springer, LNCS, 2259, pp. 199-211, 2001.

[227] S. Rinne, T. Eisenbarth, and C. Paar, “Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers,” Software Performance Enhancement for Encryption and Decryption (SPEED 2007), Amsterdam, NL, 2007.

[228] Y. Yu, Y. Yang, Y. Fan, and H. Min, “Security scheme for RFID tags,” Auto-ID Labs, Fudan University, White paper, 2006.

[229] J. Kelsey, B. Schneier, and D. Wagner, “Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, newDES, RC2, and TEA,” ICICS'97, Springer-Verlag, pp. 233-246, 1997.

[230] V.A. Reddy, “A Cryptanalysis of the Tiny Encryption Algorithm,” University of Alabama, 2003.

[231] J.-P. Kaps, “Chai-tea, cryptographic hardware implementations of xtea,” Progress in Cryptology– INDOCRYPT 2008, Springer, LNCS, 5365, pp. 363-375, 2008.

[232] J. Lu, “Related-key rectangle attack on 36 rounds of the XTEA block cipher,” International Journal of Information Security, 8(1), pp. 1-11, 2008.

[233] E. Yarrkov, “Cryptanalysis of XXTEA,” IACR Cryptology ePrint Archive, 254, 2010.

[234] A. Satoh and S. Morioka, “Hardware-focused performance comparison for the standard block ciphers AES, Camellia, and Triple-DES,” Information Security, Springer, LNCS, 2851, pp. 252-266, 2003.

[235] X. Zhao, T. Wang, and Y. Zheng, “Cache Timing Attacks on Camellia Block Cipher,” IACR Cryptology ePrint Archive, 2009.

[236] O. Tigli, “Area efficient ASIC implementation of IDEA (International Data Encryption Standard),” Best design for ASIC implementation of IDEA, GMU 2003.

[237] S. Mukherjee and B. Sahoo, “A Survey on Hardware Implementation of IDEA Cryptosystem,” Information Security Journal: A Global Perspective, 20(4-5), pp. 210-218, 2011.

[238] D. Khovratovich, G. Leurent, and C. Rechberger, “Narrow-Bicliques: Cryptanalysis of Full IDEA,” EUROCRYPT 2012, Springer, LNCS, 7237, pp. 392-410, 2012.

[239] C.H. Lim, “A Revised Version of CRYPTON: CRYPTON V1.0,” Fast Software Encryption, FSE 1999, Springer, LNCS, 1636, pp. 31-45, 1999.

[240] J.H. Park, “Security analysis of mCrypton proper to low-cost ubiquitous computing devices and applications,” International Journal of Communication Systems, 22(8), pp. 959-969, 2009.

[241] M. Renauld and F.-X. Standaert, “Algebraic side-channel attacks,” IACR Cryptology ePrint Archive, 279, 2009.

[242] L. Yang, M. Wang, and S. Qiao, "Side channel cube attack on PRESENT," Cryptology and Network Security (CANS), Springer, LNCS, 5888, pp. 379-391, 2009.

[243] O. Ozen, K. Varici, C. Tezcan, and C. Kocair, "Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT," Information Security and Privacy, Springer, LNCS, 5594, pp. 90-107, 2009.

[244] K. Jeong, Y. Lee, J. Sung, and S. Hong, "Improved differential fault analysis on PRESENT-80/128," International Journal of Computer Mathematics, Taylor & Francis, vol. 90, issue 12, pp. 2553-2563, 2013.

[245] K. Jeong, H. C. Kang, C. Lee, J. Sung, and S. Hong, "Biclique cryptanalysis of lightweight block ciphers present, piccolo and led," IACR Cryptology ePrint Archive, 2012.

[246] C. Blondeau and K. Nyberg, "Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities," EUROCRYPT 2014, Springer, LNCS, 8441, pp. 165-182, 2014.

[247] H. Cheng, H.M. Heys, and C. Wang, "PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems," 11th EUROMICRO Conference on Digital System Design Architectures - DSD 2008, Methods and Tools, Parma, Italy, pp. 383-390, 2008.

[248] G. Leander, "On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN," EUROCRYPT 2011, Springer, LNCS, 6632, pp. 303-322, 2011.

[249] C. Blondeau and B. Gerard, "Differential Cryptanalysis of PUFFIN and PUFFIN2," Workshop on Lightweight Cryptography, ECRYPT, 2011.

[250] J.-P. Aumasson, M. Naya-Plasencia, and M.-J. O. Saarinen, "Practical attack on 8 rounds of the lightweight block cipher klein," Progress in Cryptology – INDOCRYPT 2011, Springer, LNCS, 7107, pp. 134-145, 2011.

[251] Z. Ahmaadian, M. Salmasizadeh, and M. R. Aref, "Biclique cryptanalysis of the full-round KLEIN block cipher," IET Information Security, IET, pp. 8, 2015.

[252] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," Advances in Cryptology – CRYPTO 2011, Springer, LNCS, 6841, pp. 222-239, 2011.

[253] M. J. B. Robshaw, "Searching for Compact Algorithms: CGEN," Progress in Cryptology – VIETCRYPT 2006, Springer, LNCS, 4341, pp. 37-49, 2006.

[254] E. Biham, "New types of cryptanalytic attacks using related keys," Journal of Cryptology, 7(4), 1994.

[255] G. Zhao, R. Li, L. Cheng, C. Li, and B. Sun, "Differential fault analysis on LED using Super-Sbox," IET Information Security, IET, pp. 10, 2014.

[256] EPCGLOBAL, "EPC Tag Data Standard Version 1.5 EPCglobal Specification," 2010.

[257] Y. Lee, K. Jeong, C. Lee, J. Sung, and S. Hong, "Related-key Cryptanalysis on the Full PRINTcipher Suitable for IC-Printing," International Journal of Distributed Sensor Networks, Hindawi, vol. 2014, article ID 389476, pages 10, 2014.

[258] M. Walter, S. Bulygin, and J. Buchmann, "Optimizing Guessing Strategies for Algebraic Cryptanalysis with Applications to EPCBC," Information Security and Cryptology, Springer, LNCS, 7763, pp. 175-197, 2013.

[259] F. Mace, F.-X. Standaert, and J. Quisquater, "ASIC implementations of the block cipher sea for constrained applications," RFID Security (RFIDsec 2007), Malaga, Spain, pp. 103-114, 2007.

[260] C. Texcan, "The Improbable Differential Attack: Cryptanalysis of Reduced Round CLEFIA," INDOCRYPT 2010, Springer, LNCS, 6498, pp. 197-209, 2010.

[261] E. Biham, O. Dunkelman, and N. Keller, "A Related-Key Rectangle Attack on the Full KASUMI," Advances in Cryptology – ASIACRYPT 2005, Springer, LNCS, 3788, pp. 443-461, 2005.

[262] A. Bay, J.N. Jr, and S. Vaudenay, "Cryptanalysis of reduced-round MIBS Block Cipher," Cryptology and Network Security (CANS), Springer, LNCS, 6467(5005), pp. 1-19, 2010.

[263] B. Su, W. Wu, L. Zhang, and Y. Li, "Full-round differential attack on TWIS block cipher," Information Security Applications, Springer, LNCS, 6513, pp. 234-242, 2010.

[264] T. Isobe, "A single-key attack on the full GOST block cipher," Fast Software Encryption, FSE 2011, Springer, LNCS, 6733, pp. 290-305, 2011.

[265] N.T. Courtois, "An Improved Differential Attack on Full GOST," IACR Cryptology ePrint Archive, 138, 2012.

[266] Y. Wang, W. Wu, X. Yu, and L. Zhang, "Security on lblock against biclique cryptanalysis," Information Security Applications (WISA 2012), Springer, LNCS, 7690, pp. 1-14, 2012.

[267] K. Jeong, C. Lee, and J. I. Lim, "Improved differential fault analysis on lightweight block cipher LBlock for wireless sensor networks," EURASIP Journal on Wireless Communications and Networking (JWCN), EURASIP, 2013/1/151, 2013.

[268] H. Yoshikawa, M. Kaminaga, A. Shikoda, and T. Suzuki, "Secret key reconstruction method using round addition DFA on lightweight block cipher LBlock," International Symposium on Information Theory and its Applications (ISITA), Melbourne, VIC, pp. 493-496, 2014.

[269] J. Huand, S. Vaudenay, and X. Lai, "On the Key Schedule of Lightweight Block Ciphers," Progress in Cryptology– INDOCRYPT 2014, Springer, LNCS, 8885, pp. 124-142, 2014.

[270] J. Song, K. Lee, and H. Lee, "Biclique cryptanalysis on lightweight block cipher: HIGHT and Piccolo," International Journal of Computer Mathematics, Taylor \& Francis, vol. 90, issue 12, pp. 2564-2580, 2013.

[271] S. A. Azimi, Z. Ahmadian, J. Mohajeri, and M. R. Aref, "Impossible differential cryptanalysis of Piccolo lightweight block cipher," International ISC Conference on Information Security and Cryptology (ISCISC), Tehran, September, pp. 89-94, 2014.

[272] D. Hong et al., "HIGHT: A New Block Cipher Suitable for Low-Resource Device," Cryptographic Hardware and Embedded Systems, CHES 2006, Springer, LNCS, 4249, pp. 46-59, 2006.

[273] Y.-I. Lim, J.-H. Lee, Y. You, and K.-R. Cho, "Implementation of HIGHT cryptic circuit for RFID tag," IEICE Electronics Express, 6(4), pp. 180-186, 2009.

[274] B. Koo, D. Hong, and D. Kwon, "Related-key attack on the full HIGHT," Information Security and Cryptology, ICISC 2010, Springer, LNCS, 6829, pp. 49-67, 2011.

[275] L. Wen, M. Wang, A. Bogdanov, and H. Chen, "Multidimensional zero-correlation attacks on lightweight block cipher HIGHT: Improved cryptanalysis of an ISO standard," Information Processing Letters, Elsevier, 114, pp. 322-330, 2014.

[276] N. Mentens, J. Genoe, B. Preneel, and I. Verbauwhede, "A low-cost implementation of Trivium," SASC, pp. 197-204, 2008.

[277] B. Zhu and G. Gong, "Multidimensional Meet-in-the-Middle Attack and Its Applications to KATAN32/48/64," Cryptography and Communications, Springer, vol. 6, issue 4, pp. 313-333, 2014.

[278] M. Agren, “Some instant- and practical-time related-key attacks on KTANTAN32/48/64,” 18th International Conference on Selected Areas in Cryptography (SAC’11), Miri, A. and Vaudenay, S, (Eds.), Springer, pp. 213-229, 2011.
[279] M-J. O. Saarinen, “Cryptanalysis of Hummingbird-1,” Fast Software Encryption (FSE 2011), Springer, LNCS, 6733, pp.328-341, 2011.
[280] M-J. O. Saarinen, “Related-key attacks against full Hummingbird-2,” Fast Software Encryption (FSE 2014), Springer, LNCS, 8424, pp. 467-482, 2014.
[281] Standard, NIST FIPS, “Data Encryption Standard (DES),” Federal Information Processing Standards Publication, 46-3, 1999.
[282] L.R. Knudsen and H. Raddum, “On Noekeon,” Public reports of the NESSIE project, Report: NES/DOC/UIB/WP3/009/1, 2001.
[283] T. De Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, “Masking AES with d+1 Shares in Hardware,” Cryptographic Hardware and Embedded Systems (CHES 2016), Springer, LNCS, 9813, pp. 192-212, 2016.
[284] G. Leander, B. Minaud, and S. Ronjom, “A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin,” iSCREAM and Zorro, EUROCRYPT 2015, IACR, Sofia, Bulgaria, 26-30 April, 2015.
[285] L. Batina, A. Das, B. Ege, E. B. Kavun, N. Mentens, C. Paar, I. Verbauwhede, and T. Yalcin, “Dietary recommendations for lightweight block ciphers power, energy and area analysis of recently developed architectures,” IACR Cryptology ePrint Archive, 2013.
[286] J. Jean, I. Nikoli, T. Peyrin, L. Wang, and S. Wu, “Security Analysis of PRINCE,” Fast Software Encryption, FSE 2013, Springer, LNCS, 8424, pp. 92-111, 2014.
[287] H. Soleimany et al., “Reflection cryptanalysis of PRINCE-like ciphers,” Journal of Cryptology, Springer, 13 December, 2013.
[288] G. Zhao, B. Sun, C. Li, and J. Su, “Truncated differential cryptanalysis of PRINCE,” Security and Communication Networks, Wiley, 2015.
[289] H.A. Alkhzaimi and M.M. Lauridsen, “Cryptanalysis of the SIMON family of block ciphers,” IACR Cryptology ePrint Archive, 2013.
[290] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, “Differential Fault Analysis on the families of SIMON and SPECK ciphers,” IACR Cryptology ePrint Archive, 2014.
[291] R. Rabbaninejad, Z. Ahmadian, M. Salmasizadeh, and M. R. Aref, “Cube and dynamic cube attacks on SIMON32/64,” International ISC Conference on Information Security and Cryptology (ISCISC), Tehran, pp. 98-103, 2014.
[292] M. Matsui, “New Block Encryption Algorithm MISTY,” Fast Software Encryption (FSE 1997) Springer, LNCS, 1267, pp. 54-68, 1997.
[293] M. Ullrich, C. D. Canniere, S. Indesteege, O. Kucuk, N. Mouha, and B. Preneel, “Finding Optimal Bitsliced Implementations of 4×4-bit S-boxes,” Symmetric Key Encryption Workshop (SKEW), Copenhagen, DK, 2011.
[294] F. Abed, E. List, S. Lucks, and J. Wenzel, “Cryptanalysis of the SPECK family of block ciphers,” IACR Cryptology ePrint Archive, 2013.
[295] D. Lee, D.-C. Kim, D. Kwon, and H. Kim, “Efficient Hardware Implementation of the Lightweight Block Encryption Algorithm LEA,” Sensors, Open Access Journal, MDPI, vol. 14, pp. 975-994, 2014.
[296] Y. Kim and H. Yoon, “First Experimental Result of Power Analysis Attacks on a FPGA Implementation of LEA,” IACR Cryptology ePrint Archive, 999, 2014.
[297] L. Batina, J. Lano, N. Mentens, S. B. Ors, B. Preneel, I. Verbauwhede, “Energy, performance, area versus security trade-offs for stream ciphers,” The State of the Art of Stream Ciphers: Workshop Record, pp. 302-310, 2004.

[298] T. Eisenbarth, C. Paar, A. Poschmann, S. Kumar, L. Uhsadel, “A Survey of Lightweight Cryptography – Implementations,” IEEE Design and Test of Computers, IEEE, vol. 24, issue 6, pp. 522-533, 2007.

[299] N. Fournel, M. Minier, S. Ubeda, “Survey and benchmark of stream ciphers for wireless sensor networks,” IFIP WISTP 2007, Springer, LNCS, vol 4462, pp. 202-214, 2007.

[300] T. Good and M. Benaissa, “Hardware Results for Selected Stream Cipher Candidates,” SASC 2007, Bochum, Germany, pp. 191-204, 2007.

[301] M. U. Bokhari, S. Alam, F. S. Massodi, “Cryptanalysis Techniques for Stream Cipher: A Survey,” International Journal of Computer Applications, vol. 60, no. 9, pp. 29-33, 2012.

[302] G. Banegas, “Attacks in stream ciphers: a survey,” IACR, eprint, article 677, 2014.

[303] A. Klein, “Introduction to stream ciphers, Stream Ciphers,” Springer, pp. 1-13, 2013.

[304] S. M. Bellovin, “Frank Miller: Inventor of the One-Time Pad,” Cryptologia, vol. 35, issue 3, pp. 203-222, 2011.

[305] C. E. Shannon, “Communication theory of secrecy systems,” Bell system technical journal, vol. 28, issue 4, pp. 656-715, 1949.

[306] J. Daemen, J. Lano, and B. Preneel, “Chosen Ciphertext Attack on SSS,” eStream Report, article 2005/044, 2005.

[307] E. KAasper, V. Rijmen, T. E. Bjorstad, C. Rechberger, M. Robshaw, G. Sekar, “Correlated Keystreams in Moustique,” AFRICACRYPT 2008, Springer, LNCS, vol. 5023, pp. 246-257, 2008.

[308] A. Canteaut et. al, “Ongoing research areas in symmetric cryptography,” ECRYPT Information Society Technologies, Technical report D.STVL.4, 2006.

[309] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, H. Sibert, “SOSEMANUK, a Fast Software-Oriented Stream Cipher,” New Stream Cipher Designs, Springer, LNCS, vol. 4986, pp. 98-118, 2008.

[310] Y. Luo, Q. Chai, G. Gong, X. Lai, “A Lightweight Stream Cipher WG-7 for RFID Encryption and Authentication,” IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1-6, 2010.

[311] X. Fan, K. Mandal, G. Gong, “WG-8 A Lightweight Stream Cipher for Resource-Constrained Smart Devices,” 9th International Conference QShine 2013, Greader Noida, India, January 11-12, Springer, LNCS, vol. 115, pp. 617-632, 2013.

[312] A. Klapper, “A survey of feedback with carry shift registers,” SETA 2004, Springer, LNCS, vol. 3486, pp. 56-71, 2005.

[313] N. Kumar, S. Ojha, K. Jain, S. Lal, “BEAN: a Lightweight Stream Cipher,” 2nd International conference on Security of Information and Networks (SIN ‘09), pp. 168-171, 2009.

[314] G. Paul, S. Maitra, “RC4 stream cipher and its variants,” Discrete Mathematics and Its Applications, CRC press, 2011.

[315] D. J. Bernstein. “The Salsa20 family of stream ciphers,” CR.YP.TO, 2007, http://cr.yp.to/snuffle/salsafamily-20071225.pdf .

[316] B. Zhang, Z. Shi, C. Xu, Y. Yao, Z. Li, Sablier v1, “CAESAR competition,” 2014, http://competitions.cr.yp.to/round1/sablierv1.pdf .

[317] H. Wu, “The Stream Cipher HC-128,” New Stream Cipher Designs – The eSTREAM Finalists, Springer, LNCS, vol. 4986, pp. 39-47, 2008.

[318] H. Wu, “A New Stream Cipher HC-256,” FSE 2004, Springer, LNCS, vol. 3017, pp. 226-244, 2004.

[319] S. Wolfram, "Cryptography with cellular automata," Crypto-85, Springer, LNCS, vol. 218, pp. 429-432, 1986.

[320] K. Sandip, M. Debdeep, C. D. Roy, "CAvium – Strengthening Trivium Stream Cipher Using Cellular Automata," Journal of Cellular Automata, vol. 7, issue 2, pp. 179-197, 2012.

[321] S. Das, D. Roy Chowdhury, "CAR30: A New Scalable Stream Cipher with rule 30," Cryptography and Communications, vol. 5, issue 2, pp. 137-162, 2013.

[322] A. N. Pisarchik, M. Zanin, "Chaotic maps cryptography and security, Encryption: Methods, Software and Security," Nova Science Publishers, pp. 1-28, 2010.

[323] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, O. Scavenius, "Rabbit: A New High-Performance Stream Cipher," FSE 2003, Springer, LNCS, vol. 2887, pp. 307-329, 2003.

[324] M. Rosen-Zvi, E. K. Ido Kanter, W. Kinzel, "Mutual learning in a tree parity machine and its application to cryptography," Physical Review E, vol. 66, issue 6, pp. 066135-066148, 2002.

[325] T. Chen, L. Ge, X. Wang, C. Jiamei, "TinyStream: A Lightweight and Novel Stream Cipher Scheme for Wireless Sensor Networks," CIS 2010, pp. 528-532, 2010.

[326] M. Hell, T. Johansson, W. Meier, "Grain – a Stream Cipher for Constrained Environments," International Journal of Wireless and Mobile Computing, vol. 2, No 1/2007, pp. 86-93, 2007.

[327] S. Babbage, M. Dodd, "The Stream Cipher MICKEY 2.0," eStream Project, 2006, http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf .

[328] M. David, D. C. Ranasinghe, T. Larsen, "A2U2: A Stream Cipher for Printed Electronics RFID Tags," IEEE International Conference on RFID, pp. 176-183, 2011.

[329] G. Gong, A. Youssef, "Cryptographic properties of the Welch-Gong transformation sequence generators," IEEE Transactions on Information Theory, vol. 48, no. 11, pp. 2837-2846, 2002.

[330] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, H. Furuichi, "Enocoro-80: A Hardware Oriented Stream Cipher," Third International Conference on Availability, Reliability and Security (ARES 08), vol. 1294, no. 1300, pp. 4-7, 2008.

[331] Systems Development Laboratory, Hitachi, "Enocoro-128v2: A Hardware Oriented Stream Cipher," Hitachi Ltd., 2009, http://www.hitachi.com/rd/yrl/crypto/enocoro/enocoro_spec_20100222.zip .

[332] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou, C. E. Goutis, "Comparison of the hardware architecture and FPGA implementations of stream ciphers," 11th IEEE International Conference on Electronis, Circuitsand Systems (ICECS 2004), IEEE, pp. 571-574, 2004.

[333] G. Paul, S. Rathi, S. Maitra, "On Non-negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key," Designs, Codes and Cryptography, vol. 49, No 1-3, pp. 123-134, 2008.

[334] G. Paul and S. Maitra, "Permutation after RC4 Key Scheduling Reveals the Secret Key," Selected Areas in Cryptography (SAC), Springer, LNCS, vol. 4876, pp. 360-377, 2007.

[335] M. Akgun, P. Kavak, H. Demirci, "New Results on the Key Scheduling Algorithm of RC4," INDOCRYPT, Springer, LNCS, vol. 5365, pp. 40-52, 2008.

[336] A. Klein, "Attacks on the RC4 stream cipher," Designs, Codes and Cryptography, vol. 48, No 3, pp. 269-286, 2008.

[337] T. Erik, R.-P. Weinmann, A. Pyshkin, "Breaking 104 bit WEP in less than 60 seconds," Information Security Applications, Springer, LNCS, vol. 4867, pp. 188-202, 2007.

[338] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, J. C. Schuldt, "On the Security of RC4 in TLS," USENIX Security, pp. 305-320, 2013.

[339] A. Biryukov, A. Shamir, D. Wagner, "Real Time Cryptanalysis of A5/1 on a PC," Fast Software Encryption (FSE), Springer, LNCS, vol. 1978, pp. 1-18, 2001.

[340] P. Ekdahl and T. Johansson, "Another attack on A5/1," IEEE Transactions on Information Theory, vol. 49, issue 1, pp. 284-289, 2003.

[341] E. Barkan and E. Biham, "Conditional estimators: An effective attack on A5/1," Selected Areas in Cryptography, Springer, LNCS, vol. 3897, pp. 1-19, 2006.

[342] J. Shah and A. Mahalanobis, "A new guess-and-determine attack on the A5/1 stream cipher," Cryptology ePrint Archive, IACR, report 2012/208, 2012.

[343] E. Barkan, E. Biham, N. Keller, "Instant ciphertext-only cryptanalysis of GSM encrypted communication," Advances in Cryptology – CRYPTO 2003, Springer, LNCS, vol. 2729, pp. 600-616, 2003.

[344] T. Guneysu, T. Kasper, M. Novotny, C. Paar, A. Rupp, "Cryptanalysis with COPACOBANA," IEEE Transactions on Computers, vol. 57, issue 11, pp. 1498-1513, 2008.

[345] K. Nohl and C. Paget, "GSM: SRSLY," 26th Chaos Communication Congress (26C3), pp. 21-49, 2009.

[346] M. Hermelin and K. Nyberg, "Correlation properties of the Bluetooth combiner," Information Security and Cryptology – ICISC'99, Springer, LNCS, vol. 1787, pp. 17-29, 2000.

[347] S. R. Fluhrer, "Improved key recovery of level 1 of the Bluetooth encryption system," Foundations of Cryptography – Basic Tools, Cambridge University Press, 2002.

[348] Y. Lu and S. Vaudenay, "Faster correlation attack on Bluetooth keystream generator E0," Advances in Cryptology – CRYPTO 2004, Springer, LNCS, vol. 3152, pp. 407-425, 2004.

[349] Y. Lu, W. Meier, S. Vaudenay, "The conditional correlation attack: A practical attack on bluetooth encryption," Advances in Cryptology – CRYPTO 2005, Springer, LNCS, vol. 3621, pp. 97-117, (2005.

[350] NIST Special Publication 800-38A, "Recommendation for block cipher modes of operation – methods and techniques," NIST, 2001, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf .

