ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ασφαλή Ερωτήματα Σημείου στο Σύστημα CryptDB»

Χριστίνα Μαντωνανάκη

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Αντώνιος Δεληγιαννάκης, Αναπληρωτής Καθηγητής (Επιβλέπων)

Μίνως Γαροφαλάκης, Καθηγητής

Ιωάννης Παπαευσταθίου, Αναπληρωτής Καθηγητής

ΧΑΝΙΑ 2017

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας αισθάνομαι την ανάγκη να ευχαριστήσω κάποιους ανθρώπους, που ο καθένας με τον δικό του τρόπο βοήθησε στην εκπόνησή της.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή κ. Δεληγιαννάκη Αντώνιο, για την εμπιστοσύνη που μου επέδειξε στην ανάθεση του θέματος και την ευκαιρία να ασχοληθώ με ένα πολύ ενδιαφέρον θέμα, που μου χάρισε πολύτιμες γνώσεις σε αυτόν τον τομέα.

Στη συνέχεια, θα ήθελα να ευχαριστήσω τα υπόλοιπα μελή της εξεταστική επιτροπής, και συγκεκριμένα τον Καθηγητή κ. Γαροφαλάκη Μίνω και τον Αναπληρωτή Καθηγητή κ. Παπαευσταθίου Ιωάννη.

Ανεκτίμητη επίσης ήταν τόσο η βοήθεια όσο και η συμπαράσταση κάποιων ατόμων που στάθηκαν δίπλα μου όλα αυτά τα χρόνια σπουδών και θα ήθελα να τα ευχαριστώ από τα βάθη της καρδίας μου.

Ολοκληρώνοντας, ένα *μεγάλο ευχαριστώ* στους γονείς μου για την αμέριστη στήριξη και αγάπη που μου δίνουν όλα αυτά τα χρόνια. Το μεγαλύτερο ευχαριστώ όμως το οφείλω στην αδερφή μου **Αντωνία** για την πολύτιμη πολύπλευρη βοήθεια της όλα αυτά τα χρόνια και γιατί μου χάρισε τις ωραιότερες παιδικές αναμνήσεις.

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία εξετάστηκε ο τρόπος αναζήτησης ερωτημάτων σημείου (Point queries) πάνω στα δεδομένα που έχουν ανατεθεί σε ένα μη-αξιόπιστο διακομιστή (Server). Αρχικά, παρουσιάστηκε το σύστημα CryptDB και η λειτουργικότητα του, ενώ στη συνέχεια η Searchable Symmetric Encryption (SSE) τεχνική. Έτσι, εξετάστηκε το κατά πόσο μπορεί να τροποποιηθεί αυτό το σύστημα και να υποστηρίζει την τεχνική SSE. Επιπλέον, παρουσιάζονται οι τροποποιήσεις στην αρχιτεκτονική, την εισαγωγή δεδομένων και την εκτέλεση των ερωτημάτων. Η προσαρμογή του SSE στο CryptDB έγινε διότι, η κρυπτογράφηση που υποστήριζε point queries ερωτήματα στο σύστημα ήταν αποτελεσματική στις απαντήσεις της, αλλά είχε μη αποδεκτές διαρροές. Με αυτό τον τρόπο μπορούν να εκτελεστούν ερωτήματα point queries, χωρίς να αποκαλύπτεται η συχνότητα των κρυπτογραφημένων δεδομένων, καθώς με την κρυπτογράφηση DET δυο κείμενα (plaintext) είχαν ίδιο κρυπτογραφημένο κείμενο. Στην υλοποίηση της εργασίας μπορεί να θεωρηθεί οποιαδήποτε τεχνική SSE ως ένα μαύρο κουτί, και το αρχείο (file) να μετατραπεί σε πλειάδα (tuple), ενώ η λέξη κλειδί (keyword) σε δεδομένα του tuple. Επομένως, το σύστημα αυτό θα μπορέσει να λειτουργήσει αποτελεσματικά στην απάντηση point queries.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΔΙΑΓΡΑΜΜΑΤΩΝ

1. **ΕΙΣΑΓΩΓΗ**

Στο σημερινό ψηφιακό περιβάλλον οι εταιρίες συσσωρεύουν όλο και περισσότερα δεδομένα από τους χρήστες, τα όποια αποθηκεύουν σε βάσεις δεδομένων. Έχει παρατηρηθεί ότι σχεδόν κάθε χρήστης στο διαδίκτυο είναι εγγεγραμμένος σε περισσότερες από μια βάση δεδομένων. Έτσι, το γεγονός ότι αποθηκεύονται τεράστιες ποσότητες προσωπικών στοιχείων στις βάσεις δεδομένων, τις καθίστα έναν ελκυστικό στόχο για τους επιτιθέμενους που θέλουν να μάθουν ή να τροποποιήσουν στοιχεία.

Η ανάγκη για προστασία των δεδομένων οδήγησε πολλά συστήματα στο να εφαρμόσουν κρυπτογράφηση και να αποθηκεύσουν κρυπτογραφημένα στοιχεία στο Server. Κάποια συστήματα που εφαρμόζουν αυτή την τεχνική είναι τα CryptDB, SPORC, MONOMI κ.α. [2]. Συγκεκριμένα για το σύστημα CrypDB θεωρείται ως υπόθεση ότι όλα τα δεδομένα του διακομιστή (Server) θα διαρρεύσουν, ενώ στόχος του είναι η προστασία της εμπιστευτικότητας των δεδομένων, ακόμη και σε μη επιθυμητές περιπτώσεις. Η διαδικασία που ακολουθείται είναι η αποθήκευση, καθώς και η επεξεργασία των κρυπτογραφημένων δεδομένων στο Server, ώστε τα ερωτήματα που λαμβάνονται από τον χρήστη να υπολογίζονται αποτελεσματικά. Μέσω αυτού εξασφαλίζεται η εμπιστευτικότητα των δεδομένων, καθώς ο Server λαμβάνει μόνο ευαίσθητα κρυπτογραφημένα δεδομένα. Έτσι, ακόμα και στην περίπτωση που κάποιος εισβολέας αποκτήσει πρόσβαση στο Server και διαβάσει όλα τα αποθηκευμένα δεδομένα, αυτά είναι κρυπτογραφημένα και ο Server ποτέ δεν λαμβάνει το κλειδί αποκρυπτογράφησης. Τα δεδομένα παραμένουν κρυπτογραφημένα κατά τον υπολογισμό του ερωτήματος.

Το σύστημα CryptDB για να επιτύχει όλα τα παραπάνω, προσθέτει έναν έμπιστο διακομιστή μεσολάβησης (proxy-server), ο οποίος είναι υπεύθυνος για την κρυπτογράφηση και αποκρυπτογράφηση, ανάμεσα στο χρήστη και στο Server. Η τεχνική που ακολουθείται σε αυτό το σύστημα ονομάζεται *onions*, τα οποία περιέχουν στρώματα κρυπτογράφησης.

Διαρροές στο σύστημα CryptDB μπορούν να δημιουργηθούν εύκολα ανάλογα με τα ερωτήματα που θέτει ο χρήστης. Για την αντιμετώπιση των διαρροών σε ερωτήματα σημείου χρησιμοποιείται η τεχνική Searchable Symmetric Encryption (SSE). Σύμφωνα με αυτή τη τεχνική, ένας χρήστης μπορεί να κρυπτογραφεί δεδομένα με τέτοιο τρόπο, ώστε να μπορεί αργότερα να πραγματοποιήσει αναζήτηση λέξεων-κλειδιών μέσω «search token». Επιπλέον, η τεχνική SSE βασίζεται στους αλγορίθμους *KeyGen, Enc, Trpdrm, Search, Dec,* και υποστηρίζει αρχεία ή λέξεις που ενσωματώνονται σε αυτά.

Στην παρούσα διπλωματική εργασία μελετάται η ενσωμάτωση της τεχνικής SSE μέσα στο CryptDB, ώστε να μειωθεί η διαρροή δεδομένων αυτών των ερωτημάτων. Όλη αυτή η διαδικασία γίνεται μέσω του συστήματος CryptDB. Η παρούσα εργασία θα τροποποιήσει αυτή την περιγραφή, και θα απευθύνεται σε λέξεις-κλειδιά και σε typles αντί για αρχεία.

2. ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

2.1 CryptDB Σύστημα

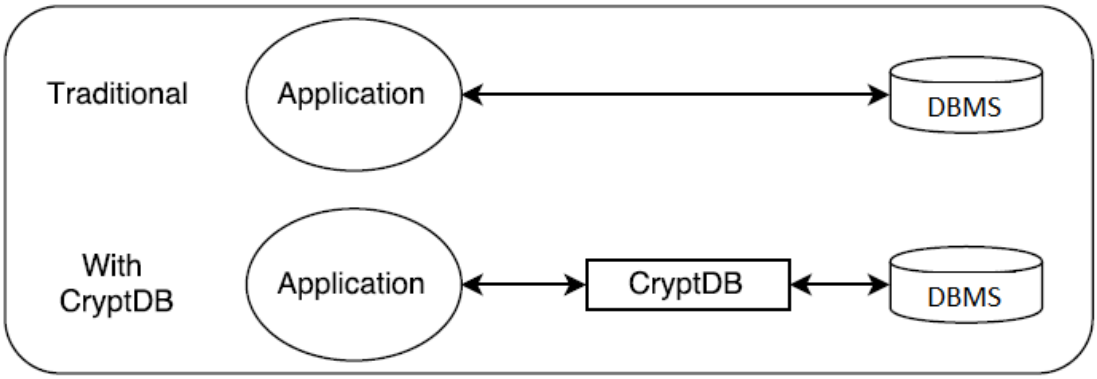
2.1.1 Εισαγωγή στο CryptDB σύστημα

Το CryptDB δημιουργήθηκε από μια ομάδα ατόμων στο πανεπιστήμιο MIT της Μασαχουσέτης, με σκοπό την προστασία των δεδομένων που βρίσκονται στο Server. Τα δεδομένα του Server βρίσκονται σε κρυπτογραφημένη μορφή, δεν περιέχονται κλειδιά που μπορούν να τα αποκρυπτογραφήσουν, και έτσι ο Server δεν έχει τρόπο να αποκρυπτογραφήσει αυτά τα δεδομένα [1].

Η προσέγγιση του CryptDB είναι να εκτελέσει ερωτήματα πάνω από τα κρυπτογραφημένα δεδομένα. Το σύστημα αυτό έχει δύο είδη απειλών. Το πρώτο είδος απειλής είναι το λεγόμενο "snooping" στο *DBMS Server*, με το οποίο ο επιτιθέμενος μπορεί να μάθει ιδιωτικά στοιχεία, όπως πχ. έγγραφα υγείας, οικονομικές καταστάσεις και προσωπικά στοιχεία. Όταν εμφανίζεται αυτό το είδος απειλής, το σύστημα εμποδίζει τη διαρροή των πληροφοριών μέσω της κρυπτογράφησης. Το δεύτερο είδος, το οποίο αναφέρεται πάλι στο "snooping", αφορά την εφαρμογή και το DBMS Server, ταυτόχρονα *(Εικόνα 2)*. Σε αυτήν την περίπτωση το σύστημα δεν μπορεί να παρέχει καμία εγγύηση στους χρήστες που βρίσκονται μέσα στην εφαρμογή (log-in) την ώρα της απειλής. Αντιθέτως, μπορεί να διασφαλίσει την εμπιστευτικότητα στα δεδομένα των χρηστών που δεν είναι συνδεδεμένοι τη στιγμή της απειλής. Οι βασικές ιδέες του συστήματος είναι η εκτέλεση SQL ερωτημάτων πάνω από τα κρυπτογραφημένα δεδομένα, και η ρυθμιζόμενη κρυπτογράφηση βάσει ερωτήματος, συμφώνα με την τεχνική "onions", η όποια θα αναπτυχθεί στη συνέχεια [2], [3].

