Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Αποκεντρωμένη Κρυπτογράφηση Βασισμένη σε Χαρακτηριστικά με Χρήση Hardware Root of Trust

Decentralized Attribute Based Encryption Leveraging a Hardware Root Of Trust

Σχεδίαση, Μελέτη και υλοποίηση

Διπλωματικη Εργασια

Στέφανος Βασιλειάδης
STEFANOS VASILEIADIS

Επιβλέπων: Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Αποκεντρωμένη Κρυπτογράφηση Βασισμένη σε Χαρακτηριστικά με Χρήση Hardware Root of Trust

Decentralized Attribute Based Encryption Leveraging a Hardware Root Of Trust

Σχεδίαση, Μελέτη και υλοποίηση

Διπλωματικη Εργασια

Στέφανος Βασιλειάδης
STEFANOS VASILEIADIS

Επιβλέπων: Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18 Ιουνίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αριστείδης Παγουρτζής	Γεώργιος Γκούμας	Νικόλαος Λεονάρδος
Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνουμε ενυπογράφως ότι είμαστε αποκλειστικοί συγγραφείς της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχουμε αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνουμε την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαστε υπόλογοι έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μας Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνουμε, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμάς προσωπικά και αποκλειστικά και ότι, αναλαμβάνουμε πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μας ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφές)

.....	.....
Στέφανος Βασιλειάδης	Stefanos Vasileiadis

18 Ιουνίου 2024

Περίληψη

Ενας απ΄ τους πιο ανερχομενους τομεις στο κλαδο της ασφαλειας των υπολογιστων σημερα ειναι το Tusted Computing, το οποιο οριζει οτι ενα στοιχειο ενος υπολογιστη θα συμπεριφερεται συστηματικα με συγκεκριμενο τροπο αφαιρωντας ακομα και απ΄τον ιδιο τον χρηστη την δυνατοτητα να αλλαξει την συμπεριφορα του ¨εμπιστου¨ στοιχειου. Λογω της διαδεδομενης χρησης του διαδικτυου και του τεραστιου ογκου πληροφοριων που ανταλλασσονται καθημερινα ειναι πολυ σημαντικο να υπαρχουν τοσο ισχυρα επιπεδα κρυπτογραφησης αυτων των δεδομενων οσο και αποδεικτικα στοιχεια οτι προηλθαν απ΄την συσκευη που το ισχυριζεται.

Στοχος αυτης της διπλωματικης εργασιας ειναι η αναπτυξη ενος πρωτοκολλου attribute based encryption που θα χρησιμοποιει ενα ¨εμπιστο¨ στοιχειο του υπολογιστη συγκεκριμενα το Trusted Platform module (TPM) και θα δινει την δινατοτητα στο χρηστη να διαχειριζεται ο ιδιος τα κρυπτογραφικα κλειδια. Προσφεροντας ετσι μια λυση κατα την οποια ο χρηστης διατηρει την ιδιοτηκοτητα, μιας και τα κρυπτογραφικα κλειδια δεν διανεμονται απο μια κεντρικη εμπιστη αρχη.

Λέξεις Κλειδιά

TPM, attributes, Trusted Computing, ECC, PCRs

Abstract

One of the hottest topics nowadays in the field of cyber-security is Trusted Computing.Trusted Computing basically dictates that a certain entity inside the device will consistently behave in expected ways, and those behaviors will be enforced by computer hardware and software. With this solution even the owner of the the device looses his primitives on having complete control over his system as he can't change the expected behavior of the trusted component. Given the extended use of the internet, as that every device is one way or another connected to it, and the huge amount of data that are transferred over the network every day it is really important to have high standards of both encryption and authentication of the data

This diploma thesis aims to the development an attribute based encryption scheme, which will leverage the use of a hardware trusted component of the device, specifically a Trusted Platform Module (TPM). The TPM is going to be responsible for the creation and the management of the cryptographic keys. This way we propose a a privacy preserving solution, as the cryptographic keys are not created and distributed by a trusted third party

Keywords

TPM, attributes, Trusted Computing, ECC, PCRs

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Αριστείδη Παγουρτζή για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Εργαστήριο Λογικής και Επιστήμης Υπολογισμών. Επίσης ευχαριστώ ιδιαίτερα τον Δρ. Αθανάσιο Γιαννέτσο για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε και συνεχίζουμε να έχουμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Ιούνιος 2024

Στέφανος Βασιλειάδης
Stefanos Vasileiadis

Περιεχόμενα

Κατάλογος Σχημάτων

Πρόλογος

Είναι γεγονός ότι ο κόσμος των υπολογιστών γίνεται όλο και πιο επικίνδυνος, καθώς συνεχώς αυξάνονται τα περιστατικά κυβερνοεπιθέσεων. Σε μια εποχή που κινείται όλο και πιο κοντά στην πραγματοποίηση του διαδικτύου των πραγμάτων, υπάρχει μεγάλη ανάγκη να διασφαλιστεί η ιδιωτικότητα του κάθε χρήστη τόσο από κακόβουλους χρήστες όσο και από κεντρικές αρχές.

Η ανάγκη για αυξημένη ασφάλεια και προστασία των προσωπικών δεδομένων γίνεται πιο επιτακτική καθώς οι συσκευές και οι εφαρμογές που συνδέονται στο διαδίκτυο αυξάνονται με ραγδαίους ρυθμούς. Οι επιθέσεις μπορεί να έχουν σοβαρές συνέπειες, από την κλοπή προσωπικών στοιχείων έως τη διατάραξη κρίσιμων υποδομών.

Με βάση αυτή τη φιλοσοφία γεννήθηκε η ιδέα του συγκεκριμένου πρωτοκόλλου κρυπτογράφησης. Στόχος του πρωτοκόλλου είναι να παρέχει μια αξιόπιστη λύση για την προστασία των δεδομένων και της επικοινωνίας στο διαδίκτυο, διασφαλίζοντας την ακεραιότητα και την εμπιστευτικότητα των πληροφοριών.

Η εργασία διεξήχθη στο Εργαστήριο Λογικής και Επιστήμης Υπολογισμών, όπου είχα την ευκαιρία να συνεργαστώ με εξαιρετικούς επιστήμονες και να αξιοποιήσω προηγμένα εργαλεία και τεχνικές για την ανάπτυξη και την αξιολόγηση του πρωτοκόλλου. Η συμβολή του εργαστηρίου ήταν καθοριστική για την επιτυχία αυτής της προσπάθειας.

Κεφάλαιο 1

Εισαγωγή

Ο Παγκόσμιος Ιστός αποτελεί χώρο διακίνησης τεράστιου όγκου πληροφοριών. Για τον λόγο αυτό είναι αναγκαία η ύπαρξη κρυπτογραφικών αλγορίθμων όπως είναι το Attribute-based encryption (ABE). Το ABE είναι ένας κρυπτογραφικός αλγόριθμος ο οποίος δίνει τη δυνατότητα στον χρήστη να επιλέγει εκείνος ποιος θα έχει πρόσβαση στα κρυπτογραφημένα, υπό συγκεκριμένες παραμέτρους (attributes), δεδομένα του (access control). Αυτή η τεχνική χρησιμοποιείται σε μεγάλο βαθμό όταν είναι απαραίτητη η ελεγχόμενη πρόσβαση σε ευαίσθητα δεδομένα. Σε συστήματα υγείας, για παράδειγμα, το ABE αποτελεί μια πολύ καλή λύση όσον αφορά την πρόσβαση στα προσωπικά δεδομένα ενός ασθενή. Διαφορετικές ιδιότητες ενός εργαζομένου του συστήματος υγείας, όπως το επάγγελμα και η ειδικότητα, μπορούν να καθορίσουν αν αυτός θα μπορεί να έχει πρόσβαση στα δεδομένα του ασθενή και εν τέλει να τα αποκρυπτογραφήσει.

Το ABE έρχεται σε διαφορετικές μορφές, είτε υπάρχει μια κεντρική αρχή η οποία είναι υπεύθυνη για τη δημιουργία και τον διαμοιρασμό των κρυπτογραφικών κλειδιών, είτε ο κάθε χρήστης/συσκευή είναι υπεύθυνος για τη δημιουργία των κρυπτογραφικών κλειδιών.

Στα πρωτόκολλα της δεύτερης κατηγορίας συναντάται ένα πολύ μεγάλο πρόβλημα. Δεν υπάρχει καμία εγγύηση ότι τα κρυπτογραφικά κλειδιά έχουν δημιουργηθεί σύμφωνα με τα παγκόσμια στάνταρ και πως δεν έχουν διαρρεύσει, μιας και είναι απλά αποθηκευμένα στη stack/heap μνήμη της διεργασίας. Σε περίπτωση λοιπόν που η συσκευή έχει εκτεθεί σε κάποιο κακόβουλο λογισμικό ή κάποιο κακόβουλο χρήστη, είναι πολύ εύκολο να αποσπαστούν, να διαγραφούν ή να αλλαχθούν.

Τη λύση σε αυτό το πρόβλημα έρχεται να δώσει το Trusted Computing μέσω του οποίου έχουμε εγγυήσεις ότι καμία ευαίσθητη πληροφορία δεν έχει διαρρεύσει και πως όλες οι λειτουργίες έχουν εκτελεστεί όπως αναμενόταν. Υπάρχουν πληθώρα επιλογών όσον αφορά την υλοποίηση μιας Trusted Computing εφαρμογής, όπως είναι τα Trusted Execution Environments (TEEs) τα οποία είναι πιο software based λύσεις που βέβαια χρησιμοποιούν το υπάρχον hardware. Ενδεικτικά τέτοιες λύσεις είναι το Gramine Intel SGX το οποίο χρησιμοποιεί λειτουργίες του επεξεργαστή του υπολογιστή για να απομονώσει μέρος της μνήμης του στην οποία θα εκτελείται μια εφαρμογή enclave, χρησιμοποιείται συνήθως σε cloud εφαρμογές, απαιτείται ωστόσο η ύπαρξη ενός Intel επεξεργαστή. Παρόμοια λειτουργία παρέχει και η λύση του Keystone το οποίο ωστόσο είναι προορισμένο αποκλειστικά για αρχιτεκτονικές RISC-V.

Εκτός από τα TEEs υπάρχουν και καθαρά Hardware based λύσεις όπως είναι το Trusted

Platform Module (TPM), το οποίο είναι και αυτό που χρησιμοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας. Το TPM είναι ένας κρυπτογραφικός μικρο-επεξεργαστής, που παρέχει πληθώρα κρυπτογραφικών συναρτήσεων. Σήμερα βρίσκεται ενσωματωμένο σε όλους τους σύγχρονους υπολογιστές, χωρίς ωστόσο να γίνεται πλήρης εκμετάλλευση των δυνατοτήτων του μιας και χρησιμοποιείται για το Secure Boot του υπολογιστή και για ασφαλή αποθήκευση ευαίσθητων πληροφοριών.

1.1 Αντικείμενο της διπλωματικής

Το βασικό ζήτημα που προκύπτει για ένα decentralized attribute-based encryption scheme είναι το πώς θα δοθούν εγγυήσεις ότι κατά τη διάρκεια της κρυπτογράφησης, κάθε διεργασία πραγματοποιήθηκε όπως αναμενόταν κι ότι καμία ευαίσθητη πληροφορία δεν έχει διαρρεύσει.

Αντικείμενο της διπλωματικής είναι η ανάπτυξη ενός συστήματος Decentralized ABE το οποίο θα χρησιμοποιεί εκτεταμένα το TPM του υπολογιστή τόσο για δημιουργία κρυπτογραφικών κλειδιών, για κρυπτογράφηση και για δημιουργία ψηφιακών υπογραφών, αφήνοντας έτσι όσο το δυνατόν λιγότερες λειτουργίες στον ¨μη εμπιστο¨ κόσμο, UnTrusted HOST.

Ένα παράδειγμα εφαρμογής του συστήματος αυτού είναι βάσεις δεδομένων στο Cloud, όπως για παράδειγμα μια βάση δεδομένων μεταξύ διαφορετικών ανώτατων εκπαιδευτικών ιδρυμάτων. Προσφέροντας έτσι στους χρήστες πλήρη έλεγχο όσον αφορά το ποιος μπορεί να έχει πρόσβαση τόσο στα ίδια τα κρυπτογραφημένα δεδομένα. Έτσι διασφαλίζεται η ιδιωτικότητα του κάθε χρήστη και του δίνεται η δυνατότητα επιλεγμένης δημοσίευσης ευαίσθητων πληροφοριών. Με αυτή τη μέθοδο μειώνεται το ρίσκο μη εξουσιοδοτημένης πρόσβασης σε ευαίσθητα δεδομένα τα οποία είναι αποθηκευμένα στη βάση δεδομένων του cloud.

1.2 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε επτά κεφάλαια:

Στο Κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο των κρυπτογραφικών αλγορίθμων που σχετίζονται με αυτή τη διπλωματική. Αρχικά περιγράφονται βασικές έννοιες συμμετρικής και μη συμμετρικής κρυπτογραφίας, συναρτήσεων κατακερματισμού (Hash functions), ψηφιακών υπογραφών, σύντομη εισαγωγή στο Direct Anonymous Attestation (DAA) και τέλος attribute-based encryption.

Στο Κεφάλαιο 3 θα γίνει ανάλυση του TPM και θα παρουσιαστούν όλες οι λειτουργίες που προσφέρει σαν ένας κρυπτογραφικός επεξεργαστής.

Στο Κεφάλαιο 4 παρουσιάζεται η ανάλυση και η σχεδίαση του συστήματος, δηλαδή η περιγραφή των υποσυστημάτων και των εφαρμογών του.

Στο Κεφάλαιο 5 θα παρουσιαστεί το μοντέλο ασφαλείας καθώς και η απόδειξη ότι το πρωτόκολλο είναι ασφαλές.

Στο Κεφάλαιο 6 παρουσιάζεται ο έλεγχος καλής λειτουργίας του συστήματος με βάση διαφορετικά σενάρια χρήσης.

Τέλος, στο Κεφάλαιο 7 δίνεται η συνεισφορά αυτής της διπλωματικής εργασίας, καθώς και μελλοντικές επεκτάσεις.

Μέρος Ι

Θεωρητικό Μέρος

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο παρουσιάζονται αναλυτικά οι βασικές κρυπτογραφικές έννοιες και οι βασικοί κρυπτογραφικοί αλγόριθμοι που σχετίζονται με την εργασία αυτή.

2.1 Συμμετρική/μη-Συμμετρική Κρυπτογραφια

2.1.1 Συμμετρική Κρυπτογραφια

Στη συμμετρική κρυπτογραφία υπάρχει ένα μόνο κλειδί, το μυστικό συμμετρικό κλειδί, το οποίο χρησιμοποιείται τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση δεδομένων. Τέτοιοι συμμετρικοί αλγόριθμοι είναι οι DES, DESX, GHOST οι οποίοι πλέον δεν θεωρούνται ασφαλείς. Ένας πιο σύγχρονος αλγόριθμος είναι ο ChaCha που ανήκει στην οικογένεια των κρυπταλγορίθμων τμήματος και έχει αποδειχθεί ασφαλής στις περισσότερες γνωστές επιθέσεις δεδομένου ότι χρησιμοποιείται σωστά. Στη συγκεκριμένη διπλωματική εργασία χρησιμοποιείται Advanced Encryption Standard (AES). Ανήκει και αυτός στην οικογένεια των κρυπταλγορίθμων τμήματος όπως ο ChaCha και ο DES και δημιουργήθηκε ουσιαστικά για να αντικαταστήσει τον DES όταν αυτός κριθεί μη ασφαλής. Προσφέρει διαφορετικές λειτουργίες, όπως CFB, CBC, υποστηρίζοντας ταυτόχρονα διαφορετικά μήκη κλειδιών (128, 192, 256) ανάλογα με το επίπεδο ασφάλειας που επιδιώκεται.

2.1.2 Μη Συμμετρική Κρυπτογραφία

Η μη-συμμετρική κρυπτογραφία, γνωστή επίσης και ως κρυπτογραφία δημοσίου κλειδιού, χρησιμοποιεί ένα ζεύγος κλειδιών για κρυπτογράφηση/αποκρυπτογράφηση και για υπογραφές/πιστοποιήσεις. Το ζεύγος κλειδιών αποτελείται από ένα δημόσιο κλειδί και ένα μυστικό κλειδί. Το δημόσιο κλειδί είναι διαθέσιμο σε οποιονδήποτε, ενώ το μυστικό κλειδί πρέπει να παραμένει μυστικό από όλους εκτός από τον κάτοχο του.

Το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση δεδομένων και την επαλήθευση υπογραφών, ενώ το μυστικό κλειδί χρησιμοποιείται για την αποκρυπτογράφηση δεδομένων και τη δημιουργία υπογραφών. Μόνο ο κάτοχος του μυστικού κλειδιού έχει τη δυνατότητα να παράγει υπογραφές και να αποκρυπτογραφεί δεδομένα με αυτό.

Παραδείγματα αλγορίθμων που χρησιμοποιούνται στη μη-συμμετρική κρυπτογραφία είναι ο RSA, ο οποίος είναι ασφαλής για κλειδιά μήκους 2048 βιτς και μεγαλύτερα και

υποστηρίζει τόσο κρυπτογράφηση όσο και ψηφιακές υπογραφές, καθώς και ο ECDSA (Ελλειπτική Καμπύλη Ψηφιακή Υπογραφή), ο οποίος βασίζεται σε συγκεκριμένες μαθηματικές ομάδες ελλειπτικών καμπυλών που έχουν καθοριστεί από το NIST και επιτρέπει τη δημιουργία ψηφιακών υπογραφών.

2.2 Συναρτήσεις Κατακερματισμού

Μια συνάρτηση κατακερματισμού αποτελεί τη μοναδική ταυτότητα για κάθε μήνυμα. Είναι ουσιαστικά η διαδικασία κατά την οποία κάθε μήνυμα μετατρέπεται σε ένα κρυπτογραφημένο μήνυμα συγκεκριμένου μεγέθους, το "κρυπτογραφημένο" μήνυμα ονομάζεται hash/digest. Το βασικό χαρακτηριστικό των συναρτήσεων κατακερματισμού είναι η ευκολία υπολογισμού του hash/digest από το αρχικό μήνυμα, ενώ είναι δύσκολο να βρεθεί το αρχικό μήνυμα, καθώς και ο υπολογισμός ενός μηνύματος το οποίο να έχει το ίδιο hash/digest, αυτό το φαινόμενο είναι γνωστό ως collision. Λέμε ότι μια συνάρτηση κατακερματισμού είναι ασφαλής όταν δεν βρίσκονται collisions. Στη συγκεκριμένη εργασία έχουν χρησιμοποιηθεί οι συναρτήσεις της οικογένειας SHA οι οποίες έχουν κριθεί ασφαλείς σε μεγάλο βαθμό και είναι πολύ διαδεδομένες. Εκμεταλλευόμενοι τις ιδιότητες των ασφαλών συναρτήσεων κατακερματισμού δημιουργούνται πολύ χρήσιμες εφαρμογές όπως το HMAC. Το HMAC είναι μια μέθοδος πιστοποίησης που χρησιμοποιεί μια ασφαλή συνάρτηση κατακερματισμού και ένα κοινό κλειδί μεταξύ των χρηστών και απαιτεί κάποιο είδος πιστοποίησης. Το HMAC είναι διαδεδομένο σε πρωτόκολλα επικοινωνίας του διαδικτύου όπως HTTPS, SFTP, FTPS.

2.3 Ψηφιακές Υπογραφές

Οι ψηφιακές υπογραφές είναι κρυπτογραφικοί μηχανισμοί που χρησιμοποιούν μη-συμμετρική κρυπτογραφία για να παρέχουν αποδείξεις ακεραιότητας για ένα ψηφιακό έγγραφο ή μήνυμα, καθώς και εγγυήσεις ότι το μήνυμα προήλθε από συγκεκριμένη πηγή. Η διαδικασία λειτουργεί με τον εξής τρόπο: Ο υπογράφων χρησιμοποιεί το ιδιωτικό του κλειδί για να υπογράψει το μήνυμα ή το έγγραφο, παράγοντας ένα ψηφιακό υπογραφή (digital signature). Ο αποδέκτης του μηνύματος χρησιμοποιεί το δημόσιο κλειδί του υπογράφοντος για να επαληθεύσει την υπογραφή και να επιβεβαιώσει την ακεραιότητα του μηνύματος ή του εγγράφου. Αυτός ο μηχανισμός επιτρέπει στους παραλήπτες να εμπιστευτούν ότι το μήνυμα δεν έχει παραβιαστεί ή παραποιηθεί και ότι προέρχεται από τον αναμενόμενο αποστολέα.

Η ασφάλεια των ψηφιακών υπογραφών βασίζεται στη δυσκολία του υπολογισμού του ιδιωτικού κλειδιού από το δημόσιο κλειδί και στην αδυναμία να παραχθεί έγκυρη ψηφιακή υπογραφή χωρίς την γνώση του ιδιωτικού κλειδιού. Η επιλογή κατάλληλου μηχανισμού υπογραφής και η χρήση ασφαλών αλγορίθμων κατακερματισμού, όπως οι συναρτήσεις της οικογένειας SHA, είναι κρίσιμες για την ασφάλεια και την αξιοπιστία του μηχανισμού ψηφιακών υπογραφών.

2.4 Direct Anonymous Attestation

Το Direct Anonymous Attestation εν γένει αποτελείται από μια κεντρική αρχή (Issuer), από ένα σύνολο υπογραφόντων (signer) και από ένα σύνολο επαληθευτών (verifiers). Η κεντρική αρχή Issuer δημιουργεί διαπιστευτήρια για κάθε έναν από τους signers. Τα διαπιστευτήρια αυτά αντιστοιχούν στην ταυτότητα του κάθε signer. Ο signer συνήθως αποτελείται από το TPM και μια κοινή διεργασία που τρέχει στο μη-εμπιστο κόσμο. Για την απόδειξη ότι ένας signer είναι μέλος του DAA δικτύου αποστέλλεται σε έναν verifier μια DAA υπογραφή, η οποία περιέχει αποδείξεις μηδενικής γνώσης, με σκοπό ο signer να αποδείξει στον verifier ότι κατέχει ένα έγκυρο διαπιστευτήριο μέλους, χωρίς όμως ο verifier να μπορεί να μάθει οτιδήποτε άλλο για την ταυτότητα του signer. Σε αντίθεση με άλλα πρωτόκολλα group signature, ο Issuer δεν μπορεί να ταυτοποιήσει τον signer μέσω μιας υπογραφής. Συνεπώς δίνεται η δυνατότητα στον signer να δημιουργεί ανώνυμες υπογραφές ενώ ταυτόχρονα πείθει τον verifier ότι κατέχει σωστά διαπιστευτήρια μέλους του DAA δικτύου, λεπτομερής ανάλυση του Direct-Anonymous-Attestation και των εφαρμογών του μπορει να βρεθεί στις παρακάτω δημοσιεύσεις [18], [19], [29].

2.4.1 Ιδιοτητες

Unforgeability: Όταν η κεντρική εμπιστευτή αρχή Issuer και όλα τα TPMs που συμμετέχουν στο δίκτυο είναι ειλικρινείς τότε κανένας αντίπαλος δεν μπορεί να δημιουργήσει μια υπογραφή εκ μέρους ενός άλλου signer.

Anonymity: Ένας αντίπαλος ο οποίος κατέχει δύο διαφορετικές DAA υπογραφές, δεν μπορεί να ξεχωρίσει αν αυτές οι υπογραφές προήλθαν από τον ίδιο ή διαφορετικούς signers.

2.5 Attribute-Based-Encryption

Το ABE, πρωτο παρουσιάστηκε από τους Sahai και Waters, παρέχοντας έναν εύκολο τρόπο να οριστούν πρωτόκολλα μη-συμμετρικής κρυπτογραφίας όσον αφορά την κρυπτογράφηση, για την εφαρμογή πολιτικών βασιζόμενη σε συγκεκριμένες ιδιότητες. Έτσι και το μυστικό κλειδί του χρήστη και το κρυπτογραφημένο μήνυμα είναι αρρηκτά συνδεδεμένα με ένα σύνολο ιδιοτήτων. Σήμερα χρησιμοποίουνται σε εφαρμογές πραγματικού κόσμου περίπλοκα σχήματα ([41], [42]), ωστόσο το Attribute-Based-Encryption έχει δύο βασικές μορφές.

Το ABE έρχεται σε δύο διαφορετικές παραλλαγές, ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE).

Στο CP-ABE, ένας χρήστης κρυπτογραφεί τα δεδομένα με βάση ένα κατηγόρημα (πολιτική εξουσιοδότησης), η οποία έχει οριστεί σύμφωνα με το σύνολο ιδιοτήτων αυτού. Έτσι μόνο ένας χρήστης ο οποίος κατέχει το μυστικό κλειδί, που είναι συνδεδεμένο με το σύνολο ιδιοτήτων και ικανοποιεί το κατηγόρημα (πολιτική εξουσιοδότησης), να μπορεί να τα αποκρυπτογραφήσει.