[351] A. Bogdanov, D. Khovratovich, and C. Rechbergerm, "Biclique Cryptanalysis of the full AES," ASIACRYPT 2011, Springer, LNCS, vol. 7073, pp. 344-371, 2011.

[352] ECRYPT, "Stream Cipher Project," ECRYPT, 2004-2008, http://www.ecrypt.eu.org/stream .

[353] D. J. Bernstein, "The Salsa20 family of stream ciphers," CR.YP.TO, 2007, http://cr.yp.to/snuffle/salsafamily-20071225.pdf .

[354] C. De Canniere, B. Prenel, "Trivium Specifications," eStream Project, 2008, http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf .

[355] W. Dai, "Crypto++ Library 5.6.2," Crypto++, 2013, http://www.cryptopp.com .

[356] M. Subhamoy, P. Goutam, M. Willi, "Salsa20 cryptanalysis: new moves and revisiting old styles," 9th International Workshop on Coding and Cryptography (WCC 2015), Paris, France, 2015.

[357] wolfSSL, "CyaSSL Embedded SSL Library," wolfSSL Inc., 2014, http://www.yassl.com/yaSSL/Products-cyassl.html .

[358] ISO/IEC 18033-4:2011, "International standard for IT Security techniques," ISO/IEC, 2011, http://webstore.iec.ch/preview/info_isoiec18033-4%7Bed2.0%7Den.pdf .

[359] M. Boesgaard, M. Vesterager, J. Christiansen, E. Zenner, “The Stream Cipher Rabbit 1,” eStream Project, 2007, http://www.ecrypt.eu.org/stream/p3ciphers/rabbit/rabbit_p3.pdf .

[360] A. Kircanski and A. M. Youssef, “Differential fault analysis of Rabbit,” Selected Areas in Cryptography, Springer, LNCS, vol. 5867, pp. 197-214, 2009.

[361] A. Kircanski and A. M. Youssef, “Differential fault analysis of HC-128,” AFRICACRYPT 2010, Springer, LNCS, vol. 6055, pp. 261-278, 2010.

[362] P. Stankovski, S. Ruj, M. Hell, T. Johansson, “Improved distinguishers for HC-128,” Design, Codes and Cryptography, Springer, vol. 63, issue 2, pp. 225-240, 2012.

[363] P. Ekdahl, T. Johansson, “A New Version of the Stream Cipher SNOW,” SAC 2002, Springer, LNCS, vol. 2595, pp. 47-61, 2003.

[364] R. Anderson, E. Biham, L. Knudsen, “Serpent: A Proposal for the Advanced Encryption Standard,” AES contest, 1998, http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf .

[365] Z. Ma and D. GU, “Improved differential fault analysis of SOSEMANUK,” 8th International Conference on Computational Intelligence and Security (CIS), IEEE, pp. 487-491, 2012.

[366] D. Otte, “AVR-Crypto-Lib,” AVR-Crypto-Lib, 2009, http://www.das-labor.org/wiki/AVR-Crypto-Lib/en .

[367] M. Hell, T. Johansson, A. Maximov, “A Stream Cipher Proposal: Grain-128,” IEEE International Symposium on Information Theory, Seattle, WA, pp. 1614-1618, 2006.

[368] A. Shamir, “SQUASH – A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags,” FSE 2008, Springer, LNCS, vol. 5086, pp. 144-157, 2008.

[369] S. Sarkar, S. Banik, S. Maitra, “Differential Fault Attack against Grain family with very few faults and minimal assumptions,” Cryptology ePrint Archive, IACR, report 2013/494, 2013.

[370] N. Mentens, J. Genoe, B. Preneel, I. Verbauwhede, “A Low-Cost Implementation of Trivium,” SASC 2008, Lausanne, Switzerland, pp. 197-204, 2008.

[371] M. S. E. Mohamed, S. Bulygin, J. Buchmann, “Improved Differential Fault Analysis of Trivium,” COSADE 2011, Darmstadt, Germany, pp. 147-158, 2011.

[372] T. Good, M. Benaissa, “Hardware Performance of eStream Phase-III Stream Cipher Candidates,” SASC 2008, Lausanne, Switzerland, pp. 163-174, 2008.

[373] S. Banik, S. Maitra, S. Sarkar, “Improved Differential Fault Attack on MICKEY 2.0,” Cryptology ePrint Archive, IACR, report 2013/029, 2013.

[374] L. Ding, J. Guan, “Cryptanalysis of MICKEY family of stream ciphers,” Security and Communication Networks, Wiley, vol. 6, issue 8, pp. 936-941, 2013.

[375] S. Babbage, M. Dodd, “The Stream Cipher MICKEY-128 2.0,” eStream Project, 2006, http://www.ecrypt.eu.org/stream/p2ciphers/mickey128/mickey128_p2.pdf .

[376] ISO/IEC 29192-3:2012, “International standard for lightweight cryptographic methods,” ISO/IEC, 2012, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56426 .

[377] M Hell and T. Johansson, “Security evaluation of stream cipher Enocoro-128v2,” CRYPTEC Technical Report, No. 2008, 2010.

[378] M. Agren, “On Some Symmetric Lightweight Cryptographic Designs,” Doctoral Thesis, Department of Electrical and Information Technology, Faculty of Engineering, LTH, Lund University, 2012.

[379] Y. Tian, G. Chen, J. Li, “Quavium - A New Stream Cipher Inspired by Trivium,” Journal of Computers, vol 7, No 5, pp. 1278-1283, 2012.

[380] Q. Chai, X. Fan, G. Gong, "An Ultra-Efficient Key Recovery Attack on the Lightweight Stream Cipher A2U2," Cryptology ePrint Archive, IACR, 2011/247, 2011.

[381] M. A. Orumiehchiha, J. Pieprzyk, R. Steinfeld, "Cryptanalysis of WG-7: A Lightweight Stream Cipher," Cryptography and Communications, vol. 4, issue 3-4, pp. 277-285, 2012.

[382] L. Ding, C. Jin, J. Guan, Q. Wang, "Cryptanalysis of lightweight WG-8 stream cipher," IEEE Transactions on Information Forensics and Security, IEEE, vol. 9, issue 4, pp. 645-652, 2014.

[383] NIST Special Publication 800-22 A, "Statistical test suit for random and pseudorandom number generators for cryptographic applications," NIST, 2010, http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf .

[384] C. Karlof, N. Sastry, D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," SenSys '04, pp. 162-175, 2004.

[385] J. Walker, "ENT - A Pseudorandom Number Sequence Test Program," ENT, 2008, http://www.fourmilab.ch/random/ .

[386] ISO/IEC 18000-6, "International standard for parameters for air interface communications at 860 MHz to 960 MHz," ISO/IEC, 2013, http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=59644 .

[387] S. Banik, S. Maitra, S. Sarkar, "A Differential Fault Attack on Grain-128a using MACs," 2nd International Conference SPACE 2012, Chennai, India, Springer, LNCS, pp. 111-125, 2012.

[388] R. Tahir, Y. Javed, A. R. Cheema, "Rabbit-MAC: Lightweight Authenticated Encryption in Wireless Sensor Networks," IEEE International Conference on Information and Automation, Zhangjiajie, China, pp. 573-577, 2008.

[389] M. Bellare, C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm," ASIACRYPT 2000, Springer, LNCS, vol. 1976, pp. 531-545, 2000.

[390] H. Wu, "ACORN: A Lightweight Authenticated Cipher," CAESAR competition, 2014, http://competitions.cr.yp.to/round1/acornv1.pdf .

[391] Intel Software Network Rev 3.01, "AES Instructions Set, Intel," 2008, https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf .

[392] M.-J. O. Saarinen, "BRUTUS: identifying cryptanalytic weaknesses in CAESAR first round candidates," Cryptology ePrint Archive, IACR, report 2014/850, 2014.

[393] X. Feng and F. Zhang, "A practical state recovery attack on the stream cipher Sablier," 8th International Conference on Network and System Security (NSS 2014), Springer, LNCS, pp. 198-208, 2014.

[394] G. Jakimoski and S. Khajuria, "ASC-1: An Authenticated Encryption Stream Cipher," Selected Areas in Cryptography 2012, Springer, LNCS, vol. 7118, pp. 356-372, 2012.

[395] A.Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, E. Tischhauser, "ALE: AES-Based Lightweight Authenticated Encryption," FSE'13, Springer, LNCS, 2013.

[396] A. Birykov, "A New 128-bit Key Stream Cipher LEX, eStream Project," 2005, http://www.ecrypt.eu.org/stream/ciphers/lex/lex.pdf .

[397] D. Khovratovich and C. Rechberger, "The LOCAL attack: Cryptanalysis of the authenticated encryption scheme ALE," Cryptology ePrint Archive, IACR, report 2013/357, 2013.

[398] S. Wu, H. Wu, T. Huang, M. Wang, W. Wu, "Leaked-State-Forgery Attack against the Authenticated Encryption Algorithm ALE," ASIACRYPT 2013, Springer, LNCS, vol. 8269, pp. 377-404, 2013.

[399] K. Ayesha, B. Deblin, P. Goutam, C. Anupam, "Optimized GPU implementation and performance analysis of HC series of stream ciphers," Information Security and Cryptology (ICISC 2012), Springer, LNCS, vol. 7839, pp. 293-308, 2013.

[400] G. Meiser, T. Eisenbarth, K. Lemke-Rust, C. Paar, "Software implementation of eSTREAM profile I ciphers on embedded 8-bit AVR microcontrollers," Workshop Record State of the Art of Stream Ciphers (SASC 07), 2007.

[401] M. Feldhofe and C. Rechberger, "A Case Against Currently Used Hash Functions in RFID Protocols," OTM 2006 Workshops, R. Meersman, Z. Tari, P. Herrero (eds.), Springer, LNCS, vol. 4277, pp. 372-381, 2006.

[402] M. O'Neill, "Low-Cost SHA-1 Hash Function Architecture for RFID Tags," RFIDSec 2008, S. Dominikus, M. Aigner, (eds.), 2008, http://events.iaik.tugraz.at/RFIDSec08/Papers/ .

[403] S. Badel, N. Dagtekin, J. Nakahara, K. Ouafi, N. Reffe, P. Sepehrdad, P. Susil, S. Vaudenay, "ARMADILLO: A Multi-Purpose Cryptographic Primitive Dedicated to Hardware," CHES 2010, S. Mangard, F.-X. Standaert, (eds.), Springer, LNCS, vol. 6225, pp. 398–412, 2010.

[404] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, "Hash Functions and RFID Tags: Mind the Gap," CHES'08 Proceedings of the 10th International workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 283-299, 2008.

[405] SHA-3 Contest, http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html .

[406] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, M. U. Sharif, "Comprehensive Evaluation of High-Speed and Medium Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs," The 3rd SHA-3 Candidate Conference, Washington, D. C., March 22-23, 2012.

[407] E. B. Kavun, T. Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications," Proceedings of the 6th Workshop on RFID Security (RFIDSec'11), Istanbul, Turkey, June 7-9, 2010

[408] A. Shamir, "SQUASH – A New MAC with Provable Security Properties for Highly Constrained Devices such as RFID Tags," FSE 2008, K. Nyberg, (ed.), Springer, LNCS, vol. 5086, pp. 144-157, 2008.

[409] T. P. Berger, J. D'Hayer, K. Marquet, M. Minier, G. Thomas, "The GLUON Family: A Lightweight Hash Function Family Based on FCSRs," AFRICACRYPT 2012, A. Mitrokotsa, and S. Vaudenay (eds.), Springer, LNCS, vol. 7374, pp. 306-323, 2012.

[410] J.-P. Aumasson, L. Henzen, W. Meier, M. Naya-Plasencia, "QUARK: A".

[411] J. Guo, T. Peyrin, A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," CRYPTO 2011, P. Rogaway (ed.), Springer, LNCS, vol. 6841, pp. 222-239, 2011.

[412] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, I. Verbauwhede, SPONGENT: A Lightweight Hash Function," CHES 2011, B. Preneel and T. Takagi (eds.), Springer, LNCS, vol. 6917, pp. 312-325, 2011.

[413] Y. Oren and M. Feldhofer, "WIPR – A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes," WISEC, D. A. Basin, S. Capkun, W. Lee (eds.), ACM, pp. 59-68, 2009.

[414] D. Hein, J. Wolkerstorfer, N. Felber, "ECC is ready for RFID – a Proof in Silicon," Selected Areas In Cryptography, Springer, LNCS, vol. 5381/2009, pp. 401-413, 2009.

[415] R. Roman, C. Alcaraz, J. Lopez, "A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes," Journal of Mobile Networks and Applications, vol. 12, issue 4, August, 2007.

[416] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, 31, 4, 469-472, 1985.

[417] M.-J. O. Saarinen, "The BlueJay Ultra-Lightweight Hybrid Cryptosystem," IEEE Symposium on Security and Privacy Workshops (SPW), pp. 27-3, May 24-25, 2012.

[418] N. Nizamuddin, S. A. Ch, W. Nasar, Q. Javaid, "Efficient Signcryption Schemes based on Hyperlliptic Curve Cryptosystem," 7th International Conference on Emerging Technologies (ICET), pp. 1-4, 2011.

[419] T. Guneysu, S. Heyse, C. Paar, "The Future of High-Speed Cryptography: New Computing Platforms and New Ciphers," Proceedings of the 21st edition of the Great Lakes Symposium on VLSI (GLSVLSI'11), 2011.

[420] S. Rohde, T. Eisenbarth, E. Dahmen, J. Buchmann, C. Paar, "Fast Hash-Based Signatures on Constrained Devices," Proceedings of the 8th Smart Card Research and Advanced Application IFIP Conference – CARDIS 2008, September 8-11, 2008.

[421] X. Shen, Z. Du, R. Chen, "Research on NTRU Algorithm for Mobile Java Security," International Conference on Scalable Computing and Communications; The Eighth International Conference on Embedded Computing 2009, SCALCOM-EMBEDDEDCOM'09, pp 366-369, 2009.

[422] A. Shoufan, T. Wink, G. Molter, S. Huss, F. Strentzke, "A Novel Processor Architecture for McEliece Cryptosystem and FPGA Platforms," Proceedings of the 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2009), pp. 98-105, 2009.

[423] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, J.-M. Chen, "Implementing Minimized Multivariate PKC on Low-resource Embedded Systems," Proceedings of the Third international conference on Security in Pervasive Computing (SPC'06), J. A. Clark, R. F. Paige, F. C. Polack, P. J. Brooke (eds.), Springer, pp. 73-88, 2006.

[424] A. A. Kamal, A. M. Youssef, "An FPGA Implementation of the NTRUEncrypt Cryptosystem," International Conference on Microelectronics (ICM), pp. 209-212, 2009.

[425] N. Howgrave-Graham, J. H. Silverman, W. Whyte, "Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3," Proceedings of the 2005 International conference on Topics in Cryptology (CT-RSA'05), A. Menezes, (ed.), Springer, pp. 118-135, 2005.

[426] wolfSSL Inc., "CyaSSL embedded ssl library," http://yassl.com/yaSSL/Products-cyassl.html .

[427] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, C. Manifavas, "A Review of Lightweight Block Ciphers," Journal of Cryptographic Engineering, Springer, vol. 7, pp. 1-44, 2017.

APPENDIX A – SURVEY AND EVALUATION OF LIGHTWEIGHT CRYPTOGRAPHIC PRIMITIVES ON EMBEDDED SYSTEMS

Pervasive computing is a growing trend for where devices with embedded microprocessors are used to enhance various aspects of our everyday lives (e.g. [12], [141], [142], [102]). Such devices operate on limited resources and therefore, data processing, communication protocols and underlying technologies must be carefully chosen to meet strict operating requirements. Considering, however, that the information they handle is in many cases private or safety critical and must be appropriately protected from malicious attackers, appliance of secure cryptographic components becomes imperative. Lightweight cryptography (LWC) investigates the integration of cryptographic primitives and algorithms into constrained devices [9], [10].

Between the two main categories of cryptographic algorithms, symmetric- and asymmetric-key, the former exhibit good performance and are widely used for confidentiality, integrity checks and authentication protocols. The tested robustness and known levels of security of well-known block ciphers, like AES [154], make them good candidates as long as they can be adjusted to meet the corresponding resource constraints. AES is the standard symmetric-key cipher used in cryptographic applications. Newer block ciphers designed specifically for this domain are gaining ground, introducing novelties and improving the overall efficiency.

In the following subsections, a survey of lightweight cryptography for embedded systems is presented as well as their implementations (in hardware and software). Similar surveys on LWC were first carried out in 2007. In [155] and [82], the authors evaluate hardware and software implementations for lightweight symmetric and asymmetric cryptography. In [156], the authors investigate the lightweight hardware and software solutions for Wireless Sensor Networks (WSNs), i.e. groups of highly-constrained hardware platforms. In [157], the authors report new trends for lightweight hardware block and stream ciphers. In [158], hardware implementations of block ciphers are examined while in [159] and [160] the authors implement and evaluate lightweight block ciphers on the same platform for fair comparison. Software implementations of lightweight block ciphers on three different platforms are presented in [161]. The latest survey on cryptanalytic attacks on lightweight block ciphers was carried out in [162].

This appendix is a comprehensive survey in LWC and includes recently proposed ciphers. We further present the latest trends in hardware/software implementations of lightweight block ciphers and summarize the state-of-the-art design directions. The latest cryptanalysis results are mentioned and the vulnerable ciphers are reported. We analyze the features of block ciphers and identify the more suitable ciphers for different types of embedded devices. Based on the devices' capabilities, we categorize the implementations in three groups: ultra-lightweight, low-cost and lightweight. Ultra-lightweight implementations fit in the most constrained devices (in terms of computation capability, memory, power), like the standard 8051 microcontroller and the ATtiny45. Low-cost devices are affordable and perform a little better than ultra-lightweight ones, like the ATmega128. Lightweight devices include the remaining devices that are reported in LWC.

Hardware-based algorithm implementations are categorized based on chip area and complexity. Ultra-lightweight implementations occupy up to 1000 logic gates, low-cost implementations occupy up to 2000 logic gates and lightweight implementations occupy up to 3000 logic gates. The software implementations are categorized based on the ROM and RAM requirements.

Ultra-lightweight implementations require up to 4KB ROM and 256 bytes RAM, low-cost implementations require up to 4KB ROM and 8KB RAM and lightweight implementations require up to 32KB ROM and 8KB RAM. Finally we record the implementation characteristics of the latest proposals in hardware/software and create benchmarks based on metrics.

A.1 Brief Overview

Embedded devices have inherent limitations in terms of processing power, memory, storage, connectivity and energy consumption [163], [164]. While ordinary cryptography focuses in providing high levels of security, lightweight cryptography also needs to consider the aforementioned restrictions, which can affect design and implementation choices.

Hardware LWC implementations try to achieve the required functionality with the minimum amount of hardware real-estate. These designs are suitable for ultra-constrained devices that perform specific tasks. The efficiency depends on a number of aspects analyzed below, like, design complexity, CMOS technology, power and energy consumption, and throughput.

The implementation's complexity is one of the most dominant factors that affect the design approach. It is determined by the logic gates that are required to implement a cipher. The relevant metric is called Gate Equivalent (GE). GE is a unit of measurement that specifies the manufacturing-technology-independent complexity of digital electronic circuits. In CMOS technologies, the silicon area of a NAND2 gate usually constitutes the technology-dependent unit area commonly referred to as gate equivalent. Except for expressing complexity, the number of the logic gates or the GE metric reflect to a portion of the chip area that is occupied by the hardware implementation. Thus, they are usually utilized as an intuitive way to express the chip area of a design. A specification in GE for a certain circuit represents a complexity measure, for which the corresponding silicon area can be deduced for a dedicated manufacturing technology. The authors in [165] and [166] suggest that approximately 250-4000 logic gates, out of the total 1000-10000 logic gates commonly present on RFID tag spaces, can be used for security related tasks. For lightweight devices, up to 3000GE can be acceptable while for even smaller devices, like low-cost RFIDs and 4-bit microcontrollers, 2000GE and 1000GE are proposed respectively [427].

CMOS technology is another factor that affects the implementation characteristics. Different technologies and standard cell libraries produce different results. For example, the same implementation of PRESENT that is presented in [167] produces 1075GE on 0.18 μm, 1169GE on 0.25μm and 1000GE on 0.35 μm CMOS technology.

Software implementations on the other hand target less constrained devices. One of the main goals of software implementations is to keep memory and CPU needs as low as possible in order to minimize power consumption and the compatible devices' cost.

Software implementations are optimized for throughput and energy savings by requiring fewer cycles, since voltage and clock frequency are usually fixed for microcontrollers. These implementations are included in cryptographic libraries for embedded systems [63], [168].

Memory restrictions, however, are bound to negatively affect performance. As small memory elements are utilized, more cycles are needed to execute an operation. While in hardware implementations the 4-bit S-boxes are a popular option ([169], [97], [98], [170], [171], [172],

[173], [174], [175], [176]) in software, the use of 8-bit S-boxes has also been proposed ([169], [177], [178], [160], [179], [180]). The reduction in cycles count is considered significant despite the use of more space.

For active devices, like wireless sensor nodes that have their own power supply, energy consumption is a significant aspect. For passive devices, like contactless smart cards and RFID tags that do not have their own power supply and must adapt to the host device's constraints, power consumption is the main concern.

Power is an important as it is related to two main issues: the power consumption of a device and attacks related to power analysis. When frequency is fixed at a low value, like 100 kHz that is usually examined in LWC research [427], the power consumption is directly connected to chip area [181]. A small area predisposes that the circuit will consume low power. Table 19 summarizes the GE and power characteristics of standard-cell libraries [182]. Power deviates by a factor of two to three across different technologies. Table 20 summarizes the general frequency, RAM, ROM, and power characteristics of different microcontroller platforms in the sensor network market [156].

Table 19 Characteristics of different CMOS technology modes for commercial standard-cell libraries

CMOS Technology Node (μm)	Gate Density (kGEs/mm^2)	Power (nW/MHz/GE)
0.35	6	18
0.18	125	15
0.13	206	10
0.09	403	7
0.065	800	5.68

Researchers aim to make the power usage profile of their implementations independent of the secret key, in order to withstand simple and differential power analysis ([183], [184], [185]). As a result, the power consumption is increased. In applications where Differential Power Analysis (DPA) [186] is a threat, researchers attempt to counter DPA while trying to respect the power constraints. In less critical applications, power consumption is considered a priority while the countermeasures for DPA are less important.

Table 20 Characteristics of different microcontroller platforms

Microcontroller platform	Frequency (MHz)	RAM (KB)	ROM (KB)	Power (mA)
8-bit	4-8	0.064-4	1.4-128	2.2-8
16-bit	4-8	2-10	48-60	1.5-2
32-bit	13-180	256-512	4000-32000	31-100

Timing requirements are also imperative for deploying several special domain applications [176], [187] and ISO/IEC standard protocols [98]. The ideal lightweight cipher should occupy a small circuit area, consume the least amount of resources and power, and perform as fast as possible.

In order to fairly compare different ciphers several factors should uphold:

- The compared implementations need to provide the same security level.
- For hardware, the benchmarking set-up, including the CMOS library, the synthesis scripts, the synthesis tools need to be fixed and not to favor any of the competing ciphers.
- Similarly, the same benchmarking machines need to be applied for software.
- There are plenty of implementation choices (e.g. serial and round-based) and the designs are optimized for specific evaluation metrics. This may lead to different architectures for different metrics.
- The reported results should ideally be reproducible by the wider community.

In order to fairly compare different ciphers, the aforementioned factors should all be considered, as it is difficult to implement all the proposals on the same platform.

On the same issue, the authors in [188] studied the impact of technology and standard cell libraries in the hardware implementation of hash functions. They conclude that accurate comparisons between different hardware implementations can be achieved when the same technology has been used, even if different standard cell libraries are used. On the other hand, significant inaccuracy could be noticed when using different technologies. Similar observations are mentioned for all other reported features, like performance and power consumption. The authors of said work also note that only latency is independent of technology, while power and area depend on technology. Throughput may depend on technology when the maximum throughput is needed, as the maximum frequency depends on the technology being used. Similar remarks can be deduced for software implementations.

A comparative study is conducted in the following subsections. The ciphers with low security level are initially indicated to draw the reader's attention. In order to provide a fair comparison, the different implementations are initially grouped, based on the deployed technology in hardware and software. Also, the implementations with the worst features (e.g. high energy consumption or low throughput) are excluded. Among the efficient ciphers, the implementations of the same technology with the same key and data-path size are retrieved and compared.