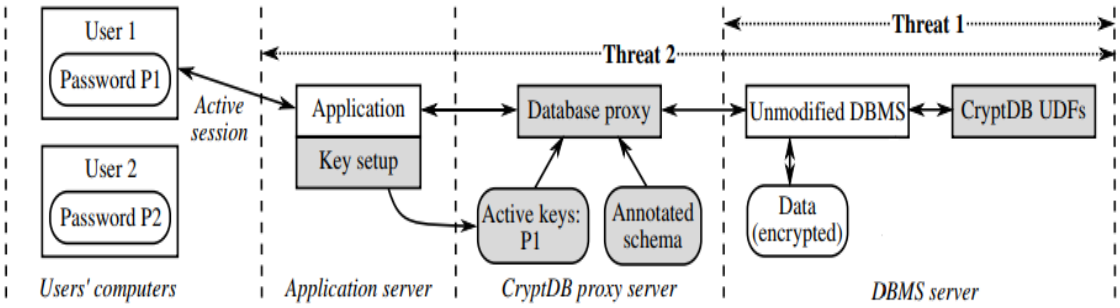
2.1.2 Αρχιτεκτονική του CryptDB συστήματος

Το CryptDB προορίζεται να λειτουργεί ως proxy-server μεταξύ της εφαρμογής και της βάσης δεδομένων, όπως φαίνεται στην *Εικόνα 1* [4]. Παραδείγματα εφαρμογών είναι ένας ιστότοπος, μια εφαρμογή κινητής συσκευής (App), μια κλασική εφαρμογή επιφάνεια εργασίας, και γενικότερα οτιδήποτε συνδέεται με μια βάση δεδομένων.



Εικόνα 1: Σχήμα επικοινωνίας μιας εφαρμογής-DBMS Server με και χωρίς το CryptDB.

Πιο συγκεκριμένα, στο παρακάτω σχήμα *(Εικόνα 2)* παρουσιάζεται αναλυτικά η αρχιτεκτονική του συστήματος. Σύμφωνα με αυτό, το CryptDB χρησιμοποιεί λειτουργίες που ορίζονται από το χρήστη (UDFs) για την εκτέλεση κρυπτογραφικών λειτουργιών στο DBMS. Τα ορθογώνια και στρογγυλά πλαίσια αντιπροσωπεύουν διαδικασίες και δεδομένα, αντίστοιχα. Η σκίαση υποδεικνύει συστατικά που προστέθηκαν από το CryptDB. Από την άλλη, οι διακεκομμένες γραμμές δείχνουν το διαχωρισμό μεταξύ των υπολογιστών των χρηστών, το application server, το Proxy-server και το DBMS Server. Το CryptDB αντιμετωπίζει δύο ειδή απειλών, όπως προαναφέρθηκε, οι οποίες απεικονίζονται σχηματικά στην *Εικόνα 2*. Επιπλέον, φαίνεται πως αποθηκεύει μόνο το σχήμα, καθώς και τα κλειδιά. Καμία άλλη πληροφορία δηλαδή, δεν αποθηκεύεται, και ούτε τα ερωτήματα τα όποια δίνονται από το χρηστή.



Εικόνα 2: Αρχιτεκτονική του CryptDB.

2.1.3 Ερωτήματα πάνω από τα κρυπτογραφημένα δεδομένα

Στην υποενότητα αυτή θα περιγραφεί ο τρόπος με τον οποίο εκτελούνται τα ερωτήματα πάνω στα κρυπτογραφημένα δεδομένα. Το CryptDB επιτρέπει στο DBMS Server να εκτελέσει SQL ερωτήματα στα κρυπτογραφημένα δεδομένα με τέτοιο τρόπο σαν να εκτελούνταν τα ίδια ερωτήματα σε μη κρυπτογραφημένα δεδομένα (plain text). Είναι σημαντικό να αναφερθεί ότι στο CryptDB proxy-server αποθηκεύονται ένα μυστικό κλειδί MK, το σχήμα της βάσης δεδομένων και τα τρέχοντα επίπεδα κρυπτογράφησης όλων των στηλών.

Η επεξεργασία ενός ερωτήματος στο CryptDB περιλαμβάνει τέσσερα βήματα:

1. Η εφαρμογή εκδίδει ένα ερώτημα, το οποίο ο proxy-server επαναγράφει. Κρυπτογραφεί, δηλαδή, κάθε όνομα πίνακα και στήλη, χρησιμοποιώντας το μυστικό κλειδί MK. Επίσης, κρυπτογραφεί κάθε σταθερά στο ερώτημα με το σχήμα κρυπτογράφησης που ταιριάζει καλύτερα στην επιθυμητή λειτουργία.

2. Ο proxy-server ελέγχει εάν θα πρέπει να δοθούν τα κλειδιά στο DBMS Server, ώστε να προσαρμόσει τα επίπεδα κρυπτογράφησης πριν από την εκτέλεση του ερωτήματος. Αν ναι, εκδίδει ένα ερώτημα UPDATE στο DBMS Server που καλεί μια UDF συνάρτηση για να προσαρμόσει το στρώμα κρυπτογράφησης στις επιθυμητές στήλες.

3. Ο proxy-server προωθεί το κρυπτογραφημένο ερώτημα στο DBMS Server, το οποίο εκτελείται χρησιμοποιώντας SQL.

4. Ο DBMS Server επιστρέφει το (κρυπτογραφημένο) αποτέλεσμα ερωτήματος, το οποίο ο proxy-server αποκρυπτογραφεί και επιστρέφει στην εφαρμογή.

Οι βασικοί τρόποι κρυπτογράφησης, τους οποίους χρησιμοποιεί το CryptDB, είναι οι παρακάτω:

Random (RND): Το σχήμα είναι πιθανοτικό, δηλαδή δυο *ίσες τιμές* αντιστοιχίζονται με *διαφορετικά κρυπτογραφημένα κείμενα*. Αυτός ο τρόπος παρέχει τη μεγίστη ασφάλεια στο CryptDB, ενώ δεν επιτρέπει την εκτέλεση αποτελεσματικών υπολογισμών στο κρυπτογραφημένο κείμενο.

Deterministic (DET): Το DET έχει ασθενέστερη εγγύηση, αλλά εξακολουθεί να παρέχει ισχυρή ασφάλεια καθώς διαρρέει μόνο όποιες *κρυπτογραφημένες τιμές* αντιστοιχούν στο *ίδιο plaintext*. Αυτό γίνεται γιατί, δυο ίδια plaintext θα έχουν το ίδιο κρυπτογραφημένο κείμενο. Έτσι, το στρώμα αυτό επιτρέπει στο server να εκτελέσει ερωτήματα ισότητας.

Order-preserving encryption (OPE): Το OPE επιτρέπει τις σχέσεις μεταξύ των δεδομένων που πρέπει να καθοριστούν με βάση την κρυπτογράφηση τους, χωρίς να αποκαλύπτει τα ίδια τα δεδομένα. Δηλαδή, αν $x<y$ τότε $OPE_K(x)<OPE_K(y)$ για κάθε μυστικό κλειδί K. Το στρώμα αυτό μπορεί να εκτελέσει ερωτήματα MIN, MAX κ.α.

Homomorphic encryption (HOM): Το HOM είναι ένα ασφαλές πιθανοτικό σχήμα, το όποιο επιτρέπει στο Server να εκτελέσει υπολογισμούς σε κρυπτογραφημένα κείμενα και να στείλει το αποτέλεσμα στο proxy για αποκρυπτογράφηση. Συγκεκριμένα, μπορεί να εκτελέσει την *πρόσθεση* (add) εφαρμόζοντας το Paillier κρυπτοσύστημα, μετατρέποντας την σε *πολλαπλασιασμό*, καθώς ισχύει η εξής ιδιότητα: $HOM_K(x) * HOM_K(y) = HOM_K(x+y)$.

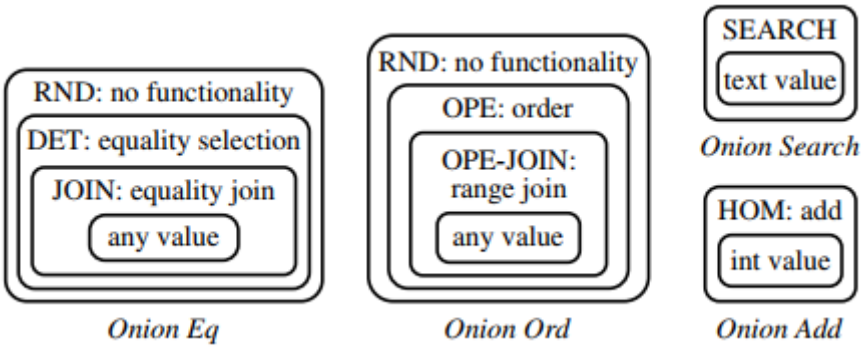
Join (JOIN and OPE-JOIN): Τα σχήματα JOIN και OPE-JOIN είναι και τα δύο "υποσχήματα" των αντίστοιχων DET του OPE. Αυτό σημαίνει ότι και τα δύο διαθέτουν τις υπολογιστικές ικανότητες του "γονικού σχήματος". Το σχήμα αυτό λειτουργεί πάνω από πολλές στήλες, επιτρέπει να προσδιοριστεί αν η τιμή της στήλης a είναι ίση με την τιμή της στήλης b για το JOIN, και αν η τιμή στη στήλη a είναι μεγαλύτερη ή μικρότερη από την τιμή στη στήλη b για το OPE-JOIN.

Word Search (SEARCH): Επιτρέπει την αναζήτηση κειμένου σε επίπεδο λέξεων-κλειδιών με τον χειριστή (operator) LIKE.

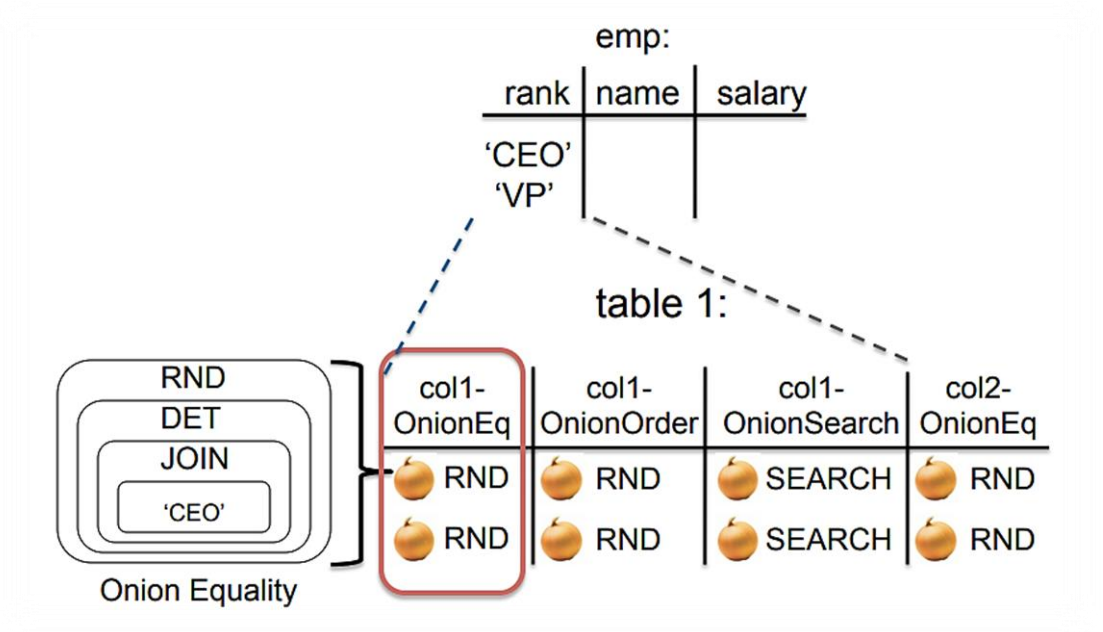
2.1.4 Ρυθμιζόμενη Κρυπτογράφηση Ερωτήματος

Ένα βασικό μέρος του σχεδιασμού του CryptDB είναι η ρυθμιζόμενη κρυπτογράφηση. Στόχος του συστήματος αποτελεί η χρήση των ασφαλέστερων επίπεδων κρυπτογράφησης, τα οποία επιτρέπουν την εκτέλεση του ερωτήματος που ζητήθηκε. Αυτό σημαίνει πως όταν η εφαρμογή δεν έχει υποβάλλει κανένα ερώτημα, η πιο ασφαλής κρυπτογράφηση για αυτά τα δεδομένα είναι η RND. Μόλις εκδοθεί ένα ερώτημα το οποίο περιέχει κάποιο *Where clause,* το RND σχήμα δε θα μπορεί να απαντήσει αποτελεσματικά αυτό το ερώτημα. Για αυτό, πρέπει να αποκτήσει πρόσβαση σε κάποιο άλλο επίπεδο (από τα προαναφερθέντα), το οποίο να παρέχει τη μεγαλύτερη ασφάλεια, ενώ παράλληλα να απαντάει σωστά στο ερώτημα.