Στο KP-ABE, έχουμε την αντίστροφη προσέγγιση. Εδώ το κρυπτογραφημένο μήνυμα είναι συνδεδεμένο με το σύνολο των ιδιοτήτων του χρήστη και το μυστικό κλειδί είναι συνδε-

δεμένο με το κατηγόρημα το οποίο έχει οριστεί σύμφωνα με το σύνολο των ιδιοτήτων.

Στην πράξη η υλοποίηση της δομής εισόδου είτε στα κλειδιά, είτε στο κρυπτογραφημένο μήνυμα (ανάλογα με το είδος που υλοποιείται), γίνεται με τεχνικές διαμοιρασμού μυστικού, το οποίο μας επιτρέπει να δημοσιεύσουμε ένα μυστικό μέσω διαφορετικών ιδιοτήτων (attributes). Η ιδέα είναι ότι το μυστικό μπορεί να ανακτηθεί μόνο εφόσον το σύνολο ιδιοτήτων (attributes) ικανοποιεί την πολιτική προσβασης.

Για παράδειγμα, αν η πολιτική προσβασης μας αποτελείται από ένα σύνολο n ιδιοτήτων, για την οποία ισχύει ότι χρειαζόμαστε τουλάχιστον t ε[1, n] εξ' αυτών ώστε να ικανοποιηθεί η πολιτική προσβασης, τότε μπορεί να εφαρμοστεί Shamir's secret sharing.

Μέρος II

Πρακτικό Μέρος

Κεφάλαιο 3

Trusted Platform Module (TPM)

Στο κεφάλαιο αυτό παρουσιάζεται αναλυτικά το TPM. Αρχικά θα περιγραφεί η αρχιτεκτονική του όσον αφορά τα κρυπτογραφικά κλειδιά που διαχειρίζεται καθώς και τις ασφαλείς μνήμες που προσφέρει. Στη συνέχεια θα παρουσιαστούν οι λειτουργίες οι οποίες χρησιμοποιήθηκαν στα πλαίσια της συγκεκριμένης εργασίας.

3.1 Ανάλυση - περιγραφή αρχιτεκτονικής του TPM: Κλειδιά

Στην ενότητα αυτή παρουσιάζεται η ανάλυση της αρχιτεκτονικής των κλειδιών, της μνήμης, καθώς και των κρυπτογραφικών δυνατοτήτων του Trusted Platform Module (TPM) ως ένας κρυπτογραφικός μικροεπεξεργαστής [38], [39], [40].

3.1.1 Διαχωρισμός κρυπτογραφικών κλειδιών: Ιεραρχία

Το Trusted Platform Module (TPM) παρέχει δύο τύπους κλειδιών. Ο διαχωρισμός αυτός γίνεται όχι ως προς τις κρυπτογραφικές δυνατότητες των εν λόγω κλειδιών, αλλά ως προς την ιεραρχία στην οποία κατατάσσονται. Το TPM εφαρμόζει μια δενδροειδή ιεραρχία στη δημιουργία των κλειδιών του. Πιο συγκεκριμένα, η ρίζα του κάθε δέντρου είναι ένα πρωτεύον κλειδί, το οποίο δημιουργείται με ασφάλεια εντός του TPM από έναν τυχαίο αριθμό που δημιουργήθηκε από τον ασφαλή γεννήτορα τυχαίων αριθμών του TPM, καθώς και από τη συνάρτηση δημιουργίας κλειδιών του. Στην περίπτωση δημιουργίας ενός πρωτεύον κλειδιού, το salt της συνάρτησης δημιουργίας κλειδιών είναι ένας μυστικός τυχαίος αριθμός ο οποίος δημιουργήθηκε κατά την κατασκευή του TPM και δεν είναι γνωστός σε κανέναν, ούτε στον κατασκευαστή, και δεν μπορεί και να αλλάξει καθ' όλη τη διάρκεια ζωής του. Το μυστικό μέρος των πρωτεύον κλειδιών δεν φεύγει ποτέ από τον εμπιστο κόσμο που εγκαθιδρύει το TPM. Τα κλειδιά τα οποία αποτελούν φύλλα του δέντρου ακολουθούν διαφορετικό αλγόριθμο ως προς τη δημιουργία τους, για να μπορεί να οριστεί και η δενδροειδής ιεραρχία. Πιο συγκεκριμένα, σε αυτήν την περίπτωση πάλι δημιουργείται ένας τυχαίος αριθμός από τον ασφαλή γεννήτορα τυχαίων αριθμών του TPM για να οριστεί ως παράμετρος της συνάρτησης δημιουργίας κλειδιών, αλλά εδώ το salt είναι το μυστικό κλειδί του εκάστοτε πρωτεύον κλειδιού. Επιπροσθέτως, σε αυτήν την περίπτωση το μυστικό κλειδί δεν αποθηκεύεται στο TPM, αλλά κρυπτογραφείται από τον πρωτεύον του κλειδιού και επιστρέφεται στον μη-ασφαλή κόσμο μαζί με το δημόσιο μέρος του κλειδιού, αν αυτό είναι μη συμμετρικό κλειδί, ή με

μια τιμή αυθεντικοποίησης, ώστε να μην μπορούν να χρησιμοποιηθούν κλειδιά τα οποία δεν δημιουργήθηκαν από το TPM.

3.1.2 Διαχωρισμός κρυπτογραφικών κλειδιών: Αλγόριθμοι

Παρακάτω δίνεται λεπτομερής περιγραφή για καθένα από τα είδη κλειδιών που υποστηρίζονται από το TPM όσον αφορά τους κρυπτογραφικούς αλγορίθμους που υποστηρίζουν. Πρέπει να σημειωθεί εδώ ότι όλοι οι αλγόριθμοι οι οποίοι υποστηρίζονται από το TPM συναντούν τις προδιαγραφές που έχουν οριστεί από το National Institute of Standards and Technology (NIST).

1. Μη Συμμετρική Κρυπτογραφία

   (α') RSA: σύστημα κρυπτογραφίας με δημόσιο κλειδί που επιτρέπει την ασφαλή μετάδοση δεδομένων και τη δημιουργία ψηφιακών υπογραφών. Παρουσιάστηκε από τους Ρον Ριεστ, Αδι Σημαιρ και Λεοναρδ Αδλεμαν το 1977. Η ασφάλεια του ΡΣΑ βασίζεται στην πρακτική δυσκολία του προβλήματος του factoring, που σημαίνει τον υπολογισμό των πρώτων παραγόντων ενός μεγάλου αριθμού. Το TPM μπορεί να δημιουργήσει RSA κλειδιά μήκους 2048 και 3072 bits, τα οποία θεωρούνται ασφαλή.

   (β') ECC: προηγμένο σύστημα κρυπτογραφίας που βασίζεται στις ιδιότητες των ελλειπτικών καμπυλών πάνω στο σύνολο των πραγματικών αριθμών. Η ECC παρέχει ισχυρή ασφάλεια με μικρότερα μεγέθη κλειδιού σε σύγκριση με άλλα παραδοσιακά συστήματα όπως το RSA. Οι ελλειπτικές καμπύλες οι οποίες είναι διαθέσιμες για δημιουργία ελλειπτικών κλειδιών στο TPM είναι οι P-256, P-384, P-521 οι οποίες παράγουν ιδιωτικά κλειδιά μήκους 256 bits και έχουν κριθεί ασφαλείς.

2. Συμμετρική Κρυπτογραφία

   (α') AES: συμμετρικός αλγόριθμος κρυπτογράφησης που χρησιμοποιείται ευρέως για την ασφαλή προστασία πληροφοριών. Ανήκει στην οικογένεια των block ciphers, εγκαθιδρύθηκε τη δεκαετία του '90 μέσω διαγωνισμού του National Institute of Standards and Technology (NIST) αντικαθιστώντας τον αλγόριθμο DES. Το TPM υποστηρίζει AES κλειδιά μήκους 128, 192 και 256 bits τα οποία έχουν κριθεί ασφαλή. Πιο συγκεκριμένα, ο AES έχει κριθεί ασφαλής ακόμα και απέναντι σε κβαντικούς υπολογιστές.

   (β') HMAC (Hash-based Message Authentication Code): δημιουργήθηκε από τους Mihir Bellare, Ran Canetti και Hugo Krawczyk το 1996. Ο σκοπός του ήταν να παρέχει έναν αποτελεσματικό μηχανισμό για τον έλεγχο ακεραιότητας και την αυθεντικοποίηση των δεδομένων. Το TPM χρησιμοποιεί τον αλγόριθμο κατακερματισμού SHA-256, ο οποίος είναι ένα ευρέως χρησιμοποιούμενο μέσο για τη δημιουργία ενός ασφαλούς κωδικού ελέγχου βασισμένου σε κατακερματισμό μηνύματος.

3.2 Ανάλυση - περιγραφή αρχιτεκτονικής του TPM : Μνημη

3.2.1 Platform Configuration Registers (PCRs)

Το Trusted Platform Module (TPM) διαθέτει ένα σύνολο από Platform Configuration Registers (PCRs). Κάθε PCR αντιστοιχεί σε έναν αριθμό (π.χ. PCR0, PCR1, κλπ.) και χρησιμοποιείται για να αποθηκεύει καταγραφικά δεδομένα σχετικά με την κατάσταση της πλατφόρμας. Αυτά τα δεδομένα είναι το αποτέλεσμα του κατακερματισμού διαφόρων παραμέτρων του συστήματος.

1. Κρυπτογραφικός Κατακερματισμός: Τα PCR χρησιμοποιούν τεχνικές κατακερματισμού, πιο συγκεκριμένα SHA256, για να δημιουργήσουν μια μοναδική "αποτύπωση" των δεδομένων που ελέγχονται.

2. Καταγραφές: Κάθε PCR καταγράφει συγκεκριμένες πληροφορίες, όπως τον κώδικα εκκίνησης, τον πίνακα περιεχομένων της μνήμης, και άλλες παραμέτρους σχετικές με την πλατφόρμα.

3. Προστασία Ακεραιότητας: Τα PCR χρησιμοποιούνται για τον έλεγχο της ακεραιότητας του λειτουργικού συστήματος και των εφαρμογών.

4. Εγγραφή και Ανάγνωση: Οι τιμές των PCR μπορούν να εγγραφούν και να διαβαστούν από εξουσιοδοτημένους χρήστες ή εφαρμογές.

3.2.2 Trusted Platform Module's Non-Volatile Memory

Η Non-Volatile memory του Trusted Platform Module (TPM) χρησιμοποιείται για την αποθήκευση δεδομένων που πρέπει να είναι προσβάσιμα από το TPM ακόμα και μετά την απενεργοποίηση του υπολογιστή. Αυτή η μνήμη είναι σημαντική για τη διατήρηση ευαίσθητων πληροφοριών, όπως κλειδιά και παραμέτρους ασφαλείας.

Η Non-Volatile memory χρησιμοποιείται κυρίως για:

1. Κλειδιά TPM: Τα μυστικά και κρυπτογραφικά κλειδιά που χρησιμοποιούνται από το TPM για διάφορες κρυπτογραφικές λειτουργίες αποθηκεύονται στη μνήμη μη-αλληλουχίας.

2. Παράμετροι Πλατφόρμας: Σημαντικές παράμετροι ασφαλείας και ρυθμίσεις του TPM αποθηκεύονται επίσης σε αυτήν τη μνήμη.

3. Αποθήκευση Αποτυπωμάτων: Τα αποτυπώματα καταγεγραμμένων δεδομένων, όπως τα Platform Configuration Registers (PCR), μπορούν επίσης να αποθηκευτούν για ασφαλή επαλήθευση της ακεραιότητας.

4. Επαναφορά Δεδομένων: Η Non-Volatile memory χρησιμοποιείται επίσης σε περιπτώσεις επαναφοράς συστήματος για την επαναφορά ευαίσθητων δεδομένων.

Η πρόσβαση στη Non-Volatile memory του TPM είναι συνήθως προστατευμένη και απαιτεί εξουσιοδότηση, εξασφαλίζοντας έτσι ότι μόνο εξουσιοδοτημένα πρόσωπα μπορούν να έχουν πρόσβαση σε αυτά τα ευαίσθητα δεδομένα.

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση και η σχεδίαση του πρωτοκόλλου, με βάση τη μελέτη που παρουσιάστηκε στα προηγούμενα κεφάλαια. Αρχικά παρουσιάζεται η αρχιτεκτονική και ο αλγόριθμος πάνω στον οποίο βασίζεται το πρωτόκολλο και στη συνέχεια τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Τέλος δίνονται οι λεπτομέρειες υλοποίησης για τους βασικούς αλγορίθμους του συστήματος καθώς και η δομή του κώδικα.

4.1 Λεπτομέρειες Αρχιτεκτονικής

Στην ενότητα αυτή παρουσιάζεται το μοντέλο πάνω στο οποίο χτίστηκε ο συγκεκριμένος Attribute Based Encryption (ABE) αλγόριθμος και περιγράφεται αναλυτικά ο ρόλος της κάθε ξεχωριστής οντότητας που συμμετέχει.

4.1.1 Οντότητες

- Attribute Master Holder (AttrMH): Είναι μια εμπιστευτική κεντρική οντότητα, η οποία είναι υπεύθυνη για τον ορισμό του χώρου των χαρακτηριστικών που ανήκουν σε κάθε χρήστη ή συσκευή που επιθυμεί να συμμετέχει σε αυτό το πρωτόκολλο και για τη δημιουργία των ελλειπτικών κλειδιών που αντιστοιχούν σε κάθε ένα από τα χαρακτηριστικά που αναφέρθηκαν πιο πάνω. Επιπλέον, ο AttrMH είναι υπεύθυνος για τον ορισμό πολιτικών, τις οποίες κάθε χρήστης/συσκευή πρέπει να εφαρμόσει έτσι ώστε να τεθεί δυνατή η επιτυχής κρυπτογράφηση και αποκρυπτογράφηση δεδομένων.

- Κρυπτογράφος: Η συσκευή/χρήστης που σκοπεύει να κρυπτογραφήσει δεδομένα με τρόπο ώστε η αποκρυπτογράφηση τους να είναι δυνατή μόνο από συσκευές/χρήστες που κατέχουν συγκεκριμένα χαρακτηριστικά, τα οποία ορίζονται κατά τη διάρκεια της κρυπτογράφησης. Τα δεδομένα που κρυπτογραφούνται μπορεί να αποτελούν ευαίσθητες πληροφορίες για τον εκάστοτε χρήστη όπως ιατρικά δεδομένα ή πληροφορίες για κάποιο είδος χρηματικής συναλλαγής.

- Αποκρυπτογράφος: Η συσκευή/χρήστης που επιθυμεί να αποκρυπτογραφήσει κρυπτογραφημένα δεδομένα χρησιμοποιώντας ABE. Για να καθιερωθεί αυτό δυνατό, ο αποκρυπτογράφος πρέπει να έχει στην κατοχή του όλα τα απαραίτητα χαρακτηριστικά που έχουν προκαθοριστεί από τον κρυπτογράφο. Πρέπει να σημειωθεί εδώ ότι

ολόκληρο το σύνολο των χαρακτηριστικών έχει οριστεί κατά την εγγραφή της εκάστοτε συσκευής/χρήστη.

Εδώ είναι η διόρθωση και η επανατύπωση του κειμένου σύμφωνα με τη σωστή γραμματική και σύνταξη:

4.2 Περιγραφή του Συνόλου Χαρακτηριστικών

Σε αυτήν την ενότητα επικεντρωνόμαστε στον καθορισμό των συνόλων όλων των δυνατών χαρακτηριστικών που μπορούν να χρησιμοποιηθούν στα πλαίσια των λειτουργιών κρυπτογράφησης και αποκρυπτογράφησης με ABE. Είναι σημαντικό να σημειωθεί ότι το συγκεκριμένο σχήμα ABE δεν δεσμεύεται από ορισμένο σύνολο χαρακτηριστικών, καθώς τα χαρακτηριστικά που θα χρησιμοποιηθούν από έναν Κρυπτογράφο επιλέγονται από τον Attribute Master Holder (AttrMH) ανάλογα με τον τύπο της συσκευής του Κρυπτογράφου και την υπηρεσία που επιθυμεί να παρέχει. Συνεπώς, δεν είναι απαραίτητο να μοιράζονται όλα τα χαρακτηριστικά μεταξύ όλων των Κρυπτογράφων ή Αποκρυπτογράφων και αντίστροφα. Αυτό σημαίνει ότι τα χαρακτηριστικά δεν χρειάζεται να είναι ούτε ίδια ούτε διαφορετικά μεταξύ τους.

Για παράδειγμα, σε ένα νοσοκομείο, ευαίσθητα ιατρικά δεδομένα μπορεί να μοιράζονται μεταξύ πολλών διαφορετικών υπολογιστών. Έτσι, όλες οι συσκευές Κρυπτογράφησης που λειτουργούν στο ίδιο επίπεδο υπάρχει ανάγκη να μοιράζονται δεδομένα θα χρησιμοποιούν το ίδιο σύνολο χαρακτηριστικών. Αντίστοιχα, συσκευές σε άλλα επίπεδα μπορεί να χρησιμοποιούν διαφορετικά σύνολα χαρακτηριστικών.

Όπως αναφέρθηκε, το συγκεκριμένο σχήμα ABE βασίζεται στα χαρακτηριστικά που κατέχει η συσκευή για να αποκρυπτογραφήσει ένα σύνολο δεδομένων που έχει κρυπτογραφηθεί με χρήση ABE, βασισμένο στις κατάλληλες πολιτικές που βασίζονται στα χαρακτηριστικά της συσκευής. Αυτές οι πολιτικές ορίζουν τα χαρακτηριστικά και τις ιδιότητες που πρέπει να επαληθευτούν από οποιαδήποτε συσκευή Κρυπτογράφησης/Αποκρυπτογράφησης, προσφέροντας τα απαραίτητα επίπεδα διασφάλισης για τον έλεγχο πρόσβασης στα δεδομένα. Τέλος, δεν καθορίζουμε τις συσκευές που μπορούν να εκτελέσουν αποκρυπτογράφηση, αλλά τα χαρακτηριστικά που πρέπει να εκδηλώνουν οι συσκευές αυτές προκειμένου να έχουν πρόσβαση σε ένα συγκεκριμένο σύνολο δεδομένων. Συγκεκριμένα, τα χαρακτηριστικά που συνδέονται με τη συμβολή του κάθε χρήστη της εκάστοτε συσκευής πρέπει να διαχειρίζονται διαφορετικά από τα χαρακτηριστικά με μακροπρόθεσμη διάρκεια, τα οποία ενδέχεται να παραμένουν ίδια καθ' όλη τη διάρκεια του λειτουργικού κύκλου ζωής της συσκευής. Αυτά συνήθως δεν αλλάζουν και, συνεπώς, μπορούν να χειριστούν διαφορετικά (για παράδειγμα, να αποθηκεύονται στα PCRs και NV-RAM του Αξιόπιστου Στοιχείου της συσκευής) για παραδειγμα ενος TPM).

4.3 Διαχείριση Χαρακτηριστικών

Σημειώνεται ότι υπάρχουν δύο είδη ιδιοτήτων που πρέπει να ληφθούν υπόψη, και κάθε είδος πρέπει να χειρίζεται διαφορετικά από το σχήμα ABE, όπως θα περιγραφεί λεπτομερώς

στο υπόλοιπο της ενότητας. Συγκεκριμένα, τα χαρακτηριστικά μπορούν να κατηγοριοποιηθούν σε στατικά και δυναμικά, τα οποία καθορίζονται ως εξής:

- **Στατικά Χαρακτηριστικά:** Αυτά καλύπτουν το σύνολο χαρακτηριστικών που κατά κανόνα δεν αλλάζουν με την πάροδο του χρόνου. Στην περίπτωση που υπάρχει αλλαγή, αυτό μπορεί να οδηγήσει σε αλλαγή στην κατάσταση μιας συσκευής, η οποία πρέπει να προστατευθεί από συχνές αλλαγές. Έτσι, επιλέγουμε να τα προσθέσουμε στα PCRs (Platform Configuration Registers) του TPM της συσκευής, ώστε να μην μπορούν να αλλάξουν χωρίς να διακοπεί η φυσιολογική λειτουργία του TPM και να απαιτείται επανεκκίνηση της συσκευής. Παραδείγματα τέτοιων ιδιοτήτων είναι το αποτύπωμα ασφαλούς εκκίνησης (Secure Boot), η έκδοση του Firmware και ο τύπος του επεξεργαστή CPU που διαθέτει η συσκευή. Κάποια στατικά χαρακτηριστικά μπορεί να προέρχονται από τον κατασκευαστή της συσκευής και να καθορίζονται κατά τη διάρκεια της κατασκευής της, αλλά μπορεί επίσης να περιλαμβάνουν ιδιότητες που καθορίζονται αργότερα όπως η ταυτότητα που λαμβάνει μια συσκευή όταν γίνεται μέλος ενός δικτύου.

- **Δυναμικά Χαρακτηριστικά:** Αυτά καλύπτουν το πλήρες φάσμα του σχεδιαστικού χώρου και περιλαμβάνουν χαρακτηριστικά του συστήματος που ενδέχεται να αλλάζουν συχνά κατά τη διάρκεια του κύκλου ζωής της συσκευής και απεικονίζουν το επίπεδο εμπιστοσύνης και την ορθότητα της λειτουργικής κατάστασης μιας συσκευής. Αυτό το είδος χαρακτηριστικών αφορά ουσιαστικά το λειτουργικό πεδίο της συσκευής και διαφέρει ανάλογα με τον τομέα εφαρμογής. Ορισμένες τέτοιες ιδιότητες μπορεί να απεικονίζουν την τρέχουσα κατάσταση της συσκευής κρυπτογράφησης/αποκρυπτογράφησης, η οποία πρέπει να βρίσκεται σε μια σωστή και αναμενόμενη κατάσταση προτού προχωρήσει σε οποιεσδήποτε επόμενες λειτουργίες.

4.4 Διαχωρισμός και Χρήση Χαρακτηριστικών

Ο διαχωρισμός αυτός πραγματοποιείται λόγω της διαφορετικής μεταχείρισης αυτών των δύο ειδών χαρακτηριστικών. Συγκεκριμένα, μόνο τα στατικά χαρακτηριστικά μετατρέπονται σε ελλειπτικά κλειδιά, ενώ τα δυναμικά χαρακτηριστικά συνδυάζονται με τα στατικά χαρακτηριστικά σε μια πολιτική περιορισμού κλειδιών. Αυτό συμβαίνει για τους εξής λόγους:

1. Τα στατικά χαρακτηριστικά πρέπει να προστατεύονται από συχνές αλλαγές.

2. Δεδομένου ότι τα στατικά χαρακτηριστικά δεν αλλάζουν με τον χρόνο, αυτά προστίθενται στο πραγματικό δέντρο χαρακτηριστικών που ένας αποκρυπτογράφος πρέπει να διασχίσει πριν είναι σε θέση να προχωρήσει στη δημιουργία των κλειδιών αποκρυπτογράφησης (λαμβάνοντας υπόψη ότι η πολιτική περιορισμού χρήσης κλειδιών είναι επίσης έγκυρη).

4.5 Decentralized ABE Επισκόπηση του Πρωτοκόλλου

Έχοντας παρουσιάσει όλα τα θεμελιώδη στοιχεία του πρωτοκόλλου καθώς και τις οντότητες που συμμετέχουν για μια επιτυχή κρυπτογράφηση και αποκρυπτογράφηση, σε αυτήν την ενότητα θα περιγραφεί μια πρώτη επισκόπηση του αλγορίθμου. Συγκεκριμένα, το σχήμα μπορεί να διακριθεί σε τρεις κύριες φάσεις, και συγκεκριμένα την Αρχικοποίηση, την Κρυπτογράφηση και την Αποκρυπτογράφηση:

- Αρχικοποίηση

    1. Εστω μια συσκευή που επιθυμεί να συμμετάσχει στο σχήμα ABE, είτε ως Κρυπτογράφος είτε ως Αποκρυπτογράφος. Αρχικά, η συσκευή αποστέλλει τα στατικά της χαρακτηριστικά στον AttrMH.
    2. Στη συνέχεια ο AttrMH δημιουργεί ελλειπτικά κλειδιά που αντιστοιχούν σε κάθε ένα από τα στατικά χαρακτηριστικά της συσκευής ή του χρήστη, καθώς και ένα κλειδί το οποίο θα εισαχθεί στην ιεραρχία κλειδιών του εκάστοτε TPM, προκειμένου να δημιουργηθούν τα κλειδιά κρυπτογράφησης/αποκρυπτογράφησης και τα κλειδιά HMAC σύμφωνα με μια πολιτική κλειδιών που περιλαμβάνει μια συνένωση των στατικών και δυναμικών χαρακτηριστικών.
    3. Αυτά τα δεδομένα κρυπτογραφούνται με το δημόσιο Endorsement κλειδί του TPM.
    4. Από την πλευρά της συσκευής, όταν αποκτά τα κρυπτογραφημένα δεδομένα, τα αποκρυπτογραφεί και αποθηκεύει τα ελλειπτικά κλειδιά που αντιστοιχούν στα στατικά χαρακτηριστικά της συσκευής μέσα στα Platform Configuration Registers (PCRs) του ενσωματωμένου TPM της, και τοποθετεί το HMAC κλειδί που παρέχεται από το AttrMH στην ιεραρχία κλειδιών του TPM.