Then, inter-technology evaluation is performed based on a subset of ciphers that are implemented in more than one technologies. For example, in hardware TWINE is only implemented in 0.09μm and RECTANGLE only in 0.13μm. PRESENT is implemented in both platforms. Based on the aforementioned analysis, PRESENT is more efficient than TWINE in 0.09μm and PRESENT is less efficient than RECTANGLE in 0.13μm. Thus, we derive that in general RECTANGLE is also more efficient than TWINE.

A.1.1 Evaluation Metrics

Several lightweight ciphers are compared based on a number of metrics presented in this subsection. Specifically, security, cost, and performance features are investigated. The best ciphers are those who provide the appropriate level of security and balance the tradeoffs between cost and performance. Some metrics are generic while others are tied to the hardware implementations (e.g. hardware technology and GE) and some to software implementations (e.g. RAM/ROM size). The metrics used for the comparison presented in this paper are the following:

- **Security level:** It is a logarithmic measure of the fastest known computational attack on an algorithm and is measured in bits. In most cases the level is identified by the key length in bits. The level of security cannot exceed the key length but it can be smaller. Initially, a cipher's developers estimate its security level, which can be updated accordingly, based on reported attacks.
- **Hardware technology:** It is the CMOS technology used to implement the cipher with the corresponding occupied area being measured in μm. The technologies most commonly used in LWC research papers are 0.13μm and 0.18μm [427]. The complexity and the area occupied by the hardware implementation are described by the Gate Equivalent (GE) metric and depend on the technology used. The GE metric is calculated by dividing the layout area of an implementation in μm^2 by the corresponding area of a NAND2 gate.
- **Throughput:** It stands for the Kb/s the cipher's encryption/decryption operation achieves at a specific frequency. Most of the research papers in LWC utilize frequencies of 100 KHz for hardware and 4 MHz for software. We record the throughput of each evaluated implementation. In cases where an implementation uses a different frequency, the reported throughput is projected at these two frequencies.
- **Latency:** It defines the number of clock cycles required to compute a single block's plaintext/ciphertext.
- **Power and energy consumption:** We evaluate the power of hardware implementations in μW. The power can be roughly estimated based on the GE and the hardware technology details referred to in Table 19. For software implementations, we consider an average power of 0.004μW and 0.00135μW for 8- and 16-bit microcontrollers in 4MHz frequency with 0.9V voltage respectively, based on Table 20. Energy consumption per bit for both hardware and software implementations is calculated by the formula:

$$Energy\ [\mu J] = \frac{Latency \left[\frac{cycles}{block}\right] \times Power\ [\mu W]}{Block\ Size [bits]} \qquad \text{(A1)}$$

  Where *latency* is the number of clock cycles that are required to encrypt a block, *power* is the μW that are consumed by the hardware or software implementation and *block size* is the size of data in bits that each cipher can process in one encryption/decryption operation.
- **RAM/ROM memory:** The amount of RAM and ROM (in bytes) the algorithm requires. RAM bytes represent the resources required to store the intermediate state. ROM bytes represent the code size which is actually the size of the implementation.
- **Efficiency:** Indicates the trade-off between performance and implementation size. The higher the value, the better. For hardware implementations, the metric is calculated by the formula:

$$Hardware\ Efficiency = \frac{Throughput\ [Kbps]}{Complexity\ [KGE]} \qquad \text{(A2)}$$

Where *throughput* is the Kb/s the cipher's encryption operation achieves at 100 KHz frequency and the *complexity* is the value of the chip area and complexity in KGE. For software implementations, the metric is calculated by the formula:

$$Software\ Efficiency = \frac{Throughput\ [Kbps]}{Code\ Size\ [KB]} \qquad \text{(A3)}$$

Where *throughput* is the Kb/s the cipher's encryption operation achieves at 4 MHz frequency and *code size* is the size of the executable code in KB.

A.1.2 Lightweight Cryptographic Mechanisms

Embedded devices often have inherent limitations in terms of processing power, memory, storage, and energy. The cryptographic functionality that ESs utilize to provide tamper resistant hardware and software security functions has direct impact on the system's:

- **Size:** Memory elements constitute a significant part of the module's surface.
- **Cost:** Directly linked to the surface of the component.
- **Speed:** Optimized code provides results faster.
- **Power Consumption:** The quicker a set of instructions is executed, the quicker the module can return to an idle state or be put in sleep mode where power consumption is minimal.

Traditional cryptography solutions focus in providing high levels of security, ignoring the requirements of constrained devices. Lightweight cryptography (LWC) is a research field that has developed in recent years and focuses in designing schemes for devices with constrained capabilities in power supply, connectivity, hardware, and software. Schemes proposed include hardware designs, which are typically considered more suitable for ultra-constrained devices, as well as software and hybrid implementations for lightweight devices.

- *Hardware designs* implement the exact functionality without redundant components. The main design goal is the reduction of the logic gates that are required to materialize the cipher. A small GE predisposes that the circuit is cheap and consumes little power. For constrained devices an implementation including up to 3000 GE can be considered acceptable while for even smaller devices, like 4-bit microcontrollers, implementations of 1000 GE are being studied [165], [166]. Energy consumption and power constraints are other significant factors. Energy consumption is important when a device is running on batteries while power constraints affect passive devices, like passive RFID tags, that must be connected to a host device to operate. Security attacks and relevant countermeasures that are correlated to power analysis are also considered in hardware designs.
- *Software implementations* typically only require a microprocessor to operate. The main design goals are the reduction of memory and processing requirements of the cipher. Implementations are optimized for throughput and power savings. Portability is their main advantage over hardware implementations.

- *Hybrid schemes* combine the two approaches exploiting the best features from both. Hardware implements the basic cipher functionality and software performs the data and communication manipulation. A common practice is the design of cryptographic coprocessors. The throughput is mostly affected by the communication bandwidth between hardware and software components. Hybrid implementations target on specific communication applications, like RFID tags, portable devices and Internet servers.

A.2 Symmetric Cryptography

Lightweight and ultra-lightweight ciphers usually offer 80 to 128 bit security [166]. 80 bit security is considered adequate for constrained devices [170], like 4-bit microcontrollers and RFID tags, while 128 bits is typical for mainstream applications [154]. For one way authentication, 64 to 80 bit security would suffice [179].

Three main approaches are followed in implementing lightweight ciphers. In the first case, researchers try to improve the performance of well-known and well-studied ciphers such as AES and DES. A state of the art AES [154] hardware implementation uses 2400 GE and is used as a benchmark for newer ciphers. In the second case, researchers design and implement new ciphers, specific for this domain. PRESENT [155] is such an example implemented for lightweight and ultra-lightweight cryptography and is one of the first ciphers that offer a 1000 GE implementation for ultra-constrained devices. In the third case, researchers mix features of several ciphers that are well studied and their individual properties are known.

The absence of decryption is another factor that can reduce the requirements of such ciphers, especially for ultra-lightweight cryptography. Hummingbird-2 [184] is a combination of cipher and protocol and adopts this strategy. This approach is suitable for devices that need only one way authentication. Furthermore, some ciphers like KTANTAN [185] propose that the key should be hardwired on the device to further reduce the GE due to the absence of key generation operations.

A.3 Block Ciphers

A.3.1 General Information

There are five basic types of block ciphers based on their inner structure: **Substitution Permutation Networks (SPNs)**, **Feistel networks**, **Add-Rotate-XOR (ARX)**, **NLFSR-based** and **hybrid**. AES is the best known cipher that adopts the SPN structure, DES is the best known Feistel type cipher, while IDEA the best known ARX cipher, KeeLoq the best known NLFSR-based cipher and the best known hybrid ciphers are those of the Hummingbird family.

SPNs process plaintext through a series of sequential substitution and permutation boxes that transform the data and prepare them for the next round. SPN ciphers include AES [154], NOEKEON [189], ICEBERG [190], mCrypton [191], PRESENT [97], PUFFIN-2 [192], PRINTcipher [187], Klein [170], LED [171], EPCBC [176], PICARO [193], PRINCE [194], Zorro [183], RECTANGLE [195], I-PRESENT^{TM} [196] and PRIDE [197].

Feistel networks perform a diffusion function on half of the data of each block, resulting in a smaller round function. Additional logic is needed to apply the diffusion function to the untransformed state, such as a bitwise XOR operation, which requires 2.5-3 GE per bit. SPNs do not have this extra overhead and, as a result, serialized SPN ciphers are likely to achieve smaller datapaths. Feistel ciphers include DES [281] and its variants [198], GOST [172], TEA [199] and its variants [200], [201], Camellia [202], SEA [203], CLEFIA [204], KASUMI [205], MIBS [206], TWIS [180], TWINE [174], LBlock [175], Piccolo [173], SIMON [207], ITUbee [208], FeW [209], Robin [210], Fantomas [210] and HISEC [211].

SPN and Feistel are the most widely-used types [427]. As will be made clear in the following subsections, many of the proposed lightweight Feistel-type ciphers suffer from security problems, in contrast to SPN-type ones. Still, Feistel networks offer both encryption and decryption with little cost, but in many tag-based applications decryption functionality is rarely required. An encryption-only SPN cipher still remains a strong competitor, if not the choice of preference, for the LWC field.

ARXs use addition, rotation and XORs with no S-boxes. The produce compact and fast implementations but their security properties are not well-studied, especially when compared to SPN and Feistel ciphers. ARX designs are IDEA [212], HIGHT [272], SPECK [207], LEA [213] and BEST-1 [214].

NLFSR-based ciphers utilize the building blocks of stream ciphers. They are mostly used for hardware implementations. The security of their inner components is based on stream cipher analysis. KeeLoq [215], KATAN and KTANTAN family [216] and Halka [217] have a stream cipher structure.

Hybrid ciphers combine the three aforementioned types to improve specific parameters, like throughput. Their analysis is determined by the selected cipher types. The Hummingbird family [218], [98] is a special case with hybrid structure of stream and block cipher. PRESENT-GRP [219] is another hybrid design that combines the PRESENT SPN with the bit permutation of a grouping permutation (GRP) structure.

A.3.2 Chronicles

We categorize the lightweight block cipher proposals in three time periods based on the general LWC features and design goals. The ciphers of each period are grouped based on their type and are referred to in chronological order.

Period 1: early lightweight applications

In 80s and 90s cryptography on mainstream computers was investigated and the first cryptographic standards were established. Embedded systems usage was limited. Lightweight security was provided by compact implementations of mainstream ciphers and some early lightweight proposals. These solutions target specific applications, like GSM security and remote keyless systems. The first attacks exposed the vulnerabilities of the lower security settings and established the cryptanalysis principles in this domain.

SPN ciphers

AES is considered a landmark in traditional cryptography and, thus, cannot be ignored in the context of LWC. It uses 128-bit blocks with 128-, 192-, and 256-bit keys through 10, 12, and 14 rounds respectively. Latest achievements in AES hardware implementations based on a serialized AES S-box [179] use 2400GE and 226 cycles per block. The S-box is based on Canright's research [220], who thoroughly investigated the hardware requirements for the AES S-box. Canright proposes a very compact S-box that is composed of smaller fields. The S-box that is used in the lightweight version is further minimized in area by the replacement of D flip-flops and MUX with scan flip-flops. The scan flip-flops are mainly utilized in the construction of the State and the Key array. Also the area requirements of the control logic are reduced by the replacement of the FSM with a LFSR. The authors also show that row-wise processing is more hardware-friendly than column-wise. The cipher remains safe with biclique cryptanalysis achieving slightly better results than exhaustive search [221].

In [160], a compact software implementation of AES is proposed. Registers are used to store the internal state and the mix column step while the key is stored in RAM. The S-box and the round constants are implemented as look-up tables. Also shift and XOR operations are used for the multiplications performed for mix column. The implementation requires 1659 bytes of ROM and needs 4557 cycles to encrypt a 128-bit block.

NOEKEON [222] uses 128-bit keys and blocks through 16 identical rounds. A related-key cryptanalysis was presented in [282] and as a result the cipher was rejected by the NESSIE project. Later, the designers of NOEKEON argued that the presented attacks were not practical and that the cipher remains safe [223]. The first hardware implementation [224] occupies 2880GE and is suitable for lightweight devices. In software [160], it requires 364 bytes of code for 21.7 Kbps throughput and is suitable for 32-bit processors. It is an early involutive cipher (uses the same datapath for encryption and decryption, allowing the same circuity to be reused for the inverse operation) whose design is explored by newer proposals.

ICEBERG [190] is a fast involutive cipher. It uses 128-bit keys with 64-bit blocks through 16 rounds. ICEBERG is optimized for reconfigurable hardware implementations as it allows changing the key at every clock cycle without any loss in performance and the deriving of the round keys on-the-fly. It produces very efficient combinations of encryption/decryption and requires 5800 gates for 400 Kbps of throughput [178]. Differential cryptanalysis on the 8-round version is the best known attack [225]. The overall design is investigated by newer involutive ciphers to provide low-cost encryption/decryption functionality.

Feistel ciphers

DES is one of the first ciphers to be investigated for LWC. It uses 56-bit keys with 64-bit blocks through 16 rounds. The disadvantage of DES compared to AES is the smaller key size (i.e. 56-bits), yielding a lower security level (also broken under linear cryptanalysis [226]). On the other hand, smaller key sizes allow for smaller inner structures and low area footprint. The **DESX** variant [198] uses key whitening to increase the security level and render brute force attacks impossible; it uses 184-bit keys for the same block size and rounds. Hardware implementations of DES and DESX are presented in [198] and their cost is 2309GE and 1848GE respectively.

Older software implementations are presented in [155]. Newer, more compact software implementations, are presented in [160] and [227].

TEA (Tiny Encryption Algorithm) [199], [228] (2355GE) uses 128-bit keys with 64-bit blocks and 64 rounds. It is notable for its efficiency in power-energy-memory, its simplicity and ease of implementation. TEA needs 648 bytes of code [160], requires 7408 cycles of encryption and does not use S-boxes. The weak points of TEA, identified in [229] and [230], are the equivalent-keys attack and its bad performance as a hash function. **XTEA** (eXtended TEA or Block TEA) [231], [200] was proposed to overcome these weaknesses. Among others, XTEA operates on arbitrary size blocks and utilizes a more complex key scheduling procedure. A related-key rectangle attack on 36 rounds of XTEA is presented in [232]. At a later stage, **XXTEA** (Corrected Block TEA) [201] was proposed, but a chosen-plaintext attack based on differential analysis against the full-round cipher was later presented [233]. Even though these ciphers are designed for software implementations, hardware implementations are reported in [231] for XTEA with 3490GE, which well exceed the 3000GE limit.

Camellia [202] is developed by Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation and is approved by ISO/IEC, IETF, the projects NESSIE and CRYPTREC, and is adopted in Japan's new e-Government Recommended Ciphers List. It became popular as it achieves similar security level and processing capabilities with AES. It uses the same block and key sizes as AES through 18 or 24 rounds. In LWC, it is mainly studied for the fast software implementations as the hardware implementation [234] exceeds the 3000GE bound (at 6511GE). In software [177], Camellia requires 1262 bytes of code and 64 cycles for encryption. Cache timing attacks in software implementations were presented in [235].

ARX ciphers

IDEA (International Data Encryption Algorithm) [212] uses 128-bit keys with 64-bit blocks through 8.5 rounds and all data operations are performed in 16-bit unsigned integers. To reduce the memory overhead, IDEA does not contain S- and P-boxes. It is based on XOR, addition and modular multiplication operations. It has been implemented in hardware for encryption in high-speed networks [236]. However, these designs are not suitable for embedded devices. Yet, IDEA performs well in embedded software and is used in PGP v2.0. Its most compact implementation occupies 596 bytes of code for 94.8 Kbps throughput [237]. Similarly to AES, biclique attacks are slightly faster than exhaustive search [238].

NLFSR ciphers

KeeLoq [215] is widely used in remote keyless entry systems. It was created by Gideon Kuhn in the 80's. The cipher is dedicated to hardware and utilizes a non-linear feedback shift register (NLFSR). It uses 64-bit keys with 32-bit blocks, a 32-bit initialization vector (IV) and process the data for 528 rounds. Practical key recovery based on a slide attack and a novel meet-in-the-middle attack are presented in [215].

Period 2: 1st LWC generation

The first LWC generation covers the years 2005-2012. In this era, embedded systems are gaining ground and adequate general purpose security becomes imperative. The foundations of LWC are set and a high variety of ciphers are designed specifically for this domain. An arm race is held where the area reduction is one of the most targeted design goals. Mostly encryption-only implementations are evaluated, with speed and power optimizations also taking place. PRESENT is the new milestone as the first lightweight cipher that reached the bound of around 1000GE area. Speed and power optimized ciphers are also proposed. Cryptanalysis is evolved as new attacks are efficiently applied in lightweight primitives. The period ends by the formal representation of LWC and the establishment of the ISO/IEC standard for LWC that includes the block ciphers PRESENT and CLEFIA.

SPN ciphers

mCrypton (miniature of Crypton) [191] is a compact edition of **Crypton** [239]. It uses smaller block size (64-bit) through 13 rounds and variable key sizes (64-, 96- and 128-bit) to comply with the new constraints in ubiquitous computing devices. It is designed for low-cost RFID tags and sensors and exhibits low-power and compact implementations in both hardware and software. A related key rectangle attack was reported in [240] for the 8-round mCrypton.

PRESENT [97] meets lightweight and ultra-lightweight requirements. It is one of the first ciphers implemented on ultra-constrained devices with almost 1000GE (encryption-only) [167]. PRESENT is a milestone in the evolution of lightweight block ciphers and is used along with AES as a benchmark for newer proposals. As with CLEFIA, PRESENT is also standardized in ISO/IEC 29192. It uses 80- and 128-bit keys with 64-bit blocks through 31 rounds. It has an SPN structure and needs 1030GE for 80-bit keys of both encryption. The main feature is the replacement of the ordinary eight S-boxes with a carefully selected single S-box. PRESENT is the first cipher that applied this serialized architecture and its design is extremely hardware efficient, since it uses a fully wired diffusion layer without any algebraic unit.

PRESENT software implementations on different platforms have also been studied [181]. The newest software proposal [160] decreases the code size, retains the throughput and performs both encryption and decryption. The code size is 1000 bytes and the algorithm requires 11342/13599 cycles for encryption/decryption on an 8-bit microcontroller. The implementation stores the round key and the state in registers. Two 256-byte tables are used for the encryption and decryption S-boxes to permit parallel lookups, while the code for permutation is utilized in both operations.

On the downside, side-channel attacks [241], [242] and a related-key attack [243] have been reported on the 17 round of PRESENT. Improved differential fault analysis was presented in [244], which recovered the key by inducing two or three 2-byte random faults. Biclique cryptanalysis on the full round cipher are slightly better than exhaustive search [245]. A truncated differential attack on the reduced 26-round cipher was presented in [246].

PUFFIN-2 [192] is based on its predecessor cipher **PUFFIN** (2303GE) [247] (the latter is broken under statistical saturation attacks [248]). It is designed to be implemented exclusively with a serialized architecture and supports the same key and block size as PRESENT through

34 rounds. PUFFIN-2 is faster and has smaller footprint (1083GE), while providing both encryption and decryption functionality; it is an involutive cipher using the same datapath for encryption and decryption. The cipher is resistant to related-key attacks, since at key scheduling the relevant permutation layers are not regularly distributed among rounds. Differential cryptanalysis on the full cipher is slightly better than exhaustive search [249].

Klein [170] uses 64-bit blocks with 64-, 80- and 96-bit keys through 12, 16, and 20 rounds respectively. It targets legacy sensor platforms. The main goal was to achieve high software-based performance while keeping a compact hardware footprint. As, in general, sensors are more powerful than RFID tags, software implementations are more practical. For its inner structure, several choices are made to balance the small area in hardware and software performance. Thus, bit-shifting operations, used by many hardware compact ciphers, are avoided. Byte-oriented matrix multiplication operations are opted for because of their software efficiency on 8-bit processors.

Practical chosen-plaintext key-recovery attacks have been successful for up to 8 rounds of Klein-64 [250]. They exploit the existence of differentials of high probability deriving from the combination of the 4-bit S-box and the byte-oriented MixColumns operation. An asymmetric biclique attack on the full version cipher exploits weaknesses of the diffusion layer and key schedule [251]. The worst case computations are $2^{62.84}$ with 2^{39} data complexity. A modified version of the attack, with higher data requirements, is slightly faster.

LED (Lightweight Encryption Device) [171] is designed to achieve small hardware footprint while keeping reasonable software performance. It uses 64-, 80-, 96- and 128-bit keys with 64-bit blocks through 32 and 48 rounds. It is an AES-like cipher, and the authors apply some newer trends from the field of lightweight hash functions based on block ciphers. LED uses no key scheduling process, the PRESENT S-box, the row-wise processing as described for lightweight AES [179] and the mix column approach of the hash function PHOTON [252]. The absence of key scheduling is also proposed in CGEN [253]. This approach offers significant chip area reduction for hardware implementations but may give rise to serious security issues, like related key attacks [254]. The authors study and address these vulnerabilities. The research for the absence of key scheduling is the main contribution of the cipher, though, as reported, further independent cryptanalysis must be performed. However, the reduced round cipher is vulnerable to biclique cryptanalysis [245]. Differential fault analysis based on Super-Sbox techniques obtain significant improvements for fault attacks [255]. The search space for the 128-bit key exhaustive search is reduced on an average of $2^{21.96}$.

Domain specific SPNs

Domain specific ciphers, like PRINTcipher [187] and EPCBC [176] are implemented to meet the cryptographic requirements of specific applications, such as Integrated Circuit (IC) printing and Electronic Product Code (EPC) encryption respectively. IC-printing is used for the production and personalization of circuits at very low cost. EPC is an industry standard by EPCglobal [256] and is considered as a replacement for barcodes using low-cost passive RFID. In its smallest form, it uses 96-bit keys.

PRINTcipher uses 80- and 160-bit keys with 48- and 96-bit blocks through 48 and 96 rounds respectively. It is developed for both domains, i.e. PRINTcipher-48 (402GE) is designed for

IC-printing applications while PRINTcipher-96 (726GE) for EPC encryption. It uses 3-bit operations. As computer architectures do not use odd number of bits, a software implementation would use redundant resources. It is noted that PRINTcipher was released in order to investigate this application domain and is not ready for actual deployment yet. The authors are not concerned about related-key attacks as they are considered unrealistic for IC printing applications. Such attacks on the full round cipher were presented in [257].

EPCBC is a PRESENT-like cipher that uses 96-bit keys with 48- or 96-bit block size through 32 rounds. The main contribution is the adjustment of an improved PRESENT version with the 96-bit keys aiming to be used for EPC encryption. The authors took the security analysis of PRESENT into consideration. Thus, EPCBC uses an optimized key scheduling procedure that is more secure against related key differential attacks. The most lightweight version of the cipher costs 1008GE and, to our knowledge, it is the most suitable cipher for EPC encryption. Reduced-round versions are vulnerable to practical algebraic cryptanalysis [258].

Ciphers like AES and the original PRESENT support smaller or larger key sizes than 96-bit. Assuming that 96-bit keys are adequate, smaller keys do not offer the required level of security while larger keys produce larger than necessary implementations due to redundant components. Other ciphers that support 96-bit keys are SEA (3758GE), mCrypton (2420GE), Klein (1528GE) and LED (1116GE). The older ciphers, i.e. SEA and mCrypton, produce larger implementations and are not suitable for EPC encryption, although mCrypton is the most efficient.

Feistel ciphers

SEA (Scalable Encryption Algorithm) [203] is designed for scalable software implementations on constrained devices, being parameterized in text, key and processor size. It uses low-cost encryption routines that are suitable for processors with a limited instruction set. The design goals were to meet low memory requirements, small code size and a limited instruction set. Other features are the efficient combination of encryption-decryption, the ability to derive keys on-the-fly and the straightforward implementation in assembly code. Its most compact software implementation on 8-bit microcontrollers [160], requires 426 bytes of code and 41604 cycles for encryption. In [259], the authors propose a hardware implementation where they demonstrate a fully generic VHDL design to achieve the algorithm's scalability. The most lightweight version requires 3758GE.

CLEFIA [169], [204] is another lightweight block cipher, known for its highly efficient hardware and software implementations. It was designed by SONY and is standardized in ISO/IEC 29192. CLEFIA uses 128-bit blocks with 128-, 192- and 256-bit keys through 18, 22, and 26 rounds respectively. The most compact implementation occupies 2488GE (encryption only) for a 128-bit key. It follows a serialized architecture without requiring any additional registers. Decryption can be implemented with an 116GE overhead. The most lightweight encryption/decryption version occupies 2604GE, which is 23% smaller than the corresponding AES-128 version. The authors apply clock gating techniques to reduce the number of multiplexers and increase the cycle counts. Furthermore, they adopt some older ideas utilized in compact AES implementations. For a compact matrix multiplier, operations are computed column by column. Scan flip-flops replace D flip-flops and MUX to further reduce area.