Η στρατηγική που ακολουθεί το σύστημα CryptDB είναι να έχει ένα προσαρμοστικό σχήμα το οποίο ρυθμίζει δυναμικά την κρυπτογράφηση των δεδομένων, ονομαζόμενο ως ***onions***. Η βασική ιδέα είναι να κρυπτογραφείται κάθε τιμή με ένα ή περισσότερα κρεμμύδια. Δηλαδή, κάθε plaintext να είναι ντυμένο σε στρώματα, ξεκινώντας από το στρώμα με τη λιγότερη ασφάλεια και καταλήγοντας στο στρώμα με τη μεγαλύτερη ασφάλεια και λιγότερη λειτουργικότητα, το οποίο είναι και το εξωτερικό περίβλημα *(Εικόνα 3).* Έτσι, κάθε στρώμα κάθε κρεμμυδιού επιτρέπει ορισμένες λειτουργίες. Για παράδειγμα, τα εξωτερικά στρώματα, όπως το RND και το HOM, παρέχουν μέγιστη ασφάλεια, ενώ τα εσωτερικά στρώματα, όπως το DET και το OPE, προσφέρουν περισσότερη λειτουργικότητα. Από τα παραπάνω γίνεται αντιληπτό πως στο Server δεν θα υπάρχουν οι πινάκες με την μορφή που δημιουργούνται από το χρήστη, αλλά θα υπάρχουν τόσες στήλες όσα τα κρεμμύδια που έχουν δημιουργηθεί για κάθε πεδίο *(Εικόνα 4).*



Εικόνα 3: Τα επίπεδα κρυπτογράφησης των Onions.



Εικόνα 4: Παράδειγμα δημιουργίας πινάκα σε χρηστή και σε server.

Στην *Εικόνα 4* φαίνεται ένα απλό παράδειγμα των όσων αναφέρθηκαν μέχρι τώρα [5]. Σύμφωνα με αυτό, όταν ένας χρηστής δημιουργεί έναν πινάκα, αυτός αποθηκεύεται με διαφορετική δομή στο Server, καθώς πλέον οι στήλες του πίνακα εκφράζονται με τη μορφή των κρεμμυδιών κάθε πεδίου. Κάθε φορά που γίνεται μια εισαγωγή δεδομένων στο σύστημα, ως προεπιλογή έχει οριστεί η κρυπτογράφηση με τις τεχνικές RND, HOM, SEARCH. Έτσι, θα αποθηκευτούν με αυτές τις κρυπτογραφημένες τιμές στο Server, καθώς δεν έχει προηγηθεί κάποιο ερώτημα από το χρήστη ώστε να αλλάξει η προεπιλογή. Παρατηρώντας το παράδειγμα, διαπιστώνεται ότι ανάλογα με τον τύπο του κάθε πεδίου δε δημιουργούνται κάποια κρεμμύδια. Όπως φαίνεται στην *Εικόνα 4*, το κρεμμύδι Add δε δημιουργήθηκε διότι ο τύπος text δεν υποστηρίζει προσθέσεις.

Για κάθε πίνακα, στήλη, κρεμμύδι και επίπεδο κρεμμυδιού χρησιμοποιείται διαφορετικό κλειδί. Με αυτόν τον τρόπο εμποδίζεται η επιπλέον διαρροή, καθώς ο server δε θα μπορεί να μάθει περαιτέρω σχέσεις. Όλα τα κλειδιά προέρχονται από το κύριο κλειδί MK. Για παράδειγμα, έστω ότι υπάρχει ένας πίνακας t, με στήλη c, με κρεμμύδι o, και με στρώμα κρυπτογράφησης ℓ, τότε ο proxy-server χρησιμοποιεί το κλειδί:

$$\mathbf{K_{t,c,o,l} = PRP_{MK}(\text{ πίνακας t, στήλη c, κρεμμύδι o, επίπεδο κρυπτογράφησης } \ell)}$$

όπου PRP είναι ψευδοτυχαία μετάθεση (pseudorandom permutation).

Καθώς λαμβάνει ο proxy-server ένα ερώτημα SQL από το χρηστή, πριν το κρυπτογραφήσει και το στείλει στο server, ελέγχει δύο παραμέτρους. Πρώτον, τι επίπεδο κρυπτογράφησης απαιτεί αυτό το ερώτημα, και δεύτερον, αν είναι σε αυτό το επίπεδο κρυπτογράφησης τα δεδομένα σε εκείνη την στήλη. Αν τα δεδομένα δε βρίσκονται στο απαιτούμενο επίπεδο κρυπτογράφησης, τότε ο proxy-server στέλνει στο Server το αντίστοιχο κλειδί για την προσαρμογή κρεμμυδιού. Την προσαρμογή κρεμμυδιού, δηλαδή την αποκρυπτογράφηση από το ένα στρώμα στο άλλο, την εκτελεί ο Server με τη βοήθεια των συναρτήσεων UDF (user-defined functions).

Για παράδειγμα, για την *Εικόνα 4*, στο onion Order για την πρώτη στήλη, η αποκρυπτογράφηση από το RND στο OPE γίνεται χρησιμοποιώντας τη συνάρτηση DECRYPT_RND UDF του Server. Η εντολή που χρησιμοποιείται είναι η εξής:

UPDATE Table1 SET

Col1-OnionOrder = DECRYPT_RND(K, Col1-OnionOrder, Col1-IV)

όπου K είναι το κλειδί που υπολογίζεται από την εξίσωση που αναφέρθηκε παραπάνω.

Στη συνέχεια, ακολουθεί αναλυτικό παράδειγμα με όσα περιγράφηκαν μέχρι αυτό το σημείο.

Παράδειγμα:

1. **Δημιουργία Πίνακα**

CREATE TABLE test1 (name text, surname text);

Στο Server, για κάθε πεδίο name & surname, δημιουργούνται τέσσερεις στήλες, εκ των οποίων οι τρεις είναι για τα Onions (Eq,Search,Ord) και η άλλη για το IV (για την κρυπτογράφηση από το RND επίπεδο). Επομένως, συνολικά δημιουργούνται 8 στήλες και για τα πεδία.

2. **Εισαγωγή στον Πίνακα**

INSERT INTO test1 VALUES (“Christina”, “Mantonanaki”);

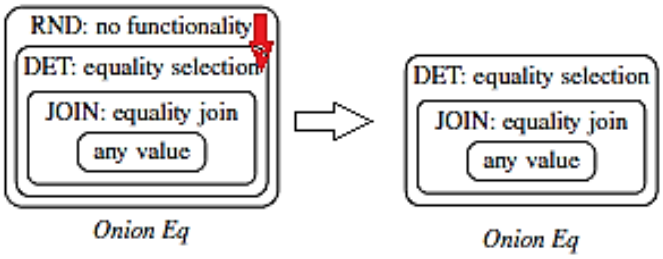
Ως προεπιλογή, το επίπεδο κρυπτογράφησης για αυτές τις τιμές θα είναι το εξωτερικό στρώμα κρεμμυδιού, δηλαδή το RND για το Eq Onion. Όποτε, με αυτήν την τεχνική κρυπτογράφησης θα αποθηκευτούν στο Server.

3. Αναζήτηση στον Πίνακα

SELECT * FROM test1 WHERE name = “Christina”;

Όταν ένα ερώτημα φτάνει στο proxy-server, αυτός συγκρίνει το επίπεδο κρυπτογράφησης που έχει εκείνη τη στιγμή η στήλη «name», με το επίπεδο που απαιτείται έτσι ώστε να απαντηθεί αυτό το ερώτημα έχοντας τη λιγότερη διαρροή. Σε αυτήν την περίπτωση, το επίπεδο που βρίσκεται είναι το RND, ενώ για να απαντηθεί αυτό το ερώτημα χρειάζεται DET επίπεδο. Έτσι, ο proxy-server εκδίδει το ερώτημα UPDATE και το στέλνει στο Server, με σκοπό εκείνη η στήλη να «κατέβει» επίπεδο κρυπτογράφησης και να αποκτήσει το επίπεδο DET *(Εικόνα 5)*.

Μετά την εκτέλεση αυτού του ερωτήματος, ο proxy-server ενημερώνει το σχήμα του για να θυμάται το νέο κρυπτογραφημένο επίπεδο που έχει εκείνη η στήλη στο συγκεκριμένο κρεμμύδι. Στη συνέχεια, κρυπτογραφεί την λέξη «Christina» με το onion Eq και με κλειδί το σχηματισμό που έχει αναφερθεί πιο πάνω. Έπειτα, στέλνει το κρυπτογραφημένο ερώτημα στο Server και λαμβάνει τα κρυπτογραφημένα αποτελέσματα από αυτόν. Τέλος, αποκρυπτογραφεί τα αποτελέσματα και τα στέλνει στο χρηστή.



Εικόνα 5: Προσαρμογή επίπεδου σε κρεμμύδι.

2.2 Searchable Symmetric Encryption Τεχνική

2.2.1 Εισαγωγή στην έννοια Searchable Symmetric Encryption (SSE)

Η τεχνική SSE επιτρέπει στο χρήστη να κρυπτογραφήσει τα δεδομένα του κατά τέτοιο τρόπο ώστε να μπορεί να τα αναζητήσει αποτελεσματικά στη συνέχεια. Χαρακτηρίζεται ως μια από τις πιο διαδεδομένες τεχνικές όσον αφορά την ασφαλή αποθήκευση των δεδομένων στο σύννεφο (cloud). Έτσι, ο χρήστης δε χρειάζεται να γνωρίζει αν ο πάροχος στο cloud είναι αξιόπιστος. Παράλληλα, η τεχνική αυτή επιτρέπει στο χρήστη να αναζητήσει δεδομένα πάνω στα κρυπτογραφημένα στοιχεία με τη δημιουργία του *search token*.

Ένα SSE σχήμα θεωρείται ασφαλές με τρεις τρόπους. Πρώτον, όταν το κρυπτογραφημένο κείμενο δεν αποκαλύπτει καμία πληροφορία για το Plaintext και δεύτερον όταν το κρυπτοκείμενο (cipher text) μαζί με το token search αποκαλύπτουν μόνο το αποτέλεσμα της αναζήτησης. Τέλος, όταν το search token μπορεί να δημιουργηθεί μόνο από το μυστικό κλειδί K [6].

Το σχήμα αυτό βασίζεται σε ένα κρυπτογραφημένο ευρετήριο (index). Ένα index μπορεί να είναι "ασφαλές", εάν η λειτουργία αναζήτησης για ένα keyword μπορεί να εκτελεστεί μόνο από τους χρήστες που έχουν το *trapdoor* για αυτό το keyword. Η δημιουργία του trapdoor βασίζεται στο μυστικό κλειδί.

2.2.2 Ορισμός SSE

Έστω D μια συλλογή από documents, όπου κάθε document $d \in D$ έχει το δικό του id και μπορεί να περιέχει οποιοδήποτε στοιχείο. Κάθε d μπορεί να έχει σαν σύνολο keywords από το λεξικό Δ. Απεικονίζεται ως id(w) όταν το id document περιέχει την λέξη w. Στόχος του σχήματος αυτού είναι να οικοδομηθεί ένα κρυπτογραφημένο index I σχετικά με τα id documents. Το πρωτόκολλο SSE περιλαμβάνει ένα χρηστή και ένα server και αποτελείται από 4 αλγορίθμους (*Keygen, BuildIndex, Trapdoor, Search*).

Αναλυτικά για τον κάθε αλγόριθμο:

- **K ← Keygen (1^{λ}):** είναι ένας πιθανοτικός αλγόριθμος δημιουργίας κλειδιού που τρέχει από το χρηστή, πριν την έναρξη του συστήματος. Παίρνει ως ασφαλή παράμετρο το λ και εξάγει το μυστικό κλειδί K.

- **I ←BuildIndex (K, D):** είναι ένας πιθανοτικός αλγόριθμος που τρέχει από το χρηστή για την δημιουργία του Index, πριν την αποστολή δεδομένων στο Server. Έχει ως είσοδο το μυστικό κλειδί K και το σύνολο documents D και παράγει ως έξοδο ένα κρυπτογραφημένο Index πάνω στα ids documents.