- Κρυπτογράφηση

    1. Εάν μια συσκευή (Κρυπτογράφος) επιθυμεί να κρυπτογραφήσει ένα σύνολο δεδομένων χρησιμοποιώντας το ABE, πρώτα ζητά από το TPM της να δημιουργήσει ένα τυχαίο αριθμό.
    2. Χρησιμοποιεί αυτόν τον τυχαίο αριθμό μαζί με το κλειδί που αποτελεί τη ρίζα του δέντρου των κλειδιών των χαρακτηριστικών μιας συσκευής καθώς και την πολιτική περιορισμού κλειδιού, για να υπολογίσει το τυχαίο seed από το οποίο θα παραχθούν τα απαραίτητα συμμετρικά κλειδιά.
    3. Στη συνέχεια, δημιουργεί το Κλειδί Κρυπτογράφησης υπό το κλειδί που παρέχεται από το AttrMH, συνδεδεμένο με την κατάλληλη πολιτική.
    4. Προκειμένου να κρυπτογραφήσει τα επιθυμητά δεδομένα, η συσκευή πρέπει να υποστεί έλεγχο ακεραιότητας, χρησιμοποιώντας τόσο τα στατικά της χαρακτηριστικά όσο και τα δυναμικά της χαρακτηριστικά.
    5. Αν η συσκευή περάσει τον έλεγχο ακεραιότητας, κρυπτογραφεί τα επιθυμητά δεδομένα και δημιουργεί το HMAC τους.

6. Στη συνέχεια, η συσκευή υπολογίζει το δημόσιο ¨μυστικό¨, χρησιμοποιώντας τα δημόσια ελλειπτικά κλειδιά που αντιστοιχούν στα στατικά χαρακτηριστικά της συσκευής και το προαναφερόμενο τυχαίο αριθμό που δημιούργησε το TPM.
7. Στη συνέχεια, η συσκευή χρησιμοποιεί το κλειδί DAA, για να δημιουργήσει μια ψηφιακή DAA υπογραφή στην πολιτική περιορισμού κλειδιού, προκειμένου να παρέχει μια απόδειξη ότι χρησιμοποιήθηκε η σωστή πολιτική περιορισμού κλειδιού κατά την κρυπτογράφηση των δεδομένων.
8. Τα κρυπτογραφημένα δεδομένα, το HMAC αυτών, το δημόσιο ¨μυστικό¨, καθώς και η ψηφιακή DAA υπογραφή αποστέλλονται σε μια βάση δεδομένων, έτσι ώστε ένας αποκρυπτογράφος να μπορεί να τα αποκρυπτογραφήσει.

- Αποκρυπτογράφηση

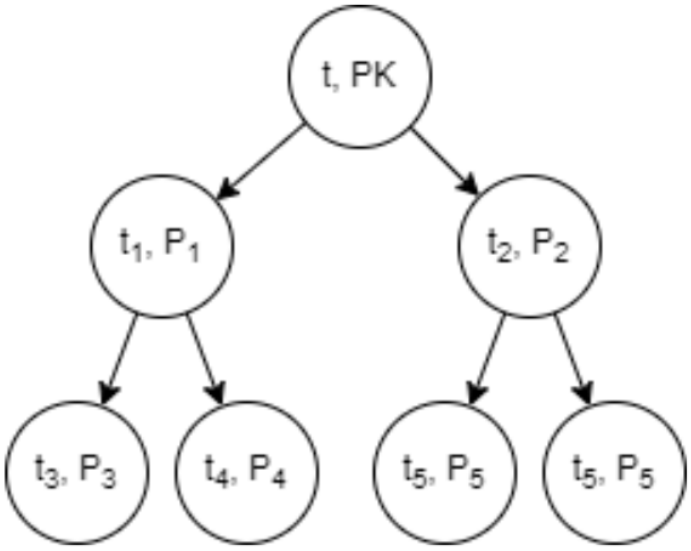
1. Όταν μια συσκευή Αποκρυπτογράφος επιθυμεί να πραγματοποιήσει αποκρυπτογράφηση σε ένα σύνολο δεδομένων που έχουν κρυπτογραφηθεί με χρήση του ABE, ανακτά πρώτα την υπογραφή DAA.
2. Χρησιμοποιεί την πολιτική που αναμενόταν να έχει υπογραφεί προκειμένου να επαληθεύσει την υπογραφή που παρέχεται από τον Κρυπτογράφο.
3. Εάν αυτός ο έλεγχος είναι επιτυχής, ο Αποκρυπτογράφος χρησιμοποιεί τα ιδιωτικά/μυστικά ελλειπτικά κλειδιά που αντιστοιχούν στα στατικά χαρακτηριστικά του για να υπολογίσει το seed με το οποίο ο κρυπτογράφος δημιούργησε τα συμμετρικά του κλειδιά.
4. Με το seed, το οποίο εξήγαγε από το δημόσιο ¨μυστικό¨, δημιουργεί ένα HMAC κλειδί για να υπολογίσει το HMAC των κρυπτογραφημένων δεδομένων.
5. Χρησιμοποιεί αυτό το HMAC προκειμένου να πραγματοποιήσει έναν έλεγχο πιστοποίησης, συγκρίνοντάς το με αυτό που παρέχεται από τον Κρυπτογράφο.
6. Εάν ο έλεγχος πιστοποίησης είναι επιτυχής, η συσκευή υποβάλλεται σε έλεγχο ακεραιότητας, χρησιμοποιώντας τα στατικά και δυναμικά χαρακτηριστικά, ώστε να μπορέσει να χρησιμοποιήσει τα δικά της κλειδιά, ικανοποιώντας την πολιτική που της έχει εφαρμοστεί από τον AttrMH κατά τη διάρκεια της αρχικοποίησής της.
7. Μετά την επιτυχή ολοκλήρωση του ελέγχου ακεραιότητας, ο Αποκρυπτογράφος αποκρυπτογραφεί τα δεδομένα.

4.6 Υλοποιηση

4.6.1 Αρχικοποιηση

Στη συνέχεια, περιγράφουμε το μηχανισμό που εκκινείται από τον AttrMH κάθε φορά που μια συσκευή, είτε κρυπτογράφος είτε αποκρυπτογράφος, επιθυμεί να συμμετάσχει στο σχήμα ABE. Ο AttrMH, που ενεργεί ως μια εμπιστευτική αρχή, κατά την εγγραφή της συσκευής δημιουργεί τα αντίστοιχα στατικά κλειδιά χαρακτηριστικών για κάθε συσκευή,

καθώς και τη δημιουργία του δέντρου ελέγχου πρόσβασης. Ένα τέτοιο δέντρο ελέγχου πρόσβασης θα φαινόταν όπως αυτό που εμφανίζεται στο σχήμα.



Σχήμα 4.1: *Οπτικοποίηση ενός δέντρου ελέγχου πρόσβασης*

Λαμβάνοντας υπόψη όλα τα παραπάνω, οι ενέργειες που ακολουθούνται στα πλαίσια της φάσης αρχικοπιησης περιγράφονται ως εξής.

1. **AttrMH αρχικοποίηση**: Ένα κλειδί κατακερματισμού (HMAC) δημιουργείται ως παιδί του Endorsement Κλειδιού του TPM του AttrMH. Αυτό το κλειδί θα μεταφερθεί σε κάθε συσκευή που εγγράφεται με ασφάλεια στον AttrMH, για να χρησιμοποιηθεί ως γονέας του κλειδιού κρυπτογράφησης και του κλειδιού κατακερματισμού (HMAC) στον ιεραρχικό πίνακα κλειδιών του TPM της συσκευής.

2. Ο χώρος των χαρακτηριστικών στο σύστημα καθορίζεται ως το σύνολο των χαρακτηριστικών $U = 1, 2, ..., n$. Για κάθε χαρακτηριστικό $i \in U$, διαλέγεται ένας αριθμός t_i τυχαία από την ομάδα $\mathbb{Z}_q$. Το δημόσιο κλειδί κάθε χαρακτηριστικού i είναι $P_i = t_i \cdot G$, όπου G είναι ο γεννήτορας της ομάδας $\mathbb{Z}_q$ και είναι γνωστός σε όλες τις συσκευές.

3. Ο AttrMH διαλέγει τυχαία t από την ομάδα $\mathbb{Z}_q$, το οποίο θα είναι και η ρίζα σε κάθε δέντρο ελέγχου πρόσβασης, δρώντας ως master κλειδί για όλες τις συσκευές. Αντίστοιχα το δημόσιο master κλειδί PK είναι $PK = t \cdot G$. Οι δημόσιες μεταβλητές ορίζονται ως $Params = PK, P_1, ..., P_U, G$.

4. Ο AttrMH ορίζει το δέντρο ελέγχου πρόσβασης Γ δημιουργώντας ένα δημόσιο πολυώνυμο $q_u(x)$, τάξης $d_u - 1$,όπου για κάθε κόμβο u του δέντρου Γ ορίζεται, με λογική από τη ρίζα του δέντρου προς τα φύλλα του, $q_u(0) = q_{parent}(index(u))$. Τέλος η ρίζα κάθε δέντρου Γ, ορίζεται ως $q_R(0) = t$.

5. **Αρχικοποίηση Συσκευής**: Υποθετούμε ότι η συσκευή έχει ήδη δημιουργήσει ένα DAA κλειδί, ώστε να μπορέσει ο AttrMH να κρυπτογραφήσει όλες τις μυστικές μεταβλητές/κλειδιά που αφορούν την συγκεκριμένη συσκευή.

6. Η συσκευή στέλνει στον AttrMH το δημόσιο μέρος του Endorsement κλειδιού του TPM της καθώς και το όνομα του DAA κλειδιού. (Όνομα ενός κλειδιού ονομάζεται το hash του δημόσιου μέρους του κλειδιού)

7. Η συσκευη στελνει στον AttrMH το δημοσιο μερος του Endorsement κλειδιου του TPM της καθως και το ονομα του DAA κλειδιου.(Ονομα ενος κλειδιου ονομαζεται το hash του δημοσιου μερους του κλειδιου)

8. Ο AttrMH δημιουργεί ένα τυχαίο συνθηματικό, το οποίο χρησιμοποιείται για διπλή κρυπτογράφηση του μυστικού κλειδιού κατακερματισμού ως ένα επιπρόσθετο επίπεδο ασφαλείας για να παρέχει φρεσκάδα κάθε φορά που μια συσκευή προσπαθεί να λάβει μέρος στο σχήμα ABE. Ο AttrMH δημιουργεί ένα τυχαίο αριθμό με τον οποίο θα δημιουργήσει ένα συμμετρικό κλειδί, με το οποίο θα κρυπτογραφηθεί το μυστικό κλειδί κατακερματισμού. Έπειτα ο AttrMH κρυπτογραφεί τον τυχαίο αυτόν αριθμό με το σημόσιο Endorsement κλειδί του TPM της συσκευής που επιθυμεί να λάβει μέρος στο ABE. Τέλος, με το τυχαίο συνθηματικό που δημιουργήθηκε από τον AttrMH, κρυπτογραφείται το ηδη κρυπτογραφημένο μυστικο κλειδι κατακερματισμου.

9. Ο AttrMH κρυπτογραφεί τα κλειδιά που βασίζονται στα στατικά χαρακτηριστικά και το τυχαίο συνθηματικό που δημιουργήθηκε στο προηγούμενο βήμα για κάθε συσκευή που επιθυμεί να λάβει μέρος στο ABE με το Endorsement κλειδί του TPM της, και στη συνέχεια τα υπολογίζει ένα μήνυμα αυθεντικοποίησης πάνω σε αυτό με βάση το όνομα του DAA κλειδιού. Αυτή η διαδικασία πραγματοποιείται εκτελώντας την εντολή **TPM2_MakeCredential**.

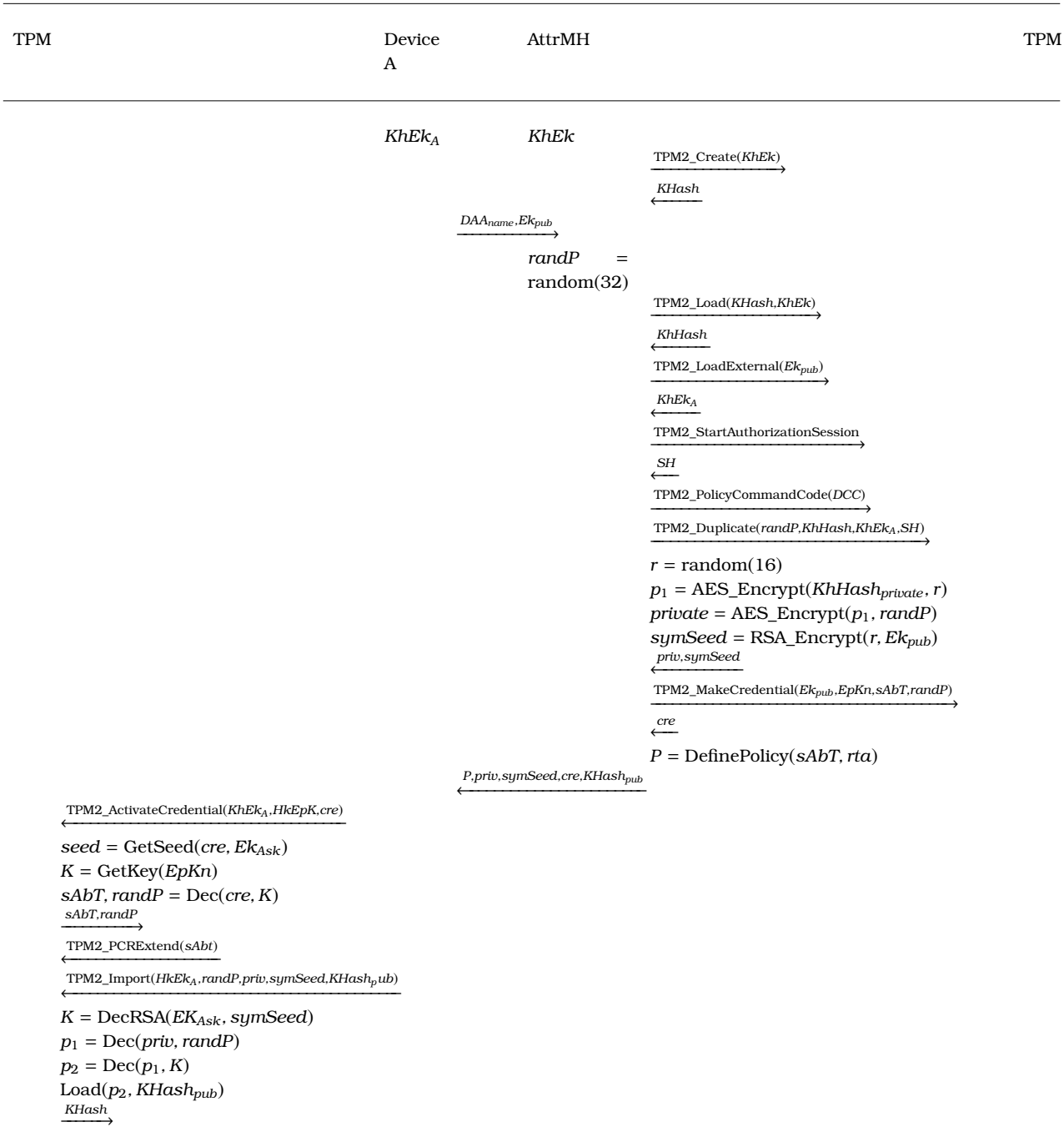
10. Ο AttrMH υπολογίζει μια πολιτική, η οποία αν εφαρμοστεί επιτυχημένα, τότε θα μπορούν χρησιμοποιηθούν τα συμμετρικά κλειδιά. Η πολιτική αυτή υπολογίζεται κατακερματίζοντας την σύνενωση όλων των στατικών και δυναμικών χαρακτηριστικών της εκάστοτε συσκευής, καθώς και τους κωδικούς κάθε εντολής που πρέπει να εκτελεστεί κατά την εκτέλεση του αλγορίθμου. Με τον τρόπο αυτό ελέγχεται αν η συσκευή εκτελεί σωστά τον αλγόριθμο.

11. Ο AttrMH στέλνει την πολιτική που υπολογίστηκε στο προηγούμενο βήμα, καθώς επίσης και το κρυπτογραφημένο δέντρο ελέγχου πρόσβασης. Μαζί με το κρυπτογραφημένο τυχαίο συνθηματικό και το διπλά κρυπτογραφημένο μυστικό κλειδί κατακερματισμού.

12. Η συσκευή αποκρυπτογραφεί το δέντρο ελέγχου πρόσβασης, το οποίο αποθηκεύεται στους Platform Configuration Registers (PCRs) του TPM της συσκευής, καθώς και το τυχαίο συνθηματικό που δημιούργησε ο AttrMH, εκτελώντας την εντολή TPM2_-ActivateCredential.

13. Η συσκευή, χρησιμοποιώντας το αποκρυπτογραφημένο τυχαίο συνθηματικό και το διπλά κρυπτογραφημένο κλειδί κατακερματισμού, εισάγει με ασφάλεια το κλειδί κατακερματισμού που διαμοιράστηκε ο AttrMH, ως παιδί του Endorsement κλειδιού του TPM της συσκευής, εκτελώντας την εντολή TPM2_Import. Το κλειδί αυτό εισάγεται σε

κάθε συσκευή για να εξουσιοδοτήσει τη δημιουργία ομοίων συμμετρικών κλειδιών σε διαφορετικά TPMs.



Σχήμα 4.2: *Διάγραμμα της φάσης της αρχικοποίησης περιλαμβάνοντας τις εντολές για το TPM 2.0.*

4.6.2 Κρυπτογραφηση

Προκειμένου να περιγράψουμε τα βήματα που πρέπει να ακολουθηθούν για την κρυπτογράφηση ενός μηνύματος, θα υποθέσουμε ότι μια συσκευή Κρυπτογράφησης (αναφερόμενη

ως Συσκευή Α) επιθυμεί να κρυπτογραφήσει ένα μήνυμα κάνοντας χρήση του ABE. Τα βήματα που πρέπει να ολοκληρώσει μια συσκευή Κρυπτογράφησης δίνονται στο αντίστοιχο σχήμα και περιγράφονται ως εξής:

1. Η Συσκευή Κρυπτογράφησης Α πρέπει να δημιουργήσει έναν τυχαίο μυστικό k που παράγεται από το TPM εκτελώντας την εντολή TPM2_GetRandom και να υπολογίσει τον πολλαπλασιασμό $k \times PK$.

2. Η Συσκευή Α λαμβάνει τη μυστική τιμή $k \times PK$, η οποία θα χρησιμοποιηθεί μαζί με μυστικό κλειδί κατακερματισμού που διαμοιράστηκε από τον AttrMH κατά την αρχικοποίηση, για να δημιουργηθεί ένα συμμετρικό κλειδί κρυπτογράφησης που είναι συνδεδεμένο με την πολιτική που ορίστηκε από τον AttrMH και ένα κλειδί κατακερματισμού. Αυτό επιτυγχάνεται με την εκτέλεση της εντολής TPM2_CreateLoaded.

3. Η Συσκευή Α εκτελεί TPM2_PolicyPCR για να φορτώσει τα στατικά χαρακτηριστικά τα οποία είναι αποθηκευμένα στα PCRs του TPM της Συσκευής Α σε μια ¨συνεδρία¨. Στη συνέχεια εκτελείται TPM2_PolicyOR με όρισμα όλα τα δυναμικά χαρακτηριστικά της Συσκευής Α προκειμένου να ελεγχθεί αν η κατάστασή της κατά την εκτέλεση του αλγορίθμου είναι η αναμενόμενη, αυτή δηλαδή που ορίστηκε ο AttrMH κατά την αρχικοποίηση.

4. Η Συσκευή Α, με το προσφάτως δημιουργημένο Κλειδί Κρυπτογράφησης, θα κρυπτογραφήσει τα δεδομένα εκτελώντας την εντολή TPM2_EncryptDecrypt2.

5. Η Συσκευή Α χρησιμοποιεί το κλειδί DAA για να υπολογίσει μια DAA ψηφιακή υπογραφή πάνω στην πολιτική που εφαρμόστηκε κατά τη διάρκεια της συγκεκριμένης κρυπτογράφησης, ως απόδειξη σωστής εφαρμογής μιας πολιτικής. Αυτή η διαδικασία τίθεται δυνατή εκτελώντας την εντολή TPM2_GetSessionAuditDigest.

6. Με το προσφάτως δημιουργημένο κλειδί κατακερματισμού (HMAC), η Συσκευή Α εκτελεί το TPM2_HMAC για να υπολογίσει τον κατακερματισμό των κρυπτογραφημένων δεδομένων.

7. Η Συσκευή Α επιλέγει ένα μονοπάτι από το δέντρο ελέγχου προσβασης και χρησιμοποιεί τα αντίστοιχα κλειδιά για να κρύψει το μυστικό με το οποίο δημιουργήθηκαν τα συμμετρικά κλειδιά, $C_i = k * P_i$ για $i \in \omega$, όπου $\omega \subset U$ είναι το σύνολο που περιέχει τα μυστικά κλειδιά για κάθε κόμβο του επιλεγμένου μονοπατιού.

8. Τέλος, η Συσκευή Α δημοσιεύει τα κρυπτογραφημένα δεδομένα, τον κατακερματισμό τους καθώς και την τιμή C_i, το επιλεγμένο μονοπάτι και την DAA ψηφιακή υπογραφή.

4.6.3 Αποκρυπτογράφηση

Κάθε συσκευή Αποκρυπτογράφησης (εφεξής αναφερόμενη ως Συσκευή Β) με αντίστοιχα χαρακτηριστικά μπορεί να ολοκληρώσει με επιτυχία την αποκρυπτογράφηση των δεδομένων που δημοσιεύτηκαν από τη συσκευή Α. Τα βήματα που ακολουθούνται για την εκτέλεση μιας

Σχήμα 4.3: *Κρυπτογραφηση με χρηση ενος TPM2.0*

διαδικασίας αποκρυπτογράφησης απεικονίζονται στο αντίστοιχο σχήμα και περιγράφονται ως εξής:

1. Η Συσκευή Β, η οποία διαθέτει τη σωστή πολιτική που έπρεπε να χρησιμοποιηθεί κατά την κρυπτογράφηση των δεδομένων, επαληθεύει την DAA ψηφιακή υπογραφή που παρείχε η Συσκευή Α. Εάν η επαλήθευση της υπογραφής είναι επιτυχής, αποδεικνύεται ότι η Συσκευή Α κρυπτογράφησε τα δεδομένα υπό τη σωστή πολιτική, επομένως η

Συσκευή Β μπορεί να συνεχίσει τον αλγόριθμο αποκρυπτογράφησης.

2. Για κάθε C_i των κόμβων u του Γ, η Συσκευή Β υπολογίζει τον αντίστοιχο κόμβο αποκρυπτογράφησης ως D: $D_u = q_u(0)/t_i$. Εδώ, $i = attr(u)$ και t_i είναι το κλειδί που έχει ανατεθεί στο συγκεκριμένο χαρακτηριστικό, απ'την ομάδα $\mathbb{Z}_q$, στη φάση της αρχικοποίησης και τέλος t_i^{-1} είναι το αντίστροφο κλειδί του t_i στην ομάδα $\mathbb{Z}_q$.