Improbable differential attacks on reduced round versions achieve the best cryptanalysis results and are slightly better than exhaustive search [260].

KASUMI [205] is implemented for cryptography in GSM, UMTS and GPRS. It uses 128-bit keys with 64-bit blocks and processes the data in 8 rounds. A differential-based related-key attack was presented in [261].

Two newer DES and DESX variants, called **DESL** (DES Lightweight) and **DESXL** [198], are also presented for the same key and block sizes and rounds. DESL reduces the gate complexity by replacing the 8 original S-boxes with a single one, thus removing 7 S-boxes and a multiplexer. The single S-box is repeated 8 times and is designed to resist against differential, linear and the Davis-Murphy attacks. It achieves a size reduction of 20%. Also, it utilizes serial hardware techniques to reduce gate complexity. DESXL is the combination of DESL and DESX and features an 18% size reduction compared to DESL. Hardware implementations of DESL and DESXL are presented in [198] and their cost is 2629GE and 2168GE respectively. Software implementations, are presented in [160], [227]. For a compact implementation of DESXL they propose the use of a function which can compute all permutations and expansions depending on the calling parameters, while producing 6-bits outputs as direct input to S-boxes. The permutation and expansion tables are stored and processed from flash memory.

MIBS [206] supports 64- and 80-bit keys with 64-bit blocks through 32 rounds. It has a Feistel structure with an SPN round function, utilizing the S-box of mCrypton [191]. The permutation is operated in nibbles and the F-function operates on half a block. The key scheduling is based on PRESENT. The authors take the side-channel and related key attacks reported for PRESENT into consideration. They secure MIBS by using a round-dependent counter to mix the contents of the key register. However, many types of attacks [262] have come close to a complete compromise, namely linear attacks up to 18 rounds, first ciphertext-only attacks on 13 rounds, differential analysis up to 14 rounds and impossible-differential attacks up to 12 rounds. Although the full cipher is not broken, these attacks reduce its security level by more than 50%.

TWIS [180] is based on the CLEFIA cipher. A differential distinguisher with probability 1 for its full-round version was reported in [263].

Attempting to create a lightweight version of the Soviet **GOST** cipher, authors in [172] managed to produce a 651GE implementation. It uses 256-bit key, 64-bit block size and is a Feistel network with 32 rounds. The main contribution of this proposal is the adaptation of the PRESENT S-box that decreases the GE metric. The authors chose not to use simple wiring for permutation to reduce area, since that would result in weaker differential and linear properties. Recently, vulnerabilities were published and the cipher is theoretically broken by an improved three-subset meet-in-the-middle attack [264]. In [265], the authors further analyze differential attacks on GOST and present a single-key attack which is the fastest attack to our knowledge.

LBlock [175] uses 80-bit keys and 64-bit blocks through 32 rounds. It produces ultra-lightweight implementations in both hardware and software. In hardware, it needs 1320GE for 200 Kbps throughput. In software, assuming an 8-bit microcontroller, it takes 3955 clock cycles to encrypt a single block. For diffusion, the authors chose to use only half of the data in each round, and a simple rotation for the other half. This is applied efficiently on hardware and software. Permutation operations are implemented with no cost of GE in hardware. A 4-bit word-wise permutation is a bitwise permutation, as it can be implemented efficiently in software. Therefore, diffusions and permutations are efficient in both hardware and software. Key scheduling was designed in a stream cipher way.

Biclique cryptanalysis has been performed against the full round LBlock [266]. The complexity of the attack is slightly lower than exhaustive key search. The modification of the key scheduling operation is proposed as a countermeasure to prevent the attack [266]. Improved differential fault attacks were presented in [267] that recover the key in a few seconds to one hour on a general PC. A round addition differential fault analysis reconstructs the key by utilizing one correct and two faulty ciphertexts [268].

TWINE [174] is a 64-bit block cipher with 80- and 128-bit keys through 36 rounds. It produces a compact hardware implementation of 1866GE. It is a GFN and implements a unified encryption and decryption functionality. The authors aim to balance performance on both hardware and software. For this reason, similar choices to LBlock have been made. Still, TWINE is an independent work and presents several concrete design advantages. One of their differences is that LBlock uses ten S-boxes while TWINE uses a single S-box. TWINE uses nibble permutation instead of bit permutation, while LBlock utilizes bit permutation for key scheduling purposes. The authors of TWINE published a saturation attack against a 22-round version of LBlock [174], lowering the security level originally claimed. While TWINE is not the most compact cipher, its performance is well balanced on both hardware and software. However, the simplified key-scheduling process can be exploited by meet-in-the-middle attacks [269].

Piccolo [173] is one of the most lightweight block ciphers. It is a variant of a Generalized Feistel Network (GFN) and supports 64-bit blocks with 80- and 128-bit keys through 25 and 31 rounds respectively. It features very compact implementations which achieve low energy consumption. The 80-bit key version is the most lightweight, requiring 432GE. Piccolo needs an additional 60GE for the decryption functionality. As a result, Piccolo remains more compact even when compared to encryption-only ciphers.

The key scheduling process is implemented by multiplexers without flip-flops (like GOST and KTANTAN) that require a large area. These values (key components) are stored in the key inputs segment, which is not hard-wired, and no registers are used for storing keys. As with other newer ciphers, Piccolo uses scan flip-flops for the data state. Also, the cipher uses AND-NOR and OR-NAND gates, which require 2GE, to perform the same functionality as XOR and XNOR respectively, which require 2.25GE. Moreover, it adopts a half-word based round permutation.

In software [159], it requires 2434 bytes of code and consumes 79 bytes of RAM. It has poor performance and achieves 7.8 Kbps of throughput.

Biclique cryptanalysis has been achieved on full round versions of Piccolo-80, [245], [270]. For Piccolo-128, they perform slightly lower than exhaustive key search. Impossible differential cryptanalysis is also performed on reduced round versions [271].

ARX ciphers

HIGHT [272] uses a compact round function, without the use of S-boxes, and all operations are simple computations. It uses 128-bit keys with 64-bit blocks through 32 rounds. The most compact hardware implementation requires 2608GE for 188Kbps throughput [273]. An impossible differential attack on 26-round HIGHT was presented in [243], a related-key attack on full-round HIGHT was presented in ICISC2010 [274], biclique cryptanalysis has been achieved on full round version [270], and zero-correlation attacks on the 26- and 27-round cipher were presented in [275].

NLFSR ciphers

The **KATAN** and **KTANTAN** [216] cipher family supports 80-bit keys and 32-, 48- and 64-bit blocks through 254 rounds. The ciphers follow the design of KeeLoq but they apply an LFSR (instead of NLFSR) and process the data for less rounds (254 rounds). They use a very simple key scheduling mechanism. KATAN achieves a small footprint of 802GE. KTANTAN authors propose to hard-wire the key on the device to further reduce the GE, due to the absence of key generation operations. KTANTAN is only suitable for devices where the key is initialized once and does not change. The most compact version of KTANTAN achieves 462GE. The authors propose that the version of KTANTAN-48 (588GE) is more suitable for RFID devices. The basic cipher resembles the structure of the stream cipher Trivium [276].

Their software implementations are not efficient since they use bit manipulations extensively. This fact is noted in [160], where a compact software implementation of KATAN is presented. The authors mainly try to decrease the code size. Although the proposal achieves a quite low code size of 338 bytes, it produces low throughput and consumes too much energy. The encryption takes 72963 cycles and the decryption 88525 cycles. Multidimensional meet-in-the-middle attacks on reduced round KATAN that are faster than exhaustive search were presented in [277]. Several related-key attacks which recover the full 80-bit key of KTANTAN with a probability of one are presented in [278].

Hybrid ciphers

Hummingbird (HB) [218] is another promising ultra-lightweight cipher, which introduces a hybrid block and stream cipher structure, with 256-bit key and 16-bit block size through 20 rounds. It features better performance than PRESENT on 4-bit microcontrollers. The first version was found vulnerable to a number of attacks [279].

Its successor, **Hummingbird-2** (HB-2) [98], optionally produces a message authentication code (MAC) for each message processed and is developed with both lightweight software and hardware implementation constraints in mind. It can actually be considered a combination of a cipher and a protocol; its design fulfills the requirements of ISO 18000-6C protocol. HB-2 uses 128-bit secret keys and 64-bit Initialization Vectors (IVs) and, as its predecessor, it has been targeted for low end microcontrollers and hardware implementations in resource-constrained devices such as RFID tags and wireless sensors. HB-2 is broken under a related key attack [280].

Period 3: 2nd LWC generation

This is the current period of LWC's lifetime. Pervasive computing is here and general purpose constrained devices are utilized in everyday activities. The competition for area optimization continues but in a less binding manner. The 1st generation of ciphers achieved a high degree of reduction, where almost 80-90% of the implementation is occupied by memory elements (e.g. the cases of PRESENT and KATAN). New challenges are now considered to meet the real application requirements. Over and above the speed-optimization, low latency is now the objective. Decryption functionality becomes essential and involutive designs gain ground. Cryptanalysis reveals that many lightweight ciphers are vulnerable to side-channel attacks. Masking techniques and ciphers that are easy to mask are proposed. Strategies for designing secure and efficient ciphers are applied (e.g. wide-trail), along with further analysis on the linear and non-linear layers of the inner structures.

SPN ciphers

The authors in [179] try to apply countermeasures against first-order side-channel attacks on AES. They implement the features proposed in [185] and increase the level of resistance. Their implementation occupies 11031GE (0.33μm). A newer implementation [283] requires 6340GE but it is deployed in larger CMOS technology (0.45μm). **PICARO** [192] is a new cipher which is specifically designed to counter such attacks. The authors implement a cipher that fits the masking constraints well, instead of integrating masking schemes to existing ciphers, like the AES proposal. They consider 4 different masking levels. PICARO's masked hardware implementation is faster than the corresponding AES. **Zorro** [183] is a newer proposal that is based on AES and provides efficient masking. It is suitable for embedded systems and it is implemented in 8-bit microcontrollers. Zorro is more efficient than PICARO as it requires less cycles to operate. However, practical invariant subspace attacks are presented in [284].

PRINCE [194] accomplishes low-latency in hardware. It is a notable proposal that applies the wide-trail strategy and uses 128-bit keys with 64-bit blocks through 12 rounds. Its lightweight implementation [285] requires 2953GE for 533.3 Kbps of throughput and has low energy consumption. Efficient software implementations are also presented [197]. Cryptanalysis in [286] and [287] reduced the security level, as attacks were performed on the reduced 6-round and the full 12-round cipher. A key recovery attack on the 7-round version based on truncated differential cryptanalysis is presented in [288].

RECTANGLE [195] is a recent lightweight SPN proposal. It uses 64-bit blocks with 80- and 128-bit keys through 24, 28 and 32 rounds respectively. The substitution is performed by 16 4×4 S-boxes in parallel setting and the permutation is executed in 3 rotations. Its authors propose a new type of S-box along with an asymmetric design of the permutation layer. These new designed criteria are motivated by the cryptanalysis of PRESENT. RECTANGLE adapts bit-slice techniques to LWC and allows lightweight hardware and fast software implementations. In hardware, its most lightweight version with 80-bit keys requires 1467GE for 246 Kbps throughput and consumes the lowest energy/bit. In software, it requires 5.38 cycles/byte for encryption of 1000 byte messages using Intel 128-bit SSE instructions (such implementations are outside the LWC scope).

I-PRESENT^{TM} [196] is an involutive version of PRESENT, using the same key and block sizes through 30 rounds. The S-box layer uses two additional 4×4 S-boxes 16 times. The involutive part is inspired by PRINCE. A ciphertext block is produced after a 15-round function, followed by a 15-round involutive function (30 rounds in total – the original cipher takes 31 rounds). The S-box from NOEKEON is used as the S-box of the involutive function. The key schedule procedure generates 30 64-bit round subkeys. Decryption is identical to encryption, except the fact that the round subkeys are inputted in the reverse order. The most compact hardware implementation requires about 2769GE, providing both encryption and decryption. The relevant encryption-only implementation of PRESENT requires 1570GE.

PRIDE [197] is a software cipher that uses 128-bit keys with 64-bit blocks through 31 rounds. The wide-trail design strategy is adopted and the linear properties of the S-box are further studied. A bit-sliced implementation is presented, with an extremely efficient linear layer. The implementation outperforms many other proposals in 8-bit microcontrollers and exhibits low latency and energy consumption.

Feistel ciphers

SIMON [207] was designed by the NSA. A performance evaluation was presented in [207]. It performs well in software and hardware. It supports several sizes for key (64, 72, 96, 128, 144, 192, 256) and block (32, 48, 64, 96, 128) and round numbers (32, 36, 42, 44, 52, 54, 68, 69, 72). Independent cryptanalysis efforts on SIMON [289] present a series of observations on the cipher's construction and attacks on reduced round versions. Differential fault attacks on SIMON are presented in [290]. Cube and dynamic cube attacks on SIMON, with 64-bit key and 32-bit block, recover the full key in a practical time complexity [291].

ITUbee [208] is a newly proposed cipher, designed for lightweight software and suitable for 8-bit software based platforms. It is based on a Feistel structure and has no key schedule, featuring 80-bit keys with 80-bit blocks. Round-dependent constants are used, instead of strong key scheduling. In its most compact form [208], it requires 586 bytes of code and 2937 cycles for encryption. Self-similarity cryptanalysis is performed in [98] that exploits the round constants and builds relations between them. The reduced round cipher is distinguishable from an ideal random permutation. A deterministic related-key differential distinguisher is presented for the 8-round version in the single-key model, reducing the security by one bit.

FeW [209] is a software oriented design for LWC. It uses 80- and 128-bit keys with 64-bit blocks through 32 rounds. The S-box of Humminbird-2 is utilized in encryption, decryption and key schedule. The key expansion process is similar to PRESENT. FeW combines a Feistel and a generalized Feistel structure to enhance security against basic cryptographic attacks. It is reported as safe against present day cryptanalytic attacks [209].

Robin and **Fantomas** [210] are LS-designs (i.e. a combination of look-up table-based L-boxes and bitslice S-boxes). Both ciphers use 128-bit keys and blocks and are based on the Feistel-type block cipher MISTY [292] (ancestor of KASUMI). They integrate LS-design techniques to provide efficient masking and protection against side-channel attacks. The ciphers use the same constants and target single-key security, excluding related or chosen key attacks. They target efficient and secure software implementations on 8-bit microcontrollers, but are considered to perform well in hardware as well.

Robin is an involutive instance of LS-design, with 16 rounds that utilize the MISTY Class13 [293]. The S-box and L-box are involutive, allowing the same circuity to be reused for the inverse operation (i.e. encryption and decryption). It aims to have similar security margins and performance to NOEKEON. These two bitslice ciphers optimized for Boolean masking are more efficient than AES, Zorro and PICARO. However, practical invariant subspace attacks are presented in [284], similarly with Zorro. A careful tweak of the cipher is proposed to prevent the attacks in case of Robin and the LS-design security is not questioned.

Fantomas is a non-involutive LS-design with 12 rounds that utilizes the MISTY 3/5-bit S-boxes [292]. It illustrates the impact of choosing involutive components with respect to inherent efficiency limits imposed by the LS-designs. Analysis indicates that involutive LS-designs require about 4/3 rounds of non-involutive LS-designs to achieve a similar level of security. Fantomas is safe against the aforementioned attacks on Robin and is also faster.

HISEC [211] is a GFN and exhibits similar characteristics as PRESENT with a different bit-permutation procedure. It uses 80-bit keys with 64-bit blocks and processes the data for 15 rounds. HISEC provides protection against the differential, integral and boomerang attacks that have been presented for other ciphers, like PRESENT, LBlock, Klein and TWINE. The hardware requirements are estimated to 1695GE. No performance details are provided.

ARX ciphers

SPECK [207] is designed by the NSA along with SIMON. As is the case with SIMON, SPECK also performs well in software and hardware. It uses the same key and block sizes through 22, 23, 26, 27, 28, 29, 32, 33, and 34 rounds. SIMON is better in hardware and SPECK is better in software. Independent cryptanalysis efforts on SPECK [294] present a series of observations on the cipher's construction and attacks on reduced versions of the cipher. Differential fault attacks are presented in [290].

BEST-1 (Better Encryption Security Technique-1) [214] is designed for ultra-lightweight implementations with low cost and power on WSN and RFID. It uses 128-bit keys with 64-bit block through 12 rounds and fits in 8-bit processors. It is an ARX proposal and the main operations are mod2^8 addition and subtraction, bitwise shift and XOR. Decryption is performed at little cost as a sequence of steps after inverting the encryption process. In hardware, BEST-1 requires around 2200GE and achieves 265.7Mbps at 80MHz frequency.

LEA (Lightweight block Encryption Algorithm) [213] is a software-oriented ARX designed by the Electronics and Telecommunication Research Institute of Korea. It targets fast encryption on common processors and all operations are 32-bit wide. It uses 128-bit blocks with 128-, 192- and 256-bit keys through 24, 28, and 32 rounds respectively. LEA has small code size and consumes low power. On ARM platforms, it requires 590 bytes of ROM and 32 bytes of RAM for 326.94 cycles per byte speed. The cipher exhibits high hardware demand with the most compact implementation requiring 3826GE for 76.19 Mbps throughput [295]. The security analysis is theoretical. Power analysis on hardware is presented in [296]. The 128-bit key is retrieved by attacking the first round. Implementing LEA with countermeasures becomes essential.

NLFSR ciphers

Halka [217] is a recently proposed cipher. It is designed for lightweight hardware but also remains software friendly. It uses 80-bit keys with 64-bit blocks through 24 rounds. The main feature is the implementation of a novel multiplicative inverse for 8-bit S-boxes using LFSR, which requires 138GE. The relevant feature of other existing standards demand more gates, like the Canright's AES S-box [220] that occupies 253GE. Thus, the total gate count of the Halka should be low. The use of 8- instead of 4-bit S-boxes is considered more secure in this design. Then, eight 8×256 bytes S-boxes are utilized to achieve super-fast software performance. The key schedule is similar to PRESENT but it is materialized with 8-bit S-box. In hardware, Halka is estimated to require 7% less GE than PRESENT but is considered more secure [217]. In software, it could be three times faster than PRESENT [217].

Hybrid ciphers

PRESENT-GRP [219] is a hybrid cipher that combines the SPN of PRESENT with a GRP (group operation) structure for bit permutation. It uses 128-bit keys with 64-bit blocks through 31 rounds. The security properties of GRP are well-studied and offer higher confusion properties than bit permutation. The P-box of the original PRESENT is replaced by GRP. The hybrid cipher with the PRESENT S-box and GRP enhances security and results in lower memory and area consumption. The hardware design requires 2125GE, which is a little larger than the corresponding PRESENT (1884GE). In software, the implementation size is slightly larger than PRESENT for the same amount of RAM.

A.3.3 Results and Discussion

Security

Security issues should be carefully considered before using a cipher. For example KeeLoq, GOST, TEA, XTEA, mCrypton, HIGHT, KASUMI, PUFFIN, Hummingbird, MIBS, KTANTAN, TWIS, PRINTcipher, Humminbird-2, LBlock, Zorro, and LEA all suffer from security vulnerabilities and should be avoided. DES 56-bit key is not adequate for nowadays applications.

Newer 2nd generation proposals, namely PICARO, SPECK, ITUbee, RECTANGLE, FeW, Halka, BEST-1, Robin, Fantomas, HISEC, I-PRESENT™, PRESENT-GRP, and PRIDE have to be further analyzed for vulnerabilities and their claimed level of security. AES, Camellia, PRESENT, CLEFIA, and DES variants are the most studied solutions and as a consequence the most acceptable ciphers. PRESENT and CLEFIA are the standardized block ciphers for lightweight cryptography.

Comparison

We examine 127 hardware and 233 software implementations. We evaluate all implementations based on the metrics and the fair comparison approach detailed in subsection A.1. In hardware, 0.09, 0.13, 0.18, and 0.35μm technologies are used. The software implementations are deployed in 8-, 16-, and 32-bit microcontrollers.

Hardware implementations

Regarding hardware implementations for constrained devices, Figure 41 to Figure 44 illustrate the best hardware implementations according to individual metrics. Implementations are summarized per technology in terms of GE, energy, power and hardware efficiency respectively. Implementations on 0.09μm technology are colored with purple, on 0.13μm technology with deep blue, on 0.18μm technology with blue, and on 0.35μm technology with green. The standardized ciphers that are utilized as comparison units (AES, PRESENT, and CLEFIA) have a yellow color and implementations with excess demands in each metric are outlined with red color. All hardware implementations are detailed in [427].

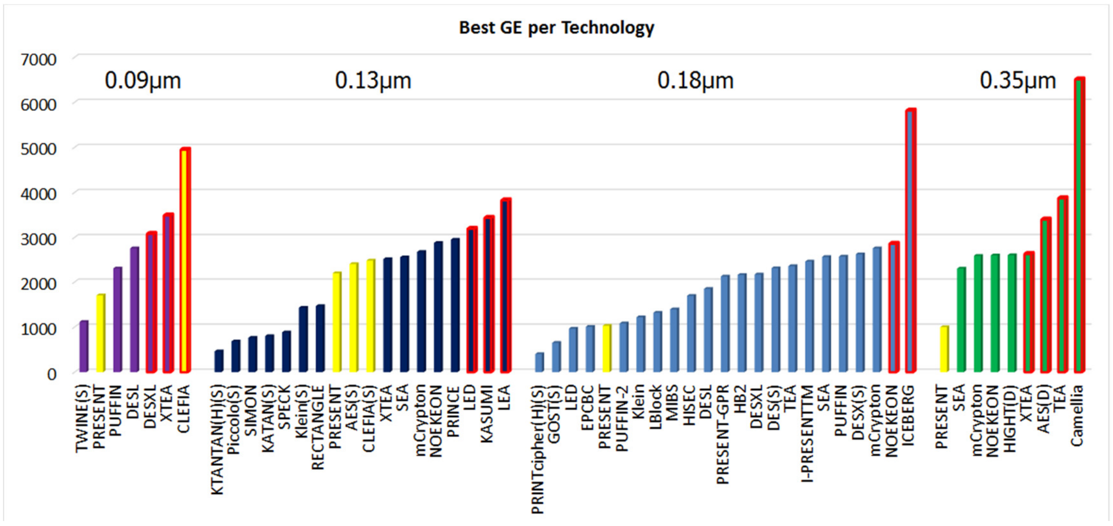


Figure 41 Best hardware implementations of block ciphers in terms of GE

All proposals for the ciphers IDEA, Camellia, ICEBERG, KASUMI, and LEA exceed the boundary of 3000GE and are, therefore, not considered efficient. NOEKEON, PUFFIN-2, LED, and EPCBC consume excessive energy per bit and produce low hardware efficiency. KTANTAN, HIGHT, HB-2, TEA, Klein, SEA, KATAN, AES, DES, DESX, EPCBC, LED, PUFFIN-2, and NOEKEON consume high energy per bit. DESX, DES, AES, KATAN, KTANTAN, EPCBC, PUFFIN-2, and NOEKEON produce high latency (144-3720 cycles per block). NOEKEON, PUFFIN-2, and EPCBC have also low throughput. DES, HB-2, I-PRESENT^{TM}, DESX, and HIGHT have higher power requirements than other ciphers.