- **T ← Trapdoor (K, w):** είναι ένας αλγόριθμος που εκτελείται από το χρήστη για την δημιουργία trapdoor, για μια δεδομένη λέξη κατά την υποβολή του ερωτήματος. Παίρνει ως είσοδο το μυστικό κλειδί K και το keyword w, ενώ εξάγει ένα token t.

- **X ← Search (I, Tw):** είναι ένας αλγόριθμος που εκτελείται από το server για να ανακτήσει όλα τα ids που περιέχουν την λέξη w. Παίρνει ως είσοδο το Index I για μια συλλογή D και ένα trapdoor Tw για λέξη w. Επιστρέφει το D(w), δηλαδή το σύνολο X των ids που είναι ίσα με w.

2.2.3 Σχήμα SSE που χρησιμοποιήθηκε

Το σχήμα SSE που μελετήθηκε περισσότερο [8], [10] βασίζεται σε ένα index που έχει την μορφή πινάκα **I** και σε δύο στατικούς πινάκες κατακερματισμού (static hash table) **TW** και **TF**. Αν ισχύει το I[i,j] = 1, τότε αυτό σημαίνει πως το Keyword_i βρίσκεται στο file_j. Σε διαφορετική περίπτωση ισχύει I[i,j] = 0.

Συγκεκριμένα, οι TW και TF περιέχουν keyword και file, αντίστοιχα. Αυτό σημαίνει πως κάθε file και keyword αποθηκεύεται στο TF και TW, οι οποίοι υπάρχουν μόνο στην πλευρά του χρηστή. Όταν δημιουργηθεί το I θα το κρυπτογραφήσει ο χρήστης και θα το στείλει στο server. Το κύριο σχήμα που μελετήθηκε υποστηρίζει αποτελεσματικά τις επιλογές ***add*** και ***search***.

Αναλυτικότερα:

- ***Επιλογή Add:***

Στην περίπτωση αυτή ο χρήστης θέλει να εισάγει ένα αρχείο. Ακολούθως, παρουσιάζονται τα βήματα που εκτελούνται για την εισαγωγή ενός αρχείου.

1. Από την πλευρά του χρηστή, παίρνει το όνομα του αρχείου και δημιουργεί Trapdoor για αυτό το αρχείο. Αφού δημιουργήσει trapdoor, προσθέτει στη θέση trapdoor του hash static TF ένα τυχαίο αριθμό από μια υπάρχουσα λίστα για την στήλη. Δηλαδή, TF[trapdoor] = random_number_1.

2. Από το random_number_1 δημιουργείται το block_Index, το όποιο στέλνεται στο server.

3. Ο Server έχει κρυπτογραφημένο Index I[i,j], όπου το i είναι για το keyword και το j είναι για το file. Ο server βρίσκει το I[*,block_Index] και το επιστρέφει στο χρηστή στην μορφή μονοδιάστατου πινάκα, ονομαζόμενο I_prime.

4. Ο χρήστης, αφού αποκρυπτογραφήσει αυτόν τον μονοδιάστατο πινάκα, ανοίγει το αρχείο που δόθηκε για add και για κάθε λέξη που διαβάζει εκτελεί παρόμοια διαδικασία με αυτή που έκανε και για το file με το TF. Μόνο που σε αυτή την περίπτωση, χρησιμοποιείται το TW και οι τυχαίοι αριθμοί δίνονται από μια άλλη λίστα για γραμμές.

5. Σειρά έχει η τροποποίηση αυτού του μονοδιάστατου πίνακα κατά τέτοιο τρόπο ώστε όσα keywords έχουν σχέση με το file να παίρνουν την τιμή 1, αλλιώς την τιμή 0. Στη συνέχεια, κρυπτογραφείται ο πινάκας με άλλο κλειδί απ' ότι προηγουμένως και στέλνεται στο Server.

6. Τέλος, ο Server παίρνει τον μονοδιάστατο πινάκα και το block_Index, τα οποία τοποθετεί στην σωστή θέση του I[*,block_Index].

- ***Επιλογή Search:***

Η λογική του αλγορίθμου Search βρίσκεται πολύ κοντά στη λογική του αλγορίθμου Add. Ο χρήστης θα δημιουργήσει το token search και μέσα από τον πίνακα TW βρίσκει το keyword_index γι αυτό. Στη συνέχεια, στέλνει το token search στο Server και παίρνει από το Server τον πίνακα I[keyword_index,*] σε μονοδιάστατη μορφή. Έπειτα, αποκρυπτογραφεί και ελέγχει τον πίνακα με σκοπό να βρει σε ποια files υπάρχει αυτό το keyword, όπως περιγράφτηκε αναλυτικά στη υλοποίηση της επιλογής add.

2.2.4 DET VS SSE in Point Queries

Όπως έχει προαναφερθεί, η κρυπτογράφηση DET χρησιμοποιείται στο CryptDB για να απαντήσει point queries. Το πρόβλημα σε αυτή τη μέθοδο είναι ότι δυο plaintext κρυπτογραφούνται με τον ίδιο τρόπο, οπότε έχουν ίδιο ciphertext.

Ο ορισμός είναι:

$$\forall\ m_0, m_1 \in M: Enc(K, m_0) = Enc(K, m_1), \text{ αν και μόνο αν } m_0 = m_1.$$

Αυτό σημαίνει πως υπάρχει μια διαρροή, διότι ο επιτιθέμενος στο Server μπορεί να αντιληφθεί τη συχνότητα κάποιων στοιχείων. Ωστόσο, το σύστημα CryptDB αυξάνει την λειτουργικότητα του (καθώς με το επίπεδο RND δεν μπορούν να απαντηθούν τέτοιου είδους ερωτήματα).

Αντίθετα με όσα έχουν ήδη αναφερθεί, η τεχνική SSE εισάγει αυστηρό ορισμό ασφάλειας, ενώ παράλληλα επιτρέπει το σχεδιασμό των συστημάτων που αποφεύγουν διαρροές που έχουν τεχνικές κρυπτογράφησης τύπου DET [7], [9], διατηρώντας υψηλή απόδοση. Η τεχνική αυτή προσπαθεί να λύσει το keyword search πρόβλημα στα κρυπτογραφημένα δεδομένα που δόθηκαν στο Server. Με ένα παρόμοιο τρόπο με το point query, οι λέξεις αντιστοιχίζονται στις τιμές που μπορούν να αναζητηθούν και τα file-id σε tuple-id που περιέχουν μια συγκεκριμένη τιμή αναζήτησης.

Στο επόμενο κεφάλαιο, θα παρουσιαστεί αναλυτικά η προσαρμογή της SSE στο σύστημα CryptDB για point queries.

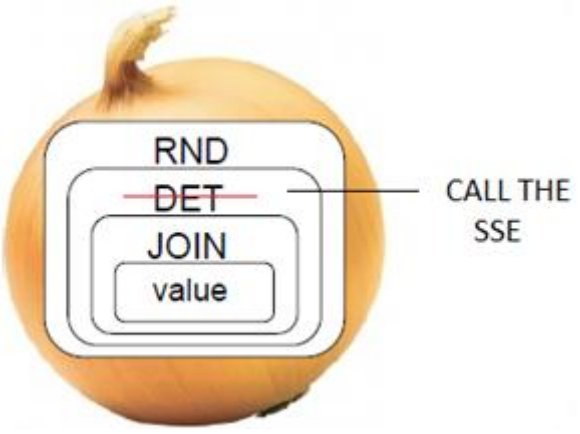
3. ΥΛΟΠΟΙΗΣΗ - ΠΑΡΑΔΕΙΓΜΑΤΑ

Στην *Ενότητα 2.1* παρουσιάστηκε αναλυτικά το σύστημα CryptDB, το οποίο αποσκοπεί στην ασφάλεια των δεδομένων σε μη αξιόπιστους παρόχους. Στις *Ενότητες 2.2 & 2.3* παρουσιάστηκε η τεχνική SSE, ενώ ταυτόχρονα αναλύθηκε γιατί θεωρείται καλύτερη από το τρόπο κρυπτογράφησης DET. Σκοπός της παρούσας διπλωματικής εργασίας είναι η απάντηση των point queries μειώνοντας σημαντικά την διαρροή αυτού του συστήματος. Έτσι, σε αυτή την ενότητα θα παρουσιαστεί αναλυτικά η υλοποίηση που ακολουθήθηκε, με στόχο να προσαρμοστεί η τεχνική SSE στο σύστημα CryptDB για την αποτελεσματική και ασφαλή απάντηση point queries. Στη συνέχεια, παρατίθεται παράδειγμα για την πλήρη κατανόηση της υλοποίησης.

3.1 Υλοποίηση

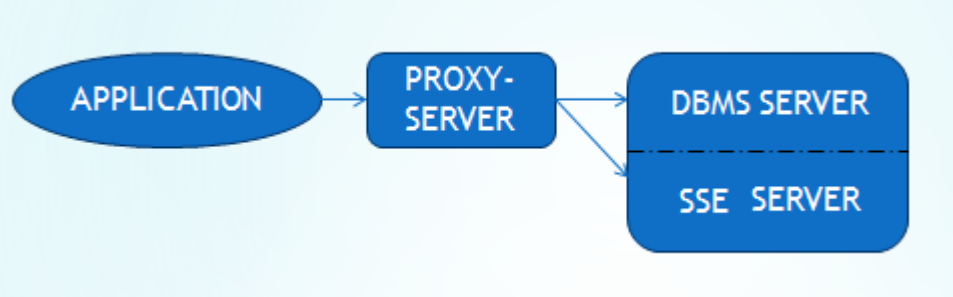
3.1.1 Γενική Ιδέα Αλγορίθμου

Το πρώτο βήμα σε αυτήν την υλοποίηση ήταν η δημιουργία ενός νέου κρεμμυδιού *MyEq* τύπου *Eq Onion*, με σκοπό την αντικατάσταση του Eq Onion με το νέο, κάθε φορά που πρέπει να κληθεί το Eq onion. Το Eq Onion *(Εικόνα 3)* περιέχει τα επίπεδα κρυπτογράφησης *RND, DET, DET-JOIN, PLAINTEXT*. Το MyEq στο επίπεδο DET θα καλεί το SSE αλγόριθμο *(Εικόνα 6).*



Εικόνα 6: MyEq Onion.

Με το να καλεί τον αλγόριθμο SSE *(Εικόνα 6)*, δε θα κατεβαίνει ποτέ το επίπεδο DET στο server και έτσι δε θα υπάρχει η διαρροή που δημιουργείται κατά τα ερώτημα ισότητας. Η αρχιτεκτονική του νέου σχήματος παρουσιάζεται στην *Εικόνα 7*. Στο νέο σχήμα που δημιουργείται, υπάρχουν δύο Server, ένας για το SSE και ένας που είχε από πριν το σχήμα CryptDB.

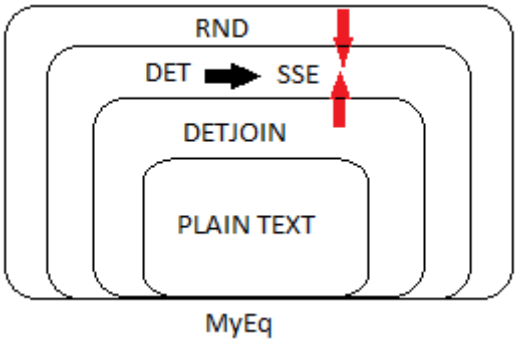


Εικόνα 7: Αρχιτεκτονική του νέου συστήματος.

3.1.2 Διαδικασία εισαγωγής ενός Tuple

Πριν την προσθήκη του SSE στο σύστημα, κατά την εισαγωγή ενός Tuple, το CryptDB έπαιρνε την κάθε τιμή πεδίου και για κάθε κρεμμύδι που είχε σχεδιαστεί για αυτό το πεδίο, κρυπτογραφούσε με το επιλεγμένο στρώμα κρυπτογράφησης. Το κομμάτι της εισαγωγής στο νέο σχήμα θα πρέπει να περιέχει και την εισαγωγή δεδομένων κάνοντας χρήση SSE. Αυτό σημαίνει ότι, όταν εξετάζεται το κρεμμύδι MyEq θα πρέπει να ελέγχεται ποιο είναι το επιλεγμένο στρώμα κρυπτογράφησης. Σε αυτό το σημείο πρέπει να τονιστεί ότι το DET δεν μπορεί ποτέ να είναι το επιλεγμένο στρώμα, καθώς όποτε χρειάζεται να κατέβει επίπεδο και να πάει στο DET θα καλείται το SSE. Σε διαφορετική περίπτωση θα συνεχίζει συμφώνα με όσα έκανε πριν.