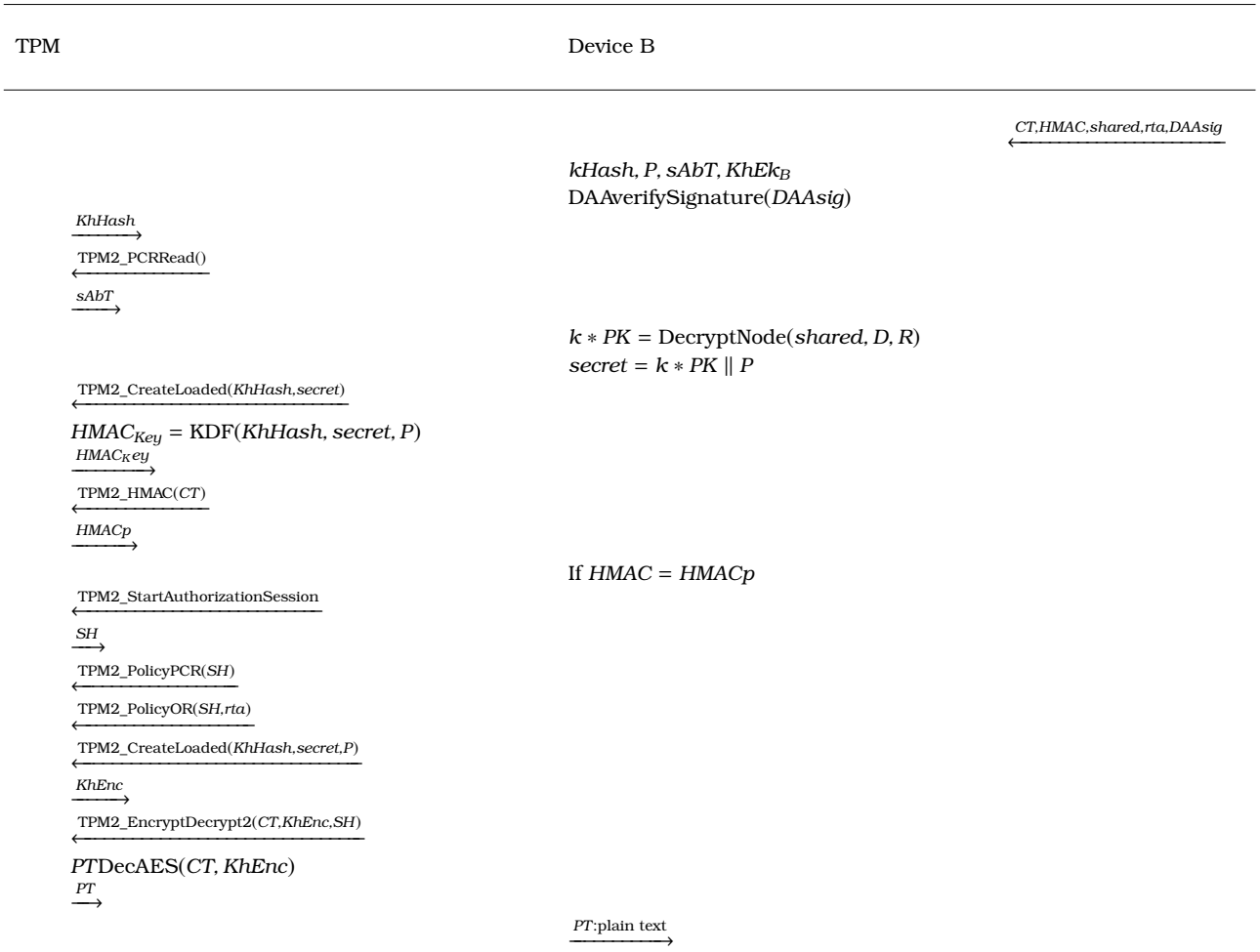
3. Η Συσκευή Β εκτελεί τον αλγόριθμο $DecryptNode(C_i, D, u)$ για κάθε κόμβο u που ανήκει στο μονοπάτι του δέντρου ελέγχου πρόσβασης. Για έναν κόμβο u, όπου το κλειδί του χαρακτηριστικού είναι καλά ορισμένο. Έστω λοιπόν $i = attr(u)$ και για αυτόν τον κόμβο το κλειδί υπολογίζεται ως $DecryptNode(C_i, D, u) = D_u * C_i = q_u(0) \cdot t_i^{-1} \cdot k \cdot P_i$. Πρέπει να σημειωθεί εδώ ότι το αποτέλεσμα της $DecryptNode(C_i, D, u)$ είναι ένα σημείο ελλειπτικής καμπύλης της ομάδας που ορίζεται από το $\mathbb{Z}_q$ ή $\perp$. Αντίστοιχα ¨ανεβαίνοντας¨ το δέντρο ελέγχου πρόσβασης για να φτάσουμε στη ρίζα του δέντρου, υπολογίζεται το $DecryptNode(C_i, D, R) = q_R(0) \cdot k \cdot G = t \cdot k \cdot G = k \cdot PK$.

4. Η Συσκευή Β με το μυστικό που ανέκτησε εκτελώντας το βήμα τρία *kPK*, εκτελεί TPM2_CreateLoaded χρησιμοποιώντας το μυστικό κλειδί κατακερματισμού που πήρε από τον AttrMH όταν εγγράφηκε σε αυτό το σχήμα ABE και το *kPK*, ώστε να δημιουργήσει τα δύο συμμετρικά κλειδιά αποκρυπτογράφησης και κατακερματισμού.

5. Η Συσκευή Β, χρησιμοποιώντας το κλειδί κατακερματισμού HMAC, υπολογίζει τον κατακερματισμό των κρυπτογραφημένων δεδομένων και τον συγκρίνει με αυτόν που παρέχεται από την Συσκευή Κρυπτογράφησης.

6. Αν ο μόλις υπολογισμένος κατακερματισμός είναι ίδιος με αυτόν που υπολόγισε ο κρυπτογράφος, η Συσκευή Β μπορεί να συνεχίσει την αποκρυπτογράφηση. Έτσι η Συσκευή Β εκτελεί TPM2_PolicyPCR ώστε να φορτώσει σε μια νέα συνεδρία με το TPM τα στατικά χαρακτηριστικά της συσκευής που είναι αποθηκευμένα στους Platform Configuration Registers (PCRs) του TPM και στη συνέχεια εκτελεί TPM2_PolicyOR για όλα τα δυναμικά χαρακτηριστικά της. Αν το αποτέλεσμα αυτών των εντολών είναι η πολιτική που έχει οριστεί κατά τη φάση της αρχικοποίησης, η Συσκευή Β είναι σε σωστή κατάσταση και μπορεί να συνεχίσει την εκτέλεση του αλγορίθμου.

7. Αν λοιπόν έχει πετύχει και το παραπάνω βήμα, με το πρόσφατα δημιουργημένο συμμετρικό κλειδί κρυπτογράφησης η Συσκευή Β εκτελεί TPM2_EncryptDecrypt2 και αποκρυπτογραφεί τα κρυπτογραφημένα δεδομένα.



Σχήμα 4.4: *Αποκρυπτογράφηση ενός κρυπτογραφημένου μηνύματος χρησιμοποιώντας TPM2.0*

Κεφάλαιο 5

Αναλυση Ασφαλειας Πρωτοκολλου

Στο κεφαλαιο αυτο γινεται η αναλυση ασφαλειας του πρωτοκολλου.

5.1 Μοντελο Ασφαλειας

Για την ανάλυση του πρωτοκόλλου ABE επιλέχθηκε το μοντέλο ασφαλείας Selective-Set, μιας και είναι το πιο διαδεδομένο μοντέλο που υιοθετείται για την ανάλυση σχημάτων Attribute-Based Encryption. Στο συγκεκριμένο μοντέλο, δύο μηνύματα που κρυπτογραφούνται από ένα KP-ABE (Key Policy Attribute-Based Encryption) είναι μη διακριτά σε επιθέσεις γνωστού επιλεγμένου κειμένου και επιλεγμένου συνόλου χαρακτηριστικών, η απόδειξη ασφαλείας θα βασιστεί στο πρόβλημα του διακριτού λογαρίθμου [46], [45], [44].

5.2 Μοντελο

Η έννοια της ασφάλειας ενος κρυπτογραφημενου μυνηματος οσον αφορα την μη διακριτοτητα του κατω απο επιθεση γνωστου επιλογμενου κειμένου στο μοντελο Selective-Set οριζεται ως το παιχνίδι που διεξάγεται μεταξύ ενός διεκδικητη $\mathcal{B}$ και ενος αντιπαλου $\mathcal{A}$.Το παιχνιδι αυτο οριζεται ως:

- Προετοιμασια: Ο αντιπαλος $\mathcal{A}$ δηλωνει ενα συνολο ιδιωτητων γ, στο οποιο θελει να επιτεθει.
- Αρχικοποιηση: Ο διεκδικητης $\mathcal{B}$ εκτελει τον αλγοριθμο αρχικοποιησης του ABE πρωτοκολλου και στελενει τις παραμετρους των δημοσιων κλειδιων που αντιστοιχουν στις ιδιοτητες στον Αντιπαλο $\mathcal{A}$.
- Φαση 1: Ο αντιπαλος $\mathcal{A}$ επιτρεπεται να κανει πολλες ερωτησεις οσον αφορα τα κλειδια αποκρυπτογραφησης για πολλες διαφορετικες δομες δεδομενων, $\mathbb{A}_j$, οπου $\mathbb{A}_j(\gamma) = 0$, για καθε j.
- Προκληση: Ο αντιπαλος $\mathcal{A}$ υποβαλει στον $\mathcal{B}$ δυο διφορετικα μυνηματα ιδιου μηκους M_0 και M_1. Ο $\mathcal{B}$ με τυχαιοτητα που καθοριζει το πεταγμα ενος νομισματος υ, κρυπτογραφει το M_υ υπο το συνολο ιδιωτητων γ που οριστηκε παραπανω. Το κρυπτογραφημενο αυτο μυνημα στην συνεχεια στελνεται στον αντιπαλο $\mathcal{A}$.

- Φαση 2: Επαναλαμβανεται η φαση 1.
- Υποθεση: Ο αντιπαλος $\mathcal{A}$ ειναι σε θεση να υπολογιση μια υποθεση v' απο το v. Στο συγκεκριμενο παιχνιδι, το πλεονεκτημα του αντιπαλου οριζεται ως

$$\varepsilon = Pr[v' = v] - 1/2$$

Ενα πρωτοκολλο KP-ABE (Key Policy Attribute-Based Encryption) χαρακτηριζεται ασφαλες απο την μη διακριτοτητα του κατω απο επιθεση γνωστου επιλογμενου κειμένου IND-CPA, στο μοντελο του Selective-Set, οταν ο αντιπαλος μπορει να κερδισει το παιχνιδι σε πολυωνιμικο χρονο με το πολυ αμελητεο πλεονεκτημα.

Πρεπει να σημειωθει εδω στον ασθενεστερο ορισμο ασφαλεις OWE-CPA (One-Way-Encryption Chosen Plaintext Attack στο Selective Set, ο Αντιπαλος $\mathcal{A}$ λαμβανει ενα κρυπτογραφημενο μυνημα CM απο την φαση της προκλησης, οπως οριστηκε στο παιχνιδι παραπανω. Ο Αντιπαλος $\mathcal{A}$ λοιπον, κερδιζει αν καταφερει να ανακτησει το σωστο μυνημα απο το οποιο προεκυψε το συγκεκριμενο κρυπτοκειμενο. Η νικη αποφασιζεται στην φαση της υποθεσης του παραπανω παιχνιδιου και οριζεται ως $M = Decrypt(CM)$.

5.3 Αποδειξη Ασφαλειας

Αφού οριστηκε το μοντέλο ασφάλειας του πρωτοκόλλου ABE, μπορούμε τώρα να προχωρήσουμε στην απόδειξη ασφάλειας του προτεινόμενου σχήματος, εκμεταλλευόμενοι τις υποθέσεις του Διακριτού Λογαρίθμου (DL) και της Decisional Diffie–Hellman (DDH). Αυτό μπορεί να εκφραστεί ως εξής:

- Ορισμος 1: Η υποθεση του διακριτου λογαριθμου οριζει οτι δεδομενου ενος $y \in G$, πρεπει να βρεθει x τετοιο ωστε $g^x = y$. Αυτο το προβλημα θεωρειται οτι ειναι υπολογιστικα δυσκολο σε συγκεκριμενες ομαδες αριθμων.
- Ορισμος 2: Η υποθεση Decisional Diffie-Hellman (DDH) οριζει οτι δεδομενου ενος g^a και ενος g^b, οπου οι αριθμοι $a, b \in \mathbb{Z}_q$ επιλεχτηκαν τυχαια και ανεξαρτητα, ενας υπολογιστηκα περιορισμενος αντιπαλος $\mathcal{A}$ δεν μπορει να ξεχωρισει g^{ab} απο κανενα αλλο στοιχειο g^r το οποιο ανηκει στην ιδια ομαδα αριθμων.

Βασιζομενοι στις παραπανω υποθεσεις και ορισμενες εννοιες , η ασφάλεια του προτεινόμενου σχήματος ABE αποδεικνύεται στο μοντέλο Selective-Set. Όπως και στα υπάρχοντα σχήματα ABE, η μέθοδος της αντίφασης χρησιμοποιείται για να αποδειχθεί η ασφάλεια του σχήματος. Δεδομένου ότι η ασφάλεια του σχήματός μας εξαρτάται από τη δυσκολια του προβληματος Decisional Diffie-Hellman (DDH) πανω σε ελλειπτικες καμπυλες, , γίνεται μείωση της δυσκολίας της υπόθεσης ECDDH.

- Αποδειξη: Αρχικα ο $\mathcal{B}$ οριζει την ομαδα που οριζεται απο την μια ελλειπτικη καμπυλη G_E ταξεως q πανω απο πεπερασμενο πεδιο $\mathbb{F}_p$ και γεννητορα G. Στην συνεχεια ο $\mathcal{B}$ ριχνει ενα δικαιο δυαδικό νομισμα μ και τυχαια επιλεγει a, b, z απο το $\mathbb{Z}_q^*$. Αν $\mu = 0$, οριζεται $(A, B, Z) = (c\&G, d\&G, c\&d\&G)$· αλλιως, οριζεται $(A, B, Z) = (c\&G, d\&G, z\&G)$. Υποθετουμε οτι το συνολο των ιδιωτητων U ειναι ορισμενο.

- Προετοιμασια: Ο $\mathcal{B}$ λαμβανει το συνολο των ιδιωτητων γ, στο οποιο ο Αντιπαλος $\mathcal{A}$ θελει να επιτεθει. Υποθετουμε οτι το δεντρο που οριζεται απ'τα κλειδια που αντιστοιχουν στις ιδιοτητες για το συγκεκριμενο συνολο ιδιωτητων γ ειναι το Γ .
- Αρχικοποιηση: Ο $\mathcal{B}$ εκτελει το αλγοριθμο της προετοιμασιας ωστε να οριστουν οι παραμετροι των δημοσιων κλειδιων του συστηματος και τους στελνει στον Αντιπαλο $\mathcal{A}$.
  - $\mathcal{B}$ διαλεγει $KHash \leftarrow \mathbb{Z}_q$
  - $\mathcal{B}$ οριζει τις παραμετρους του συστηματος $Y = A = c\&G$
  - $\mathcal{B}$ οριζει Y_i συμφωνα με τον παρακατω κανονα $\forall i \in U$
    1. ΑΝ $i \in \gamma$, $\mathcal{B}$ τυχαια διαλεγεται $r_i \leftarrow \mathbb{Z}_q^*$ και οριζει $Y_i = r_i\&G$ και $y_i = r_i$
    2. Αν $i \in U - \gamma$, $\mathcal{B}$ τυχαια διαλεγει $\beta_i \leftarrow \mathbb{Z}_q^*$, και οριζει $Y_i = \beta_i\&B = d\&\beta_i\&G$ και $y_i = d\&\beta_i$
  - $\mathcal{B}$ στελνει τις παραμετρους των δημοσιων κλειδιων του συστηματος $\{Y, Y_i, i \in U\}$ στον Αντιπαλο $\mathcal{A}$. Ειναι προφανες οτι τα Y, Y_i αντιστοιχουν στα $\{PK, P_i\}$ του προτεινομενου σχηματος ABE.

- Φαση 1: Ο αντίπαλος $\mathcal{A}$ μπορεί να κάνει πολλές ερωτήσεις για το κλειδί αποκρυπτογράφησης που αντιστοιχεί σε οποιαδήποτε δομή πρόσβασης Γ^*, όπου το σύνολο πρόκλησης γ δεν ικανοποιεί τη Γ^*, δηλαδή $\Gamma^*(\gamma) = 0$. Προκειμένου ο αντίπαλος $\mathcal{A}$ να μπορεί να ανακατασκευάσει το κλειδί αποκρυπτογράφησης για τη δομή πρόσβασης Γ^*, ο $\mathcal{B}$ πρεπει να ορισει ενα πολυωνυμο Q_u βαθμου d_u σε καθε κομβο u στο δεντρο που οριζεται απ'τα κλειδια των ιδιοτητων Γ^*. Το πολυωνυμο Q_u για καθε κομβο u στο Γ^* οριζεται ως $Q_u(x)$ και θα αναφερεται ως $q_u(x)$ στο προτεινομενο σχημα. Το κοινό μυστικό ορίζεται να είναι η σταθερά του πολυωνύμου της ρίζας, το οποιο ειναι $Q_R(0) = c$. Το μυστικο κλειδι που αντιστοιχει σε καθε κομβο x που ειναι φυλλο του δεντρου Γ^* δινεται απ'το πολυωνυμο του υπολογιζοντας:

  $D_x = Q_x(0)/r_i$=

  - $q_x(0)/r_i$, ΑΝ $(att(x) \in \gamma)$
  - $q_x(0)/\beta_i d$, ΑΝ $(att(x) \notin \gamma)$

  οπου $i = attr(x)$.

  Το συνολο $(D_x, x \in \Gamma^*)$ ειναι το μυστικο κλειδι για την δομη προσβασης Γ^* , το οποιο στελνεται στον Αντιπαλο $\mathcal{A}$.

- Προκληση: Ο Αντιπαλος $\mathcal{A}$ θα υποβαλει δυο μυνηματα στον $\mathcal{B}$. Ο $\mathcal{B}$ με χρηση ενος δικαιου δυαδικου νομισματος τ υπολογιζει το κρυπτοκειμενο M_τ. Για να κρυπτογραφησει το μυνημα M_τ, $\mathcal{B}$ διαλεγει τυχαια k απο $\mathbb{Z}_q^*$ ακι υπολογιζει τον συμμετρικο κλειδι κρυπτογραφησης και το κλειδι αυθεντικοποιησης κανοντας χρηση της συναρτησης $PRF(kY, KHash) = (K_x, K_y)$, αν K_x ή K_y ειναι 0, ξανα διαλεγουμε το k. Οριζεται SHARED $= kY$, $C_i = r_i B$, $i \in \gamma$. $\mathcal{B}$ υπολογιζει $C = ENC(M_\tau, K_x)$ και $MAC = HMAC(M_\tau, K_y)$, και μεταδιδει το κρυπτοκειμενο $(\gamma, C, MAC, C_i,\ i \in \gamma)$ το $\mathcal{A}$.

1. Αν $\tau = 0$, τοτε $Z = c\&d\&G$. Αν k ειναι το d, τοτε θα επρεπε να ειναι $\texttt{SHARED} = k\&Y = d\&c\&G = Z$, και $C_i = k\&Y_i = d\&Y_i = d\&r_i\&G = r_i\&B$, οπου, $i \in \gamma$.
2. Αν $\tau = 1$, τοτε $Z = z\&G$. Αν k ειναι το z, τοτε αποδεικνυεται οτι $\texttt{SHARED} = z\&G$.

Αφου c, d και z ειναι τυχαιοι αριθμοι, ο SHARED θα ειναι ενα τυχαιο στοιχειο της ομοαδας G_E για τον αντιπαλο $\mathcal{A}$ και δεν μπορει να εκμαιευση καποια ευαισθητη/ σημαντικη πληροφορια M_τ απο το κρυπτοκειμενο.

- Φαση 2: Και ο $\mathcal{B}$ και ο $\mathcal{A}$ συμπεριφερονται ακριβως οπως συμπεριφερθηκαν στην Φαση 1.
- Υποθεση: Ο Αντιπαλος $\mathcal{A}$ στελνει την υποθεση του τ' απο το τ στον $\mathcal{B}$.
  1. Αν $\tau' = \tau$, ο $\mathcal{B}$ γυρναει σαν αποτελεσμα $\mu' = 0$ για να δειξει οτι του δωθηκε μια εγκυρη τριπλετα ECDDH (A, B, Z).
  2. Αν $\tau' \neq \tau$, ο $\mathcal{B}$ γυρναει σαν αποτελεσμα $\mu' = 1$ για να δειξει οτι του δωθηκε μια τυχαια τριπλετα ECDDH (A, B, Z).

  Πρεπει να σημειωθει εδω οτι η πιθανοτητα του μ' αν ειναι ιδιο με το μ που ορίζεται στη φάση της αρχικοποίησης είναι η μιση, καθώς αυτή είναι μια τυχαία εκτίμηση από τον $\mathcal{B}$.

  $Pr[\tau \neq \tau'] = Pr[\tau = \tau'] = 1/2$ οταν $\mu = 1$ αφου αυτη ειναι μια τυχαια μαντεψια αν στον $\mathcal{B}$ δωθηκε μια τυχαια τριπλετα (A, B, Z).

  Στο ορισμενο ECDDH παιχνιδι,

$$Pr[\tau' = \tau] = 1/2Pr[\mu' = \mu = 0]+1/2Pr[\mu' = \mu = 1] = 1/2(1/2+\varepsilon)+(1/2)(1/2) = 1/2+\varepsilon/2$$

  Αποδεικνυεται λοιπον οτι το συνολικο πλεονεκτημα του $\mathcal{B}$ απεναντι στον Αντιπαλο $\mathcal{A}$ στο ECDDH παιχνιδι ειναι

$$(Pr[\tau' = \tau] - 1/2 = 1/2 + \varepsilon/2 - 1/2 = \varepsilon/2)$$

  Το οποιο ερχεται σε αντιπαραθεση με το γεγονος οτι το προβλημα ECDDH ειναι δυσκολο να λυθει για αμελητεα πιθανοτητα ε. Οληκληρωνοντας ετσι την αποδειξη.

Κεφάλαιο 6

Έλεγχος

Στο κεφάλαιο αυτό γίνεται ο έλεγχος καλής λειτουργίας και αξιολογησης του συστήματος. Στο πλαίσιο της υλοποίησης του κώδικα διευρευνήθηκαν τα εργαλεία ασφαλής επικοινωνίας με ενα [47], [48] και επιλέχθηκε αυτο της IBM.

6.1 Μεθοδολογία Ελέγχου

Το προτεινόμενο πρωτόκολλο Attribute Based Encryption λειτουργεί σε συσκευές που υιοθετούνται σε ενσωματωμένα συστήματα οι οποίες έχουν σχετικά χαμηλή υπολογιστική δύναμη όπως Raspberry PIs, Beaglebone, Arduino. Οι συσκευές αυτές επιλέγονται καθώς προσφέρουν τη δυνατότητα το υπολογιστικό σύστημα να χρησιμοποιήσει ένα TPM. Κάνοντας λοιπόν την υπόθεση ότι το πρωτόκολλο θα λειτουργεί σε περίπου ίδιου τύπου συσκευές, όσον αφορά την υπολογιστική τους δύναμη, η αξιολόγηση του επικεντρώνεται αλλού. Η αποτελεσματικότητα του πρωτοκόλλου επηρεάζεται κυρίως από τον αριθμό των χαρακτηριστικών με τα οποία κρύβεται ο τυχαίος αριθμός με τον οποίο δημιουργήθηκαν τα τυχαία συμμετρικά κλειδιά και φυσικά το μέγεθος των δεδομένων που πρέπει να κρυπτογραφηθούν. Λόγω περιορισμών από το TPM, μπορούμε να κρυπτογραφήσουμε μόνο 1KB ανά εκτέλεση του αλγορίθμου.

6.1.1 Ανάλυση Λειτουργιών

Το σχήμα ABE ο τρόπος λειτουργίας του οποίου περιγράφεται λεπτομερώς στο κεφάλαιο 4, απαιτεί την παρουσία τριών στοιχείων: (ι) Η Αξιόπιστη Αρχή (Trusted Third Party), η οποία είναι υπεύθυνη για τον καθορισμό του χώρου των χαρακτηριστικών και είναι σε θέση να δημιουργήσει τα αντίστοιχα κλειδιά των χαρακτηριστικών η οποία αναφέρεται στο πρωτόκολλο ως Attribute Master Holder, (ιι) Ο Κρυπτογράφος που είναι εξοπλισμένος με ένα TPM και κρυπτογραφεί δεδομένα και (ιιι) Ο Αποκρυπτογράφος, που είναι σε θέση να αξιοποιήσει το TPM του για να αποκρυπτογραφήσει τα δεδομένα που έχουν κρυπτογραφηθεί με ABE. Το πρωτόκολλο ABE λοιπόν μπορεί να διαχωριστεί σε τρεις κύριες φάσεις:

1. Φάση Αρχικοποίησης: Λαμβάνοντας υπόψη μια συσκευή που επιθυμεί να συμμετέχει στο σχήμα ABE, είτε ως Κρυπτογράφος είτε ως Αποκρυπτογράφος, σε αυτή τη φάση ορίζεται ο χώρος των χαρακτηριστικών και δημιουργούνται τα ελλειπτικά κλειδιά που

αντιστοιχούν στα χαρακτηριστικά αυτά και φυσικά τη δημιουργία των κλειδιών που αποθηκεύονται στην ιεραρχία κλειδιών του TPM.

2. Φάση Κρυπτογράφησης: Περιλαμβάνει τη δημιουργία του κλειδιού κρυπτογράφησης ABE υπό το κλειδί που παρέχεται από το Attribute Master Holder, που δεσμεύεται από την κατάλληλη πολιτική. Η πολιτικη αυτη περιλαμβανει έναν έλεγχο ακεραιότητας της συσκευής για να επαληθευθει η ορθοτητα των στατικών και δυναμικών της χαρακτηριστικών. Και τελος, την κρυπτογράφηση των δεδομένων.