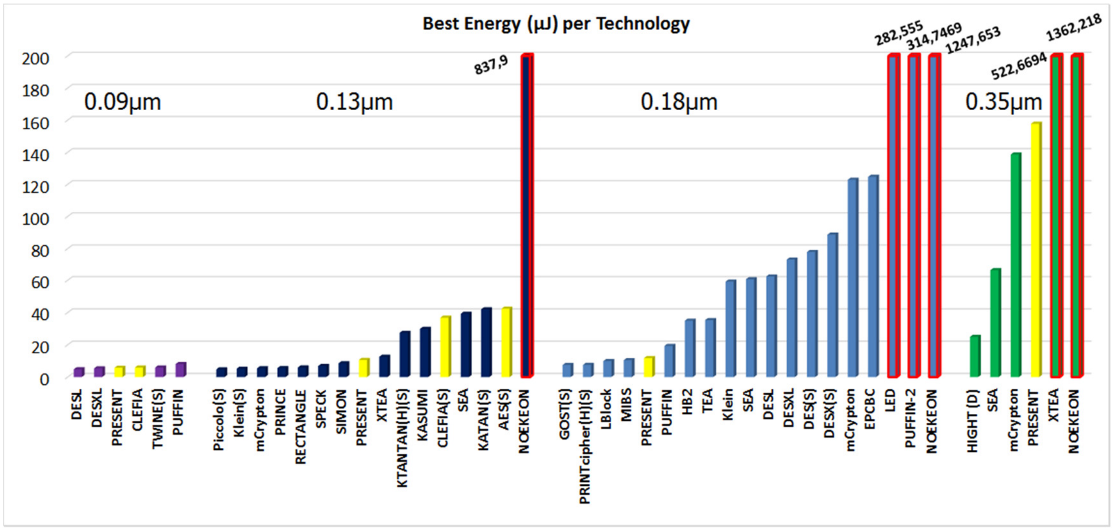


Figure 42 Best hardware implementations of block ciphers in terms of energy consumption (axis y is limited at 200µJ)

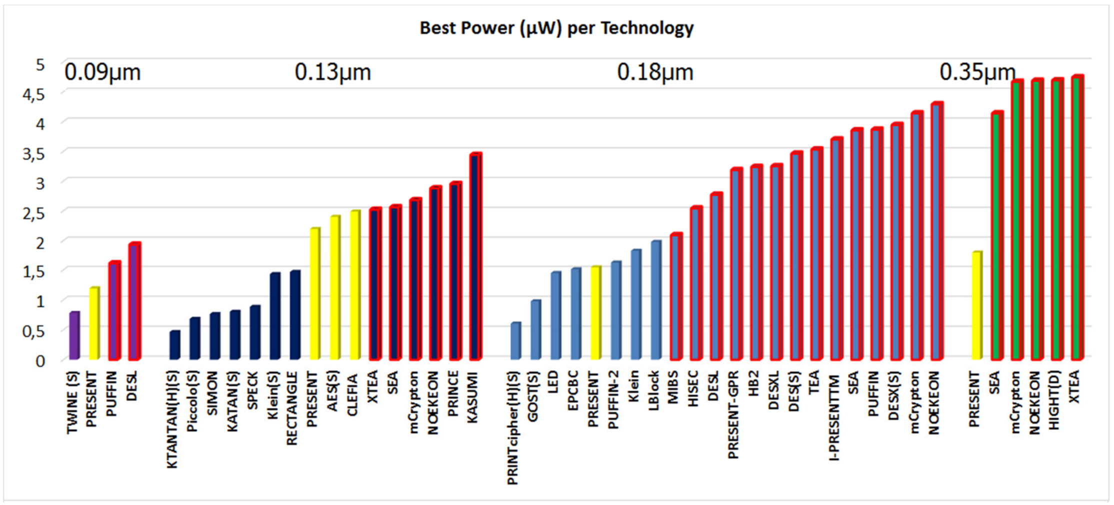


Figure 43 Best hardware implementations of block ciphers in terms of power requirements

KTANTAN, PRINTcipher, Piccolo, SIMON, TWINE, KATAN, SPECK, and GOST produce compact implementations with the lowest power requirements (less than 1µW). Klein, LED, RECTANGLE, EPCBC, PRESENT, PUFFIN, PUFFIN-2, DESL, LBlock, MIBS, AES, CLEFIA, XTEA, HISEC, and SEA consume also low power (around 2.5µW). Piccolo, DESL, Klein, DESXL, mCrypton, PRINCE, PRESENT, CLEFIA, TWINE, RECTANGLE, SPECK, GOST, PRINTcipher, PUFFIN, SIMON, LBlock, and MIBS consume low energy per bit (less than 10 µJ per bit).

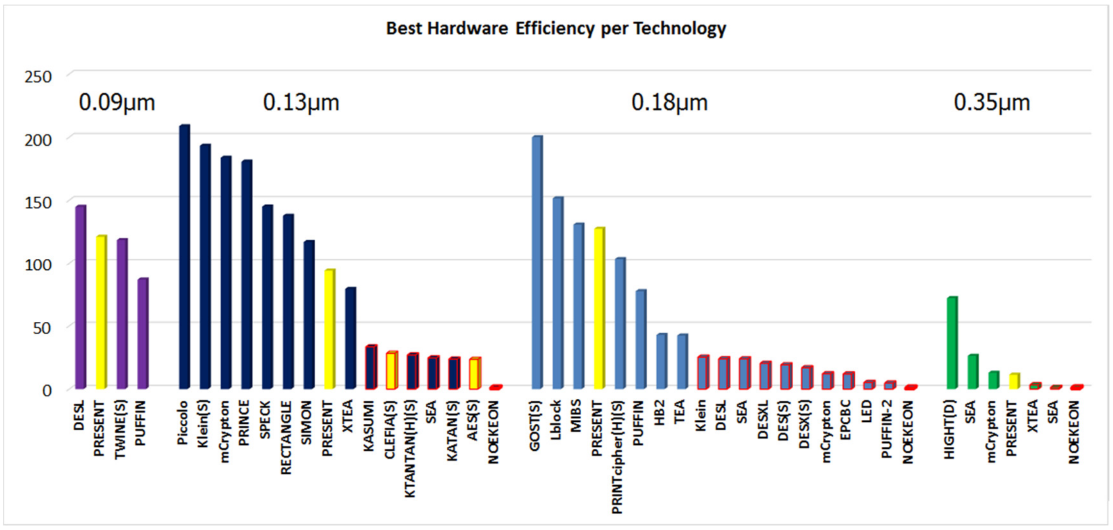


Figure 44 Best hardware implementations of block ciphers in terms of hardware efficiency (Hardware Efficiency = Throughput [Kbps]/Complexity [KGE])

Figure 45 illustrates the best overall hardware implementations. Based on the criteria adopted in this work, Piccolo evaluates best compared to all the other proposals for many key sizes. PRESENT and TWINE perform similarly and achieve a good overall status. TWINE, as a Feistel network, can offer the decryption functionality at low cost. RECTANGLE, PRINCE, SPECK, and mCrypton exhibit low-latency and efficient implementations. SEA also offers low latency but it is quite slower. For lower security, DESL performs well.

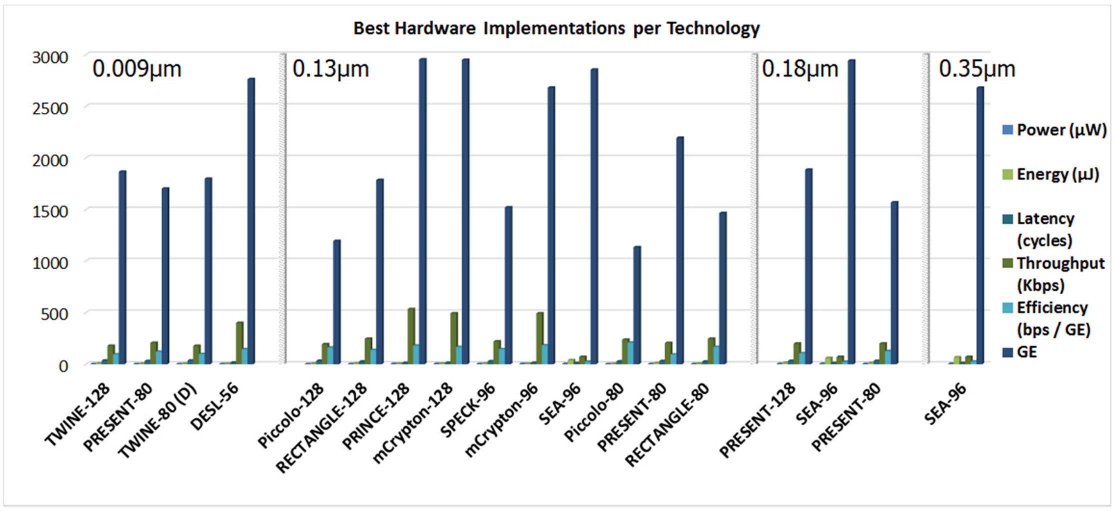


Figure 45 Best overall hardware implementations of block ciphers per technology

Low-cost RFID devices require about 2000GE and 80-bit keys. From the ciphers that support 80-bit keys, LED, PUFFIN-2, KATAN, Klein, EPCBC, and LBlock produce low hardware efficiency or consume a lot of energy per bit. Klein typically targets wireless sensors which are more powerful devices and PUFFIN-2 requires more than 2000GE. Piccolo achieves the best overall status compared to all the other ciphers. It occupies a small chip area even when both encryption and decryption are implemented. Piccolo has higher throughput and hardware efficiency, while it consumes the least energy. SIMON, SPECK, and RECTANGLE also achieve a good overall status and consume low energy but are newly proposed ciphers. GOST and PRINTcipher and perform well in this category with higher key sizes. KTANTAN, MIBS, TWINE, and PRESENT perform reasonably on such devices.

Suitable implementations for ultra-constrained devices require less than 1000GE and 64-bit keys. Klein, LED, and MIBS provide exactly 64-bit keys. Among these three ciphers, only LED can be implemented within the GE limit but with high energy cost. The most compact ciphers that require around 1000GE and provide key sizes between 80- and 256-bits are the GOST, PRINTcipher, KTANTAN, Piccolo, SIMON, KATAN, SPECK, PRESENT, and LED. GOST, PRINTcipher, and KTANTAN perform the best.

Some serial implementations have achieved very small chip areas. The ciphers PRINTcipher, GOST, Piccolo, KTANTAN, and SIMON can be implemented with less than 800 GEs.

Among the novelties and well-known practices proposed to enhance hardware implementations are the use of a scan flip-flop instead of a D flip-flop and a 2-to-1 MUX, AND-NOR gates instead of XOR gates, and OR-NAND gates instead of XNOR gates. Furthermore, the storage of memory elements in registers instead of flip-flops consumes less energy and occupies less chip area. Bit-slice techniques are also utilized to reduce the area and design complexity requirements. Key scheduling is simplified to enhance latency and throughput. Involutive designs are suggested for low cost encryption/decryption functionality.

Software implementations

For software implementations, Figure 46 to Figure 48 illustrate the best software implementations according to individual metrics. The ciphers AES, Camellia, IDEA, NOEKEON, KASUMI, GOST, HIGHT, PRESENT, PRESENT-GRP, CLEFIA, HB, HB-2, Piccolo, TEA, XTEA, SEA, mCrypton, MIBS, TWINE, LED, Klein, KATAN, KTANTAN, ITUbee, SIMON, SPECK, PRINTcipher, LBlock, PRINCE, PRIDE, Zorro, Robin, Fantomas, LEA, and DES and its variants are examined. All software implementations are detailed in [427].

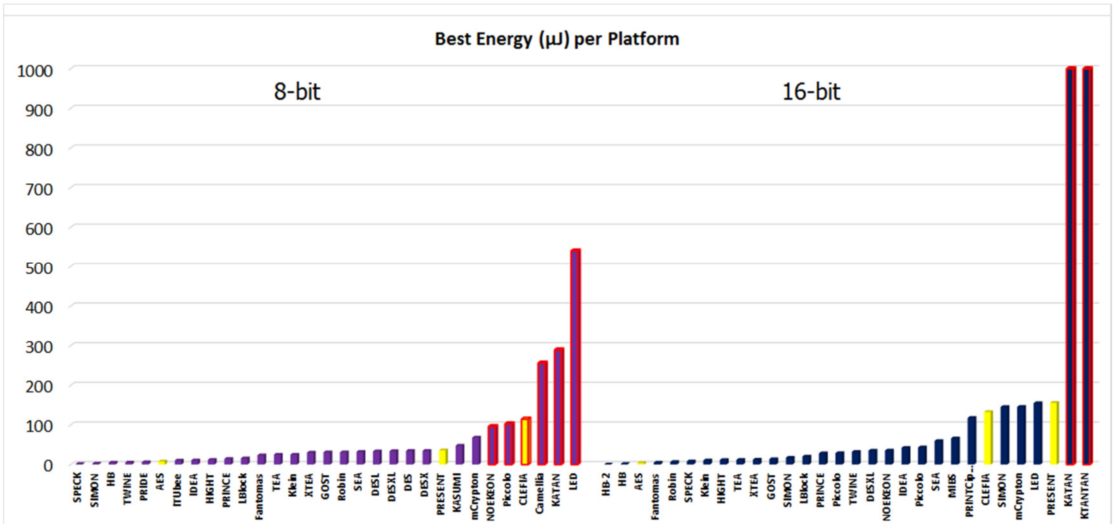


Figure 46 Best software implementations of block ciphers in terms of energy consumption

KTANTAN, LED, KATAN, and Camellia consume high energy. KTANTAN, KATAN, PRINTcipher, LED, and Piccolo are too slow. The standardized PRESENT and CLEFIA has also low throughput as the lightweight proposals for mCrypton, PRINCE, KASUMI, MIBS, SEA, DESX, DES, DESXL, and DESL. KASUMI, mCrypton, TWINE, NOEKEON, Piccolo, Camellia, IDEA, MIBS, PRINTcipher, CLEFIA, LED, KATAN, and KTANTAN exhibit high latency in software (more than 10000 cycles). KTANTAN, KATAN, PRINTcipher, GOST, LED, MIBS, Camellia, DESX, and DES achieve the worst overall performance based on the software efficiency metric in all three microcontroller technologies.

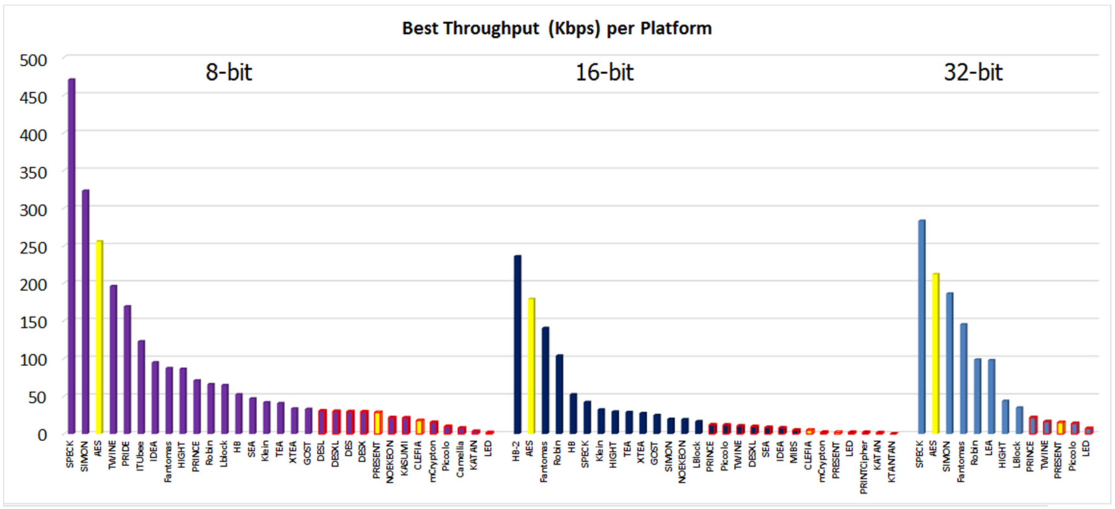


Figure 47 Best software implementations of block ciphers in terms of throughput

A speed optimized implementation of HB exceeds the boundary of 8KB RAM for both low-cost and lightweight devices and is not considered efficient for LWC. For ultra-constrained devices only, KTANTAN, PRESENT-GRP, and Zorro exceed the boundary of 256 byte RAM and cannot be applied. For ultra-constrained and low-cost devices, some of the implementations for GOST, PRINTcipher, KTANTAN, PRINCE, DES, and DESX exceed the boundary of 4KB ROM with HB-2, DESL, DESXL, HB, LBlock, and KATAN coming close.

The most compact implementations (in terms of code size) are reported for SPECK, SIMON, PRIDE, SEA, KATAN, NOEKEON, AES, ITUbee, HIGHT, and XTEA (less than 0.5KB). The implementations with the higher throughput are those of SPECK, SIMON, AES, HB-2, TWINE, PRIDE, Fantomas, ITUbee, Robin, LEA, IDEA, and HIGHT (more than 85 Kbps). The lowest latency is achieved by HB-2, SPECK, SIMON, HB, TWINE, PRIDE, AES, ITUbee, IDEA, and HIGHT (less than 3000 cycles). HB-2, SPECK, HB, SIMON, AES, Fantomas, TWINE, PRIDE, and Robin have low energy consumption (less than 10 μJ per bit).

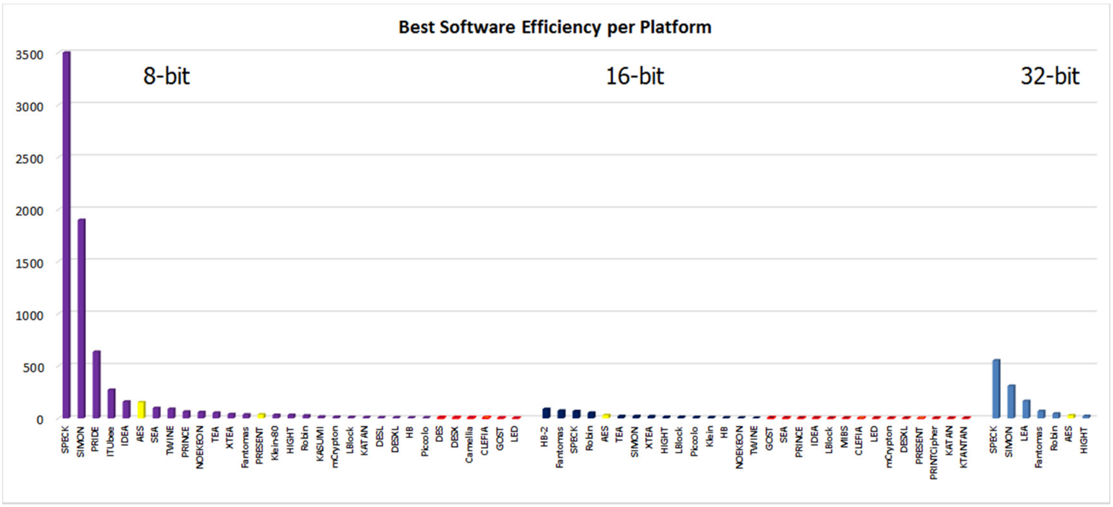


Figure 48 Best software implementations of block ciphers in terms of software efficiency (Software Efficiency = Throughput [Kbps] / Code size [KB])

Figure 49 illustrates the best overall software implementations for the three microcontroller platforms and different keys. Based on the criteria adopted in this work, SPECK and PRIDE evaluate best compared to all the other proposals for many key sizes. Fantomas and Robin perform similarly well, providing additional security against side-channel attacks. AES, SEA, and ITUbee achieve a good overall status. PRESENT is slower with high latency and energy consumption. SPECK and SEA also offer efficient encryption/decryption implementations with low cost. DESXL provides higher key size, while for lower security DESL performs well.

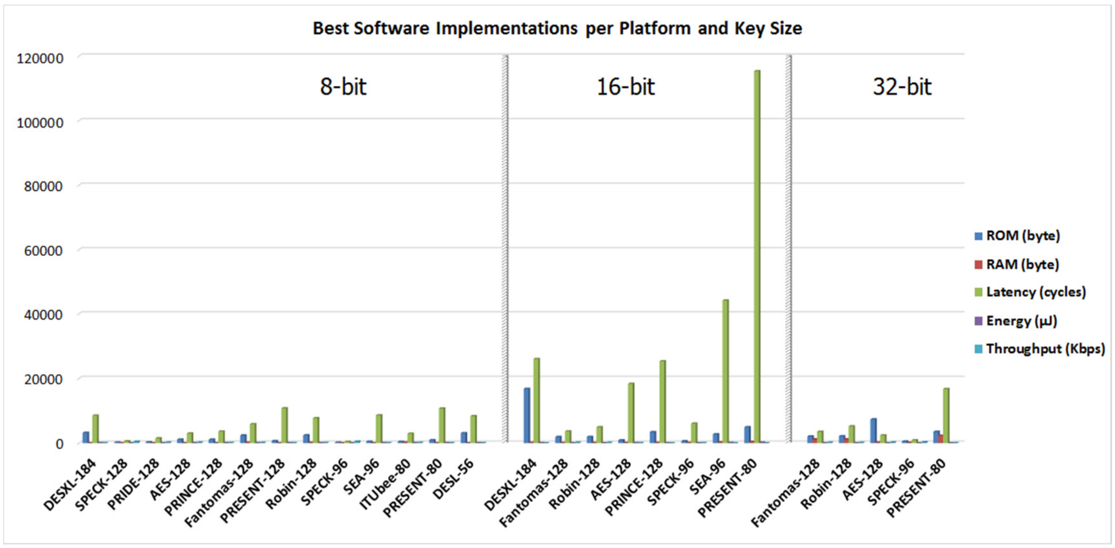


Figure 49 Best overall software implementations of block ciphers per platform

In software implementations, code size reduction can be achieved by using low-level languages, such as the device's assembly language, and by developing segments of code that will facilitate code reuse. To reduce the memory requirements it is suggested to reuse storage space for keeping different elements, whenever it is possible. To achieve better performance it is preferred to use the device's word size as a processing unit (usually byte words). Word-wise permutations (4- or 8-bit) are preferred to bitwise permutations. Moreover, word-wise matrix multiplication operations are utilized.

The ciphers SIMON, SPECK, TWINE, Klein, LED, and LBlock were designed both for lightweight hardware and compact software implementations. SIMON and SPECK shine in both domains and TWINE in hardware, while the rest of the ciphers perform reasonably well in both domains and do not excel in any of them.

Design directions

The maintenance of the internal state is the most challenging aspect of managing space requirements. The state has to be saved at each round which typically requires at least half of the total chip area and considerable power. As memory is limited in low-cost devices and read/write operations are power-demanding, the intermediate state is often kept in registers. The registers are materialized by flip-flops that have high area and power needs. Thus, lightweight ciphers try to minimize the storage requirements.

Space reduction can be achieved by lowering the block and key size. A small block or key size preserves small memory complexity. On the other hand, various security issues, such as birthday attacks, are introduced when we use block sizes that are less than 32 bits. Popular parameters for lightweight ciphers are 64-bit blocks and 80-bit keys size.

Absence of decryption functionality and hard-wired keys are techniques that can also improve the performance of applications that utilize cryptography. Some proposals tend to apply simple wiring while others avoid this approach to harden differential analysis. For applications where the implementation cost is more important than the security level, Feistel networks and simple key schedule mechanisms are a good choice. For applications were moderate security is needed, SPNs with more robust key schedule mechanisms are preferred.

S-box

S-boxes are a critical component of block ciphers, typically used to introduce nonlinearity. In software, they are implemented using look-up tables (LUT). However, these LUTs, when implemented in hardware, may have a large footprint or introduce other technical complications. Combinatorial solutions [155] are usually more efficient, where the input bits affect the complexity of the equations and the output bits affect the number of the Boolean equations required. If a combinatorial solution does not deal with the inner structure of the S-box, the occupied area will grow rapidly with the number of input and output bits. High nonlinearity demands more area, but also offers higher resistance to differential and linear analysis.

The size of the S-box is another important parameter. Large S-boxes can achieve high levels of security but consume many resources for both their hardware and software implementations. On hardware, the gate count increases exponentially with the size. However, very small S-boxes cannot provide an acceptable level of security. A good choice that balances efficiency and security, and is adopted by the majority of the lightweight ciphers, is the 4×4 S-box (as an alternative to typical 8×8 S-boxes). If higher levels of security are required, it is preferable to increase the number of rounds.

A.4 Stream Ciphers

A.4.1 General Information

Stream ciphers have been applied in several real-time applications (e.g. mobile phone telephony) due to high performance in hardware. In 2004, a performance analysis of these mainstream ciphers was presented in [297]. The main stream cipher designs and the basic hardware-efficiency and security trade-offs were identified.

In 2007, the development of the lightweight block cipher PRESENT signified a milestone in LWC with several lightweight designs being proposed afterwards. A first survey on LWC was held in the same year [298], reviewing several symmetric and asymmetric ciphers for embedded hardware and software.

The eStream project advanced the research efforts for compact stream cipher designs. A survey and software benchmark of stream ciphers for wireless sensor networks was presented in 2007 [299], focusing on eStream candidates. Several ciphers were deployed and evaluated on the same micro-controller platform. Similarly, hardware results for selected eStream candidates in ASIC are presented in [300].

The ISO/IEC standardization effort in 2012, enhanced cryptanalysis results for secure stream cipher design. Cryptanalysis techniques for stream ciphers and relevant attacks are presented in [301] and [302] respectively.