Ελέγχοντας, λοιπόν, ποιο είναι το επιλεγμένο στρώμα σε αυτό το κρεμμύδι οι δυνατές επιλογές είναι δύο: RND ή DET-JOIN. Έτσι, πριν εισαχθεί το συγκεκριμένο Tuple στο SSE, θα πρέπει να κρυπτογραφηθεί σε επίπεδο DET *(Εικόνα 8)* και μετά να δοθεί για εισαγωγή στον αλγόριθμο εισαγωγής του SSE. Αυτός ο αλγόριθμος εισαγωγής θα κρυπτογραφήσει το tuple, πριν το εισάγει στο Server του (όπως εξηγήθηκε στην *Ενότητα 2.2*), και έτσι δεν θα υπάρχουν οι διαρροές.



Εικόνα 8: Σχηματική απεικόνιση της αλλαγή του επίπεδου κρυπτογράφησης για την εισαγωγή του στο αλγόριθμο εισαγωγής SSE.

Επεξηγώντας τα παραπάνω, το DET είναι η μονή μορφή που επιτρέπει σύγκριση ισότητας με την λιγότερη διαρροή. Επομένως, είναι η κατάλληλη επιλογή ως όρισμα για την εισαγωγή του αλγόριθμου SSE. Επιπλέον, κάθε στήλη ενός πινάκα έχει διαφορετικό επίπεδο κρυπτογράφησης, που σημαίνει ότι ένα επίπεδο μιας στήλης μπορεί να είναι RND και ένα άλλο DETJOIN (για MyEq). Σκοπός όμως για την ομαλή κρυπτογράφηση των δεδομένων είναι να παρέχονται όλα ως ορίσματα από τον αλγόριθμο εισαγωγής με το *ίδιο επίπεδο κρυπτογράφησης*.

Ο αλγόριθμος εισαγωγής του SSE παίρνει ως ορίσματα το *keyword* και το *tuple*. Ως keyword δίνεται η *τιμή κάθε πεδίου + το όνομα του πεδίου + όνομα του πινάκα* και ως tuple οι τιμές όλων των πεδίων διαχωριζόμενες από το string “!!!”.

Παράδειγμα 1: Πως δημιουργείτε το tuple, keyword που στέλνονται ως όρισμα.

CREATE TABLE t1 (id integer,name text);

INSERT INTO t1 VALUES (1, “alice”);

1,id,t1 και 1!!!alice
alice,name,t1 και 1!!!alice

Στο αρχικό σύστημα, όπως έχει ήδη αναφερθεί, όταν ένα ερώτημα δεν μπορούσε να απαντηθεί με το ήδη υπάρχων στρώμα κρυπτογράφησης, ο proxy έστελνε Update ερώτημα στο DBMS Server. Με τη βοήθεια των UDF συναρτήσεων, ο Server έκανε Update εκείνη τη στήλη στο επίπεδο που έπρεπε, για να απαντηθεί το ερώτημα. Αυτή η διαδικασία είναι αδύνατη από την πλευρά της τεχνικής SSE, καθώς παίρνει ως όρισμα εισαγωγής ολόκληρο το tuple. Όμως, με την επανακρυπτογράφηση/ αποκρυπτογράφηση στο επίπεδο DET δεν είναι αναγκαίο κάτι τέτοιο.

Η επανακρυπτογράφηση είναι μια βελτίωση την όποια προτείνει το ίδιο το σύστημα Cryptdb μόνο για πολύ ευαίσθητα δεδομένα, όπως αριθμοί καρτών, στα όποια όμως τα ερωτήματα δεν είναι τόσο συχνά. Θεωρείται χρονοβόρα διαδικασία, η οποία αποφεύγεται, καθώς ένας πινάκας μπορεί να έχει παρά πολλά tuples. Αντιθέτως, στην περίπτωση του σχήματος που η εισαγωγή γίνεται σε ένα tuple τη φορά, και όχι σε μια συλλογή από tuples, δεν αποτελεί το ίδιο χρονοβόρα διαδικασία. Παρακάτω παρουσιάζεται ο αλγόριθμος εισαγωγής με ότι περιγράφτηκε έως τώρα.

<u>Αλγόριθμος εισαγωγής</u>

<u>Ορίσματα</u>: οι τιμές που έδωσε ο χρήστης για εισαγωγή στον πίνακα.

1. Δημιουργία του tuple το όποιο θα είναι κενό στην αρχή, tuple = " "
2. Δημιουργία ενός διανύσματος (vector) για τα Keywords
3. Για κάθε τιμή
4. Κρυπτογράφησε την με όλα τα κρεμμύδια που έχει εκείνο το πεδίο
5. Αν το Keyword vector δεν είναι empty
6. tuple = tuple + "!!!"
7. Για το κρεμμύδι *MyEq* δημιούργησε τιμή encrypt_det στο επίπεδο DET
8. Keyword. push_back (encrypt_det+ ","+ field_name +","+table_name)
9. tuple = tuple + encrypt_det
10. Για κάθε value_keyword του Vector Keyword
11. Κάλεσε τη συνάρτηση εισαγωγής SSE με ορίσματα value_keyword,tuple
12. Στείλε τα δεδομένα που έχουν κρυπτογραφηθεί από το βήμα 4 στο DBMS Server.

Ίσως να μην έχει γίνει ακόμα αντιληπτό γιατί να ανεβαίνουμε επίπεδο (δηλαδή από DET-JOIN σε DET), αφού όταν η στήλη έχει επίπεδο DETJOIN, τα δεδομένα απαντιούνται από το DBMS Server. Στη συνέχεια, στο *Παράδειγμα 2* δίνεται η επακριβής εξήγηση αυτού του προβληματισμού.

Παράδειγμα 2: Γιατί ανεβαίνει επίπεδο από DETJOIN σε DET.

CREATE TABLE t1 (id integer,name text);

CREATE TABLE t2 (id intger,name text);

INSERT INTO t1 VALUES (1,"alice"); ***RND για το DBMS Server & DET για τον αλγόριθμο εισαγωγής SSE***

INSERT INTO t2 VALUES (1,"alice");

SELECT * FROM t1,t2 WHERE t1.id=t2.id; ***Προσαρμογή επίπεδου: DETJOIN***

INSERT INTO t2 VALUES (2,"alice"); ***Το id είναι DETJOIN επίπεδο, όρισμα ως DET στο SSE***

SELECT * FROM t1 where name=" alice"; ***Κλήση SSE σε select star ερώτημα το οποίο θέλει και το id που ήταν σε επίπεδο DETJOIN, γι αυτό τον λόγο αποθηκεύεται και αυτό στο SSE ενώ έχει κατέβει ήδη επίπεδο.***

3.1.3 Διαδικασία Αναζήτησης Ερωτημάτων

Κατά την αναζήτηση στο αρχικό σχήμα, όταν ο proxy-server λαμβάνει ένα ερώτημα αναζήτησης για να το απαντήσει ακολουθεί μια συγκεκριμένη διαδικασία. Ελέγχει να δει αν το επίπεδο κρυπτογράφησης, εκείνης της στήλης που χρειάζεται, είναι ίδιο με το επίπεδο που απαιτείται για να απαντηθεί αποτελεσματικά το ερωτήματα.

Στην περίπτωση που το ερώτημα θέλει επίπεδο κρυπτογράφησης DET, για να απαντηθεί αποτελεσματικά, ο proxy-server καλεί τη συνάρτηση αναζήτησης του SSE (η περιγραφή της οποίας βρίσκεται στην *Ενότητα 2*). Αυτή η συνάρτηση παίρνει ως όρισμα το keyword. Αυτό έχει ως αποτέλεσμα ότι η στήλη στο DBMS Server δε θα κατέβει επίπεδο, αλλά θα παραμείνει στο RND στρώμα. Από το ερώτημα που έχει λάβει ο Proxy-server θα πάρει την λέξη που πρέπει να αναζητήσει, θα την κωδικοποιήσει με τον ίδιο τρόπο, όπως περιγράφηκε παραπάνω, και θα δημιουργήσει το σωστό keyword. Επιπλέον, θα αναζητηθεί στο SSE Server η απάντηση αυτού του ερωτήματος.

Σε αντίθετη περίπτωση, αν έρθει ένα ερώτημα και το επίπεδο κρυπτογράφησης που απαιτείται για να απαντηθεί αυτό το ερώτημα είναι DET-JOIN, τότε εκείνη η στήλη

του ερωτήματος θα κατέβει επίπεδο και θα αναζητήσει την απάντηση, όπως έκανε παλιότερα.

Στην περίπτωση που το ερώτημα πρέπει να λάβει απάντηση από το SSE, θα επιστρέψει κρυπτογραφημένα tuples και όχι πινάκα. Το αποτέλεσμα του ερωτήματος γράφεται σε ένα αρχείο. Πριν ξεκινήσει την αποκρυπτογράφηση ο proxy-server, θα ξεχωρίσει τα Tuples και θα τα βάλει σε έναν πινάκα, με σκοπό να τα στείλει στο χρήστη. Στη συνέχεια, θα ακολουθήσει την ιδία διαδικασία που εκτελούσε και πριν, για την αποκρυπτογράφηση του πινάκα.

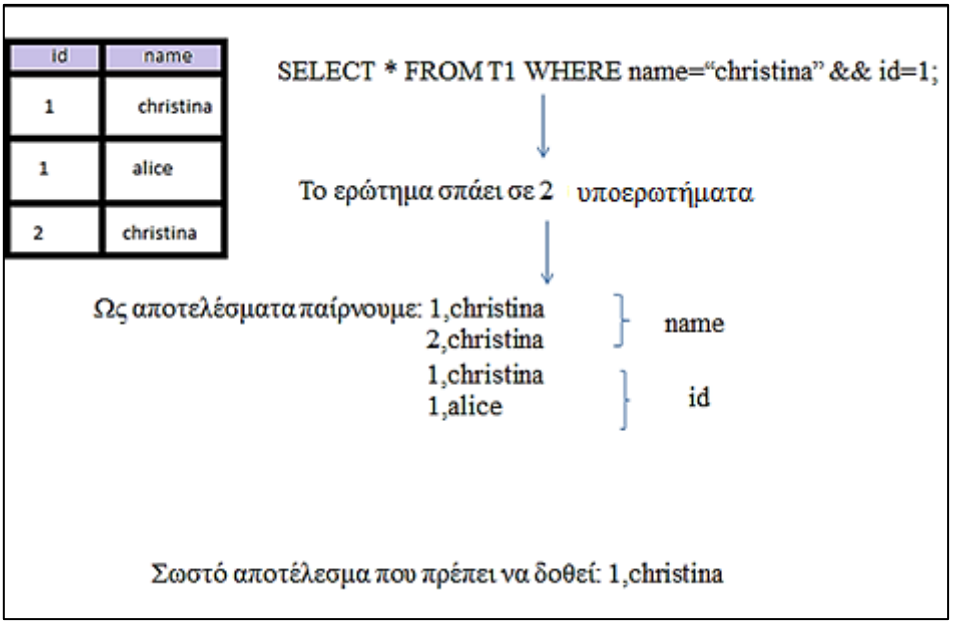
Αλγόριθμος Αναζήτησης

1. Αν το επίπεδο που απαιτείται για να απαντηθεί το ερώτημα είναι DET τότε
2. Να μην γίνει προσαρμογή σε αυτό το επίπεδο
3. Η τιμή που θέλει ο χρήστης να αναζητήσει, να κρυπτογραφηθεί με μορφή DET. DET_value
4. Δημιουργία του Keyword = DET_value+“,” field_name+“,”+able_name
5. Κλήση της συνάρτησης αναζήτησης SSE δίνοντας ως όρισμα το Keyword
6. Διαφορετικά αν το επίπεδο κρυπτογράφησης δεν μπορεί να απαντήσει το ερώτημα
7. Προσαρμογή κρεμμυδιού
8. Κρυπτογράφηση ερωτήματος και αποστολή στο DBMS Server
9. Διαφορετικά
10. αποστολή στο DBMS Server

3.1.4 Ερωτήματα Αναζήτησης με τελεστές AND και OR

Στα ερωτήματα με τελεστές, η διαδικασία αλλάζει, όσον αναφορά στον αλγόριθμο του SSE αναζήτησης. Λόγω τελεστών, θα δημιουργούνται παραπάνω από ένα Keyword με την ίδια λογική όπως πριν. Έτσι, τα Keywords θα αποθηκεύονται σε ένα διάνυσμα (vector), και ο αλγόριθμος SSE θα καλείται για όσα στοιχεία έχει μέσα το vector. Αυτό σημαίνει ότι, ένα ερώτημα με τελεστή θα χωρίζεται σε υποερωτήματα, και κάθε υποερώτημα θα λειτούργει όπως πριν. Υλοποιώντας αυτή τη μέθοδο, το αποτέλεσμα μπορεί να έχει ένα tuple παραπάνω φορές από όσες χρειάζεται, ή κάποια tuples να πρέπει να αφαιρεθούν από το αποτέλεσμα για να είναι σωστό. Στη συνέχεια, ανάλογα με τον τελεστή που έχει δοθεί στο ερώτημα γίνεται ο κατάλληλος διαχωρισμός.