3. Φάση Αποκρυπτογράφησης: Αναφέρεται στην αποκρυπτογράφηση των δεδομένων που έχουν κρυπτογραφηθεί με BE και περιλαμβάνει τον υπολογισμό του τυχαίου αριθμού με τον οποίο ο κρυπτογράφος δημιούργησε τα συμμετρικά κλειδιά του, τον υπολογισμό του κλειδιού αποκρυπτογράφησης, έναν έλεγχο ακεραιότητας της συσκευής και την πραγματική αποκρυπτογράφηση των δεδομένων.

Στον παρακάτω πίνακα παρέχονται οι εσωτερικές λειτουργίες του σχήματος ABE που θα αξιολογηθούν σε όλο αυτό το κεφάλαιο, καθώς και η ¨θέση¨ όπου εκτελούνται (η οποία μπορεί να είναι η συσκευή ως μέρος μιας απλής διεργασίας, η συσκευή TPM ή ο Attribute Master Holder). Παρέχονται επίσης λεπτομέρειες για κάθε λειτουργία, καθώς και το αναμενόμενο αντίκτυπο ή το πρόσθετο φορτίο που θα έχει κάθε λειτουργία στην απόδοση του πρωτοκόλλου συνολικά.

6.1.2 Καιρια σημεια για την αξιολογηση

Για να αξιολογήσουμε τον αντίκτυπο της χρήσης ενός TPM ως ασφαλούς στοιχείου σε μια υπάρχουσα υποδομή, μετριεται ο χρόνος εκτέλεσης κάθε εντολής TPM.Επιπλέον, εξετάζονται οι δυνατότητες του πρωτοκόλλου κρυπτογράφησης με βάση τα χαρακτηριστικά, αλλάζοντας το μέγεθος των επιθυμητών κρυπτογραφημένων δεδομένων και τον αριθμό των χαρακτηριστικών που μπορεί να έχει η συσκευή του. Παρουσιαζονται αναλυτικα στους πινακες 6.1, 6.2, 6.3 και 6.4.

6.1.3 Αξιολογηση ABE

Σχετικά με το πρωτόκολλο κρυπτογράφησης με βάση τα χαρακτηριστικά, διαφοροποιούμετε μεταξύ του αριθμού των χαρακτηριστικών και του μεγέθους των κρυπτογραφημένων δεδομένων. Στις πειραματικές μας δοκιμές, φτάσαμε μέχρι 8192 διαφορετικά χαρακτηριστικά ξεκινώντας από 32 χαρακτηριστικά. Ανάλογα με την περίπτωση χρήσης, κρυπτογραφήσαμε και αποκρυπτογραφήσαμε δεδομένα τάξης KB, MB GB. Επιπλέον, παρείχαμε μια βελτιωμένη λύση όπου χρησιμοποιούμε το κλειδί DAA ή ένα ψευδώνυμο για να υπογράψουμε την πολιτική περιορισμού χρήσης του κλειδιού ως επιπλέον επίπεδο ελεγξιμότητας κατά τη διάρκεια της συνεδρίας κρυπτογράφησης. Επιπλέον, όλα τα πειράματά μας διενεργήθηκαν με Software Based TPM το οποιο εχει σημιουργηθει απ' την IBM καθως και με Hardware Based TPM, συγκεκριμενα το SLB9673 FW 16.0 το οποιο ειναι και το πρωτο TPM που υποστηριζει συμμετρικη κρυπτογραφηση, στις πλατφόρμες RPI 3/4.

Λειτουργια	**HOST/TPM/AttrMH**	**Περιγραφη/Εκτιμωμενη Επιδραση**
Αναγνωση του δημοσιου Endorsement κλειδιου	TPM	Δεν προβλέπεται κανένα επιπλέον φορτίο ή αντίκτυπος στο πρωτόκολλο.
Δημιουργια Χαρακτηριστικων	HOST	Γραμμική λειτουργία με πολυπλοκότητα O(ν). Αναμένεται ελάχιστο ή μηδενικό αντίκτυπο, εξαρτάται από τον αριθμό των χαρακτηριστικών.
Duplication του HASH κλειδιου	AttrMH	Ο AttrMH θεωρητικα εχει μεγαλη υπολογιστικη δυναμη, έτσι η τριπλή κρυπτογράφηση που πραγματοποιείται δεν έχει κάποιο σημαντικό αντίκτυπο.
Δημιουργια των κλειδιων χαρακτηριστικων	AttrMH	Οπως αναφερθηκε και παραπανω ο AttrMH εχει μεγαλη υπολογιστικη δυναμη.Η πολυπλοκότητα αυτής της λειτουργίας είναι πάλι O(ν), δεν προβλέπουμε ένα σημαντικό αντίκτυπο στον αλγόριθμο, αλλά εξαρτάται από τον αριθμό των χαρακτηριστικών.
Ορισμος των πολιτικων χρησης των κλειδιων	AttrMH	Ο AttrMH συνενώνει τα χαρακτηριστικά για να τα κατακερματίσει και να δημιουργήσει μια αναπαράσταση του χώρου των χαρακτηριστικών. Χρησιμοποιούμε το SHA-256 το οποίο έχει πολυπλοκότητα Θ(ν). Εξαρτάται από τον αριθμό των χαρακτηριστικών.
Make Credential για καθε ενα TPM που θελει να συμμετεχει στο σχημα	AttrMH	Ο AttrMH χρησιμοποιεί το δημόσιο Ενδορσεμεντ κλειδί και το όνομα του DAA κλειδιού για να μετασχηματίσει το τυχαίο κλειδί κρυπτογράφησης που χρησιμοποιείται για να κρυπτογραφηθεί το HASH κλειδι. Λόγω του οτι ο AttrMH διαθέτει πολλούς υπολογιστικους πόρους, δεν αναμένεται να έχει κάποιο σημαντικό αντίκτυπο αυτή η λειτουργία.

Πίνακας 6.1: *Καιρια σημεια*

Λειτουργια	**HOST/TPM/AttrMH**	**Περιγραφη/Εκτιμωμενη Επιδραση**
Activate Credential	TPM	Εδώ ζητείται από το TPM να πραγματοποιήσει μια διπλή αποκρυπτογράφηση και αποτελεί μία από τις πιο βαριές λειτουργίες.
Import του HASH κλειδιου του AttrMH	TPM	Το TPM καλείται να τοποθετήσει το HASH κλειδι του AttrMH υπό τη δική του ιεραρχία κλειδιών. Αυτή είναι επίσης μία από τις πιο βαριές λειτουργίες, αλλά δεν προβλέπεται καμία αλλαγή ανάλογα με τον αριθμό των χαρακτηριστικών.
Αποθηκευση μιας απεικονισης των στατικων χαρακτηριστικων στα PCRs	TPM	Το TPM καλείται να αποθηκεύσει στα PCRs την αναπαράσταση κατακερματισμού των χαρακτηριστικών. Μια ελαφριά λειτουργία χωρίς εξάρτηση από τον αριθμό των χαρακτηριστικών.
Δημιουργια τυχαιου αριθμου	TPM	Το TPM καλείται να δημιουργήσει μια τυχαία τιμή που θα χρησιμοποιηθεί ως για να δημιουργηθουν τα συμμετρικά κλειδιά. Μια σχετικά ελαφριά λειτουργία, ανεξάρτητη από τον αριθμό των χαρακτηριστικών.
Δημιουργια συμμετρικου κλειδιου κρυπτογραφησης	TPM	Το TPM καλείται να παράγει ένα κλειδί AES-128 (16 bytes) υπό το HASH κλειδι. Μια σχετικά ελαφριά λειτουργία, ανεξάρτητη από τον αριθμό των χαρακτηριστικών.

Πίνακας 6.2: *Καιρια σημεια*

Λειτουργια	**HOST/TPM/AttrMH**	**Περιγραφη/Εκτιμωμενη Επιδραση**
Δημιουργια συμμετρικου κλειδιου HMAC	TPM	Το TPM καλείται να παράγει ένα κλειδί HMAC SHA-256 (32 bytes) υπό το HASH κλειδι. Μια σχετικά ελαφριά λειτουργία, ανεξάρτητη από τον αριθμό των χαρακτηριστικών.
Κρυπτογραφηση δεδομενων	TPM	Το TPM καλείται να χρησιμοποιήσει το συμμετρικό κλειδί κρυπτογράφησης για να κρυπτογραφήσει ένα τμήμα δεδομένων. Αυτή η λειτουργία εξαρτάται πλήρως από το μέγεθος των δεδομένων που πρέπει να κρυπτογραφηθούν.
Υπολογισμος κατακερματισμου των κρυπτογραφημενων δεδομενων	TPM	Το TPM καλείται να χρησιμοποιήσει το συμμετρικό κλειδί HMAC για να υπολογίσει το αυθεντικοποιητικό κατακερματισμό των κρυπτογραφημένων δεδομένων. Η υπολογιστική πολυπλοκότητα πρέπει να είναι η ίδια με αυτή της SHA-256. Εξαρτάται από το μέγεθος των κρυπτογραφημένων δεδομένων αλλά δεν πρέπει να έχει τόσο μεγάλο αντίκτυπο.
Υπολογισμος του δημοσιου μυστικου	HOST	Σε αυτήν τη λειτουργία, ο HOST χρησιμοποιεί τα κλειδιά των χαρακτηριστικών για να αποκαλύψει τον τυχαίο αριθμο προκειμένου να είναι σε θέση να δημιουργήσει τα ίδια συμμετρικά κλειδιά με τη συσκευή του κρυπτογράφου. Αυτή η λειτουργία έχει μια υπολογιστική πολυπλοκότητα της τάξης O(n) και εξαρτάται πλήρως από τον αριθμό των χαρακτηριστικών.

Πίνακας 6.3: *Καιρια σημεια*

Λειτουργια	**HOST/TPM/AttrMH**	**Περιγραφη/Εκτιμωμενη Επιδραση**
Αποκρυπτογραφηση δεδομενων	TPM	Το TPM καλείται να χρησιμοποιήσει το συμμετρικό κλειδί κρυπτογράφησης για να αποκρυπτογραφήσει ένα τμήμα δεδομένων. Αυτή η λειτουργία εξαρτάται πλήρως από το μέγεθος των δεδομένων που πρέπει να αποκρυπτογραφηθούν.

Πίνακας 6.4: *Καιρια σημεια*

Εντολη	**Μ.Ο.**	**Ελαχιστο**	**Μεγιστο**
START UP	0.001806477	0.001616844	0.002026291
PCR EXTEND	0.0004972938	0.000417826	0.000546921
CREATE PRIMARY	0.1471225642	0.079638235	0.304354623
CREATE PRIMARY	0.0094450972	0.009013945	0.009823077
FLUSH CONTEXT	0.000673492	0.000452252	0.001218767
ACTIVATE CREDENTIAL	0.0219752618	0.004698937	0.047928039
IMPORT	0.01820128	0.014817145	0.016367571
Συνολο	0.0677746842	0.065299151	0.071461789

Πίνακας 6.5: *Αξιολογηση Αρχικοποιησης με SW TPM*

Αξιολογηση ABE με SW Based TPM

Σε αυτη την υπο ενοτητα παρεχουμε τα πειραματικα αποτελεσματα που προεκυψαν με την χρηση του IBM Software Based TPM. Τα αποτελεσματα παρουσιαζονται στους πινακες 6.5, 6.6 και 6.7

Αξιολογηση ABE με HW Based TPM

Σε αυτή την υποενότητα παρουσιάζουμε τα πειραματικά αποτελέσματα που προέκυψαν με τη χρήση ενός Hardware Based TPM, model SLB9673. Τα αποτελεσματα παρουσιαζονται στους πινακες 6.8, 6.9, 6.11, 6.13, 6.10, 6.12, 6.14

6.1.4 Κριτικη των αποτελεσματων

Κατά την ανάλυση της αξιολόγησης του επιπέδου ελεγξιμότητας, παρατηρούμε ξεκάθαρα ότι η χρήση μόνο ψευδωνύμων για την υπογραφή των πολιτικών περιορισμού χρήσης των κλειδιών δεν επιφέρει σχεδόν μεγάλη επιβάρυνση σε σύγκριση με την έκδοση που δεν παρέχει καμία μορφή υπογραφής. Η επιπλέον απόδοση που επιφέρει αυτή η προσέγγιση μπορεί να είναι αποδεκτή ακόμα και για ενσωματωμένες συσκευές χαμηλής απόδοσης. Από την άλλη πλευρά, όταν χρησιμοποιείται το κλειδί DAA για να πραγματοποιηθούν ανώνυμες υπογραφές των πολιτικών περιορισμού χρήσης των κλειδιών, η εκτέλεση του πρωτοκόλλου για μόλις 1 kB δεδομένων περίπου διπλασιάζεται. Για ακόμα και συσκευές υψηλής απόδοσης όπως ένα

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
LOAD	0.0009230926667	0.000845326	0.001074051
GET RANDOM	0.01903196433	0.00030335	0.044239122
PCR READ	0.0006313993333	0.000348294	0.001057939
CREATE LOADED	0.019700908	0.000530126	0.045169725
START POLICY AUTHORIZE	0.02368135822	0.000967327	0.042588566
POLICY PCR	0.0006754995556	0.000609255	0.000779013
POLICY OR	0.0005865598889	0.000485665	0.000654624
ENCRYPT	0.08884107646	0.087919944	0.091833439
FLUSH CONTEXT	0.00349614917	0.002622204	0.005791849
CREATE	0.09180251848	0.087675965	0.092345573
LOAD	0.02324042261	0.022435828	0.026651316
SIGN	0.07149048336	0.070680185	0.07568829
FLUSH CONTEXT	0.00354593473	0.003116219	0.006353845
FLUSH CONTEXT	0.00374777247	0.002662666	0.004601228
CREATE LOADED	0.02592730014	0.024651796	0.030766548
HMAC	0.01803300808	0.016208222	0.02026925
FLUSH CONTEXT	0.00357389292	0.002695426	0.005772479
FLUSH CONTEXT	0.00392067652	0.002960443	0.006724028
Κρυπτογραφηση 1KB	0.44589795382	0.441093494	0.452517415

Πίνακας 6.6: *Αξιολογηση Κρυπτογραφησης με SW TPM*

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
LOAD External	0.06912259656	0.068294565	0.071962119
VERIFY SIGNATURE	0.08799303295	0.087483809	0.088093295
FLUSH CONTEXT	0.00389590536	0.003007108	0.006602936
PCR READ	0.00482037692	0.0037104	0.006050366
LOAD	0.02371775912	0.022314367	0.02779681
CREATE LOADED	0.02557767668	0.024941624	0.028230862
HMAC	0.01802036485	0.016667311	0.018728946
FLUSH CONTEXT	0.00326377912	0.00273226	0.006051069
CREATE LOADED	0.02405196354	0.023153709	0.027134257
START AUTHORIZATION SESSION	0.00491497729	0.003906528	0.017907562
POLICY PCR	0.00519736359	0.004580524	0.007287728
POLICY PCR	0.00519736359	0.004580524	0.007287728
POLICY OR	0.00448814842	0.003518865	0.006962286
DECRYPT	0.08519258747	0.042579312	0.095610274
FLUSH CONTEXT	0.00402843378	0.002826684	0.005926052
FLUSH CONTEXT	0.00342636117	0.00279387	0.006819694
FLUSH CONTEXT	0.00357389292	0.002695426	0.005772479
FLUSH CONTEXT	0.00332857582	0.00258626	0.005278964
Αποκρυπτογραφηση 1KB	0.25911463217	0.233931767	0.269651723

Πίνακας 6.7: *Αξιολογηση Αποκρυπτογραφησης με SW TPM*

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
START UP	0.054478722	0.053818009	0.054796667
PCR EXTEND	0.020487969	0.020536014	0.022402312
CREATE PRIMARY	15.460215689	7.647699493	27.267419680
CREATE PRIMARY	0.139640442	0.139472453	0.139742327
FLUSH CONTEXT	0.168251138	0.167923780	0.169285000
ACTIVATE CREDENTIAL	0.020483542	0.020479384	0.020509740
IMPORT	0.219318005	0.219017455	0.220773822
Συνολο	0.411866918	0.411793550	0.412198287

Πίνακας 6.8: *Αξιολογηση Αρχικοποιησης με HW TPM*

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
LOAD	0.064231741625	0.063589357	0.068186607
GET RANDOM	0.018575923875	0.017732887	0.019184045
PCR READ	0.020374650	0.020250576	0.021133811
CREATE LOADED	0.05628028487	0.056100792	0.05724012
START AUTHORIZATION SESSION	0.02307832150	0.022964225	0.023254002
POLICY PCR	0.0171784285	0.017062597	0.017204836
POLICY OR	0.021594027625	0.021590326	0.021598215
ENCRYPT	0.22280223275	0.222697161	0.223190863
FLUSH CONTEXT	0.01338006587	0.013279842	0.01392756
FLUSH CONTEXT	0.01316756937	0.012953547	0.013224953
CREATE LOADED	0.04949998975	0.049381113	0.049623889
HMAC	0.1253899745	0.125342135	0.125439133
FLUSH CONTEXT	0.01358314537	0.013233101	0.015892751
FLUSH CONTEXT	0.01320502275	0.012967622	0.013279656
Κρυπτογραφηση 1KB	0.69625747350	0.693888425	0.703841488

Πίνακας 6.9: *Αξιολογηση Κρυπτογραφησης με ελεγξιμοτητα 0 με HW TPM*

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
PCR READ	0.02033982725	0.020309834	0.020382093
LOAD	0.06355289162	0.063504431	0.063682392
CREATE LOADED	0.049490205	0.04947591	0.049506871
HMAC	0.12539417337	0.125376057	0.125463279
FLUSH CONTEXT	0.01328024025	0.013266583	0.013305602
CREATE LOADED	0.05608198274	0.056041737	0.056117347
START AUTHORIZATION SESSION	0.02386526537	0.022957243	0.029403241
POLICY PCR	0.01713156737	0.016929485	0.017220262
POLICY OR	0.02159571962	0.021589178	0.021617511
DECRYPT	0.23088128125	0.230736582	0.23098672
FLUSH CONTEXT	0.01323415925	0.012996455	0.01332136
FLUSH CONTEXT	0.01320134712	0.013009529	0.013243231
FLUSH CONTEXT	0.01323262912	0.013213361	0.013262545
Αποκρυπτογραφηση 1KB	0.68519183512	0.68261128	0.692449547

Πίνακας 6.10: *Αξιολογηση Αποκρυπτογραφησης με HW TPM*

Εντολη	**Μ.Ο.**	**Ελαχιστο**	**Μεγιστο**
LOAD	0.06385716754	0.063630174	0.065127326
GET RANDOM	0.01851370962	0.017528159	0.019271692
PCR READ	0.02034716404	0.020318888	0.020768029
CREATE LOADED	0.05648617920	0.056306512	0.059989133
START AUTHORIZATION SESSION	0.02330704956	0.022862191	0.029494342
POLICY PCR	0.01723798784	0.016959437	0.017799446
POLICY OR	0.02171233148	0.021673081	0.022531624
ENCRYPT	0.22413580900	0.223756927	0.225781643
FLUSH CONTEXT	0.01333127924	0.013305154	0.013470898
CREATE	0.17903034042	0.178707213	0.18024078
LOAD	0.06128426984	0.061093457	0.065473111
SIGN	0.06589900328	0.065448463	0.06697195
FLUSH CONTEXT	0.01332395392	0.013297449	0.013723916
FLUSH CONTEXT	0.01323126844	0.013175485	0.013302909
CREATE LOADED	0.04967804194	0.049558522	0.050221507
HMAC	0.12613483248	0.125882179	0.129469907
FLUSH CONTEXT	0.01330663478	0.013261678	0.013949348
FLUSH CONTEXT	0.01328120016	0.013018016	0.013946181
Κρυπτογραφηση 1KB	1.01790200378	1.015456676	1.026116725

Πίνακας 6.11: *Αξιολογηση Κρυπτογραφησης με ελεγξιμοτητα 1 με HW TPM*

Εντολη	**Μ.Ο.**	**Ελαχιστο**	**Μεγιστο**
LOAD External	0.07232978435	0.071181336	0.076818888
VERIFY SIGNATURE	0.055401559	0.054963402	0.056387911
FLUSH CONTEXT	0.01344804601	0.013186225	0.015444494
PCR READ	0.02042695407	0.02014471	0.021859783
LOAD	0.06378582236	0.063640632	0.064767284
CREATE LOADED	0.04977072752	0.049368525	0.051974658
HMAC	0.12635904948	0.12593152	0.131074576
FLUSH CONTEXT	0.0133762779	0.013283569	0.016729035
CREATE LOADED	0.05633319596	0.056249652	0.057236349
START AUTHORIZATION SESSION	0.02318112128	0.022893894	0.023747929
POLICY PCR	0.01722901668	0.016954807	0.018027908
POLICY	0.02172203664	0.021666352	0.023028139
DECRYPT	0.23252885728	0.231925864	0.236820959
FLUSH CONTEXT	0.01334086056	0.013092576	0.014396222
FLUSH CONTEXT	0.01330732412	0.01297149	0.01445803
FLUSH CONTEXT	0.01330800862	0.013188823	0.014361962
Αποκρυπτογραφηση 1KB	0.77311945612	0.769498729	0.781709385

Πίνακας 6.12: *Αξιολογηση Αποκρυπτογραφησης με ελεγξημοτητα 1 με HW TPM*

Εντολη	**Μ.Ο.**	**Ελαχιστο**	**Μεγιστο**
LOAD	0.0636730641	0.063572902	0.064057158
GET RANDOM	0.01792732250	0.0177252	0.019179634
PCR READ	0.0202833979	0.020309257	0.020256443
CREATE LOADED	0.05618761079	0.056096266	0.056388337
START AUTHORIZATION SESSION	0.02393339609	0.023038517	0.030299173
POLICY PCR	0.01718019559	0.017054074	0.01723661
POLICY OR	0.02164068170	0.021585415	0.021945932
ENCRYPT	0.23146202570	0.230919426	0.233942239
FLUSH CONTEXT	0.01336657429	0.013204729	0.013505116
FLUSH CONTEXT	0.01322774369	0.013204951	0.013321932
READ PUBLIC	0.03977967289	0.039584298	0.040050369
START AUTHORIZATION SESSION	0.023335765	0.023132424	0.024545785
POLICY CC	0.01499886	0.014916218	0.015116902
POLICY OR	0.02163143159	0.021605915	0.021736692
LOAD EXTERNAL	0.0713436718	0.070989093	0.07260801
VERIFY SIGNATURE	0.0559790474	0.054959032	0.056472281
POLICY AUTHORIZE	0.0343608286	0.034298629	0.034479905
COMMIT	0.09511273569	0.09458183	0.096286837
HASH	0.0340403104	0.033896631	0.034131686
START AUTHORIZATION SESSION	0.023197664	0.023179664	0.024196972
POLICY PCR	0.017144963	0.017054074	0.017218629
POLICY OR	0.021620304	0.021588156	0.021721581
POLICY AUTHORIZE	0.034362962	0.034307888	0.034444280
SIGN	0.03977967289	0.039584298	0.040050369
CREATE LOADED	0.0495503802	0.049485105	0.049609234
HMAC	0.1254128258	0.125374204	0.125481852
FLUSH CONTEXT	0.0135696767	0.013247359	0.01634669
FLUSH CONTEXT	0.0132746064	0.013233192	0.013473801
Κρυπτογραφηση 1KB	1.2905215482	1.28731308	1.2950982

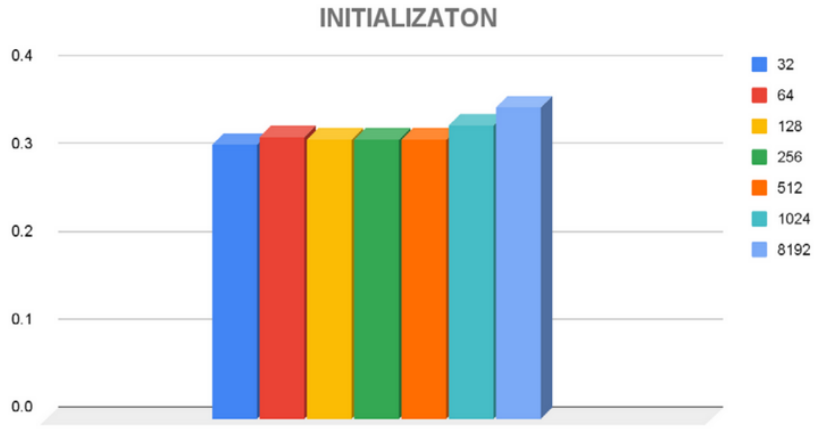
Πίνακας 6.13: *Αξιολογηση Κρυπτογραφησης με ελεγξιμοτητα 2 με HW TPM*

Εντολη	Μ.Ο.	Ελαχιστο	Μεγιστο
PCR READ	0.0204087266	0.020304664	0.02106579
LOAD	0.0643255693	0.063356274	0.06651016
CREATE LOADED	0.0496615254	0.049463086	0.050125544
HMAC	0.12555463079	0.125307352	0.126197124
FLUSH CONTEXT	0.0133759558	0.013265599	0.013982075
CREATE LOADED	0.0562400124	0.056057728	0.057545386
START AUTHORIZATION SESSION	0.0235449304	0.022816519	0.024539396
POLICY PCR	0.0173413763	0.017041223	0.018073734
POLICY OR	0.02179275650	0.021586545	0.022771426
DECRYPT	0.23136355810	0.230865631	0.233620649
FLUSH CONTEXT	0.0133208412	0.013253302	0.013386525
FLUSH CONTEXT	0.01325696009	0.013235452	0.013299117
FLUSH CONTEXT	0.01323760469	0.013219636	0.013255692
Αποκρυπτογραφηση 1KB	0.68561567760	0.682935934	0.689144231

Πίνακας 6.14: *Αξιολογηση Αποκρυπτογραφησης με ελεγξημοτητα 2 W TPM*

Raspberry Pi, αυτή η προσέγγιση δεν είναι βιώσιμη όταν τα δεδομένα που πρόκειται να κρυπτογραφηθούν είναι μεγαλύτερα από μερικά kB.