In 2013, a comparative analysis of newer stream cipher implementations for embedded systems was conducted in [160]. The most efficient implementations in embedded hardware and software were evaluated based on a set of predefined metrics.

In this subsection, the main traditional and lightweight stream ciphers are surveyed along with the latest design and cryptanalysis results in the field. We include authenticated encryption schemes and relevant CAESAR candidates that are based on stream ciphers. We indicate the safe ciphers and perform a benchmark analysis on embedded hardware and software platforms.

A.4.2 Stream Cipher Design

General features

Stream ciphers are an alternative type of symmetric cryptosystem to block ciphers [303]. They are based on the idea of the One-Time Pad (OTP) cipher, known as Vernam cipher [304]. OTP utilizes a completely random keystream and each digit of the keystream is combined with one digit from plaintext to produce a digit of ciphertext. OTP was proved to be unbreakable [305], however, the keystream digits have to be completely random and, moreover, the keystream must have the same length as the plaintext. Thus, OTPs introduce significant practical management and maintenance problems, and are considered impractical for wide use.

Stream ciphers address these issues by sacrificing a degree of security: they apply a secret key, which is used to generate a pseudorandom keystream. The secret key is the symmetric key and the pseudorandom keystream generated is used in the same way as the keystream in OTPs. However, the fact that the keystream is not completely random introduces security concerns. The keystream must be random enough to ensure that if an attacker knows the keystream, he cannot recover the secret key or derive the cipher's internal state.

The keystream period is a significant security factor for stream ciphers. It stands for the number of encrypted bits or blocks that are processed before the keystream is repeated. If this is exhausted and the keystream is repeated, cryptanalysis could potentially decrypt the sequential ciphertext bits [303]. Thus, the keystream period should be larger than the size of the plaintext that will be encrypted. If the keystream period elapses, a different key or nonce must be used to re-initialize the cipher; a longer period implies fewer instances that this will have to take place.

After initialization, all stream ciphers require a few clock cycles to encrypt a bit. Yet, the initialization process is also essential in determining the applicability of a stream cipher. Stream ciphers with fast initialization phase are suitable for applications where many short messages have to be processed. On the other hand, when few and large messages must be encrypted, stream ciphers with fast encryption are appropriate. Thus, stream ciphers cannot be easily classified in terms of performance, as it depends on both the initialization and en/decryption operations.

Cryptanalysis and attacks

Attackers aim to exploit stream cipher designs in order to discover the key that is used for de/encryption. Then, they can recover part or the whole communication processed by this key.

Passive attacks exploit the output or the initialization/resynchronization phases. They mainly analyze design flaws. Active attacks insert, delete or replay ciphertext bits (e.g. fault attack) and try to misuse the communication channel. They are countered by strong data integrity and authentication mechanisms.

Exhaustive key search is a brute force attack where an attacker tries out all possible combinations until she finds out the key. The computational complexity can't exceed the key size in bits but can also be lower. All ciphers are vulnerable to these attack. The lower bound of the exhaustive key search complexity for a cipher denotes the intuitive limit that all the other attack types try to improve upon.

The re-initialization process of a cipher design is another target of attacks. If the input is related to the internal state without sufficient non-linearity when the new keys are produced, an attacker can determine the *related keys* and the initial one. *Chosen-IV* attacks exploit weaknesses in key scheduling and extract the initial state from the memory.

Side-channel attacks measure electromagnetic emission or power consumption during the processing of the data from the cipher. The goal is to derive useful information about the algorithm's internal processes.

In *distinguishing* attacks, the attacker tries to distinguish the key stream from a purely random sequence. Such attacks can turn into complete key recovery.

Correlation attacks analyze the linear functions of the cipher and determine the produced keystream by observing the output bits. *Linear* attacks correlate the linear functions of selected keystream bits or linear functions of selected keystream and state bits. Similarly, *algebraic* attacks resolve systems of algebraic equations that are utilized for the production of the key or state bits. One popular stream cipher design approach utilizes non-linear components with masking of block ciphers. *Linear masking* distinguishes a non-linear characteristic that exhibits some bias. Missing linear combinations, derived from the linear functions of the cipher, are applied to the output and discover the traces of this distinguished characteristic.

Cube attacks are a relatively new attack type and can be applied to almost any cryptosystem. The attack takes advantage of the existence of a low degree polynomial representation of a single output bit as a function of the key and plaintext bits. These bits are summed over all possible values of a subset of the plaintext bits to detect linear equations in the key bits, which can be efficiently solved.

Time/Memory/Data tradeoffs explore the general structure of a cipher and summarize the analysis results in large tables. At attack phase, these precomputed tables are utilized in order to retrieve the key of the actual data that is being processed.

Guess-and-determine attacks guess a part of the state and try to reconstruct the whole state based on the keystream that is actually observed.

Slide attacks correlate two copies of the same encryption process that are one round out of order. The two relevant plaintexts are lined up, trying to find a match. The attack reveals the key in the state update of the cipher.

In *divide and conquer* strategy, the cipher is disassembled into basic components. A few bits are determined in each state and the most vulnerable components are attacked first. Designs that exhibit high correlation immunity are less vulnerable to such attacks.

A survey of cryptanalysis techniques for stream ciphers is presented in [301], where the exhaustive key search, related key, side channel analysis, distinguishing, correlation, algebraic, linear masking, time-memory tradeoff and guess-and-determine attacks are explained. The fault, chosen-IV, cube and slide attacks are additionally described in a later survey [302], along with the proposed mechanisms to mitigate them.

Synchronous and self-synchronous ciphers

Stream ciphers generate keystream digits based on their internal state. There are two categories of stream ciphers based on the update operation of the internal state.

Synchronous stream ciphers update the internal state independently from the plaintext or ciphertext data. A sender and a receiver must be synchronized before encryption/decryption takes place, and if part of the message is added or removed, this synchronization is lost. A re-synchronization mechanism is, thus, required to maintain communication, introducing complexity and computational overhead. On the other hand, if a bit of the ciphertext message is altered, only a single plaintext bit is affected, i.e. the error is not propagated to the rest message. This characteristic is very important in applications like wireless communications, where the error rate is high, but it also makes it difficult to detect errors. Figure 50, illustrates the operation of synchronous stream ciphers. Synchronous stream ciphers are vulnerable to active attacks, as an attacker could disrupt a communication and discover correlations between the altered ciphertext bits and the corresponding plaintext bits. Most of the cryptanalysis efforts focus on known-plaintext attacks.

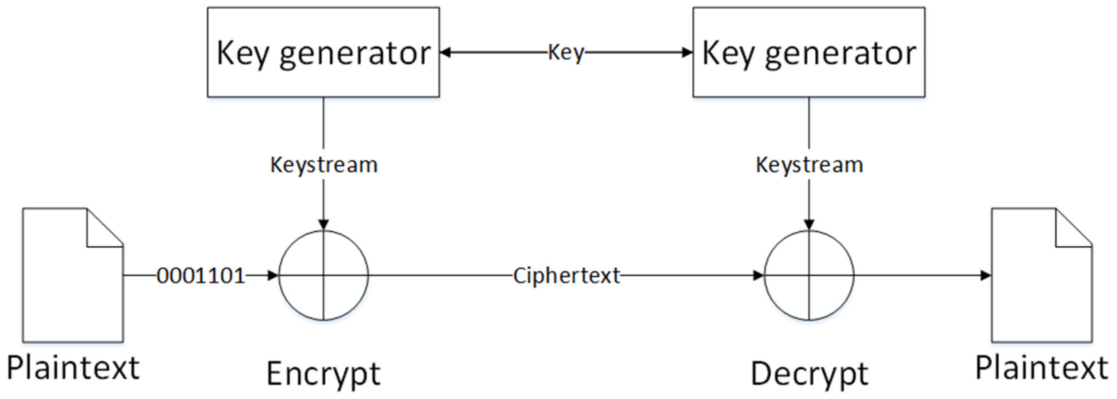


Figure 50 Synchronizing stream cipher design

Self-synchronizing stream ciphers update the internal state based on the *N* previous ciphertext bits; Figure 51 illustrates their functionality. With self-synchronizing stream ciphers the receiver can automatically synchronize herself with the sender after receiving *N* ciphertext digits. This makes it easier to recover the message even if some bits are added or removed. If a bit is altered, *N* plaintext bits will be affected at most. Thus, it is more difficult to perform active attacks. However, in a cryptographic context this is also a disadvantage, as up to *N* bits will be decrypted incorrectly for every altered bit. Moreover, self-synchronizing ciphers suffer from important security issues. As the key is updated based on previous bits, statistical regularities in plaintext are disclosed in the keystream. Self-synchronizing ciphers are mainly vulnerable to chosen ciphertext attacks and once the attacker decrypts any part of the message, she can decipher more. These features make the design of secure self-synchronizing stream ciphers a difficult task, which also explains the rarity of new proposals. In eSTREAM project, only the ciphers SSS [306] and MOUSTIQUE [307] are self-synchronizing but both are broken. As stated in [308], there is little interest in self-synchronization as they are not supported by the industry today.

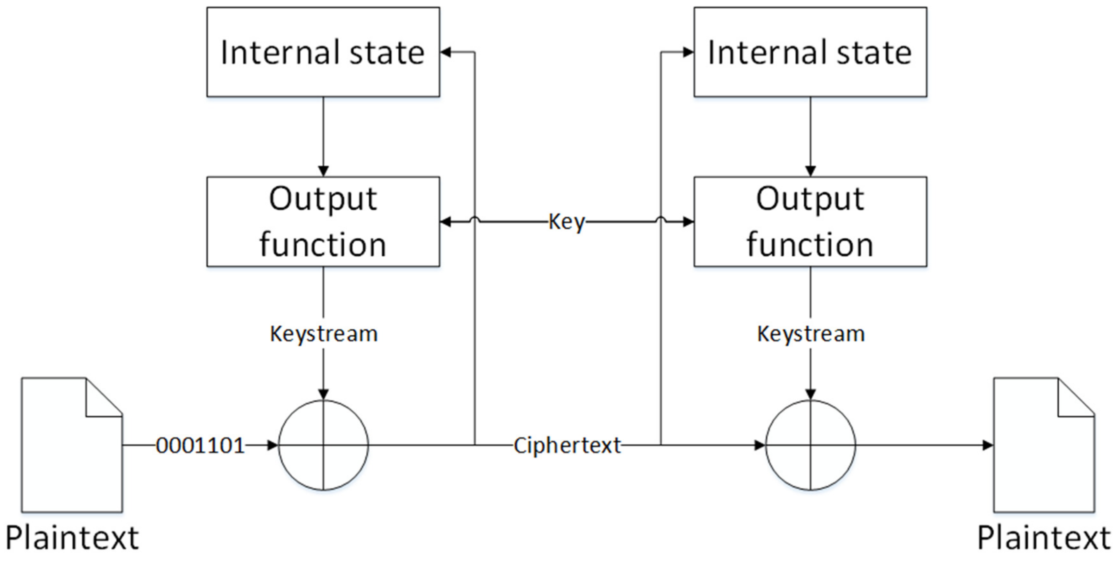


Figure 51 Self-synchronous stream cipher design

Both synchronous and self-synchronizing ciphers can be used for encryption/decryption, providing confidentiality, but also as pseudorandom number generators (PRNG) by encrypting a long sequence of zeros. Integrity and authenticity are additional features that are accomplished by more advanced primitives. Authenticated encryption (AE) schemes utilize the pure ciphers (block or stream) and achieve confidentiality, integrity and authenticity. AE processes a message and produces the ciphertext, along with a message authentication code (MAC) for authenticity and integrity check. AE schemes for embedded system are detailed in a subsection below.

<u>Design taxonomy</u>

Several stream cipher designs are proposed in the literature. In this subsection we introduce the most suitable of them for LWC. Figure 52 illustrates the stream cipher design taxonomy.

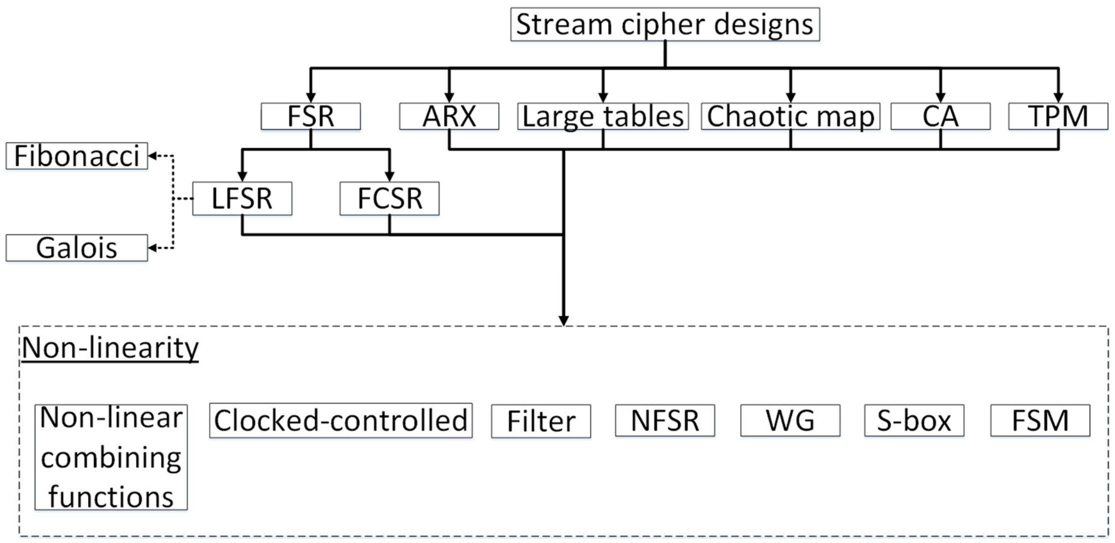


Figure 52 The taxonomy of the stream cipher designs

Main design types

Stream ciphers are usually built using Feedback Shift Registers (FSRs). At each cycle, an FSR gets one bit as input and produces one bit at the output. The input bit is a function of the previous state. There are two types of FSRs based on the feedback function: the Linear Feedback Shift Registers (LFSRs) and the Feedback with Carry Shift Registers (FCSRs).

LFSRs are a common choice for stream ciphers (e.g. [309], [310], [311]). They are easy and fast to implement in hardware and their properties can be studied and analyzed mathematically. However, their linear nature means they are not secure unless they are used along with additional components to introduce nonlinearity. There are two styles of LFSR according to the shift operation. *Fibonacci* LFSR is the typical style, where each bit is copied to its right neighbor and the bits are right shifted. The rightmost bit is the output. The new leftmost bit that is inserted is the parity of some special bits, called taps. *Galois* LFSR is the alternative style, where the bits are right shifted, except from the tap bits. The rightmost bit of the taps is XOR-ed with the previous output bit before the copy is done. The original rightmost bit from the taps is both the new leftmost bit that is inserted and the output. Galois LFSRs are considered more efficient in hardware than Fibonacci LFSRs as they introduce lower gate delay and thus lower clock cycle time.

FCSRs are arithmetic analogs of LFSRs [312]. They are similar to LFSRs but with additional memory for maintaining a carry from one stage to the next, forming a finite device. They can be efficiently implemented in a parallel architecture. However, FCSRs are periodic in nature and cannot be directly applied in cryptography [313].

Except from the stream ciphers that are based on FSRs, there are several alternative design types. Ciphers that use modular **add, rotate and XOR operations (ARX)** are popular due to their fast and cheap implementation (e.g. [314], [315], [316]). These operations run in constant time, reducing the feasibility of timing attacks. ARXs produce fast and compact implementations but their security properties are not well-studied.

In contrast to ARX ciphers that use simple operation, **large table** designs are proposed for high speed in software (e.g. [317], [318]). The state is maintained in large tables and the content is updated at each round by non-linear functions. This approach exhibits among the fastest results at the cost of high memory consumption in both hardware and software.

A **Cellular Automata (CA)** [319] is a regular grid of cells, where each one has a finite number of states. The cells update their state according to a fixed rule. The rule determines the new state of each cell based on the current state of a cell and its neighbors. Given the rule, it is easy to determine the future states but very difficult to estimate the previous ones. In cryptography, CAs can act as a one-way function whose inverse is hard to find (e.g. [320], [321]). Their hardware implementation is simple, while word-wise implementations can be efficient in software. Moreover, parallel transformations are also possible, allowing to increase throughput.

Chaotic maps [322] are maps that exhibit a chaotic behavior. They can be parameterized by a discrete-time parameter, like an iterated function. In cryptography, their ordinary setting is the mixture of the plaintext with a keystream using bitwise operations [323]. The optimum outcome is a pseudorandom keystream with good statistical properties. Their main advantage is the level of confusion and diffusion that they achieve. Moreover, they can be directly implemented in hardware.

A **Tree Party Machine (TPM)** [324] is a multi-layer feed-forward neural network. Although it is not a common choice in cryptography, it has been applied to stream cipher design [325] due to the level of randomness that it achieves. In the common topology, two TPMs update their weight vectors by mutual learning based on each other's output in order to achieve the weight space synchronization. The common inputs vary dynamically at each learning iteration, but kept identical throughout all mutual learning time. The TPM properties guarantee that given the mutually exchanged TPM output over an open channel, an eavesdropper cannot exactly determine which internal weight vectors has been updated each time. Thus, the eavesdropper cannot derive the final synchronized weight vectors. These final weight vectors are mapped to a shared key.

Non-linearity

Several schemes have been proposed to increase the security of the main LFSR design and introduce the needed nonlinearity. Examples include the non-linear combining functions, the clock-controlled generators and the filter generators. With a **non-linear combining function**, the outputs of several parallel LFSRs are passed to a non-linear Boolean function. **A clock-controlled generator** uses two LFSRs. In the ordinary setting, a LFSR is stepped regularly. The generator implies that the LFSR must be clocked irregularly according to the output of the second LFSR. When **filter generators** are used, the entire state of a LFSR is fed into a non-linear filtering function. Combinations of all these schemes are also examined in the literature (e.g. [326], [327]).

Non-linear Feedback Shift Registers (NFSRs) are another core component, utilized by stream ciphers to provide nonlinearity (e.g. [326], [327], [328]). Furthermore, they are more resistant to cryptanalytic attacks. Yet, only a few theoretical results exist for NFSRs. The security level of ciphers that make use of NFSRs depends on the difficulty of analyzing the design itself. In hardware, they are typically slightly more complex than LFSRs and FCSRs. However, it is difficult to design a good NFSR, as there are no simple and precise guidelines on building strong non-linear Boolean functions. These functions are used in NFSRs to determine the balance between the zeroes and ones in the output, the nonlinearity, the algebraic degree and the correlation immunity.

The **Welch-Gong (WG)** functions are another family of structures which, in contrast to NFSRs, can be analyzed mathematically [329]. They are proven to guarantee a variety of randomness properties, like ideal two-level autocorrelation, balance, long period, ideal tuple distribution and high and exact linear complexity. These properties satisfy security requirements for encryption and authentication [310].

The **Substitution-boxes (S-box)** are implemented in the form of lookup tables which map *m* input bits to *n* output bits. They are a common and well-studied choice in block ciphers, used to obscure the relationship between the key and the ciphertext. S-boxes are also utilized by many stream ciphers (e.g. [98], [218], [309], [313], [330], [331]) in order to enhance their nonlinearity features.

Finite state machines (FSMs) are mathematical models for computing sequential logic. At each time point, the FSM is in one of a finite number of states. A transition to a new state is performed as a result of a triggering event or condition. In stream cipher design, FSMs perform well in software and enhance protection against guess-and-determine attacks [309].

Design guidelines

The maintenance of the internal state is the most space-consuming part of a cipher. This burden is even heavier in stream ciphers, as, comparatively, a larger percentage of the hardware implementation is used for storing the internal values. For example, a hardware implementation of the Grain stream cipher, presented in [326], requires 1294 gates, with the bulk of them being used for this task. In more detail, the internal state must be at least twice the security level in bits [326]. The nonlinearity functionality, on the other hand, can be compact in hardware.

Thus, reducing the area factor of a stream cipher is a difficult task. Stream cipher designers typically aim to reduce the size of the registers and the complexity of the Boolean and filtering functions.

In hardware, an LFSR uses flip-flops to maintain its state and XOR gates to compute the feedback. In order to decrease the hardware area, designers can decrease the size of the internal state and the number of XORs. However, a cipher with short registers and a simple feedback function can be vulnerable to cryptanalysis.

An NFSR uses more complex Boolean operations or S-boxes to form the feedback function. The area occupied by these Boolean operations is larger than the one of the XORs of an LFSR, but the resulting number of required flip-flops is smaller.

Word size is another important factor. In this regard, bit- and binary-oriented stream ciphers are efficiently implemented in hardware, while word oriented ciphers are preferable in software.

In terms of hardware implementations, the most popular approach is to exploit LFSRs and NFSRs ([326], [327], [328]). For software implementations, using table-driven ciphers ([317],

[323]) or building blocks from block ciphers ([98], [218], [309], [313], [330], [331]) are suggested when aiming to achieve high throughput rates. To increase the provided level of security, larger internal states and mixing of algebraic domains ([310], [311]) are often adopted. In general, it is proposed to use simple operations that are implemented by the native processor's instruction set to increase speed.

A.4.3 Stream Cipher in LWC

In this subsection, stream ciphers that have been applied in embedded systems are surveyed. The widely used ciphers and their vulnerabilities are highlighted, then focusing on recent research efforts and standards for LWC.

Traditional stream ciphers

Traditional ciphers for stream encryption and pertinent security issues are presented in [303]. As referred in said work, ciphers RC4 [314], A5/1 [332] and E0 [332], although not completely compromised, suffer from serious security vulnerabilities and should not be used in new applications.

RC4 is the most widely used software stream cipher and is included in popular protocols and standards, like WEP, WPA and TLS. It has a simple design and is remarkably fast. It relies solely on byte manipulations, without an LFSR. RC4 is efficient in both hardware and software. However, security issues have been reported due to its dated design, including criticism for its tenuous key scheduling process. Specifically, RC4 doesn't take a separate nonce with the input key, leaving it up to the communication protocol to specify how to combine these two parameters and produce the keystream. One solution is a strong MAC that hashes the key with a nonce, but analysis revealed biased outputs of the keystream [333]. Thus, the keystream bytes can be correlated with the key ([334], [335]), a fact that was exploited, along with related effects, to break WEP ([336], [337]). The latest attack on RC4 requires 224 connections [338]. Although there are no practical attacks on the cipher itself, the latest results speculate that well-equipped agencies (e.g. state-funded entities) may have better attacks that render the cipher insecure.

A5/1 was designed to provide over-the-air communication privacy for the GSM cellular telephone standard has been widely used in GSM telephony in Europe and the USA. In its design, three combined LFSRs with irregular clocking are used to encrypt bursts of traffic, as is required in GSM. A5/1 is nowadays considered insecure, as attacks both in the cipher itself and its implementations in GSM have been presented. Most of the cryptanalysis was based on time-memory tradeoff and other known-plaintext attacks, with many of them breaking the cipher in a few minutes or seconds ([339], [340], [341], [342]). In GSM, active attacks and ciphertext-only analysis prove the protocol to be insecure even when in the presence of a secure cipher [343]. The latest attacks utilize parallel computing platforms, like FPGAs and GPGPUs, to launch practical attacks on A5/1 and GSM at runtime ([344], [345]).

E0 is used in Bluetooth. It uses 4 shift registers (SHR) to produce a sequence of pseudorandom numbers which are XOR-ed with the data. It is also vulnerable and practical attacks have been reported both for the cipher and the Bluetooth protocol. Typically, E0 uses 128-bit key.

However, analysis has decreased the security level to 65 bits even when longer keys are used ([346], [347]). In Bluetooth, statistical and known-plaintext attacks have further decreased the security to 38 bits ([348], [349]).

Block ciphers can operate as stream ciphers too. The NIST modes of operation [350] indicate three ways to implement this functionality: the cipher feedback (CFB) mode for implementing self-synchronized ciphers, and the output feedback (OFB) and counter (CTR) modes for implementing synchronized ciphers. The implementation involves the encryption functionality for the most part, as the decryption is almost identical under these modes. Security issues on these stream modes are discussed in [302]. AES is the most investigated block cipher in terms of its stream versions. The cipher is safe with biclique cryptanalysis achieving slightly better results than exhaustive search [351]. AES in CTR mode (AES-CTR) is currently the only secure and widespread solution for stream encryption [303]. Figure 53 illustrates the encryption process of AES-CTR. PRESENT, the standardized block cipher for LWC (ISO/IEC 29192), has also been studied but was found to be inferior to the dedicated stream ciphers.