Στην *Εικόνα 9* φαίνεται η διαδικασία που ακολουθείται όταν ο χρήστης δώσει ένα ερώτημα που περιέχει τελεστή. Στην *Εικόνα 10* παρουσιάζεται ο αλγόριθμος που ακολουθείται όταν έχουμε τελεστές, δηλαδή πως από τον πινάκα που περιέχει παραπάνω Tuples, παράγουμε το επιθυμητό-σωστό αποτελέσματα.



Εικόνα 9: Διαδικασία Ερωτήματος με Τελεστή.

Αξιοσημείωτο είναι ότι όταν σπάει το ερώτημα σε υποερωτήματα, το ένα υποερώτημα μπορεί να απαιτεί DET επίπεδο (δηλαδή κλήση SSE αλγορίθμου), ενώ η στήλη του άλλου ερωτήματος που απευθύνεται, μπορεί να είναι σε DET-JOIN επίπεδο. Με όσα έχουν προαναφερθεί, το δεύτερο ερώτημα απαντιέται από το DBMS Server. Για να αποφευχθεί η διαδικασία του μπερδέματος και των δυο server, και μόνο σε αυτές τις περιπτώσεις (ερωτήματα με τελεστές), τα ερωτήματα απαντώνται μόνο από τον SSE Server. Όταν τρέξουν τα υποερωτήματα και το αποτέλεσμα τοποθετηθεί σε πινάκα, εκτελείται ο αλγόριθμος της *Εικόνας 10*, ο όποιος αφορά τους τελεστές AND ή OR ανάλογα με το ερώτημα.

Αλγόριθμος για Τελεστές

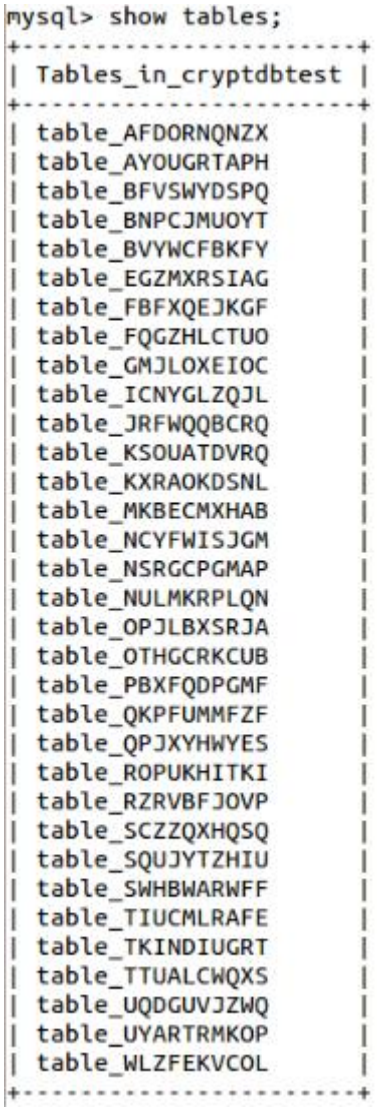
1. Για κάθε στοιχείο Α του Πινακα-1
2. temp = false
3. Για κάθε στοιχείο Β του Πινάκα ξεκινώντας από το Α+1
4. Αν Α==Β και Α!=”!!!” τότε
5. Αποθήκευση όλου του Tuple
6. Β = “!!!”
7. temp = true
8. Break
9. Αν ο τελεστής == OR και Α!=”!!!” και temp == false τότε
10. Αποθήκευση όλου το tuple
11. Αν τελεστής == OR και Πινάκα[size-1]!=”!!!” τότε
12. Αποθήκευση όλου του tuple

Εικόνα 10: Αλγόριθμος δημιουργίας σωστού αποτελέσματος στην περίπτωση των τελεστών.

3.2 Παραδείγματα

Σε αυτήν την ενότητα, παρουσιάζονται ερωτήματα που τρέχουν από το χρήστη, καθώς και η λειτουργία του νέου συστήματος όσον αφορά στα Point queries.

Στο DBMS Server, τα δεδομένα αποθηκεύονται κρυπτογραφημένα και οι πινάκες με κρυπτογραφημένα ονόματα. Στην *Εικόνα 11* φαίνεται πως εμφανίζονται οι πινάκες, όταν ο χρήστης πληκτρολογήσει την εντολή *show tables;* στη Mysql.



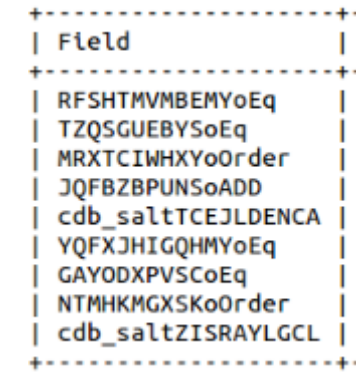
Εικόνα 11: Τα κρυπτογραφημένα ονόματα των πινάκων.

Τρέχοντας, λοιπόν, ο χρήστης ένα ερώτημα δημιουργίας πινάκα, ο DBMS Server δημιουργεί τόσες στήλες, όσες είναι τα κρεμμύδια του κάθε πεδίου. Έτσι, προκύπτουν τα παρακάτω:

CREATE TABLE test (id integer, name text);

```
NEW QUERY: create table table_WLZFEKVCOL (RFSHTMVMBEMYoEq BIGINT unsigned, TZQSG
UEBYSoEq BIGINT unsigned, MRXTCIWHXYoOrder BIGINT unsigned, JQFBZBPUNSoADD VARBI
NARY(256), cdb_saltTCEJLDENCA BIGINT(8) unsigned, YQFXJHIGQHMYoEq BLOB, GAYODXPV
SCoEq BLOB, NTMHKMGXSKoOrder BIGINT unsigned, cdb_saltZISRAYLGCL BIGINT(8) unsig
ned) AUTO_INCREMENT=0 ENGINE=InnoDB
```

Εικόνα 12: Κρυπτογραφημένο ερώτημα δημιουργίας πινάκα



Εικόνα 13: Δομή του πινάκα που δημιουργείτε στο DBMS Server.

Στη συνέχεια, ο χρήστης τρέχει ένα ερώτημα εισαγωγής σε αυτόν τον πινάκα που μόλις δημιούργησε. Η λογική που ακολουθεί ο Proxy-server είναι να κρυπτογραφεί το plaintext με τα στρώματα που κρυπτογράφησης που έχει.

INSERT INTO test VALUES (1, "bob");

Αυτό που κάνει το σύστημα είναι να παίρνει τιμή-τιμή και να την κρυπτογραφεί συμφώνα με τα σωστά Onions.

Για παράδειγμα για την τιμή bob:

Plain Text: bob
DETJOIN[bob] = DetJoin_Result
DET[DetJoin_Result] = *Det_Result*
RND[Det_Result] = RND_Result

Έπειτα, αφού κρυπτογραφηθούν όλες οι τιμές, ο proxy-server στέλνει κρυπτογραφημένο ερώτημα εισαγωγής στο DBMS Server με τις κρυπτογραφημένες τιμές.

```
NEW QUERY: insert into `cryptdbtest`.`table_WLZFEKVCOL` values (1601080497516905
6310, 14420422466232759519, 4382398224217044178, '??????-D???????Q?T)?DK??O???L?
???"in???)J???????:X#??uZ*?x?\n?ZM???A??c????????]??+?Nn?h ??0?x ??????yJM????\\?
?<?B???>????S??"??????;?N?l??:r?]?????\n??????????g?????? ?6R?????wh2N?G?M?V????
!JA??\rRg?P?Q?f??q=???X?;w?C-?:??E^??)2?????I????????%????????r??????#', 6258147
858853295395, '?4?????J?oy??\n??6????9jeud"?II}?5?A???!?6;?T???U', '?02?6????rc?
`?w8c.?]Bn?4?????1Z?l?NEA?????G?????', 15408966840474158219, 5961744111291347996
)
```

Εικόνα 14: Κρυπτογραφημένο ερώτημα εισαγωγής που στέλνετε στο DBMS Server.

Παράλληλα, όλες οι τιμές στο επίπεδο DET στέλνονται στον αλγόριθμο εισαγωγής SSE για την αποθήκευση τους σε εκείνο τον Server. Στην *Εικόνα 16* δίνεται η διαδικασία που ακολουθεί ο αλγόριθμος εισαγωγής SSE, ο όποιος θα κληθεί δυο φορές, μια για το πεδίο id και μια για το πεδίο name. Τα ορίσματα που δίνονται για αυτόν τον αλγόριθμο παρουσιάζονται στην *Εικόνα 15*.

```
the tuple is: 1814899306514963523!!!◆zlI◆◆◆v◆b%◆(◆◆ztQ◆◆0d]◆i9D◆$◆4
the keyword is: ◆zlI◆◆◆v◆b%◆(◆◆ztQ◆◆0d]◆i9D◆$◆4,name,table_WLZFEKVCOL
the tuple is: 1814899306514963523!!!◆zlI◆◆◆v◆b%◆(◆◆ztQ◆◆0d]◆i9D◆$◆4
the keyword is: 1814899306514963523,id,table_WLZFEKVCOL
```

Εικόνα 15: Ορίσματα για τον αλγόριθμο Εισαγωγής SSE

Εικόνα 16: Εισαγωγή του tuple για το πρώτο πεδίο στο SSE

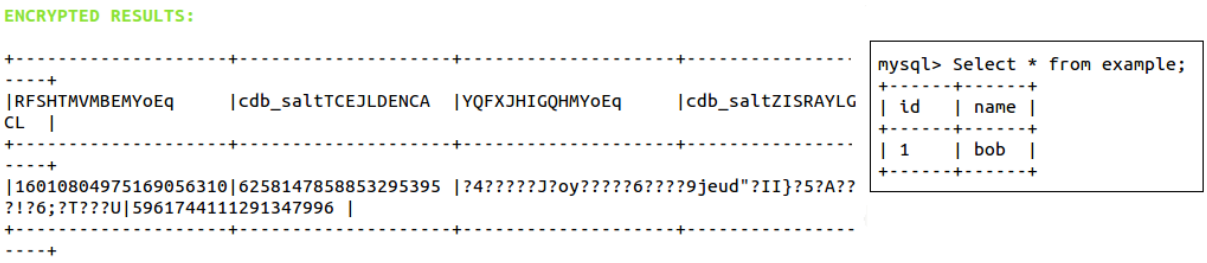
Στη συνέχεια, ο χρήστης εκτελεί ένα ερώτημα αναζήτησης.

Select * from test;

```
NEW QUERY: select `cryptdbtest`.`table_WLZFEKVCOL`.`RFSHTMVMBEMYoEq`,`cryptdbtes
t`.`table_WLZFEKVCOL`.`cdb_saltTCEJLDENCA`,`cryptdbtest`.`table_WLZFEKVCOL`.`YQF
XJHIGQHMYoEq`,`cryptdbtest`.`table_WLZFEKVCOL`.`cdb_saltZISRAYLGCL` from `cryptd
btest`.`table_WLZFEKVCOL`
```

Εικόνα 17: Κρυπτογραφημένο ερώτημα Select star.