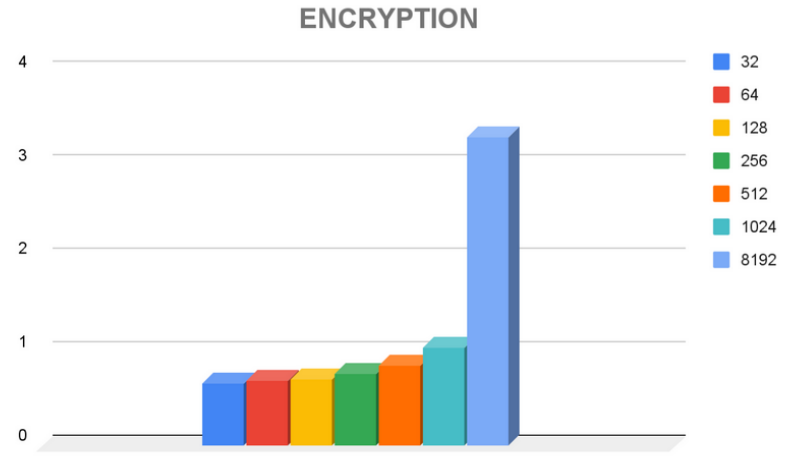
Όπως αναφέρθηκε στις προηγούμενες υποενότητες, εκτός από το επίπεδο ελεγξιμότητας, διεξήχθησαν πειράματα όσον αφορά το μέγεθος των δεδομένων που πρόκειται να κρυπτογραφηθούν/αποκρυπτογραφηθούν καθώς και τον αριθμό των χαρακτηριστικών που συνέβαλαν στην κρυπτογράφηση και αποκρυπτογράφηση. Όπως φαίνεται στο σχήμα 6.1 Ανεξάρτητα από τον αριθμό των χαρακτηριστικών που χρησιμοποιούμε, σε πραγματικά σενάρια χρήσης, στη φάση αρχικοποίησης του πρωτοκόλλου σχεδόν πάντα παρατηρούμε κοντά στο μηδέν απόκλιση.



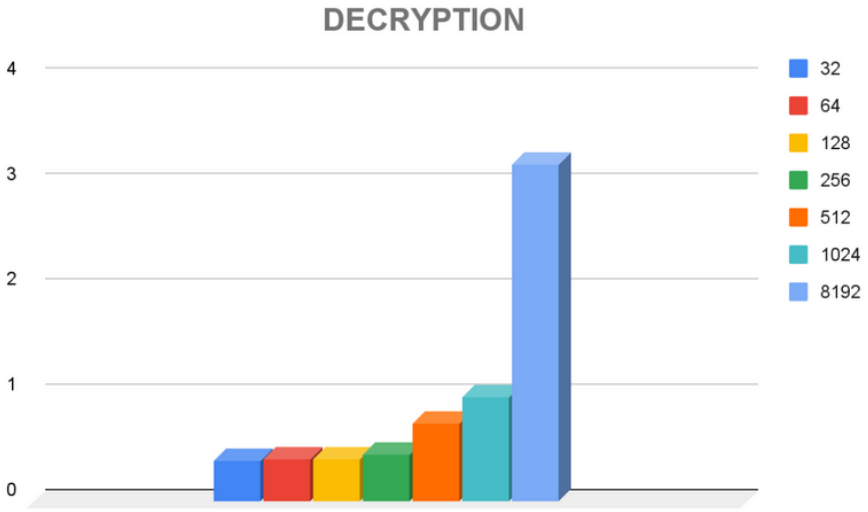
Σχήμα 6.1: *ABE Αρχικοποιηση για διαφορετικο αριθμο χαρακτηριστικων*

Ωστόσο, το ίδιο δεν ισχύει για την κρυπτογράφηση/αποκρυπτογράφηση των δεδομένων. Στο Σχήμα 6.2 και 6.3, παρατηρούμε εδώ ξεκάθαρα ότι ο αριθμός των χαρακτηριστικών επηρεάζει το κόστος της κρυπτογράφησης ενός κομματιού δεδομένων (1 KB). Ωστόσο, πρέπει

να σημειωθεί ότι, προκειμένου να παρατηρήσουμε μια επιβάρυνση στην απόδοση του πρωτοκόλλου, πρέπει να χρησιμοποιήσουμε έναν απραγματικό αριθμό χαρακτηριστικών.

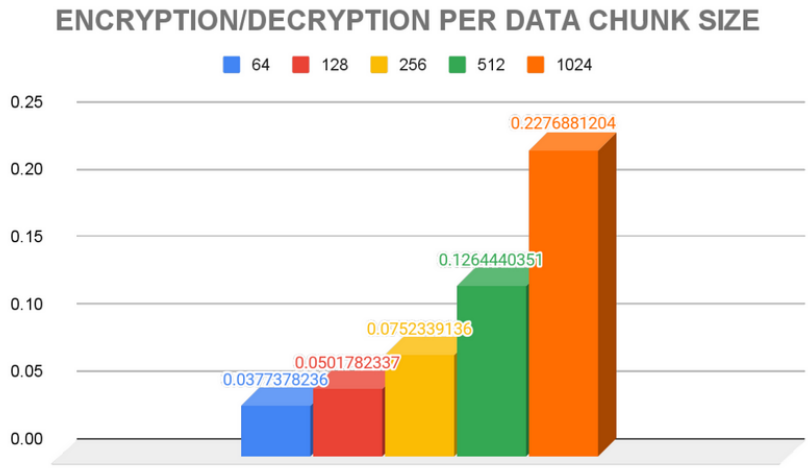


Σχήμα 6.2: *ABE Κρυπτογραφηση για διαφορετικο αριθμο χαρακτηριστικων*



Σχήμα 6.3: *ABE Αποκρυπτογραφηση για διαφορετικο αριθμο χαρακτηριστικων*

Στο σχημα **;;**, παρεχουμε την αξιολογηση του πρωτοκολλου για κρυπτογραφηση και αποκρυπτογραφηση διαφορων μεγεθων δεδομενων, σε Hardware Based TPM. Παρατηρούμε λοιπον, ότι το σχήμα κρυπτογράφησης ABE εξαρτάται από το μέγεθος των δεδομένων που πρέπει να κρυπτογραφηθούν/αποκρυπτογραφηθούν. Με βάση αυτές τις μετρήσεις, συμπεραίνουμε ότι ολόκληρη η εκτέλεση της διαδικασίας κρυπτογράφησης και αποκρυπτογράφησης μπορεί να μειωθεί σε περίπου 190ms, λαμβάνοντας υπόψη τον ίδιο αριθμό χαρακτηριστικών.



Σχήμα 6.4: *ABE Κρυπτογραφηση/Αποκρυπτογραφηση για διαφορετικο μεγεθος δεδομενων*

Λειτουργια	**FABEO**	**ABE**
Δημιουργια κλειδιων/Αρχικοποιηση	0.200s	0.409s
Κρυπτογραφηση	0,032s	0.696
Αποκρυπτογραφηση	0.031s	0.685

Πίνακας 6.15: *Αξιολογηση ABE εναντιον FABEO*

6.1.5 Αξιολογηση ABE εναντιον του FABEO

Σε αυτήν την ενότητα, παρέχουμε συγκριτικά πειραματικά αποτελέσματα του σχήματος ABE έναντι του σχήματος FABEO, το οποίο παρέχει ελεγχόμενη πρόσβαση υψηλής ευκρίνειας στα κρυπτογραφημένα δεδομένα **ABAC**, χωρίς περιορισμούς στον τύπο πολιτικής των χαρακτηριστικών, και στοχεύει στην επίτευξη βέλτιστης, προσαρμόσιμης ασφάλειας με πολλαπλά κρυπτογραφημένα κείμενα προκλήσεων. Η κύρια διαφορά με το σχήμα ABE είναι ότι το FABEO δεν λαμβάνει υπόψη τη χρήση ενός Hardware Root Of Trust και συνεπώς δεν παρέχει Hardware Based εγγυήσεις όσον αφορά την ασφάλεια, την ιδιωτικότητα και την εμπιστευτικότητα του σχήματος. Στον πίνακα 6.15 παρουσιάζονται οι χρόνοι που απαιτούνται για τη δημιουργία κλειδιών, την κρυπτογράφηση και την αποκρυπτογράφηση για τα δύο πρωτόκολλα.

Από αυτά τα αποτελέσματα, παρατηρούμε ότι το σχήμα FABEO είναι πολύ ταχύτερο από το σχήμα ABE, ειδικά στην περίπτωση των διαδικασιών κρυπτογράφησης και αποκρυπτογράφησης, όπου το ABE είναι περίπου εικοσι φορές πιο χρονοβόρο. Αν και αυτό μπορεί να υποδηλώνει χειρότερη απόδοση του σχήματος ABE, στην πραγματικότητα αυτό εισάγει έναν συμβιβασμό μεταξύ της απόδοσης και της παροχής εγγυήσεων βασισμένων στο Hardware το οποιο χρησιμοποιειται. Με άλλα λόγια, το κυριότερο συμπέρασμα είναι ότι η χρήση ενος Hardware Root of Trust είναι σε θέση να παράσχει εγγυήσεις ασφάλειας, ιδιωτικότητας και εμπιστοσύνης, οι οποιες δεν παρεχονται από το FABEO, καθιστώντας το πιο κατάλληλο για περιπτώσεις χρήσης με αυστηρές απαιτήσεις ασφάλειας και ιδιωτικότητας. Ειδικότε-

ρα, στο FABEO, η φάση Δημιουργίας Κλειδιών/Αρχικοποίησης περιλαμβάνει τη δημιουργία των κλειδιών κρυπτογράφησης από μια εμπιστη οντότητα για κάθε χρηστη/συσκευη που θελει να συμμετεχει στο συγκεκριμενο σχημα, αντίθετα με το σχήμα ABE αρχικοποιούμε το TPM σε μια συγκεκριμένη κατάσταση όπου μπορεί να παράγει συμμετρικά κλειδιά, και να ελεγχει κατά τη διάρκεια της εκτέλεσης του αλγοριθμου την χρηση τους έναντι μιας προκαθορισμένης αναπαράστασης του χώρου κλειδιών χαρακτηριστικών. Επιπλέον, στη φάση Κρυπτογράφησης/Αποκρυπτογράφησης, με το σχήμα FABEO η συσκευή του χρησιμοποιεί τα κλειδιά κρυπτογράφησης που παρέχονται από το TTP, ενώ στο σχήμα ABE για κάθε συνεδρία κρυπτογράφησης/αποκρυπτογράφησης παράγουμε νέα τυχαία συμμετρικά κλειδιά εντός του TPM (Trusted World), τα οποία στη συνέχεια χρησιμοποιούνται για την κρυπτογράφηση/αποκρυπτογράφηση των δεδομένων και τον υπολογισμό τον υπολογισμο του κατακερματισμου αυθεντικοποιησης. Συμπερασματικα, ενώ η υπολογιστική πολυπλοκότητα του σχήματος ABE είναι σημαντικά υψηλότερη, παρέχει επίσης αρκετά περισσότερες λειτουργίες, όπως ιδιωτικότητα και αποκεντρωμένη διαχείριση κλειδιών.

Μέρος III

Επίλογος

Κεφάλαιο 7

Επίλογος

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα συμπεράσματα τα οποία διευκρινίστηκαν από τη συγκεκριμένη διπλωματική, καθώς και μελλοντικές επεκτάσεις πάνω στην συγκεκριμένη θεματολογία.

7.1 Συμπεράσματα

Μέσω της συγκεκριμένης διπλωματικής δημιουργήθηκε ένα Attribute Based Encryption σχήμα το οποίο μπορεί να χρησιμοποιηθεί σε υψηλά κατανεμημένες και/ή απομακρυσμένες συσκευές με σημαντικά περιορισμένους πόρους, όσον αφορά την υπολογιστική τους ισχύ. Ειδικότερα, το σχήμα Κρυπτογράφησης με Βάση τα Χαρακτηριστικά (Attribute-Based Encryption - ABE) επιτρέπει διαφορετικά επίπεδα ασφάλειας, επιτρέποντας σε συσκευές που έχουν εγγραφεί στο σχήμα ABE να κρυπτογραφούν τα δεδομένα τους με τρόπο που τους επιτρέπει να ελέγχουν ποιος μπορεί να έχει πρόσβαση σε αυτά αποκρυπτογραφώντας τα. Επιπλέον, το ABE επιτρέπει στη συσκευή κρυπτογράφησης να κρυπτογραφεί δεδομένα με έναν τρόπο που εξασφαλίζει ότι καμία οντότητα, που δεν έχει το δικαίωμα, δεν μπορεί να εξάγει οποιαδήποτε πληροφορία σχετικά με το ποια χαρακτηριστικά χρησιμοποιήθηκαν για την κρυπτογράφηση, προσφέροντας έτσι τις απαιτούμενες εγγυήσεις ασφάλειας και ιδιωτικότητας της συσκευής. Να σημειωθεί εδώ ότι ακόμα κι αν κατά κάποιο τρόπο διαρρεύσουν τα κλειδιά χαρακτηριστικών από τη συσκευή που θα πραγματοποιήσει την κρυπτογράφηση σε μια συνεδρία κρυπτογράφησης, ένας αντίπαλος δεν μπορεί να εξάγει καμία πληροφορία για το κλειδί κρυπτογράφησης, καθώς αυτό έχει δημιουργηθεί από έναν μυστικό αριθμό ο οποίος δεν φεύγει ποτέ εκτός του TPM.

Οπως ειδαμε και στο αναλυτικο benchmarking του προηγουμενου κεφαλαιου το συγκεκριμενο σχημα ειναι αρκετα πιο αργα απο τα περισσοτερα τα οποια εξετασθηκαν για αυτη την διπλωματικη εργασια, ωστοσο δεν βρεθηκε αλλο σχημα ABE το οποιο να προσφερει παρομοιες ιδιοτητες οπως το συγκεκριμενο. Με αλλα λογια με την χρηση του συγκεκριμενου σχηματος ενας χρηστης θα θυσιασει την αποδοση του συστηματος του, με τα επιπεδα ασφαλειας που του παρεχονται απ το συγκεκριμενο ABE. Σε εφαρμογες οπου υπαρχουν υψηλες απαιτησεις οσον αφορα την ευαισθησια των δεδομενων που ειτε χρειαζεται να αποθυκευτουν σε μια κεντρικη βαση δεδομενων, ειτε να σταλουν σε καποιον αλλον χρηστη μεσω του διαδικτυου το συγκεκριμενο σχημα ειναι ιδανικο συγκριτικα με τα υπαρχον, εφοσον τα δεδομενα τα οποια χρειαζονται να κρυπτογραφηθουν να ειναι σχετικα μικρου μεγεθους. Τελος, ειναι

δυσκολο να υιοθετηθει αυτο το σχημα σε υπολιστικα συστηματα γενικης χρησης, γιατι οι εμπορικοι υπολογιστες διαθετουν ακομα παλιες εκδοσεις TPM που δεν υποστηριζουν συμμετρικη κρυπτογραφια και φυσικα γιατι η εγκατασταση του νεου TPM (SLB 9673) ειναι πολυ κοστοβορα για καποιον ο οποιος δεν εχει την αναγκη να εφαρμοσει να τετοιο σχημα.

7.2 Μελλοντικες Επεκτασεις

Σε αυτήν την υποενότητα θα παρουσιαστούν οι μελλοντικές επεκτάσεις όσον αφορά που θα πραγματοποιηθούν ερευνητικά πάνω στο συγκεκριμένο σχήμα.

7.2.1 Επεκτασεις πανω στο Trusted Computing

Όπως αναλύθηκε στα προηγούμενα κεφάλαια, το συγκεκριμένο σχήμα υλοποιήθηκε χρησιμοποιώντας TPMs. Επειδή τα TPMs ρίχνουν πάρα πολύ την απόδοση του συγκεκριμένου πρωτοκόλλου, θα διερευνηθεί η υλοποίηση του συγκεκριμένου πρωτοκόλλου χρησιμοποιώντας διαφορετικά Hardware Root of Trust. Συγκεκριμένα, έχουν επιλεγεί το GRAMINE, το οποίο είναι ένας UniKernel για αρχιτεκτονικές x86, x86-64 και χρησιμοποιεί μια επέκταση των Intel επεξεργαστών (Intel Software Guard eXtension), το οποίο μπορεί να βρεθεί σε όλους τους καινούργιους επεξεργαστές XEON καθώς και σε κάποιες γενιές επεξεργατών i7 και i9. Σύμφωνα με τα πειράματα που έχουν διεξαχθεί πάνω στο GRAMINE, το κόστος του να εκτελείς μια Trusted εφαρμογή μέσω του GRAMINE υπολογίζεται ανάμεσα στο $100\% - 500\%$ παραπάνω από την εκτέλεσή τους ως μια απλή Linux διεργασία (ELF). Το παραπάνω κόστος εξαρτάται με τη γλώσσα με την οποία υλοποιήθηκε η Trusted εφαρμογή, με την C και RUST να παρουσιάζουν τα καλύτερα αποτελέσματα και οι Python και GoLang τα χειρότερα. Πέρα από το GRAMINE, θα διερευνηθεί και η χρήση του OP-TEE, το οποίο είναι και το πιο διαδεδομένο Trusted Execution Environment και προορίζεται για ARM αρχιτεκτονικές. Το OP-TEE χρησιμοποιεί επίσης ένα Hardware Extension το οποίο βρίσκεται σε κάποιους ARM επεξεργαστές το οποίο λέγεται TrustZone. Το OP-TEE παρουσιάζει επίσης πολύ καλά αποτελέσματα όσον αφορά την εκτέλεση Trusted εφαρμογών, αλλά περιορίζεται στη γλώσσα υλοποίησης της εφαρμογής καθώς περιορίζεται μόνο σε υλοποίηση με C και εκτελείται ως μέρος του Kernel της συσκευής.

7.2.2 Κρυπτογραφικες Επεκτασεις

Πέρα από τις επεκτάσεις που θα διερευνηθούν πάνω στο Trusted Computing, θα πραγματοποιηθεί έρευνα και σε διάφορα κρυπτογραφικά πρωτόκολλα τα οποία μπορούν να εφαρμοστούν. Συγκεκριμένα, θα πραγματοποιηθεί έρευνα ως προς το πώς μπορεί να διαμοιραστεί ο τυχαίος αριθμός ο οποίος συμβάλλει στην δημιουργία των συμμετρικών κλειδιών. Βάσει αυτού έχει εξεταστεί η εφαρμογή μιας παραλλαγής του SHAMIR Secret Sharing όπου δεν θα μπορεί μόνο μια συσκευή που θέλει να αποκρυπτογραφήσει κάποια δεδομένα, αλλά θα πρέπει να συνεργαστούν πολλές συσκευές μαζί ώστε να εξάγουν αυτόν τον τυχαίο αριθμό. Με αυτό τον τρόπο, ακόμα και μια συσκευή που θέλει να αποκρυπτογραφήσει κάποια δεδομένα έχει παραβιαστεί, θα της είναι αδύνατον να πραγματοποιήσει την αποκρυπτογράφηση των δεδομένων αυτών μόνη της. Εκτός από την εφαρμογή του SHAMIR Secret Sharing,

θα διερευνηθούν και πρωτόκολλα τα οποία αντέχουν σε επιθέσεις από κβαντικούς υπολογιστές. Τέτοιες τεχνικές είναι για παράδειγμα το Isogeny Key Exchange για να μπορεί να διαμοιραστεί με ασφάλεια ένα κοινό μυστικό καθώς και τα πρωτόκολλα Kyber και Dilithium για την δημιουργία ψηφιακών υπογραφών. Δεν χρειάζεται να γίνει κάποια επέκταση ισχύος αφορά τους συμμετρικούς αλγορίθμους που χρησιμοποιούνται καθώς έχει αποδειχθεί ότι αντέχουν σε επιθέσεις από κβαντικούς υπολογιστές αν αυξηθεί το μήκος του κλειδιού το οποίο χρησιμοποιείται.

Αξιοσημείωτες κρυπτογραφικές τεχνικές

Όπως είδαμε στην ανάλυση του συγκεκριμένου αλγορίθμου, παρέχουμε διαφορετικά επίπεδα ελεγξιμότητας πάνω στην πολιτική που εφαρμόστηκε κατά τη διάρκεια μιας συνεδρίας κρυπτογράφησης, μέσω ψηφιακών υπογραφών. Από τα πειραματικά μας αποτελέσματα, παρατηρήσαμε ότι ο αλγόριθμος Direct Anonymous Attestation αν και έχει παρα πολύ ισχυρές κρυπτογραφικές ιδιότητες κάνει την συνολική χρονική εκτέλεση του αλγορίθμου ABE έως και διπλάσια σε ορισμένες περιπτώσεις. Αυτή η παρατήρηση μας οδηγεί να εξερευνήσουμε και άλλα σχήματα ψηφιακών υπογραφών τα οποία ενώ δεν έχουν όλες τις κρυπτογραφικές ιδιότητες του Direct Anonymous Attestation, προστατεύουν την ιδιωτικότητα ενός χρήστη. Τέτοια σχήματα υπογραφών είναι, για παράδειγμα, τα Publicly Auditable Conditional Blind Signatures - PACBS [51], ανώνυμες υπογραφές η εγκυρότητα των οποίων εξαρτάται από συνθήκες που έχει αποφασίσει ένας ορισμένος ελεγκτής και που αξιοποιούνται και στην περιοχή του e-voting [52, 53, 54], καθώς και το σχήμα υπογραφών Designated-Verifier Linkable Ring Signatures (DVLRS) [49, 50], μια καινοτόμο κρυπτογραφική πρωτογενή που συνδυάζει τις κρυπτογραφικές ιδιότητες του καθορισμένου ελεγκτή και των linkable υπογραφών δακτυλιού. Ο στόχος μας είναι να εξασφαλίσουμε την ανωνυμία των υπογραφόντων και να παρέχουμε στον καθορισμένο ελεγκτή τη δυνατότητα να προσθέτει ¨θόρυβο¨ χρησιμοποιώντας προσομοιωμένες υπογραφές που είναι δημοσίως επαληθεύσιμες, αυξάνοντας έτσι τη συνολική ιδιωτικότητα. Συγκεκριμένα για τις υπογραφές Designated-Verifier Linkable Ring Signatures (DVLRS) πραγματοποιήθηκαν πειράματα σε Trusted Execution Environment (GRAMINE INTEL SGX) με πολύ καλά αποτελέσματα, γεγονός που μας οδηγεί στην εξερεύνηση λειτουργίας αυτού του σχήματος χρησιμοποιώντας TPMs. Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα του συγκεκριμένου πειράματος 7.1, οπου πραγματοποιήθηκαν σε αρχιτεκτονική X86-64 και υπολογίστηκε ο μέσος όρος της κάθε λειτουργίας του σχήματος.

Λειτουργία	**DVLRS**
Δημιουργία κλειδιών	0.00151s
Υπογράφη	0.121s
Πιστοποίηση	0.00006905s

Πίνακας 7.1: *Αξιολόγηση DVLRS σε GRAMINE*

Βιβλιογραφία

[1] Bryan Parno, *Bootstrapping Trust in a "Trusted" Platform*, Proceedings of the 3rd Conference on Hot Topics in Security, HOTSEC'08, pp. 9:1–9:6, 2008. http://dl.acm.org/citation.cfm?id=1496671.1496680.

[2] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, *Privacy in Inter-Vehicular Networks: Why Simple Pseudonym Change Is Not Enough*, Seventh International Conference on Wireless On-demand Network Systems and Services (WONS), pp. 176–183, Feb. 2010.

[3] David Förster, Hans Löhr, Jan Zibuschka, and Frank Kargl, *REWIRE - Revocation Without Resolution: A Privacy-Friendly Revocation Mechanism for Vehicular Ad-Hoc Networks*, Trust and Trustworthy Computing, 2015.

[4] J. Petit, F. Schaub, M. Feiri, and F. Kargl, *Pseudonym Schemes in Vehicular Networks: A Survey*, IEEE Communications Surveys Tutorials, vol. 17, no. 1, pp. 228–255, 2015.

[5] Petar Maymounkov and David Mazières, *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*, Peer-to-Peer Systems, edited by Peter Druschel, Frans Kaashoek, and Antony Rowstron, Springer Berlin Heidelberg, 2002, pp. 53–65. https://doi.org/10.1007/3-540-45748-8$_5$.