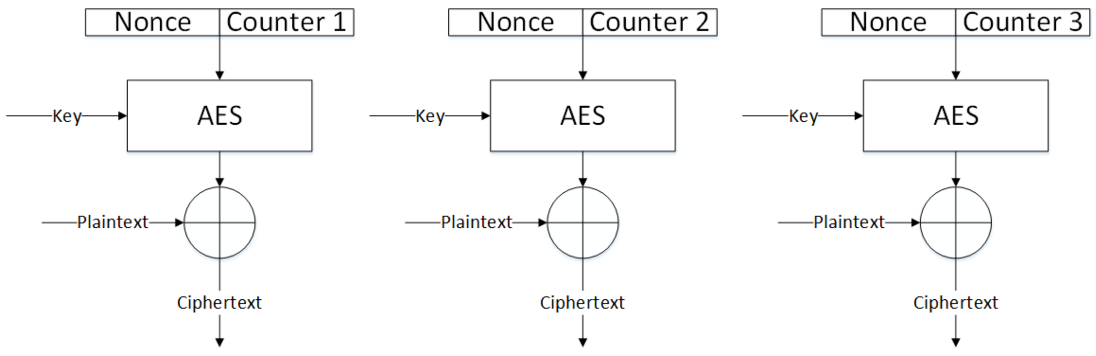


Figure 53 AES in CTR mode of operation design

The eSTREAM project

The eSTREAM [352] project was part of the European Network of Excellence for Cryptology II (ECRYPT II) within the Information & Communication Technologies (ICT) Programme of the European Commission's Seventh Framework Programme (FP7) and delivered a small portfolio of promising stream ciphers.

Two profiles of ciphers for software and hardware implementations were defined. Profile 1 consists of ciphers with high throughput in software that are faster than the 128-bits AES-CTR. Finalist ciphers include Salsa20/12 [353], Rabbit [323], HC-128 [317] and SOSEMANUK [309]. Profile 2 consists of ciphers that are suitable for highly constrained environments and are more compact in hardware than the 80-bits AES. The finalists include Grain [326], Trivium [354] and MICKEY 2.0 [327].

Several ciphers were submitted and rejected due to security vulnerabilities or lower overall performance. From a total of 34 ciphers, only two self-synchronizing stream ciphers were submitted and both were proven to be insecure. The finalists were also cryptanalyzed and were found to be secure against all attacks that are faster than the exhaustive key search attack.

However, newer cryptanalysis attempts have reported some tangible results for some of these ciphers.

In LWC, the most notable ciphers are the two finalists of Profile 2, i.e. Grain and Trivium. They are very attractive primitives and, thus, have been extensively investigated, as their compact and efficient hardware implementation can be applied to ultra-constrained devices. Nevertheless, the finalists of Profile 1 (i.e. Salsa and Rabbit) are also extensively examined for compact embedded software applications as they are fast and do not consume significant resources. Thus, the above four ciphers, along with AES-CTR, are used as a benchmark for newer stream cipher proposals.

Profile 1 – software-oriented ciphers

Salsa20/r, where *'r'* stands for the iterations of the round function, is an eSTREAM finalist for software implementations. It uses 256-bit keys and 128-bit IVs. Three variants have been proposed to cover the tradeoff between security and performance, catering for the different application needs. Salsa20/20 is designated for encryptions in typical cryptographic applications, while the variants Salsa20/12 and Salsa20/8 offer faster but less secure operation. Its design is based on simple operations of addition, modulo 232, bit rotation and bitwise XOR (ARX), which are efficiently implemented in software. The encryption and decryption operations are identical, allowing for a more compact implementation. Salsa20 is included in the Crypto++ cryptographic library [355] as a high speed stream cipher.

The most lightweight software implementation [298] requires 1452 bytes of code and 18400 cycles for encryption. The most compact hardware implementation [300] occupies 12126 GE, which is far beyond the scope of LWC.

Attacks have been reported for the reduced versions of Salsa20 [356]. For the variants Salsa20/20 and Salsa20/12 no better attack than the exhaustive key search has been reported.

Rabbit is a synchronous stream cipher that was designed for high software performance, with very fast key setup and encryption functionality. It is applicable to internet protocols and other applications which process large amounts of data or a large number of packets; it is also one of the most efficient software proposals of eSTREAM. Rabbit was patented by its designers but the algorithm is free to use. It is included in the cryptographic library for embedded systems CyaSSL [357] and is standardized in the ISO/IEC 18033-4:2011 [358] standard for IT security techniques.

Rabbit uses simple and basic operations, taking advantage of the features of modern processors. It provides strong non-linearity that is not based on NFSR and S-Boxes; Rabbit is based on the ideas of chaotic maps.

The key size is 128 bits and the IV is 64 bits. Its software implementation on a 1.7GHz Pentium 4 system consists of 1976 bytes of code and needs 486 cycles for key setup and 5.1 cycles for encrypting a byte [359]. Rabbit was also a candidate for the eSTREAM Profile 2. Its most compact hardware implementation requires 3800 GE, achieving 88 Kbps of throughput [359].

Practical fault analysis attacks require around 128-256 faults and precomputed table of size 241.6 bytes to recover the complete internal state in about 238 steps [360].

HC is a cipher that has two main variants HC-128 and HC-256 [318] for 128- and 256-bit keys respectively, with 128-bit IVs. It uses two large secret tables, each one with 512 32-bit elements, and operates on 32-bit words. At each step, an element is updated using a non-linear feedback function and a 32-bit output is generated by a non-linear output filtering function. HC is suitable for parallel computing and modern superscalar microprocessors as three consecutive steps can be computed in parallel. The feedback and output functions of each step can be performed simultaneously. It is now included in the CyaSSL crypto library.

As a table-driven cipher, it can yield impressive performance in software when large streams of data are encrypted. However, due to the initialization overhead, when small streams are processed, the performance is low (as frequent re-initialization is required). It is estimated that, in hardware, 52400 GE would be occupied just to cover the memory requirements [300].

No significant cryptanalytic advances have been reported on HC [361], [362], and the cipher is considered safe. Its authors estimate a keystream period larger than 2256 bits.

SOSEMANUK is a synchronous stream cipher. The key size varies between 128 and 256 bits and the IV size is 128 bits. However, the provided security level is the same, at 128 bits, for all key sizes. It adopts some of the design principles of the stream cipher SNOW 2.0 [363] and the block cipher Serpent [364]. SOSEMANUK performs better than SNOW 2.0 as it has a faster IV setup phase and needs less amount of static data. It uses an LFSR and a finite state machine (FSM) and operates on 32-bit words. For the setup phase, a 24 rounds Serpent is used to fill the LFSR and the FSM. At the keystream generation phase, 4 output words of the FSM are fed to Serpent's third S-Box and then XOR-ed to the LFSR's output words.

In software, its implementation takes about 2000-5000 bytes of code depending on the platform and the compiler and it achieves a throughput of 360 Kbps [309], while its hardware implementation occupies 18819 GE and achieves 3200 Kbps of throughput [300].

An improved differential fault analysis attack requires around 4608 faults, 235.16 SOSEMANUK iterations and 223.46 bytes storage to recover the inner state [365]. It takes about 11.35 hours on a PC to perform the attack.

Profile 2 – hardware-oriented ciphers

Grain was designed with an easy and compact hardware implementation in mind. It is a synchronous stream cipher and utilizes both LFSR and a non-linear filtering function. The LFSR ensures a minimum keystream period and balances the output. The filtering function is considered as a type of NFSR and introduces nonlinearity to the cipher. The output of the LFSR is masked with the input of this NFSR to balance its state.

Grain adopts a bit-oriented architecture, which is appropriate for constrained hardware implementations. The simplest implementation produces 1 bit/cycle. However, it also offers the option to increase its speed by increasing the length of the processing word; one can increase the number of bits at the expense of more hardware (the rate can be increased up to 16 bits/cycle). It provides higher levels security compared to the well-known A5/1 and E0 ciphers, while providing small hardware complexity.

Grain uses 80-bit keys and 64-bit IVs. It requires about 1294 GE for 1bit/cycle and 3239 GE for 16bits/cycle [326]. In [366] a software implementation is reported for the 1 bit/cycle rate that occupies 778 bytes of code and requires 107366 cycles for initialization and 617 cycles to generate the output.

Grain-128 [367] is a version of the Grain cipher that supports 128-bit keys and 96-bit IVs, being designed for applications that require higher level of security. The minimum word length is 1 bit and the maximum is 32 bits (the latter being a popular choice for software implementations). It requires 1857 GE for 1 bit/cycle and 4617 GE for 32 bits/cycle. Furthermore, Grain-128 is used as a building block for the state of the art hash function SQUASH [368].

Several cryptanalysis results have been reported since its selection in eSTREAM. The latest study [369] presents a differential fault attack on the Grain family of ciphers, under reasonable assumptions, which exploits certain properties of the Boolean functions and corresponding choices of the taps from the LFSR. An attacker injects a small number of faults and recovers the secret key.

Trivium is an eSTREAM Profile 2 finalist and a standardized stream cipher for LWC (ISO/IEC 29192-3:2012). Its designers intended to explore how far a stream cipher can be simplified without sacrificing its security, speed or flexibility. It is a synchronous, bit-oriented, stream cipher, which uses 80-bit keys and 80-bit IVs. It applies three SHR and forms a nonlinear internal state to avoid building nonlinearity mechanisms for the keystream output.

In hardware [370], its implementation in standard CMOS technology requires 2017 GE. An implementation in a custom design with dynamic logic and C2MOS flip-flops only occupies 749 GE.

Even though it was not designed for software applications, it performs reasonably well in software as well [348]. The initialization process requires 2050 cycles, the key setup 55 cycles and the stream generation 12 cycles/byte.

Due to its simplicity, several attacks have been reported. An improved differential fault analysis attack can recover the inner state of the cipher by just injecting 2 faults and using 420 keystream bits [371].

MICKEY 2.0 (Mutual Irregular Clocking KEYstream generator) is the third finalist of eSTREAM Profile 2. It uses a Galois LFSR and a NFSR with irregular clocking to introduce nonlinearity as well as some novel techniques to guarantee period and pseudo-randomness. It is worth noting that there were 80 stages for the two registers in the first version of the cipher, but, due to security issues that were noticed during the early phases of eSTREAM, the states were later increased to 100. Its key size is 80 bits and the IV can vary from 0 to 80 bits. The 240 keystream bits can be generated from each pair of key/IV and each key can be used with up to 240 different IVs of the same length. A hardware implementation occupies 3188 GE [372]. As with Grain and Trivium, a differential fault attack on MICKEY 2.0 was demonstrated in [373]. The attack requires around 214.7 faults, as MICKEY 2.0 is more complex than the other two finalists. Related key attacks with 65 key/IV pairs break MICKEY 2.0 with 0.9835 probability [374].

The **MICKEY-128 2.0** [375] version was designed to provide higher security. The key size is 128 bits and the IV varies from 0 to 128 bits. In this version, the stages of the two registers were increased from 128 to 160. At 170 MHz clock frequency the cipher achieves a maximum throughput of 170 Mbps. The hardware implementation [372] requires 5039 GE (higher than the 3000 GE limit for LWC). The cipher performs well on hardware. Yet, the irregular clocking mechanism indicates that the cipher cannot be directly parallelized. Similarly with MICKEY 2.0, MICKEY-128 2.0 is vulnerable to related key attacks [374]. The success probability for 113 related key/IV pairs is 0.9714.

The ISO/IEC standard for LWC

ISO/IEC 29192-3:2012 [376] standardized stream ciphers for LWC. The standard includes two stream ciphers: Trivium and Enocoro [330]. Figure 54 illustrates the two designs (Trivium [354], Enocoro [331]).

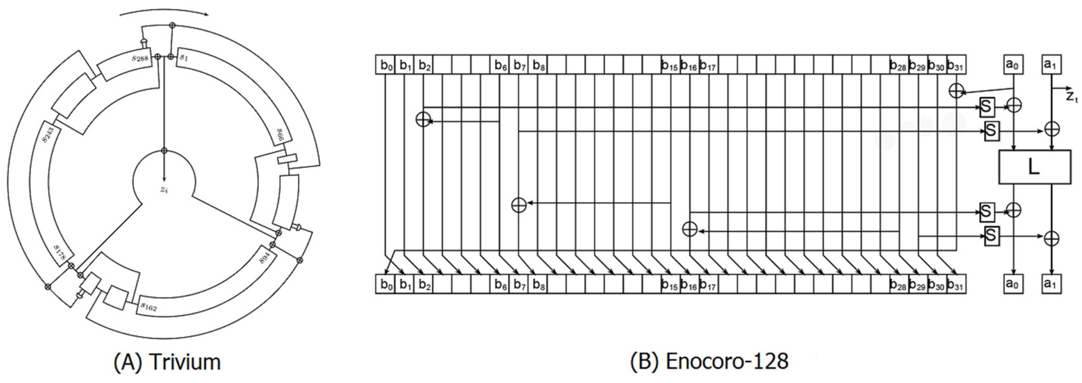


Figure 54 The standardized designs of the stream ciphers (A) Trivium and (B) Enocoro

The eSTREAM Profile 2 finalist Trivium, as described in the previous subsection, is efficient in hardware and performs reasonable in software. **Enocoro** was designed by Hitachi in 2007 and is their fourth cryptographic algorithm to become a standard. Enocoro achieves the encryption process of AES with 1/10th the amount of power consumption. This is also the main advantage of the cipher, which makes it appropriate for providing basic security functionality in compact control equipment and sensors.

The Enocoro family consists of Enocoro-80 and Enocoro-128 [331], for 80- and 128-bit keys respectively. Enocoro uses 64-bit IVs and adopts a byte-oriented design with an S-box which performs well both in hardware and software. It produces 1 byte per round and up to 264 bytes for each pair of key and IV.

In hardware, Enocoro-80 requires 2700 logic gates [330] which is comparable to the relevant implementations of the other eSTREAM Profile 2 finalists. Enocoro-128 requires 4100 logic gates for 3520 Mbps of throughput at 440 MHz [331].

In software, Enocoro-128 requires 4869.5 cycles to initialize and achieves a 46.3 cycles/byte throughput [331]. The initialization phase is long, like Trivium, but the encryption is faster than AES-CTR and Grain.

No practical attacks have been reported and the cipher is considered secure [377].

Other lightweight stream ciphers

Lightweight stream ciphers inspired by eSTREAM finalists

BEAN [313] is a newer proposal that is based on Grain. Its key size is 80 bits and it is more compact than Grain. BEAN uses two FCSRs and an S-box. The two FCSRs use different primitive polynomials to update themselves, while the S-box introduces better diffusion properties for keystream generation and amends cryptanalysis issues introduced by the FCSRs. It supports binary output production without additional hardware, which is another improvement over Grain. For software implementations, BEAN uses the same amount of memory as Grain but requires less time to generate the keystream bits. However, an efficient distinguisher attack and a state-recovery attack are demonstrated in [378], which are made possible due to the weak output function of BEAN.

Quavium [379] is proposed as a scalable extension of Trivium and it supports the same key (80 bits), IV (80 bits) and internal state (288 bits) sizes as Trivium. It is based on 4-round Trivium-like SHRs and k-order primitive polynomials. Instead of the three SHRs in series connection used in the original Trivium, Quavium uses four Trivium-like SHRs in coupling connection. It can also work with either 2 or 3 rounds, as the coupling connection retains the primitiveness of characteristic polynomials.

In hardware, Quavium requires 3496 GE and the 3-round version requires 2372 GE, while in software it is as fast as Trivium, with its 3-round version achieving improved performance [379]. In more detail, a Trivium C++ implementation on an Intel Core 2 Duo 2.00 GHz achieves 16.8 cycles/byte, while the equivalent Quavium implementation achieves 17 cycles/byte and the 3-round Quavium 12 cycles/byte.

No cryptanalysis results are presented. However, we consider that the aforementioned improved differential fault analysis attack that is performed on Trivium [371] can affect the security of Quavium due to the Trivium-like SHRs.

CAvium [320] is another new proposal based on Trivium. It utilizes CA with both nonlinear and linear rules, producing a secure design that can reach the desired setup state of a cipher much faster than the equivalent LFSR and NFSR structures. Thus, CAvium significantly reduces the long key setup phase of Trivium (from 1152 to 144 rounds) and counters the cryptanalysis attacks on the reduced round versions, while retaining comparable complexity in hardware. No independent cryptanalysis results are presented.

Other proposals

A2U2 [328] is a domain specific stream cipher. It was designed for the extremely resource limited environment of printed electronic RFID tags. In this application field, the area occupied for security has to be, at most, around 500 GE, while the power consumption is limited to a few tens of μWs. Throughput should also be reasonable to permit real time interactions with a large numbers of tags.

A2U2 is a synchronous stream cipher that uses 56-bit keys. Principles from both stream and block ciphers were taken into account during its design phase. In order to achieve a small hardware area, A2U2 uses short-length registers supported by compact functional blocks and reuse of hardware components. Its implementation is strongly based on efficient hardware design principles introduced by the KATAN [216] block cipher. More specifically, it adopts an LFSR-based counter and a combination of two NSFRs, as proposed by KATAN. The LFSR operates as a counter during the initialization process and then continues to operate as an LFSR. The feedback function of each NFSR provides the feedback to the other NFSR. Moreover, A2U2 uses irregular changes in the feedback and filter functions.

A2U2 is the most compact stream cipher of the ones examined in this study. It was estimated that the smaller version would require 284 GE and achieve 50 Kbps throughput.

However, an ultra-efficient chosen plaintext attack is presented in [380], which fully recovers the secret key of A2U2. It queries the encryption function on the victim tag twice and solves 32 spare systems of linear equations; it only takes 0.16 seconds to break the cipher on a laptop computer.

WG-7 [310] is an updated version of the WG cipher which was a candidate in eSTREAM Profile 2 and which was faster than the other candidates and consumed less memory. WG-7 uses 80-bit keys, 81-bit IVs and is parameterized for RFID tags. It is a synchronous stream cipher and utilizes an LFSR of 23 stages over finite fields F27. The LFSR feeds a nonlinear filtering function which is based on a WG transformation.

However, a distinguishing attack is presented in [381]. The polynomial of the LFSR allows an attacker to distinguish the keystream that is generated by the cipher from a truly random keystream due to the small number of tap positions in the LFSR.

WG-8 [311] is an updated version of WG-7 that counters the abovementioned attack and improves its performance. It uses 80-bit keys and IVs, while the characteristic polynomial of the LFSR consists of 8 tap positions (F28). WG-8 inherits the good randomness properties of the WG cipher family and is resistant to most common attacks against stream ciphers. The cipher performs well in embedded software and exhibits low energy consumption. In comparison to other lightweight stream ciphers, WG-8 is 2-15 times faster while consuming 2-220 times less energy. A key recovery attack requires 253 chosen IVs to recover the key with 253.32 complexity, for specifically selected key-IV pairs [382]. The provided security level is still adequate for WSN and RFID tags when these key-IV pairs are avoided.

CAR30 [321] is a novel stream cipher that, like CAvium, makes use of CA. It utilizes the classical Rule 30 of CA along with a maximum length linear hybrid CA. This combination of a linear and a non-linear CA, and their operation over a number of rounds, reduces the linearity with the adjacent sequence at the production of the keystream, alleviating the relevant security issues of CA-based ciphers. Moreover, the proposed solution can use different key and IV sizes without changing the design and the structure of the cipher. CAR30 provides configurable security and extensibility to any key and IV length. The proposed setting uses 128-bit keys and 120-bit IVs for producing 128-bit blocks of keystream in each iteration. Its statistical randomness properties were successfully evaluated against the NIST statistical test suite for random and pseudorandom number generators for cryptographic applications [383]. No independent cryptanalysis results are presented.

The design can be implemented efficiently in both hardware and software. In hardware, CAR30 achieves higher throughput than Grain and Trivium. In software, it is faster than Rabbit. Moreover, the initialization process is fast (160 cycles), which makes CAR30 suitable for encrypting small messages.

TinyStream [325] is a novel 128-bit stream cipher for wireless sensor networks (WSN), based on TinySec [384]. It employs a TPM to achieve keystream randomness, which is the main feature of the cipher. The 128-bit keystream is generated for each state rotation and this approach passes the full ENT randomness test [385]. In software, it requires 65024 bytes of ROM and 1659184 bytes of RAM. TinyStream is more efficient than TinySec in terms of computation and power consumption. No independent cryptanalysis results are presented.

Lightweight authenticated encryption

Authenticated encryption (AE) is a cryptographic operation that simultaneously provides confidentiality, integrity and authenticity over processed data. The encryption produces the ciphertext along with an authentication tag. The decryption is combined in a single step with integrity validation. It retrieves the plaintext and produces an error if the authentication tag doesn't match the ciphertext. Figure 55, illustrates the generic AE scheme. Authenticated encryption is imperative in communication protocols, especially in on-line applications, in order to prevent an attacker from intercepting, tampering, or submitting ciphertexts to the receiver. If the attacker lunches such attacks (e.g. chosen ciphertext attacks), he can decrypt messages and completely reveal the communication data. AES-GCM is currently the most accepted solution for authenticated encryption in mainstream applications. The evolution of pervasive and ubiquitous computing leads to the development of relevant lightweight schemes for resource constrained devices. The CEASAR competition is expected to advance our knowledge in deploying secure and efficient solutions both for mainstream and lightweight schemes.

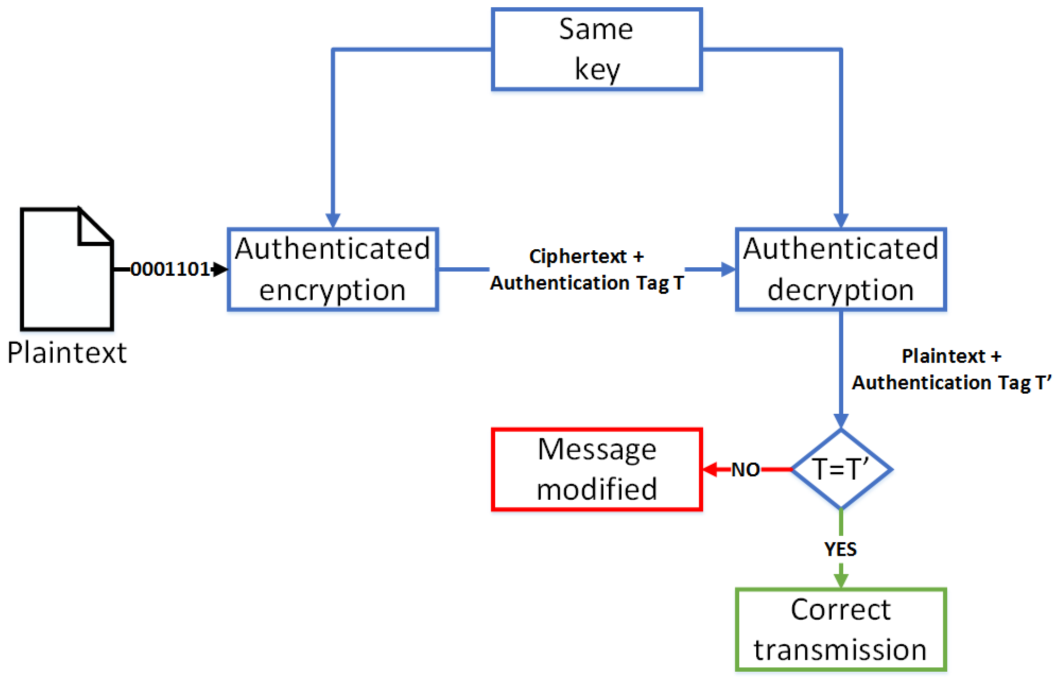


Figure 55 Authenticated encryption scheme

WG-7 was presented along with a mutual authentication protocol [310] between a reader and several RFID tags, which was based on the cipher. The protocol aims to provide un-traceability, resistance to tag impersonation, reader impersonation and Denial of Service attacks. However, as mentioned above, the cipher itself is vulnerable to attacks.

Hummingbird [218] is an ultra-lightweight cipher with a hybrid structure of block and stream cipher, with 256-bit key and 16-bit block sizes. It features better performance than PRESENT on 4-bit microcontrollers. However, it was found to be vulnerable to some types of attacks [279]. Its successor, **Hummingbird-2** [98], was developed with both software and hardware lightweight implementations in mind. It can optionally produce a MAC for each message processed, which can be used to form an authentication protocol fulfilling the requirements of the ISO 18000-6C protocol [386]. Hummingbird-2 has a 128-bit key and a 64-bit IV and, as its predecessor, is targeted to low-end microcontrollers and hardware implementations in resource-constrained devices, such as RFID tags and wireless sensors. However, its security can be compromised by a related-key attack on the full cipher [280].