Το προηγούμενο ερώτημα, επειδή δεν έχει καμία συνθήκη, μπορεί να απαντηθεί αποτελεσματικά με το επιλεγμένο επίπεδο RND. Έτσι, ο DBMS Server θα φορτώσει όλα τα στοιχεία του πινάκα και θα τα στείλει στο proxy-server για αποκρυπτογράφηση. Έπειτα, αυτά τα στοιχεία θα σταλθούν στο χρηστή *(Εικόνα 18).* Στην αριστερή πλευρά της εικόνας, φαίνεται το αποτέλεσμα που πήρε ο proxy-server από τον server, το όποιο θα αποκρυπτογραφήσει, ενώ στη δεξιά πλευρά φαίνεται το αποτέλεσμα που δόθηκε στον χρηστή.

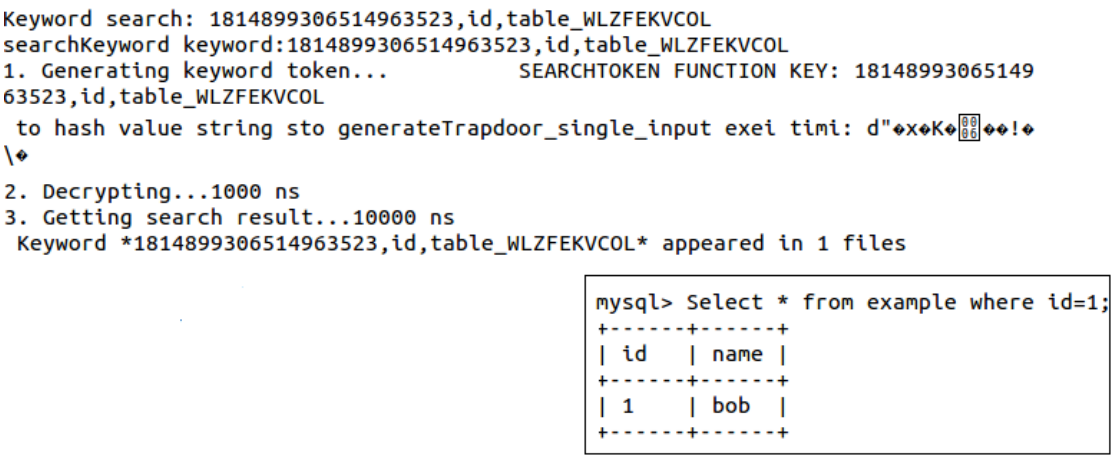


Εικόνα 18: Η απάντηση στο ερώτημα select star.

Τέλος, ο χρήστης έδωσε ένα point query ερώτημα το όποιο απαντιέται πλέον μέσω τεχνική SSE.

Select * from test where id=1;

Στην *Εικόνα 19* φαίνεται το keyword που στάλθηκε στον αλγόριθμο αναζήτησης SSE. Ο αλγόριθμος αυτός στέλνει το keyword token στο Server, και δέχεται τον Πινάκα I[keyword_index,*] ως μονοδιάστατο I_Prime. Στη συνέχεια, αποκρυπτογραφεί για να ελέγξει αν υπάρχει αυτή η λέξη σε κάποιο Tuple.



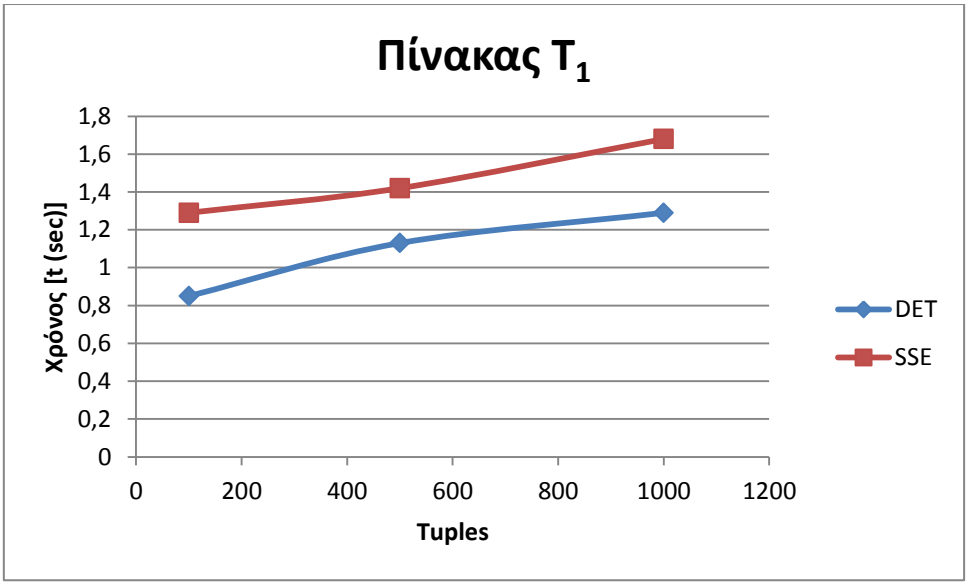
Εικόνα 19: Point query με SSE.

4. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ – ΤΡΟΠΟΠΟΙΗΣΗ ΣΧΗΜΑΤΟΣ

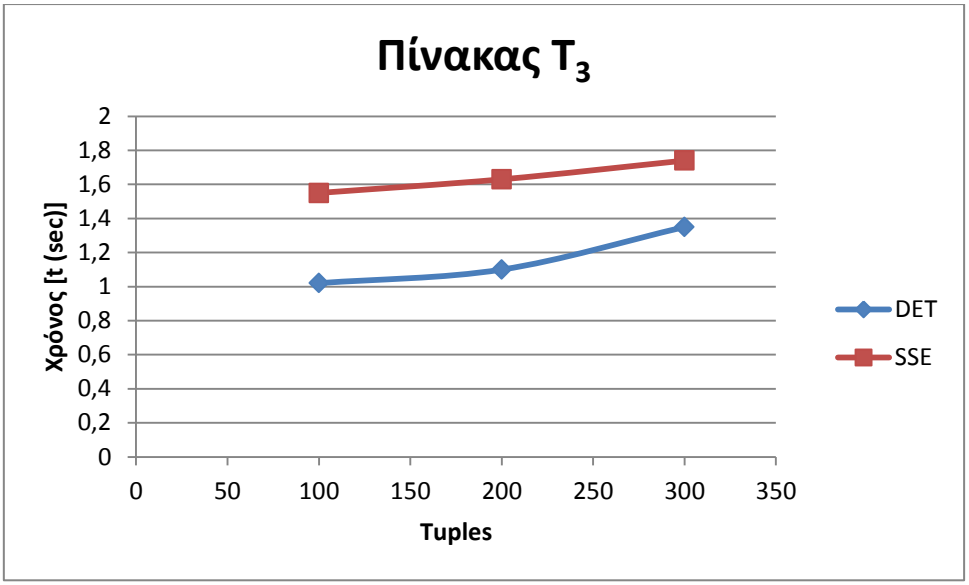
4.1 Πειράματα

Σε αυτήν την ενότητα παρουσιάζονται τα πειραματικά αποτέλεσμα στα point queries, συγκρίνοντας τις τεχνικές DET και SSE, εκ των οποίων η δεύτερη προσαρμόστηκε στο CryptDB σύστημα στα πλαίσια της παρούσας διπλωματικής εργασίας.

Για την πειραματική αξιολόγηση, δημιουργήθηκαν δύο πινάκες ο T_1 με ένα attribute μόνο και ο T_3 με τρία attributes. Ο πινάκας T_1 αντιστοιχεί στο χρόνο αν υπήρχε ένα Index για κάθε attribute και tuple και ο T_3 για το Index που έχει το σχήμα. Προστέθηκαν tuples σε αυτούς τους πίνακες και μετρήθηκε ο χρόνος που χρειάζεται για να απαντήσουν τα Point Queries ερωτήματα, τα οποία θέλουν ως αποτέλεσμα όλο το tuple (select star ερωτήματα). Παρακάτω εμφανίζονται τα δύο διαγράμματα για τους πίνακες T_1 και T_3, καθώς και παρατηρήσεις πάνω σε αυτά. Ο κάθετος άξονας απεικονίζει τον χρόνο (σε sec), ενώ ο οριζόντιος άξονας των αριθμό των tuples που δόθηκαν ως εισαγωγή.



Διάγραμμα 1:Χρόνος απάντησης με SSE & DET για τον πινάκα με ένα attribute



Διάγραμμα 2: Χρόνος απάντησης με SSE & DET για τον πινάκα με τρία attribute

Από τα παραπάνω διαγράμματα φαίνεται ότι ο χρόνος απάντησης ερωτημάτων είναι μεγαλύτερος κατά την τεχνική SSE σε σύγκριση με την κρυπτογράφηση DET. Η SSE τεχνική χρειάζεται τη δημιουργία trapdoors για το ξεκλείδωμα κόμβων του inverted index που περιέχουν τα tuples. Αυτά προσθέτουν overhead. Έτσι η SSE τεχνική είναι μεν πιο αργή, αλλά παρέχει μεγαλύτερη ασφάλεια, και γι αυτό προτιμάται. Διαπιστώνεται ακόμα ότι ο *Πίνακας* T_3 κάνει περισσότερο χρόνο να απαντήσει ένα ερώτημα με την τεχνική SSE και DET. Για την DET κρυπτογράφηση αυτό οφείλεται στο γεγονός ότι τα ερωτήματα ήταν select star, άρα θα ελεγχτούν και τα τρία attribute σε τι επίπεδο πρέπει να είναι τα δεδομένα σε εκείνες τις στήλες. Στην περίπτωση αυτή, ο proxy-server για να απαντήσει το ερώτημα απαιτεί το attribute του where clause σε DET μορφή και τα άλλα σε RND. Για την τεχνική SSE, αυτό οφείλεται στο γεγονός ότι όλα τα attributes διαφορετικών πινάκων αποθηκεύονται στο ίδιο ανάστροφο ευρετήριο. Αυτό πρακτικά σημαίνει ότι η αναζήτηση στον T_3 με το SSE σχήμα επεξεργάζεται περισσότερα δεδομένα απ' ότι στα δεδομένα του T_1. Ο *Πίνακας* T_1 περιέχει το κόστος του SSE σε ένα attribute, αν είχε δημιουργηθεί ξεχωριστό SSE index ανά attribute και πίνακα (table).

4.2 Τροποποίηση σχήματος SSE

Το σχήμα SSE όπως έχει είδη αναφερθεί είναι υπεύθυνο για την απάντηση point queries ώστε να μην υπάρχουν οι διαρροές της τεχνικής DET. Υπάρχει ένα κοινό SSE ευρετήριο για όλους τους πίνακες, attributes. Αυτό επιλέχθηκε διότι κρύβει το queried attribute και το table κατά τη διάρκεια της αναζήτησης. Στην ενότητα αυτή περιγράφουμε τις απαιτούμενες τροποποιήσεις ώστε να δημιουργηθεί ένα ξεχωριστό ευρετήριο ανά attribute και table. Σε αυτή την περίπτωση θα αλλάξουν κάποια κομμάτια στον κώδικα.

Αρχικά, κάθε φορά που θα δημιουργείται ένας πινάκας, θα δημιουργούνται και τα αντίστοιχα Index για αυτόν τον πινάκα (τα μη κρυπτογραφημένα Index), τα οποία θα κρυπτογραφούνται στη συνεχεία. Το ευρετήριο θα έχει όνομα έχει όνομα που θα παράγεται από μία συνάρτηση (πχ, χρησιμοποιώντας HMAC ως ένα PRF) του attribute και του πίνακα. Έπειτα, κάθε φορά που έρχεται ένα ερώτημα εισαγωγής θα δίνονται ως όρισμα το tuple και το keyword όπως παλιά. Το keyword θα έχει ως τιμή μόνο την τιμή του πεδίου (και όχι το όνομα του attribute και του πινάκα). Τέλος, θα χρειαστεί και ένα επιπλέον όρισμα, το όνομα του Index, δηλαδή το όνομα του attribute και πινάκα μαζί, ώστε να ξέρει ακριβώς το σύστημα σε ποιο Index να κάνει την εισαγωγή.