[6] ISO/IEC JTC 1/SC 27, *IT Security and Privacy — A Framework for Identity Management — Part 1: Terminology and Concepts*, ISO/IEC 24760-1:2019, 2019.

[7] Sebastian Clauß and Marit Köhntopp, *Identity Management and Its Support of Multilateral Security*, Computer Networks, vol. 37, no. 2, pp. 205–219, Aug. 2001.

[8] *Privacy-Enhancing Identity Management*, Information Security Technical Report, vol. 9, no. 1, pp. 35–44, 2004.

[9] Ahmad Sabouri, Ioannis Krontiris, and Kai Rannenberg, *Trust Relationships in Privacy-ABCs Ecosystems*, International Conference on Trust, Privacy and Security in Digital Business, Springer International Publishing, 2014, pp. 13–23.

[10] Benjamin Larsen, Thanassis Giannetsos, Ioannis Krontiris, and Kenneth Goldman, *Direct Anonymous Attestation on the Road: Efficient and Privacy-Preserving Revocation in C-ITS*, Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '21, New York, NY, USA, 2021, pp. 48–59. https://doi.org/10.1145/3448300.3467832.

[11] Jorden Whitefield, Liqun Chen, Thanassis Giannetsos, Steve A. Schneider, and Helen Treharne, *Privacy-Enhanced Capabilities for VANETs Using Direct Anonymous Attestation*, VNC, IEEE, 2017, pp. 123–130. http://dblp.uni-trier.de/db/conf/vnc/vnc2017.htmlWhitefieldCGST17.

[12] Georgios Fotiadis, José Moreira, Thanassis Giannetsos, Liqun Chen, Peter B. Rønne, Mark D. Ryan, and Peter Y. A. Ryan, *Root-of-Trust Abstractions for Symbolic Analysis: Application to Attestation Protocols*, Security and Trust Management, edited by Rodrigo Roman and Jianying Zhou, Springer International Publishing, Cham, 2021, pp. 163–184. https://doi.org/10.1007/978-3-030-91859-0$_9$.

[13] Petar Maymounkov and David Mazières, *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*, Peer-to-Peer Systems, edited by Peter Druschel, Frans Kaashoek, and Antony Rowstron, Springer Berlin Heidelberg, 2002, pp. 53–65. https://doi.org/10.1007/3-540-45748-8$_5$.

[14] ISO/IEC JTC 1/SC 27, *IT Security and Privacy — A Framework for Identity Management — Part 1: Terminology and Concepts*, ISO/IEC 24760-1:2019, 2019.

[15] Sebastian Clauß and Marit Köhntopp, *Identity Management and Its Support of Multilateral Security*, Computer Networks, vol. 37, no. 2, pp. 205–219, Aug. 2001.

[16] *Privacy-Enhancing Identity Management*, Information Security Technical Report, vol. 9, no. 1, pp. 35–44, 2004.

[17] Ahmad Sabouri, Ioannis Krontiris, and Kai Rannenberg, *Trust Relationships in Privacy-ABCs Ecosystems*, International Conference on Trust, Privacy and Security in Digital Business, Springer International Publishing, 2014, pp. 13–23.

[18] Benjamin Larsen, Thanassis Giannetsos, Ioannis Krontiris, and Kenneth Goldman, *Direct Anonymous Attestation on the Road: Efficient and Privacy-Preserving Revocation in C-ITS*, Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '21, New York, NY, USA, 2021, pp. 48–59. https://doi.org/10.1145/3448300.3467832.

[19] Jorden Whitefield, Liqun Chen, Thanassis Giannetsos, Steve A. Schneider, and Helen Treharne, *Privacy-Enhanced Capabilities for VANETs Using Direct Anonymous Attestation*, VNC, IEEE, 2017, pp. 123–130. http://dblp.uni-trier.de/db/conf/vnc/vnc2017.htmlWhitefieldCGST17.

[20] Georgios Fotiadis, José Moreira, Thanassis Giannetsos, Liqun Chen, Peter B. Rønne, Mark D. Ryan, and Peter Y. A. Ryan, *Root-of-Trust Abstractions for Symbolic Analysis: Application to Attestation Protocols*, Security and Trust Management, edited by Rodrigo Roman and Jianying Zhou, Springer International Publishing, Cham, 2021, pp. 163–184. https://doi.org/10.1007/978-3-030-91859-0$_9$.

[21] Georgios Fotiadis, Jose Moreira-Sanchez, Thanassis Giannetsos, Liqun Chen, Peter B. Ronne, Mark Ryan, and Peter Y.A. Ryan, *Root-of-Trust Abstractions for Symbolic*

Analysis: Application to Attestation Protocols, STM 2021: Security and Trust Management, 2021.

[22] M. Carbone, M. Nielsen, and V. Sassone, *A Formal Model for Trust in Dynamic Networks*, First International Conference on Software Engineering and Formal Methods, 2003. Proceedings., IEEE, Sep. 2003, pp. 54–61. https://doi.org/10.1109/SEFM.2003.1236207.

[23] Michael E. Locasto, Steven J. Greenwald, and Sergey Bratus, *Trust Distribution Diagrams: Theory and Applications*, Proceedings of the 4th Layered Assurance Workshop (LAW 2010), Austin, TX, 2010.

[24] Alessandro Aldini, *A Formal Framework for Modeling Trust and Reputation in Collective Adaptive Systems*, arXiv preprint arXiv:1607.02232, 2016.

[25] Munirul M. Haque and Sheikh I. Ahamed, *An Omnipresent Formal Trust Model (FTM) for Pervasive Computing Environment*, 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), vol. 1, IEEE, 2007, pp. 49–56.

[26] Sergei Skorobogatov, *Physical Attacks and Tamper Resistance*, Introduction to Hardware Security and Trust, Springer New York, 2012, pp. 143–173.

[27] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Trans. Inf. Theor., vol. 22, no. 6, pp. 644–654, Nov. 1976. https://doi.org/10.1109/TIT.1976.1055638.

[28] Zhang Xiong, Hao Sheng, WenGe Rong, and Dave E. Cooper, *Intelligent Transportation Systems for Smart Cities: A Progress Review*, Science China Information Sciences, 2012.

[29] Ernest F. Brickell, Jan Camenisch, and Liqun Chen, *Direct Anonymous Attestation*, ACM Conference on Computer and Communications Security, CCS, 2004.

[30] Philippe Golle and Kurt Partridge, *On the Anonymity of Home/Work Location Pairs*, Proceedings of the 7th International Conference on Pervasive Computing, Pervasive '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 390–397. https://doi.org/10.1007/978-3-642-01516-$8_2$9.

[31] ETSI TS 103 097 V1.3.1, *Intelligent Transport Systems (ITS); Security; Security Header and Certificate Formats*, October 2017.

[32] ETSI, *Trust and Privacy Management*, ETSI Technical Specification, 2012. http://www.etsi.org/deliver/etsi_ts/102900_102999/102941/01.01.01_60/ts_102941v010101p.pdf [Online; accessed 26-August-2017].

[33] Stylianos Gisdakis, Marcello Lagana, Thanassis Giannetsos, and Panos Papadimitratos, *SEROSA: SERvice Oriented Security Architecture for Vehicular Communications*, VNC, IEEE, 2013, pp. 111–118.

[34] Jorden Whitefield, Liqun Chen, Thanassis Giannetsos, Steve Schneider, and Helen Treharne, *Privacy-Enhanced Capabilities for VANETs Using Direct Anonymous Attestation*, 2017 IEEE Vehicular Networking Conference (VNC), IEEE, 2017, pp. 123–130. https://doi.org/10.1109/VNC.2017.8275615.

[35] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos, *SPPEAR: Security & Privacy-Preserving Architecture for Participatory-Sensing Applications*, Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, WiSec '14, ACM, New York, NY, USA, 2014, pp. 39–50. https://doi.org/10.1145/2627393.2627402.

[36] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah, *Trusted Execution Environment: What It Is, and What It Is Not*, 2015 IEEE Trustcom, IEEE, 2015, pp. 57–64.

[37] Ανυπαμ Δαττα ετ αλ., *Α Λογις οφ Σεχυρε Σψστεμς ανδ Ιτς Αππλιχατιον το Τρυστεδ Ὅμπυτινγ*, 30τη ΙΕΕΕ Σψμποσιυμ ον Σεχυριτψ ανδ Πριαχψ, ΙΕΕΕ, 2009, πп. 221–236.

[38] Reiner Sailer et al., *Design and Implementation of a TCG-Based Integrity Measurement Architecture*, USENIX Security Symposium, 2004, pp. 223–238.

[39] Trusted Platform Module (TPM), *Trusted Computing Group*, 2020. https://trustedcomputinggroup.org/work-groups/trusted-platform-module, Accessed: 2020-12-01.

[40] Trusted Platform Module (TPM) 2.0: A BRIEF INTRODUCTION, *Trusted Computing Group*, 2015. https://trustedcomputinggroup.org/wp-content/uploads/TPM-2.0-A-Brief-Introduction.pdf, Accessed: 2020-12-01.

[41] Xuanxia Yao, Zhi Chen, and Ye Tian, *A Lightweight Attribute-Based Encryption Scheme for the Internet of Things*, Future Generation Computer Systems, vol. 49, pp. 104–112, 2015, Elsevier.

[42] Syh-Yuan Tan, Kin-Woon Yeow, and Seong Oun Hwang, *Enhancement of a Lightweight Attribute-Based Encryption Scheme for the Internet of Things*, IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6384–6395, 2019, IEEE.

[43] Jan Camenisch, Manu Drijvers, and Anja Lehmann, *Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited*, Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9, Springer, pp. 1–20, 2016.

[44] ′ Dan Boneh, *The Decision Diffie-Hellman Problem*, International Algorithmic Number Theory Symposium, Springer, pp. 48–63, 1998.

[45] Dan Boneh, *The Decision Diffie-Hellman Problem*, Algorithmic Number Theory, edited by Joe P. Buhler, Springer Berlin Heidelberg, pp. 48–63, 1998.

[46] Dan Boneh, *The Decision Diffie-Hellman Problem*, Algorithmic Number Theory: Third International Symposium, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings, Springer, pp. 48–63, 2006.

[47] IBM's TPM 2.0 TSS, *Ken Goldman*, 2020. https://sourceforge.net/projects/ibmtpm20tss/, Accessed: 2020-12-01.

[48] Linux TPM2 & TSS2 Software, *Frauenhofer*, 2020. https://github.com/tpm2-software, Accessed: 2020-12-01.

[49] Pourandokht Behrouz, Panagiotis Grontas, Vangelis Konstantakatos, Aris Pagourtzis, and Marianna Spyrakou, *Designated-Verifier Linkable Ring Signatures*, 24th International Conference on Information Security and Cryptology - ICISC 2021, LNCS, vol. 13218, pp. 51–70, 2022. https://doi.org/10.1007/978-3-031-08896-4_3.

[50] Danai Balla, Pourandokht Behrouz, Panagiotis Grontas, Aris Pagourtzis, Marianna Spyrakou, and Giannis Vrettos, *Designated-Verifier Linkable Ring Signatures with Unconditional Anonymity*, 9th International Conference on Algebraic Informatics, CAI 2022, LNCS, vol. 13706, pp. 55–68, 2022. https://doi.org/10.1007/978-3-031-19685-0_5.

[51] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang, *Publicly Auditable Conditional Blind Signatures*, Journal of Computer Security, vol. 29, no. 2, pp. 229–271, 2021. https://doi.org/10.3233/JCS-181270.

[52] Panagiotis Grontas, Aris Pagourtzis, and Alexandros Zacharakis, *Coercion Resistance in a Practical Secret Voting Scheme for Large Scale Elections*, ISPAN-FCST-ISCC 2017, Exeter, United Kingdom, June 21-23, 2017, pp. 514–519, IEEE Computer Society, 2017. https://doi.org/10.1109/ISPAN-FCST-ISCC.2017.79.

[53] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang, *Towards Everlasting Privacy and Efficient Coercion Resistance in Remote Electronic Voting*, Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Revised Selected Papers, LNCS, vol. 10958, pp. 210–231, Springer, 2018. https://doi.org/10.1007/978-3-662-58820-$8_1$5.

[54] Panagiotis Grontas and Aris Pagourtzis, *Anonymity and Everlasting Privacy in Electronic Voting*, International Journal of Information Security, vol. 22, no. 4, pp. 819–832, 2023. https://doi.org/10.1007/S10207-023-00666-2.