Grain-128a [99] is an updated version of the Grain-128 cipher, enhancing the security of the original proposal and having built-in support for authentication. It uses slightly different non-linear functions, in order to counter the attacks reported on Grain-128. Grain-128a utilizes 128-bit keys and 96 bit IVs, and supports variable tag sizes of up to 32 bits. It consists of an LFSR, an NFSR and a pre-out function. The shift registers are clocked regularly and the cipher outputs one bit per cycle or one bit per 2 cycles when authentication is used. In hardware, the smaller, 32-bit tag, version requires about 2769.5 GE. Without authentication, the cipher requires 2145.5 GE, while the original Grain-128 requires 2133 GE under the same conditions.

Other than the aforementioned attack to the Grain family ciphers [369], a differential fault attack is also feasible, as presented in [387]. Said attack recovers the key of Grain-128a by observing the correct and faulty MACs of specific chosen messages. The attack exploits certain properties of the Boolean functions and the relevant choices of taps from the LFSR. The attack requires less than 211 fault injections and invocations of less than 212 MAC generation routines.

Rabbit-MAC [388] is a newly proposed scheme for authenticated encryption based on the stream cipher Rabbit. It is designed for WSNs and provides authenticity, integrity, and confidentiality at the link layer. Thus, except from end-to-end security between two end nodes of the WSN, Rabbit-MAC also achieves peer-to-peer security among the intermediate nodes of a communication path. The scheme assumes that the symmetric keys have been pre-distributed by a public-key cryptosystem. The IVs are included in the packet header to reduce the communication overhead introduced from exchanging secure IVs; moreover, these are unique for each different message to enhance security. The MAC component utilizes the next-state function of the cipher to preserve the implementation requirements; it follows the Encrypt-then-MAC approach [389] and its value is computed over the encrypted data and the packet header. The scheme fulfils the security requirements and is energy efficient. No independent cryptanalysis results are presented. However, we consider that the scheme is affected by the aforementioned practical fault analysis attacks that is performed in the Rabbit cipher [360].

ACORN [390] is a newly proposed stream cipher for lightweight authenticated encryption, and one of the ciphers submitted in CAESAR. ACORN uses a 128-bit key and IV and produces an authentication tag of a maximum of 128 bits. The cipher takes 1792 steps to initialize, while the state's size is 293 bits and consists of six concatenated LFSRs. There are three functions for generating the keystream bit from the state, computing the overall feedback bit, and updating the state respectively. The tag is generated after processing all plaintext bits. The decryption follows the same approach. In order to counter known-plaintext or chosen plaintext attacks the ciphertext and the tag are not given as output when the verification fails.

ACORN is bit-wise and is efficient in both hardware and software. The hardware cost is slightly higher than Trivium. The software implementation is fast and, moreover, 32 steps can be computed simultaneously and the cipher can be parallelized. In comparison to AES-GCM, ACORN is more hardware efficient. In software, it produces smaller code size and is more efficient in general computing devices without AES-NI [391] capability.

Automated analysis on CAESAR candidates indicates that ACORN represents significantly elevated adaptive chosen plaintext attack risks [392], but no attack is presented so far.

Sablier [316] is another CAESAR candidate. It is a hardware-oriented stream cipher with built-in authentication. The latter does not decelerate encryption due to a carefully designed leak extraction strategy that is applied on the internal state. The cipher uses 80-bit keys and IVs. It takes 64 rounds to initialize and produced tag is 32 bits. A new internal structure is proposed to produce the keystream, consisting of only bitwise XOR, bitwise logical AND, and bitwise intra-word rotations. Sablier’s constrained device hardware implementation is 16 times faster than Trivium and requires 1925 GE. A practical state recovery attack for Sablier v1 is presented in [393].

ASC-1 [394] is an authenticated encryption stream cipher that utilizes AES with 128-bit key as a building block. In more detail, it uses 128-bit keys to encrypt three 128-bit blocks of plaintext, using a 56-bit IV for each block. Thus, the plaintext is processed in a CFB-like mode. However, this encryption strategy is also its main drawback as it only encrypts messages that consist of three 128-bit blocks. Also, its designers consider that the cipher is safe when the security problem is bounded to the case of distinguishing if the round keys are uniformly random or are generated by a key scheduling algorithm.

ALE [395] is another AES-based lightweight authenticated encryption scheme. It is also inspired by ASC-1 and the stream cipher LEX [396] (an eStream candidate based on AES). ALE uses 128-bit keys and IVs and encrypts plaintext of up to 245 bytes. It utilizes AES in order to benefit from the high security of the cipher and the performance of the AES-NI assembly instruction set. However, its security has already been compromised by cryptanalysts [397], [398].

A.4.4 Evaluation

<u>Discussion</u>

RC4, A5/1, E0, Trivium, Rabbit, SOSEMANUK, MICKEY 2.0, BEAN, WG-7, A2U2, ALE, Sablier, and the Grain and Hummingbird cipher families are vulnerable to attacks or not recommended for use in new applications and, thus, should be avoided. ASC-1 security is bounded as it requires a mechanism to ensure that round keys are uniformly random. Concerning the newer cipher proposals, i.e. Quavium, CAvium, CAR30, and TinyStream, further independent cryptanalysis must be conducted to verify their security properties before they can be endorsed.

AES-CTR is the typical and widely-accepted choice for stream encryption. It is efficient on many embedded devices, and, where applicable, it should be preferred. It is used as a benchmark for stream encryption and its performance is the aim of new proposals.

However, AES-CTR is not always an option, especially in constrained devices (e.g. low-power embedded systems and RFIDs) and in applications where high throughput is required. Enocoro can be applied in such cases. The cipher is designed and supported by a large industrial entity in embedded electronics and is a standardized cipher for LWC. It is efficient in hardware and performs reasonably well in software. Enocoro is optimized for low power consumption and can be used in compact control equipment and low cost sensors. The encryption process consumes 1/10th the amount of power of AES and is faster than AES-CTR. For RFID tags, WG-8 is another attractive solution, as it is fast and consumes low energy.

Cryptographic libraries suitable for embedded systems, like Crypto++ and CyaSSL, adopt the eSTREAM Profile 1 finalists Salsa20 and HC. Both of them were designed to outclass the AES-CTR in software. The main advantage of these ciphers is the fast encryption stage. Salsa20 is fast and has a short initialization phase, which makes it suitable for Internet protocols and applications where a large number of packets are processed. On the other hand, HC can be parallelized, and is thus appropriate for parallel computing applications and superscalar microprocessors [399]. The cipher excels in domains where large streams must be processed.

For authenticated encryption, ACORN is an attractive choice, as it surpasses AES-GCM in embedded hardware and software. It can be parallelized, complying with the latest norm of parallel computing, and, as AES, supports the AES-NI for higher efficiency. As a candidate in the CEASAR competition, it has undergone detailed cryptanalysis, enhancing its acceptability status. For WSNs and RFID tags, WG-8 is the best choice as it fortifies security and improves energy efficiency. It is specifically designed with wireless sensor and RFID applications in mind. It has short initialization phase and fast de/encryption process making it suitable for encrypting many short messages, as is usually the case in these domains.

Self-synchronizing ciphers seems to be of no interest to the industry and there is limited effort in designing such ciphers. Secure synchronous ciphers constitute the main candidates for embedded systems. The designing of the non-linearity part is an important factor along with efficient sophisticated functions for computing the algebraic immunity of the cipher for a large number of inputs.

In authenticated encryption schemes the production of a robust authentication tag is also a core target. Other than the discrete attacks on the cipher or the tag themselves, attention must also be paid in types of attacks that exploit vulnerabilities in both primitives to disclose data (e.g. [387]). Several schemes that encrypt the message and produce the MAC simultaneously are vulnerable to attacks (e.g. [218], [99], [395]). To this end, the encrypt-then-MAC approach (where the message is encrypted first and then the authentication tag is produced) is widely adopted (e.g. [388], [390]).

Nevertheless, it should be pointed out that choosing a secure cipher does not guarantee the security of the underlying application. As practice has shown, the design of the protocol itself is equally important. Potential oversights in protocol design can compromise the security of the implementation, a characteristic case of this being the WEP protocol.

Performance evaluation

Providing a fair comparative analysis of the presented ciphers is not a trivial task, as they are implemented in different platforms and the measured parameters aren't correlated in a straightforward manner. However, the authors held a constrained benchmark analysis in hardware and software, focusing on the secure ciphers identified above (AES, Enocoro, WG-8, Salsa20, HC). The comparison contributes to the classification and the conclusions mentioned above.

In general, the new ciphers are designed with AES-CTR performance in mind and are more efficient. However, the confidence on AES and its tested robustness is an obstacle new proposals have to overcome in order to be widely adopted.

Hardware

The potential hardware solutions are limited, as presented in the previous subsections. The authors implement the five ciphers in Verilog. The cipher designs are evaluated on a low-cost Xilinx Spartan3 XC3S1000 FPGA (7680 slices, 630 MHz, 55KB RAM).

The metrics of throughput, latency, maximum frequency, power, and occupied flip-flops (FF) and slices are used to characterize each cipher. Throughput counts the Mbps that the cipher's encryption operation achieves. Latency is the number of clock cycles that are required to process a single block. Maximum frequency in MHz is the frequency limit that is accomplished by an implementation. The Xilinx Power Estimator is utilized for the power consumption estimation in Watts. FF constitute the amount of memory elements that are used. Slices represent the hardware area of the implementation, consisting of all the memory and computational components. The overall efficiency and performance/size tradeoff is estimated by the throughput/slices metric (the higher the value, the better).

Table 21, summarizes the best FPGA implementations. The implementations were based on the latest embedded hardware designs for each cipher (AES [179], Enocoro-128 [331], WG-8 [311], Salsa20 [300], HC-256 [300]). The authors implement the most efficient variants of each cipher, based on the throughput/slice metric.

Table 21 Hardware implementations of stream ciphers on FPGA

Cipher	**Key size (bits)**	**Block size (bits)**	**IV (bits)**	**Latency (cycles/ block)**	**Throughput (MBps)**	**Max frequency (MHz)**	**Power (W)**	**FFs**	**Slices**	**Throughput / Slices**
Better is				**Lower**	**Higher**		**Lower**	**Lower**	**Lower**	**Higher**
WG-8	80	1	80	1	2112	192	0.016	207	398	0.66
Enocoro-128	128	1	64	1	900	118	0.008	239	292	0.38
Enocoro-128	128	1	64	1	1200	149	0.008	362	442	0.33
AES	128	128	-	226	8754	77	0.078	781	5948	0.18
WG-8	80	11	80	1	190	190	0.005	85	137	0.17
Salsa20	256	32	64	2	990	19.4	0.012	1286	2036	0.06
HC-128	128	512	128	-	-	-	-	-	>> 262000	-

The relevant features in ASIC are also summarized, based on the original implementation details. The metrics of throughput, gate equivalent (GE) and figure of merit (FOM) [403] are used to characterize each cipher in ASIC. The throughput metric is the same as in FPGA. GE corresponds to the number of logic gates that are required to implement the cipher and represents the complexity and occupied area (similar to the slices metric in FPGAs). FOM is calculated as $throughput/GE^2$. It is aggregate metric for estimating the size and performance tradeoff (as the throughput/slices in FPGAs. The most efficient implementations in ASIC are presented by the FOM metric (the higher the value, the better). Table 22, summarizes the best ASIC implementations.

Table 22 Hardware implementations of stream ciphers on ASIC

Cipher	**Key size (bits)**	**Block size (bits)**	**IV (bits)**	**Latency (cycles/block)**	**Throughput at 100 KHz (MBps)**	**Tech (μm)**	**Area (GE)**	**FOM**
Better is				**Lower**	**Higher**		**Lower**	**Higher**
WG-8	80	11	80	1	1100	0.65	3942	0.00884
Enocoro-128	128	1	64	1	800	0.09	4100	0.00594
WG-8	80	1	80	1	100	0.65	1786	0.00391
Enocoro-80	80	1	80	1	-	0.09	2700	-
AES	128	128	-	226	56.64	0.35	2400	0.00122
Salsa20	256	32	64	2	99	0.13	12126	0.00008
HC-256	256	-	256	-	-	0.13	>> 524000	-

WG-8, Enocoro, and AES achieve lightweight implementations that are suitable for embedded systems (less than 3000 GE). WG-8 is the most compact and more efficient. Enocoro also performs well and is the only secure standardized stream cipher for LWC. AES has moderate performance but high power consumption. The secure eStream ciphers, Salsa20 and HC, produce large implementations and are not appropriate for embedded hardware. The internal memory of HC128 alone requires around 524KB, resulting in about 262000 slices in FPGA or 524000 GE in ASIC.

Software

Similarly, in software, the ciphers are implemented in C and evaluated on a low-cost credit-card-sized BeagleBone embedded device (AM3359 ARM Cortex A8 single core CPU, 720 MHz, 256MB RAM, Ubuntu OS). All implementations are assessed over a common benchmark suite.

The metrics of throughput, ROM, RAM, and combine metric (CM) [160] were used. Throughput is the same as in hardware. ROM and RAM measures the code and runtime memory requirements in KB. CM is calculated as $(ROM \times encryption\ cyles) / block\ size$.

The software implementations are based on the latest embedded software designs for each cipher (AES [400]/AES-CTR [331], Enocoro-128 [331] WG-8 [311], Salsa20 [298], HC-128 [400]). The best reported variants of each cipher are implemented, based on the CM metric. Table 23, aggregates the most attractive software implementations.

Table 23 Software implementations of stream ciphers on BeagleBone embedded devices

Cipher	**Key size (bits)**	**Block size (bits)**	**IV (bits)**	**Initialization (cycles)**	**Encryption (cycles)**	**ROM (KB)**	**RAM (KB)**	**Throughput at 720 MHz (MBps)**	**CM**
Better is				**Lower**	**Lower**	**Lower**	**Lower**	**Higher**	**Lower**
Salsa20	128	512	64	460	29491	4.8	8.27	12.5	276
Salsa20	256	512	64	460	29729	4.8	8.35	12.4	278
Enocoro-128	128	1	64	4870	138	3.8	7.56	5.2	526
WG-8	80	1	80	1379	69	6.6	0.59	10.4	456
HC-128	128	512	128	2082876	17388	77.2	16.58	21.2	2621
AES	128	128	-	6953	20480	22.2	88	4.5	3552
AES-CTR	128	128	128	469.6	21942	22.3	88.5	4.2	3822

Salsa20 achieves the best performance, being fast and with a short initialization phase. Enocoro and WG-8 also perform well with low code and memory requirements. HC is the fastest cipher, while AES is the slowest cipher.

A.4.5 Future Work

Key drivers in stream cipher evolution come from the competition with block ciphers and the intrinsic requirements of different application domains. Stream ciphers will remain in the foreground due to their high efficiency and ability to destroy statistical properties in natural language, in contrast to block ciphers. The evolution of the Internet-of-things implies the interconnection of a large number of embedded devices, usually with constrained computational capabilities, and their interaction with users [141], [142]. Smart cites and social mobility applications are expected to include distributed structures to process and transmit high amounts of streams. Examples of industrial focus include but are not limited to telephony, urban surveillance with smart cameras, vehicular ad hoc networks (VANET), green networking, and ambient environment monitoring. Although stream ciphers seem the natural choice for streaming applications, ongoing research should enhance their usage over well-analyzed and reputable block ciphers. In hardware, LFSR and NFSR-based designs are more efficient. In software, table-driven or ciphers with chaotic and/or LFSR structures are appropriate.

Parallel computing architectures and models are widely used today and are often appropriate for embedded applications. Thus, secure stream ciphers with a higher degree of parallelism constitute a desirable goal.

A.5 Hash Functions

Hash functions are another research field of LWC. The standardized or widely-used MD5 (8001 GE) [401], SHA-1 (5527 GE) [402], and SHA-2 (10868 GE) [401] and ARMADILLO (4353 GE) [403] are too large to fit in hardware constrained devices (i.e. more than 3000 GE).

After the release of the PRESENT cipher there were many efforts to build novel lightweight hash functions based on PRESENT design principles [404], like C-PRESENT (4600 GE), H-PRESENT (2330 GE), and PRESENT-DM (1600 GE).

The NIST's SHA-3 competition [405] in 2012 defined a new function to replace the older SHA-1 and SHA-2. The finalists [406] were Blake, Grostl, JH, Skein, and Keccak, with the latter being the winner. Unfortunately, the SHA-3/Keccak and the other finalists are not much more compact than the previous SHA functions. At this time, all SHA-3 finalists require more than 12000GE for 128 bit security. The SHA-3 competition has helped our understanding of hash functions significantly and led to a new design trend of hash functions with sponge constructions. Keccak is such a function and a lightweight implementation of a constrained Keccak version [407] was later announced at 2520-5090 GE.

Other new lightweight hash functions with sponge constructions are SQUASH (6328 GE) [408], GLUON (2071 GE) [409], Quark (1379 GE) [410], Photon (1120 GE) [411] and SPONGENT [412]. SPONGENT is the most lightweight hash function family known so far. Its smallest implementations require 738, 1060, 1329, 1728 and 1950 GE for 88, 128, 160, 224 and 256 bit respectively. It is based on a sponge construction instantiated with a PRESENT-type permutation, following the hermetic sponge strategy.

All the state of the art hash functions that are mentioned should be extensively tested for security vulnerabilities before being widely used.

A.5.1 Evaluation

In hardware, hash functions perform efficiently in low-cost and lightweight devices. For ultra-lightweight devices the hash functions SPONGENT and PHOTON are implemented. SPONGENT is the most lightweight choice but PHOTON produces higher FOM. Targeting low-cost devices, the hash functions DM-PRESENT, D-QUARK, and U-QUARK are implemented (as well as the ones for ultra-lightweight devices). SPONGENT is the best while DM-PRESENT, D-QUARK, and U-QUARK produce worse FOM metrics. For lightweight devices, the hash functions H-PRESENT, Keccak, S-QUARK and SQUASH, in addition to the ones for ultra-lightweight and low-cost devices. The hash function DM-PRESENT achieves by far the best FOM for all the relevant proposals. Keccak as the new SHA-3 function can also be used. S-QUARK and SQUASH produce poor performance.

In software, hash functions are embodied to key exchange schemes and communication protocols that are implemented in cryptographic libraries. Compact libraries, like CyaSSL [426] which is specifically designed for embedded devices, are suitable for lightweight implementations.

A.6 Asymmetric Cryptography

Asymmetric algorithms and protocols must also be adapted to operate on devices with the aforementioned resource limitations. This is an elaborate task, since asymmetric ciphers are computationally far more demanding than their symmetric counterparts and are usually used with powerful hardware. The performance gap is wider on constrained devices such as 8-bit microcontrollers. Even an optimized asymmetric algorithm e.g. elliptic-curve cryptography (ECC) is 100 to 1000 times slower than a standard symmetric algorithm like AES which correlates to two or three orders-of-magnitude higher power consumption.

A.6.1 Traditional Asymmetric Cryptosystems

Traditional public key cryptography is based on one-way trapdoor functions. These functions are based on a set of hard mathematical problems. There are three well established cryptosystems:

1. **RSA, Rabin [413]:** based on the Integer Factorization Problem (FP)
2. **ECC [414] / HECC [415]:** based on Elliptic Curve Discrete Logarithm Problem (ECDLP)
3. **ElGamal [416]:** based on the Discrete Logarithm Problem In Finite Fields (DLP)

RSA is the most popular algorithm for asymmetric cryptography and supports key sizes from 1024 to 4096 bits. As such, it is used as a benchmark for the various public key cryptosystems researchers propose. However its large hardware footprint and its resource demanding implementations led researchers to seek for other algorithms for applications in constrained devices.

Rabin is quite similar to RSA. One main difference is the complexity of the factorization problems that they rely upon. Rabin is proven to be as hard as the integer factorizations problem, while RSA is not. Also, the encryption for Rabin is faster but the decryption is less efficient. WIPR [413] is a low-resource implementation of Rabin in hardware. The implementation shares several architectural principles with the SQUASH hash function. It requires 4682 GE and fits on RFID tags and wireless sensor nodes. BluJay [417] is a hybrid Rabin-based scheme that is suitable for lightweight platforms and is based on WIPR and Hummingbird-2. The encryption is significantly faster and more lightweight than RSA and ECC for the same level of security. The hardware implementation requires less than 3000 GE.

ECC [414] and HECC [415] are considered the most attractive cryptosystems for embedded systems. They present smaller operand lengths and relatively lower computational requirements. Their main advantage is the fact that for the same level of security they offer shorter keys compared to RSA, which leads to smaller internal state requirements. As the level of security increases, RSA key sizes grow much faster than ECC. ECC also produces lightweight software implementations due to its memory and energy savings. The most known software implementations [418] are the TinyECC and the WMECC.

HECC is a generalization of elliptic curves. A hyper elliptic curve of genus 1 is an elliptic curve. As the genus increases, the arithmetic of encryption gets more complicated, but it needs fewer bits for the same level of security. HECC's operand size is at least a factor of two smaller than the ECC one. The curves of genus 2 are of great interest for the research community as higher genus curves suffer from security attacks. HECC has better performance than ECC and is more attractive in resource constrained devices.

ElGamal [416] is of no interest for resource constrained platforms. The computation is more intensive than RSA and encryption produces a 2:1 expansion in size from plaintext to ciphertext. It is also considered vulnerable to some types of attacks, like chosen ciphertext attacks.

A.6.2 Alternative Asymmetric Cryptosystems

Alternative public key cryptosystems (APKCs) [419] that are based on other mathematical features have become popular due to their performance and their resistance against quantum computing. These alternative cryptosystems are based on:

- **Hash-Based Cryptography:** The Merkle signature scheme (MSS) [420] is a crypto-system which uses typical hash functions
- **Lattice-Based Cryptography:** NTRU [421] is the most popular scheme which is based on the Shortest Vector Problem
- **Code-Based Cryptography:** McEliece [422] is a popular scheme based on error-correcting codes
- **Multivariate-Quadratic (MQ) Cryptography:** MQ [423] is based on the problem of solving multivariable quadratic equations over finite fields

An MSS implementation with the AES-based hash function [420] has smaller code size and faster verification process than RSA and ECC. Moreover, the signature generation is faster than RSA and comparable to ECC. MSS may gain ground in lightweight asymmetric cryptosystems due to the evolution of lightweight hash function design.

NTRU [421], [424] is the most promising cryptosystem of all APKCs. Encryption and decryption use only simple polynomial multiplications, which makes them very fast compared to traditional cryptosystems. NTRU is highly efficient, suitable for embedded systems and provides a level of security comparable to RSA and ECC. In hardware implementations [424], NTRU is 1.5 times faster compared to ECC for the same level of security and only has 1/7 of its memory footprint. The hardware implementation requires almost 3000 GE. In software implementations [421], NTRU is 200 times faster in key generation, almost 3 times faster in encryption and about 30 times faster in decryption compared to RSA. On the other hand, NTRU produces larger output, which may impact the performance of the cryptosystem if the number of transmitted messages is crucial. It is considered safe when the recommended parameters are used [425]. NTRU can be efficiently used in embedded systems because of its easy key generation process, its high speed and its low memory usage. The system is now adopted by the IEEE P1363 standards under the specifications for lattice-based public-key cryptography as well as IEEE P1363.1 and ANSI X9.98 Standard for use in the financial services industry.

The main drawback of McEliece [422] and MQ [423] cryptosystems is the use of large keys. In comparison to 1924 bit RSA, MQ requires 9690 bytes for the public key and 879 bytes for the private key. Key sizes impact on the computations that are performed, the speed, the key storage and the output's size. The advantage of these systems is the fast encryption and decryption process that makes them suitable for high performance applications where messages must be assigned in real time.

A.6.3 Evaluation

In hardware, asymmetric cryptography is feasible only in lightweight devices. For ultra-lightweight and low-cost devices the key establishment mechanisms based only on symmetric cryptography can be applied. For lightweight devices, the asymmetric cryptosystems NTRUencrypt and GPS-4/4-F are implemented. For the asymmetric cryptosystems, NTRU appears to be the most suitable.

In software, asymmetric cryptography materializes specific key exchange schemes and communication protocols, like SSL. Due to the complexity of implementing this functionality, cryptographic libraries are utilized to enhance the robustness of an application. Hash functions are embodied to these schemes and protocols.