Κατά την αναζήτηση, θα δίνονται δυο ορίσματα, αντί για ένα όπως με το υπάρχον σχήμα. Το πρώτο όρισμα θα είναι το keyword με την τιμή του πεδίου και το δεύτερο όρισμα θα είναι το όνομα του Index. Έτσι, το σύστημα θα ξέρει ακριβώς σε ποιο Index να ψάξει.

5. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΣΤΑΣΕΙΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

5.1 Συμπεράσματα

Στην παρούσα εργασία προτάθηκε ένας πιο ασφαλής τρόπος αναζήτησης point-queries, από τους είδη υπάρχοντες. Μελετώντας το σύστημα CryptDB και τη τεχνική που απαντάει point-queries σε αυτό το σύστημα, προστέθηκε η τεχνική SSE στο CryptDB σύστημα. Έτσι κάθε φορά που τα ερωτήματα point queries απαιτούσαν DET επίπεδο για να απαντηθούν, πλέον θα απαντιούνται από την SSE τεχνική. Με αυτόν τον τρόπο θα μειώσει τις διαρροές στο Server. Αυτό οφείλεται στο γεγονός ότι δε θα εμφανίζεται η σχέση των δεδομένων όπως πριν, δηλαδή με την κρυπτογράφηση DET, όπου δυο plaintext αντιστοιχούν στο ίδιο κρυπτογραφημένο κείμενο.

5.2 Διδάγματα (Lessons Learned)

Η πιο σημαντική δυσκολία κατά την εκπόνηση της διπλωματικής ήταν η ανάγνωση και κατανόηση του κώδικα CryptDB, το οποίο περιέχει 18.000 γραμμές κώδικα C++ και Lua module 150 γραμμές. Γνωρίζοντας τη λειτουργία του κώδικα και τα σημαντικά σημεία, στη συνέχεια θα έπρεπε να δημιουργηθεί ο αλγόριθμος απάντησης ερωτημάτων σημείου, και η προσθήκη κατάλληλου κώδικα στα σωστά σημεία. Αυτό ήταν και το πιο χρονοβόρο κομμάτι της εργασίας αυτής.

5.3 Προτάσεις για μελλοντική έρευνα

Μετά την ολοκλήρωση αυτής της εργασίας γίνονται προτάσεις για περαιτέρω μελέτη και έρευνα αυτού του αντικειμένου. Αυτά που προτείνονται είναι τα εξής:

- ✓ Να προστεθεί η τεχνική Dynamic SSE, η οποία επιτρέπει την ενημέρωση (update) και τη διαγραφή (delete) στα ερωτήματα που μελετήθηκαν.

- ✓ Μια επιπλέον επέκταση του συστήματος, η οποία θα μπορούσε να είναι η μετατροπή των Range Queries (στρώμα κρυπτογράφησης OPE) σε αναζήτηση πολλαπλών λέξεων κλειδιών (multi-keyword search). Με τον τρόπο αυτό θα μειωθεί η διαρροή της διάταξης των δεδομένων που δημιουργείται από την κρυπτογράφηση OPE, μέσω της τεχνικής SSE.

- ✓ Τέλος, προτείνεται η υποστήριξη της επανακρυπτογράφηση στην περίπτωση του JOIN ερωτήματος, μόνο όμως όταν οι στήλες περιέχουν ευαίσθητα δεδομένα, τα οποία δεν ζητούνται συχνά από τον χρηστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Cryptdb System https://css.csail.mit.edu/cryptdb/

[2] Popa R.A., Redfield C.M.S., Zeldovich N., Balakrishnan H., CryptDB: Protecting Confidentiality with Encrypted Query Processing, 23rd ACM Symposium on Operating Systems Principles *(SOSP)*, (2011).

[3] Popa R.A., Building Practical Systems that Compute on Encrypted Data, Ph.D. thesis, (2014).

[4] Skiba M., Analysis of Encrypted Databases with CryptDB, Bachelor thesis, Ruhr-University Bochum, Germany, (2015).

[5] Popa R.A., Zeldovich N., Balakrishnan H., CryptDB: Processing Queries on an Encrypted Database, Communication of the ACM, 55 (9), 103- 111, (2012).

[6] Kamara S., Papamanthou C., Roeder T., Dynamic Searchable Symmetric Encryption, ACM Conference on Computer and communication security, CCS ’12, 965- 976, (2012).

[7] Demertzis I., Papadopoulos S., Papapetrou O., Deligiannakis A., Garofalakis M. Practical Private Range Search Revisited, International Conference on Management of Data, SIGMOD '16, 185- 198, (2016).

[8] Attila A.Y., and Guajardo J., Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Updates on Commodity Hardware, 22nd International Conference on Selected Areas in Cryptography, SAC 2015, Vol. 9566, 241- 259, (2015).

[9] Demertzis I., Practical Secure and Efficient Range Search, Master thesis, Technical University of Crete, Greece, (2015).

[10] SSE τεχνική https://github.com/thanghoang/IM-DSSE

ΠΑΡΑΡΤΗΜΑ

Σε αυτό το σημείο παρατίθενται επιπλέον χρήσιμα στοιχεία για τους κώδικες. Αρχικά πρέπει να αναφερθεί πως η λήψη του κώδικα CryptDB έγινε από την επίσημη σελίδα του συστήματος [1]. Έχει σχεδιαστεί για να τρέχει σε Ubuntu και συγκεκριμένα μέχρι την έκδοση 13.04. Ωστόσο, στο GitHub υπάρχουν πολλές επεκτάσεις του ώστε να λειτουργεί και πιο νέες έκδοσης Ubuntu (έως και 16.04). Στην εργασία το σύστημα αυτό εγκαταστάθηκε στα Linux 12.04. Ο κώδικας SSE πάρθηκε από το GitHub [10]. Εγκαταστάθηκε και αυτός στην ίδια έκδοση Linux και μπορεί να τρέχει για οποιαδήποτε έκδοση.

Α. Σημαντικά Κομμάτια Κώδικα

Στις *Εικόνες 20 & 21* φαίνονται τα Onions και τα επίπεδα κρυπτογράφησης που έχει το Onion Eq.

```
typedef enum onion {
    MYoDET,
    oDET,
    oOPE,
    oAGG,
    oSWP,
    oPLAIN,
    oBESTEFFORT,
    oINVALID,
} onion;
```

Εικόνα 20: Onions

```
{oDET, std::vector<SECLEVEL>({SECLEVEL::DETJOIN, SECLEVEL::DET,
                              SECLEVEL::RND})},
```

Εικόνα 21: Κρεμμυδι Eq

Στην *Εικόνα 22* φαίνεται ένα κομμάτι κώδικα που αφορά την προσαρμογή επιπέδου. Ελέγχει να δει αν το επίπεδο που απαιτείται για να απαντηθεί το ερώτημα είναι μικρότερο από το ήδη υπάρχον επίπεδο, και αν το επίπεδο που απαιτείται δεν είναι DET. Αν ισχύουν και οι δυο περιορισμοί γίνεται προσαρμογή επιπέδου. Σε διαφορετική περίπτωση δε χρειάζεται να γίνει προσαρμογή επιπέδου, είτε επειδή το ερώτημα θα απαντηθεί από το SSE Server, είτε επειδή το επίπεδο που έχει εκείνη η στήλη επαρκεί για την απάντηση του ερωτήματος που έχει στείλει ο χρήστης.

```
//check if we need onion adjustment
const OnionMeta &om =
    a.getOnionMeta(db_name, plain_table_name, i.field_name,
                   constr.o);
const SECLEVEL onion_level = a.getOnionLevel(om);
assert(onion_level != SECLEVEL::INVALID);
if (constr.l < onion_level && TypeText<SECLEVEL>::toText(constr.l)!="DET")
{
    //need adjustment, throw exception

   std::cout << "onion_level: " <<  TypeText<SECLEVEL>::toText(onion_level) << std::endl;
   std::cout << "constr.l: " <<  TypeText<SECLEVEL>::toText(constr.l) << std::endl;
   std::cout << "constr.onion: " << TypeText<onion>::toText(constr.o) << std::endl;



    const TableMeta &tm =
        a.getTableMeta(db_name, plain_table_name);
    throw OnionAdjustExcept(tm, fm, constr.o, constr.l);
}
else
{
  std::cout << "not adjustment\n";
}
```

Εικόνα 22: Έλεγχος για προσαρμογή κρεμμυδιού

Στις *Εικόνες 23 & 24* φαίνεται η κλήση του SSE για την εισαγωγή και την αναζήτηση, αντίστοιχα. Ως όρισμα δίνονται τα στοιχεία tuple, keyword για την εισαγωγή, ενώ το Keyword για την αναζήτηση.

```
ENOSI enosi;
std::cout << "SSE\n";
std::cout << "-------------BEGIN----------------" << std::endl;
enosi.addTuple(Sendtuple,*it);
std::cout << "-------------END----------------" << std::endl;
```

Εικόνα 23: Κλήση SSE αλγορίθμου εισαγωγής.

```
for (std::vector<std::string>::iterator it = a.keyword_SEND.begin(); it != a.keyword_SEND.end(); ++it)
{
   std::string keyword = *it+","+tm.getAnonTableName();
   std::cout << "CALL THE SSE\n";
   enosi.searchKey(keyword);
}
```

Εικόνα 24: Κλήση SSE αλγορίθμου αναζήτησης.

Στην *Εικόνα 25*, παρουσιάζεται η δομή του Keyword σε μορφή κώδικα. Το Keyword περιλαμβάνει την τιμή του πεδίου, το όνομα του πεδίου, καθώς και το όνομα του πίνακα, τα οποία διαχωρίζονται με κόμμα μεταξύ τους.

```
keywords.push_back(encrypt+","+fmVec[count-1]->getFieldName()+","+tm.getAnonTableName());
```

Εικόνα 25: Κλειδί για το SSE.

Β. Εγκατάσταση CryptDB

Οι εντολές που χρειάζονται για να εγκατασταθεί με επιτυχία αυτό το σύστημα στα *Linux 12.04* είναι οι εξής:

1. sudo apt-get update
2. sudo apt-get install git ruby
3. git clone -b public git://g.csail.mit.edu/cryptdb
4. cd cryptdb
5. sudo ./scripts/install.rb .
6. Στο αρχείο .bashrc, export EDBDIR=/full/path/to/cryptdb/

Γ. Εγκατάσταση SSE

Για να λειτουργήσει σωστά αυτός ο κώδικας έπρεπε να εγκατασταθούν οι εξής βιβλιοθήκες:

1. ZeroMQ Library
2. Libtomcrypt Library
3. Google sparsehash Library
4. Intel AES-NI Library (προαιρετικό)

Δ. Σημαντικά κομμάτια κώδικα που τροποποιήθηκαν

Υπάρχουν *σημαντικές* συναρτήσεις οι οποίες τροποποιήθηκαν και αξίζει να αναφερθούν:

- Η συνάρτηση *do_rewrite_type* στο αρχείο *rewrite_field.cc*. Αυτή η συνάρτηση χρησιμοποιείται για την προσαρμογή του κρεμμυδιού, το οποίο είναι υπεύθυνο για την απάντηση του ερωτήματος. Προστέθηκε κώδικας ώστε να μην κατεβαίνει επίπεδο DET αλλά να καλείται η SSE Τεχνική.

- Η συνάρτηση *rewrite* στο αρχείο *dml_handler.cc*, η οποία είναι υπεύθυνη για την εισαγωγή. Σε αυτή την συνάρτηση, προστέθηκε κώδικας ώστε να μπορεί να δημιουργείται σωστά το κρυπτογραφημένο ζευγάρι tuple,keyword και να στέλνεται στην SSE Τεχνική.

- Η συνάρτηση *gather* στο αρχείο *dml_handler.cc* καλεί την SSE τεχνική κάθε φορά που είναι να απαντηθεί το ερώτημα με το DET επίπεδο, αφού πρώτα έχει δημιουργηθεί σωστά το keyword.

- Η συνάρτηση *decryptResults* στο αρχείο *rewrite_main.cc*, η οποία είναι υπεύθυνη για την αποκρυπτογράφηση. Αποκρυπτογραφεί τα αποτελέσματα, τα οποία είτε έχει λάβει από το DBMS Server ή τον SSE Server.