Παραρτήματα

Παράρτημα Α΄

Σημαντικες Συναρτησεις

Στο συγκεκριμενο παραρτημα παρουσιαζονται σημαντικες συναρτησεις, οι οποιες ειναι μερος της εφαρμογης που υλοποιηθηκε.

```
EC_KEY * generate_Ecc(void){


  int eccgrp, rc;
  EVP_PKEY *pkey = NULL;
  EC_KEY *myecc = NULL;
  // const EC_GROUP *ecgrp;

    eccgrp = OBJ_txt2nid("prime256v1");
    myecc = EC_KEY_new_by_curve_name(eccgrp);
    if(myecc == NULL){
      handle_ssl_error(0);
    }

    EC_KEY_set_asn1_flag(myecc, OPENSSL_EC_NAMED_CURVE);

  rc = EC_KEY_generate_key(myecc);
  handle_ssl_error(rc);

  pkey = EVP_PKEY_new();

    rc = EVP_PKEY_assign_EC_KEY(pkey,myecc);
    handle_ssl_error(rc);

  myecc = EVP_PKEY_get1_EC_KEY(pkey);
  ////////////////////////////////////////////////////////////
    EVP_PKEY_free(pkey);

  return myecc;

}

EC_GROUP *get_ec_group_bnp256(void) {

    int ok = 0;
    EC_GROUP *curve = NULL;
    EC_POINT *generator = NULL;
```

```
38      BN_CTX *ctx = BN_CTX_new();
39      BIGNUM *tmp_1 = NULL, *tmp_2 = NULL, *tmp_3 = NULL;
40
41      // Added those
42      unsigned char bnp256_a[1] = {0x00};
43      unsigned char bnp256_b[1] = {0x03};
44      unsigned char bnp256_gX_[1] = {0x01};
45      unsigned char bnp256_gy_[1] = {0x02};
46
47
48      if ((tmp_1 = BN_bin2bn(bnp256_p, 32, NULL)) == NULL)
49          goto err;
50      if ((tmp_2 = BN_bin2bn(bnp256_a, 1, NULL)) == NULL)
51          goto err;
52      if ((tmp_3 = BN_bin2bn(bnp256_b, 1, NULL)) == NULL)
53          goto err;
54      if ((curve = EC_GROUP_new_curve_GFp(tmp_1, tmp_2, tmp_3, NULL)) == NULL)
55          goto err;
56
57
58      /* build generator */
59      generator = EC_POINT_new(curve);
60      if (generator == NULL)
61          goto err;
62      if ((tmp_1 = BN_bin2bn(bnp256_gX_, 1, tmp_1)) == NULL)
63          goto err;
64      if ((tmp_2 = BN_bin2bn(bnp256_gy_, 1, tmp_2)) == NULL)
65          goto err;
66      if (1 != EC_POINT_set_affine_coordinates_GFp(curve, generator, tmp_1, tmp_2, ctx))
67          goto err;
68
69      if ((tmp_1 = BN_bin2bn(BNP256_ORDER, 32, tmp_1)) == NULL)
70          goto err;
71      BN_one(tmp_2);
72      if (1 != EC_GROUP_set_generator(curve, generator, tmp_1, tmp_2))
73          goto err;
74
75      ok = 1;
76
77      err:
78      if (tmp_1)
79          BN_free(tmp_1);
80      if (tmp_2)
81          BN_free(tmp_2);
82      if (tmp_3)
83          BN_free(tmp_3);
84      if (generator)
85          EC_POINT_free(generator);
86      if (ctx)
87          BN_CTX_free(ctx);
88      if (!ok) {
89          printf("[-] Error in creating curve\n");
```

```
90          EC_GROUP_free(curve);
91          curve = NULL;
92      }
93      return (curve);
94  }
95
96
97  EC_POINT *Create_Public_Master_Attribute(uint8_t *Master_Attribute){
98
99      unsigned char BNP256_ORDER[32] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xF0, 0xCD, 0
        x46, 0xE5, 0xF2, 0x5E, 0xEE, 0x71,
100                                       0xA4, 0x9E, 0x0C, 0xDC, 0x65, 0xFB, 0x12, 0x99, 0x92,
         0x1A, 0xF6, 0x2D, 0x53, 0x6C,
101                                       0xD1, 0x0B, 0x50, 0x0D};
102     EC_GROUP *ecgrp = NULL;
103     EC_POINT *generator; //@2
104     uint8_t p1[2] = {0x01, 0x02};
105     BIGNUM *multiplier = BN_new();
106     BN_CTX *ctx = BN_CTX_new();
107     EC_POINT *resultingPoint;
108     // point_conversion_form_t form;
109
110     ecgrp = get_ec_group_bnp256();
111     generator = EC_POINT_new(ecgrp);
112     get_ECC_POINT(generator, &p1[0], &p1[1], ecgrp, 1);
113     resultingPoint = EC_POINT_new(ecgrp);
114     multiplier = BN_Mod(Master_Attribute, BNP256_ORDER);
115
116     if (1 != EC_POINT_mul(ecgrp, resultingPoint, NULL, generator, multiplier, ctx)) {
117         printf("[-] There was an error during multiplication\n");
118
119
120         BN_CTX_free(ctx);    // @1
121         EC_GROUP_free(ecgrp);
122
123         return NULL;
124     }
125     BN_free(multiplier);
126     EC_GROUP_free(ecgrp);
127     EC_POINT_free(generator);
128     BN_CTX_free(ctx);    // @1
129
130     return resultingPoint;
131 }
132
133
134
135 int Create_Key_Seed(GetRandom_Out K, EC_POINT *PK, uint8_t *seed_x, uint8_t *seed_y){
136
137   unsigned char BNP256_ORDER[32] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xF0, 0xCD, 0
      x46, 0xE5, 0xF2, 0x5E, 0xEE, 0x71,
```

```
138                                 0xA4, 0x9E, 0x0C, 0xDC, 0x65, 0xFB, 0x12, 0x99, 0x92,
       0x1A, 0xF6, 0x2D, 0x53, 0x6C,
139                                 0xD1, 0x0B, 0x50, 0x0D};
140
141   EC_GROUP *ecgrp = NULL;
142   EC_POINT *temp_point;
143   BIGNUM *multiplier = BN_new();
144
145   BN_CTX *ctx = BN_CTX_new();
146   TPM_RC rc = 0;
147
148
149     ecgrp = get_ec_group_bnp256();
150     temp_point = EC_POINT_new(ecgrp);
151
152   multiplier = BN_Mod(K.randomBytes.b.buffer, BNP256_ORDER);
153
154   if (1 != EC_POINT_mul(ecgrp, temp_point, NULL, PK, multiplier, ctx)) {
155         printf("[-] There was an error during multiplication\n");
156
157
158         BN_CTX_free(ctx);    // @1
159         EC_GROUP_free(ecgrp);
160
161         return EXIT_FAILURE;
162     }
163   if(1 != point2bb(seed_x, seed_y, temp_point)){
164     printf("[ABE]\t ERROR IN convverting point to bb\n");
165     BN_CTX_free(ctx);
166     BN_free(multiplier);
167     EC_POINT_free(temp_point);
168     EC_GROUP_free(ecgrp);
169     return 0;
170
171   }
172
173   BN_CTX_free(ctx);
174   BN_free(multiplier);
175   EC_POINT_free(temp_point);
176   EC_GROUP_free(ecgrp);
177   return rc;
178 }
179
180
181 EC_POINT* create_shared_secret(TPM2B_DIGEST attribute, TPM2B_DIGEST Key_Seed){
182
183   unsigned char BNP256_ORDER[32] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xF0, 0xCD, 0
       x46, 0xE5, 0xF2, 0x5E, 0xEE, 0x71,
184                                 0xA4, 0x9E, 0x0C, 0xDC, 0x65, 0xFB, 0x12, 0x99, 0x92,
       0x1A, 0xF6, 0x2D, 0x53, 0x6C,
185                                 0xD1, 0x0B, 0x50, 0x0D};
186   unsigned const char *attrfile = "rolesoutput.bin";
```

```
187  uint8_t attrbuffer[298];
188  FILE *fp;
189  uint8_t *zerobuff=malloc(32);
190
191  BIGNUM *BN_attribute = BN_new();
192  BIGNUM *BN_Random = BN_new();
193  BN_CTX *ctx = BN_CTX_new();
194
195  ////////////////////////////////////////////////////////////
196  //Update
197  ////////////////////////////////////////////////////////////
198  EC_GROUP *ecgrp = NULL;
199    EC_POINT *generator; //@2
200    EC_POINT *resultingPoint;
201    EC_POINT *temp_point;
202    uint8_t p1[2] = {0x01, 0x02};
203    ecgrp = get_ec_group_bnp256();
204    generator = EC_POINT_new(ecgrp);
205    temp_point = EC_POINT_new(ecgrp);
206    get_ECC_POINT(generator, &p1[0], &p1[1], ecgrp, 1);
207    resultingPoint = EC_POINT_new(ecgrp);
208
209  BN_attribute = BN_Mod(attribute.b.buffer, BNP256_ORDER);
210
211
212  if (1 != EC_POINT_mul(ecgrp, resultingPoint, NULL, generator, BN_attribute, ctx)) {
213        printf("[-] There was an error during multiplication\n");
214        BN_free(BN_attribute);
215        EC_POINT_free(resultingPoint);
216        EC_GROUP_free(ecgrp);
217        EC_POINT_free(generator);
218        EC_POINT_free(temp_point);
219        BN_CTX_free(ctx);
220  }
221  EC_POINT_free(generator);
222    BN_free(BN_attribute);
223    BN_Random = BN_bin2bn(Key_Seed.b.buffer, 32, NULL);
224
225  if (1 != EC_POINT_mul(ecgrp, temp_point, NULL, resultingPoint, BN_Random, ctx)) {
226    printf("[-] There was an error during multiplication\n");
227    EC_POINT_free(temp_point);
228        BN_free(BN_attribute);
229        EC_POINT_free(resultingPoint);
230        EC_GROUP_free(ecgrp);
231        EC_POINT_free(generator);
232        BN_CTX_free(ctx);
233  }
234
235  memset(zerobuff,0x00,32);
236  memset(attrbuffer, 0x00, 298);
237  fp = fopen(attrfile, "rb");
238  fread(attrbuffer, 298, 1, fp);
```

```
239   fclose(fp);
240     for(int i=0; i<9; i++){
241     if (memcmp(&attrbuffer[32*i],zerobuff,32) != 0){
242
243       BN_attribute = BN_new();
244       BN_attribute = BN_Mod(&attrbuffer[32*i], BNP256_ORDER);
245       if (1 != EC_POINT_mul(ecgrp, temp_point, NULL,temp_point, BN_attribute, ctx)) {
246         printf("[-] There was an error during multiplication\n");
247         BN_free(BN_attribute);
248         EC_POINT_free(resultingPoint);
249         EC_GROUP_free(ecgrp);
250         EC_POINT_free(generator);
251         EC_POINT_free(temp_point);
252         BN_CTX_free(ctx);
253         }
254       BN_free(BN_attribute);
255     }
256   }
257   EC_POINT_free(resultingPoint);
258     BN_free(BN_Random);
259     EC_GROUP_free(ecgrp);
260     BN_CTX_free(ctx);
261
262   return temp_point;
263 }
264
265
266 int compute_Key_seed(uint8_t *MasterAttribute[65], uint8_t *key_seed_x, uint8_t *
      key_seed_y, TPM2B_DIGEST attribute, EC_POINT *shared_secret){
267   unsigned const char *attrfile = "rolesoutput.bin";
268   uint8_t *zerobuff = malloc(32);
269   uint8_t attrbuffer[298];
270   FILE *fp;
271   unsigned char BNP256_ORDER[32] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xF0, 0xCD, 0
      x46, 0xE5, 0xF2, 0x5E, 0xEE, 0x71,
272                                     0xA4, 0x9E, 0x0C, 0xDC, 0x65, 0xFB, 0x12, 0x99, 0x92,
       0x1A, 0xF6, 0x2D, 0x53, 0x6C,
273                                     0xD1, 0x0B, 0x50, 0x0D};
274
275   BIGNUM *BN_attribute = BN_new();
276   BIGNUM *BN_inverse = BN_new();
277   BIGNUM *BN_order = BN_new();
278   BIGNUM *BN_Master = BN_new();
279   BN_CTX *ctx = BN_CTX_new();
280
281
282   EC_GROUP *ecgrp = NULL;
283     EC_POINT *resultingPoint;
284     EC_POINT *finalPoint;
285     ecgrp = get_ec_group_bnp256();
286     resultingPoint = EC_POINT_new(ecgrp);
287   BN_attribute = BN_bin2bn(attribute.b.buffer, attribute.b.size, NULL);
```

```
288  BN_order = BN_bin2bn(BNP256_ORDER, 32, NULL);
289  BN_inverse = BN_mod_inverse(NULL, BN_attribute, BN_order, ctx);
290  BN_Master = BN_Mod(MasterAttribute, BNP256_ORDER);
291  if (1 != EC_POINT_mul(ecgrp, resultingPoint, NULL, shared_secret, BN_inverse, ctx)) {
292    printf("[-] There was an error during multiplication\n");
293    BN_free(BN_attribute);
294    BN_free(BN_Master);
295    BN_free(BN_order);
296    BN_free(BN_inverse);
297    BN_CTX_free(ctx);
298    EC_POINT_free(resultingPoint);
299    EC_GROUP_free(ecgrp);
300  }
301    finalPoint = EC_POINT_new(ecgrp);
302
303  if (1 != EC_POINT_mul(ecgrp, finalPoint, NULL, resultingPoint, BN_Master, ctx)) {
304    printf("[-] There was an error during multiplication\n");
305    BN_free(BN_attribute);
306    BN_free(BN_Master);
307    BN_free(BN_order);
308    BN_free(BN_inverse);
309    BN_CTX_free(ctx);
310    EC_POINT_free(resultingPoint);
311    EC_POINT_free(finalPoint);
312    EC_GROUP_free(ecgrp);
313
314  }
315 memset(zerobuff,0x00,32);
316 memset(attrbuffer, 0x00, 298);
317 fp = fopen(attrfile, "rb");
318 fread(attrbuffer, 298, 1, fp);
319 fclose(fp);
320 BN_free(BN_attribute);
321 BN_free(BN_inverse);
322 for (int i = 0; i < 9; i++){
323   if (memcmp(&attrbuffer[32*i],zerobuff,32) != 0){
324     printf("1\n");
325
326     BN_attribute= BN_new();
327     BN_attribute = BN_bin2bn(&attrbuffer[32*i], 32, NULL);
328     BN_inverse= BN_new();
329     BN_inverse = BN_mod_inverse(NULL, BN_attribute, BN_order, ctx);
330
331
332     if (1 != EC_POINT_mul(ecgrp, finalPoint, NULL,finalPoint, BN_inverse, ctx)) {
333       printf("[-] There was an error during multiplication\n");
334       BN_free(BN_attribute);
335
336       EC_GROUP_free(ecgrp);
337       EC_POINT_free(finalPoint);
338
339       BN_CTX_free(ctx);
```

```
340         }
341       BN_free(BN_attribute);
342       BN_free(BN_inverse);
343     }
344
345
346   }
347   if(1 != point2bb(key_seed_x, key_seed_y, finalPoint)){
348     printf("[-] There was an error converting point to bb\n");
349     BN_free(BN_attribute);
350     BN_free(BN_Master);
351     BN_free(BN_order);
352     BN_free(BN_inverse);
353     BN_CTX_free(ctx);
354     EC_POINT_free(resultingPoint);
355     EC_POINT_free(finalPoint);
356     EC_GROUP_free(ecgrp);
357   }
358   BN_free(BN_attribute);
359   BN_free(BN_Master);
360   BN_free(BN_order);
361   BN_free(BN_inverse);
362   BN_CTX_free(ctx);
363   EC_POINT_free(resultingPoint);
364   EC_POINT_free(finalPoint);
365   EC_GROUP_free(ecgrp);
366
367   return 0;
368 }
369
370 CreatePrimary_Out create_primary(TSS_CONTEXT *tssContext, TPM2B_PUBLIC public_template,
       TPMI_RH_HIERARCHY Hierarchy){
371
372   CreatePrimary_In in_create;
373   CreatePrimary_Out out_create;
374   struct timespec start, finish, delta;
375
376   TPMI_RH_HIERARCHY primaryHandle = Hierarchy;
377   const char *parentPasswordPtr = NULL;
378   TPMI_SH_AUTH_SESSION  sessionHandle0 = TPM_RS_PW;
379   unsigned int  sessionAttributes0 = 0;
380   TPMI_SH_AUTH_SESSION  sessionHandle1 = TPM_RH_NULL;
381   unsigned int  sessionAttributes1 = 0;
382   TPMI_SH_AUTH_SESSION  sessionHandle2 = TPM_RH_NULL;
383   unsigned int  sessionAttributes2 = 0;
384
385   TPM_RC rc = 0;
386
387   in_create.primaryHandle = primaryHandle;
388   in_create.inSensitive.sensitive.userAuth.t.size = 0;
389   in_create.inSensitive.sensitive.data.t.size = 0;
390
```

```
391   in_create.inPublic = public_template;
392
393   in_create.outsideInfo.t.size = 0;
394   in_create.creationPCR.count = 0;
395
396   clock_gettime(CLOCK_REALTIME, &start);
397   rc = TSS_Execute(tssContext,
398         (RESPONSE_PARAMETERS *)&out_create,
399         (COMMAND_PARAMETERS *)&in_create,
400         NULL,
401         TPM_CC_CreatePrimary,
402         sessionHandle0, parentPasswordPtr, sessionAttributes0,
403         sessionHandle1, NULL, sessionAttributes1,
404         sessionHandle2, NULL, sessionAttributes2,
405         TPM_RH_NULL, NULL, 0);
406   clock_gettime(CLOCK_REALTIME, &finish);
407     sub_timespec(start, finish, &delta);
408     printf("[TIMING]\tCREATE PRIMARY takes:%d.%.9ld\n", (int)delta.tv_sec, delta.
      tv_nsec);
409
410   if(rc != TPM_RC_SUCCESS){
411     handle_TPM_error(rc);
412   }
413
414   return out_create;
415 }
416
417
418 GetRandom_Out GetRandom(TSS_CONTEXT *tssContext){
419     GetRandom_In     in;
420     GetRandom_Out      out;
421     struct timespec start, finish, delta;
422     TPM_RC  rc = 0;
423     uint32_t       bytesRequested = 32;
424     uint32_t       bytesCopied;
425   size_t br;
426
427     int        noZeros = FALSE;
428
429
430     TPMI_SH_AUTH_SESSION       sessionHandle0 = TPM_RH_NULL;
431     unsigned int     sessionAttributes0 = 0;
432     TPMI_SH_AUTH_SESSION       sessionHandle1 = TPM_RH_NULL;
433     unsigned int     sessionAttributes1 = 0;
434     TPMI_SH_AUTH_SESSION       sessionHandle2 = TPM_RH_NULL;
435     unsigned int     sessionAttributes2 = 0;
436
437
438     for (bytesCopied = 0 ; (rc == 0) && (bytesCopied < bytesRequested) ; ) {
439     if (rc == 0) {
440         in.bytesRequested = bytesRequested - bytesCopied;
441     }
```

```
442     if (rc == 0) {
443       clock_gettime(CLOCK_REALTIME, &start);
444         rc = TSS_Execute(tssContext,
445               (RESPONSE_PARAMETERS *)&out,
446               (COMMAND_PARAMETERS *)&in,
447               NULL,
448               TPM_CC_GetRandom,
449               sessionHandle0, NULL, sessionAttributes0,
450               sessionHandle1, NULL, sessionAttributes1,
451               sessionHandle2, NULL, sessionAttributes2,
452               TPM_RH_NULL, NULL, 0);
453         clock_gettime(CLOCK_REALTIME, &finish);
454         sub_timespec(start, finish, &delta);
455         printf("[TIMING]\tGET RANDOM takes:%d.%.9ld\n", (int)delta.tv_sec, delta.
    tv_nsec);
456         if(rc != 0){
457           handle_TPM_error(rc);
458         }
459     }
460     if (rc == TPM_RC_SUCCESS) {
461         for (br = 0 ; (br < out.randomBytes.t.size) && (bytesCopied < bytesRequested) ;
     br++) {
462
463       if (!noZeros || (out.randomBytes.t.buffer[br] != 0)) {
464           bytesCopied++;
465       }
466         }
467     }
468     }
469     return out;
470
471 }
472
473
474
475 CreateLoaded_Out create_loaded(TSS_CONTEXT *ctx, TPM2B_PUBLIC public_template,
    TPM_HANDLE parent, TPM2B_DIGEST seed1, TPM2B_DIGEST seed2){
476
477     CreateLoaded_In      in;
478     CreateLoaded_Out     out;
479     struct timespec start, finish, delta;
480     TPM_RC rc = 0;
481     TPMS_DERIVE InitSeed;
482   const char       *parentPassword = NULL;
483   uint16_t written = 0;
484   uint16_t written_Der = 0;
485   uint32_t size = sizeof(in.inPublic.t.buffer);
486
487   uint8_t *buffer = in.inPublic.t.buffer;
488   uint8_t *MarshaledSeed = in.inSensitive.sensitive.data.t.buffer;
489
490
```

```
491
492     TPMI_SH_AUTH_SESSION sessionHandle0 = TPM_RS_PW;
493   TPMI_SH_AUTH_SESSION sessionHandle1 = TPM_RH_NULL;
494   TPMI_SH_AUTH_SESSION sessionHandle2 = TPM_RH_NULL;
495   unsigned int sessionAttributes0 = 0;
496   unsigned int sessionAttributes1 = 0;
497   unsigned int sessionAttributes2 = 0;
498
499
500     in.parentHandle = parent;
501     InitSeed.label.b.size = 32;
502     InitSeed.context.b.size = 32;
503     memcpy(InitSeed.label.b.buffer, seed1.b.buffer, seed1.b.size);
504     memcpy(InitSeed.context.b.buffer, seed2.b.buffer, seed2.b.size);
505
506     in.inSensitive.sensitive.userAuth.t.size = 0;
507     rc = TSS_TPMS_DERIVE_Marshalu(&InitSeed, &written_Der, &MarshaledSeed, &size);
508
509     in.inSensitive.sensitive.data.t.size = written_Der;
510
511   rc = TSS_TPMT_PUBLIC_D_Marshalu(&public_template.publicArea, &written, &buffer, &size
      );
512   in.inPublic.t.size = written;
513
514   clock_gettime(CLOCK_REALTIME, &start);
515   rc = TSS_Execute(ctx,
516       (RESPONSE_PARAMETERS *)&out,
517       (COMMAND_PARAMETERS *)&in,
518       NULL,
519       TPM_CC_CreateLoaded,
520       sessionHandle0, parentPassword, sessionAttributes0,
521       sessionHandle1, NULL, sessionAttributes1,
522       sessionHandle2, NULL, sessionAttributes2,
523       TPM_RH_NULL, NULL, 0);
524   clock_gettime(CLOCK_REALTIME, &finish);
525     sub_timespec(start, finish, &delta);
526     printf("[TIMING]\tCREATE LOADED takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec
      );
527   if(rc != TPM_RC_SUCCESS){
528     handle_TPM_error(rc);
529   }
530     return out;
531
532 }
533
534 EncryptDecrypt_Out EncryptDecrypt(TSS_CONTEXT *ctx, TPM2B_MAX_BUFFER data, TPM_HANDLE
      KeyHandle, TPM_HANDLE *session,TPMI_YES_NO flag){
535
536     EncryptDecrypt2_In     in2;
537     EncryptDecrypt_Out     out;
538     struct timespec start, finish, delta;
539     TPM_RC rc = 0;
```

```
540     const char       *keyPassword = NULL;
541   TPMI_SH_AUTH_SESSION sessionHandle0 = session == NULL ? TPM_RS_PW : *session;
542   TPMI_SH_AUTH_SESSION sessionHandle1 = TPM_RH_NULL;
543   TPMI_SH_AUTH_SESSION sessionHandle2 = TPM_RH_NULL;
544   unsigned int sessionAttributes0 = session == NULL ? 0 : 1;
545   unsigned int sessionAttributes1 = 0;
546   unsigned int sessionAttributes2 = 0;
547   in2.keyHandle = KeyHandle;
548   in2.decrypt = flag;
549   in2.mode = TPM_ALG_NULL;
550   in2.ivIn.t.size = MAX_SYM_BLOCK_SIZE;
551   memset(in2.ivIn.t.buffer, 0, MAX_SYM_BLOCK_SIZE);
552   in2.inData.t.size = (uint16_t)data.b.size;
553   memcpy(in2.inData.t.buffer, data.b.buffer, data.b.size);
554
555   clock_gettime(CLOCK_REALTIME, &start);
556   rc = TSS_Execute(ctx,
557            (RESPONSE_PARAMETERS *)&out,
558            (COMMAND_PARAMETERS *)&in2,
559            NULL,
560            TPM_CC_EncryptDecrypt2,
561            sessionHandle0, keyPassword, sessionAttributes0,
562            sessionHandle1, NULL, sessionAttributes1,
563            sessionHandle2, NULL, sessionAttributes2,
564            TPM_RH_NULL, NULL, 0);
565   clock_gettime(CLOCK_REALTIME, &finish);
566     sub_timespec(start, finish, &delta);
567     printf("[TIMING]\tENCRYPT DECRYPT takes:%d.%.9ld\n", (int)delta.tv_sec, delta.
      tv_nsec);
568
569   if (rc != TPM_RC_SUCCESS)
570   {
571     handle_TPM_error(rc);
572   }
573   return out;
574 }
575
576
577 HMAC_Out HMAC(TSS_CONTEXT *ctx, TPM2B_MAX_BUFFER data, TPM_HANDLE KeyHandle){
578
579   TPM_RC rc = 0;
580     HMAC_In        in;
581     HMAC_Out       out;
582     struct timespec start, finish, delta;
583
584
585     TPMI_SH_AUTH_SESSION       sessionHandle0 = TPM_RS_PW;
586     unsigned int     sessionAttributes0 = 0;
587     TPMI_SH_AUTH_SESSION       sessionHandle1 = TPM_RH_NULL;
588     unsigned int     sessionAttributes1 = 0;
589     TPMI_SH_AUTH_SESSION       sessionHandle2 = TPM_RH_NULL;
590     unsigned int     sessionAttributes2 = 0;
```

```
591     const char       *keyPassword = NULL;
592 
593     in.handle = KeyHandle;
594     in.hashAlg = TPM_ALG_SHA256;
595     in.buffer.t.size = (uint16_t)data.b.size; /* cast safe, range tested above */
596     memcpy(in.buffer.t.buffer, data.b.buffer, data.b.size);
597 
598     clock_gettime(CLOCK_REALTIME, &start);
599   rc = TSS_Execute(ctx,
600         (RESPONSE_PARAMETERS *)&out,
601         (COMMAND_PARAMETERS *)&in,
602         NULL,
603         TPM_CC_HMAC,
604         sessionHandle0, keyPassword, sessionAttributes0,
605         sessionHandle1, NULL, sessionAttributes1,
606         sessionHandle2, NULL, sessionAttributes2,
607         TPM_RH_NULL, NULL, 0);
608   clock_gettime(CLOCK_REALTIME, &finish);
609     sub_timespec(start, finish, &delta);
610     printf("[TIMING]\tHMAC takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec);
611 
612   if (rc != TPM_RC_SUCCESS)
613   {
614     handle_TPM_error(rc);
615   }
616 
617   return out;
618 }
619 
620 Duplicate_Out Duplicate(TSS_CONTEXT *ctx, TPM_HANDLE KeyHandle, TPM_HANDLE newHandle,
      TPMI_SH_POLICY session, TPM2B_DIGEST SymKey){
621 
622     Duplicate_In     in;
623     Duplicate_Out      out;
624     struct timespec start, finish, delta;
625     TPM_RC rc;
626 
627     TPMI_DH_OBJECT     newParentHandle = newHandle;
628     TPMI_SH_AUTH_SESSION       sessionHandle0 = session;
629     unsigned int     sessionAttributes0 = 0;
630     TPMI_SH_AUTH_SESSION       sessionHandle1 = TPM_RH_NULL;
631     unsigned int     sessionAttributes1 = 0;
632     TPMI_SH_AUTH_SESSION       sessionHandle2 = TPM_RH_NULL;
633     unsigned int     sessionAttributes2 = 0;
634 
635   in.symmetricAlg.algorithm = TPM_ALG_AES;
636     in.symmetricAlg.keyBits.aes = 256;
637     in.symmetricAlg.mode.aes = TPM_ALG_CFB;
638   in.objectHandle = KeyHandle;
639   in.newParentHandle = newParentHandle;
640   memcpy(in.encryptionKeyIn.b.buffer, SymKey.b.buffer, SymKey.b.size);
641   in.encryptionKeyIn.b.size = 32;
```

```
642
643   clock_gettime(CLOCK_REALTIME, &start);
644   rc = TSS_Execute(ctx,
645         (RESPONSE_PARAMETERS *)&out,
646         (COMMAND_PARAMETERS *)&in,
647         NULL,
648         TPM_CC_Duplicate,
649         sessionHandle0, NULL, sessionAttributes0,
650         sessionHandle1, NULL, sessionAttributes1,
651         sessionHandle2, NULL, sessionAttributes2,
652        TPM_RH_NULL, NULL, 0);
653   clock_gettime(CLOCK_REALTIME, &finish);
654     sub_timespec(start, finish, &delta);
655     printf("[TIMING]\tDUBLICATE takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec);
656   if(rc != TPM_RC_SUCCESS){
657     handle_TPM_error(rc);
658   }
659   return out;
660 }
661
662 LoadExternal_Out Load_External(TSS_CONTEXT *ctx, TPM2B_PUBLIC public){
663
664   LoadExternal_In in;
665   LoadExternal_Out out;
666   struct timespec start, finish, delta;
667   TPM_RC rc;
668
669
670   TPMI_RH_HIERARCHY    hierarchy = TPM_RH_ENDORSEMENT;
671
672   TPMI_SH_AUTH_SESSION       sessionHandle0 = TPM_RH_NULL;
673   unsigned int     sessionAttributes0 = 0;
674   TPMI_SH_AUTH_SESSION       sessionHandle1 = TPM_RH_NULL;
675   unsigned int     sessionAttributes1 = 0;
676   TPMI_SH_AUTH_SESSION       sessionHandle2 = TPM_RH_NULL;
677   unsigned int     sessionAttributes2 = 0;
678
679   in.inPrivate.t.size = 0;
680   in.inPublic = public;
681   in.hierarchy = hierarchy;
682
683
684   clock_gettime(CLOCK_REALTIME, &start);
685   rc = TSS_Execute(ctx,
686         (RESPONSE_PARAMETERS *)&out,
687         (COMMAND_PARAMETERS *)&in,
688         NULL,
689         TPM_CC_LoadExternal,
690         sessionHandle0, NULL, sessionAttributes0,
691         sessionHandle1, NULL, sessionAttributes1,
692         sessionHandle2, NULL, sessionAttributes2,
693        TPM_RH_NULL, NULL, 0);
```

```
694   clock_gettime(CLOCK_REALTIME, &finish);
695     sub_timespec(start, finish, &delta);
696     printf("[TIMING]\tLOAD EXTERNAL takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec
      );
697
698   if(rc != TPM_RC_SUCCESS){
699     handle_TPM_error(rc);
700   }
701
702
703   return out;
704 }
705
706
707 Import_Out import(TSS_CONTEXT *ctx, TPMI_DH_OBJECT parentHandle, TPM2B_DIGEST
      encryptionKey, TPM2B_PRIVATE duplicate, TPM2B_PUBLIC objectPublic,
      TPM2B_ENCRYPTED_SECRET SymSeed){
708
709     Import_In         in;
710     Import_Out        out;
711     struct timespec start, finish, delta;
712     TPM_RC rc = 0;
713
714     TPMI_SH_AUTH_SESSION        sessionHandle0 = TPM_RS_PW;
715     unsigned int      sessionAttributes0 = 0;
716     TPMI_SH_AUTH_SESSION        sessionHandle1 = TPM_RH_NULL;
717     unsigned int      sessionAttributes1 = 0;
718     TPMI_SH_AUTH_SESSION        sessionHandle2 = TPM_RH_NULL;
719     unsigned int      sessionAttributes2 = 0;
720
721     in.symmetricAlg.algorithm = TPM_ALG_AES;
722     in.symmetricAlg.keyBits.aes = 256;
723     in.symmetricAlg.mode.aes = TPM_ALG_CFB;
724     in.objectPublic = objectPublic;
725   memcpy(in.encryptionKey.b.buffer, encryptionKey.b.buffer, encryptionKey.b.size);
726   in.encryptionKey.b.size = 32;
727   in.parentHandle = parentHandle;
728   memcpy(in.duplicate.b.buffer, duplicate.b.buffer, duplicate.b.size);
729   in.duplicate.b.size = duplicate.b.size;
730   memcpy(in.inSymSeed.b.buffer, SymSeed.b.buffer, SymSeed.b.size);
731   in.inSymSeed.b.size = SymSeed.b.size;
732
733   clock_gettime(CLOCK_REALTIME, &start);
734   rc = TSS_Execute(ctx,
735         (RESPONSE_PARAMETERS *)&out,
736         (COMMAND_PARAMETERS *)&in,
737         NULL,
738         TPM_CC_Import,
739         sessionHandle0, NULL, sessionAttributes0,
740         sessionHandle1, NULL, sessionAttributes1,
741         sessionHandle2, NULL, sessionAttributes2,
742         TPM_RH_NULL, NULL, 0);
```

```
743   clock_gettime(CLOCK_REALTIME, &finish);
744     sub_timespec(start, finish, &delta);
745     printf("[TIMING]\tIMPORT takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec);
746
747   if (rc != TPM_RC_SUCCESS)
748   {
749     handle_TPM_error(rc);
750   }
751   return out;
752 }
753
754
755 MakeCredential_Out Make_Credentials(TSS_CONTEXT *ctx, TPM_HANDLE KeyHandle, TPM2B_NAME
      Ephimeral_Name, TPM2B_DIGEST Attribute){
756
757     MakeCredential_In      in;
758     MakeCredential_Out     out;
759     struct timespec start, finish, delta;
760     TPM_RC rc;
761
762     TPMI_SH_AUTH_SESSION       sessionHandle0 = TPM_RH_NULL;
763     unsigned int     sessionAttributes0 = 0;
764     TPMI_SH_AUTH_SESSION       sessionHandle1 = TPM_RH_NULL;
765     unsigned int     sessionAttributes1 = 0;
766     TPMI_SH_AUTH_SESSION       sessionHandle2 = TPM_RH_NULL;
767     unsigned int     sessionAttributes2 = 0;
768
769     in.handle = KeyHandle;
770     memcpy(in.credential.b.buffer, Attribute.b.buffer, Attribute.b.size);
771     in.credential.b.size = Attribute.b.size;
772
773     memcpy(in.objectName.b.buffer, Ephimeral_Name.b.buffer, Ephimeral_Name.b.size);
774     in.objectName.b.size = Ephimeral_Name.b.size;
775
776     clock_gettime(CLOCK_REALTIME, &start);
777     rc = TSS_Execute(ctx,
778         (RESPONSE_PARAMETERS *)&out,
779         (COMMAND_PARAMETERS *)&in,
780         NULL,
781         TPM_CC_MakeCredential,
782         sessionHandle0, NULL, sessionAttributes0,
783         sessionHandle1, NULL, sessionAttributes1,
784         sessionHandle2, NULL, sessionAttributes2,
785         TPM_RH_NULL, NULL, 0);
786     clock_gettime(CLOCK_REALTIME, &finish);
787     sub_timespec(start, finish, &delta);
788     printf("[TIMING]\tMAKE CREDENTIAL takes:%d.%.9ld\n", (int)delta.tv_sec, delta.
      tv_nsec);
789
790     if (rc != TPM_RC_SUCCESS)
791     {
792   handle_TPM_error(rc);
```

```
793   }
794
795   return out;
796 }
797
798
799 ActivateCredential_Out Activate_Credential(TSS_CONTEXT *ctx, TPM_HANDLE ActivateHandle,
        TPM_HANDLE KeyHandle, TPM2B_ID_OBJECT CredentialBlob, TPM2B_ENCRYPTED_SECRET
      secret){
800
801     ActivateCredential_In   in;
802     ActivateCredential_Out  out;
803     struct timespec start, finish, delta;
804     TPM_RC rc;
805
806     TPMI_SH_AUTH_SESSION      sessionHandle0 = TPM_RS_PW;
807     unsigned int     sessionAttributes0 = 0;
808     TPMI_SH_AUTH_SESSION      sessionHandle1 = TPM_RS_PW;
809     unsigned int     sessionAttributes1 = 0;
810     TPMI_SH_AUTH_SESSION      sessionHandle2 = TPM_RH_NULL;
811     unsigned int     sessionAttributes2 = 0;
812
813   in.activateHandle = ActivateHandle;
814   in.keyHandle = KeyHandle;
815   memcpy(in.credentialBlob.b.buffer, CredentialBlob.b.buffer, CredentialBlob.b.size);
816   in.credentialBlob.b.size = CredentialBlob.b.size;
817
818   memcpy(in.secret.b.buffer, secret.b.buffer, secret.b.size);
819   in.secret.b.size = secret.b.size;
820
821   clock_gettime(CLOCK_REALTIME, &start);
822   rc = TSS_Execute(ctx,
823         (RESPONSE_PARAMETERS *)&out,
824         (COMMAND_PARAMETERS *)&in,
825         NULL,
826         TPM_CC_ActivateCredential,
827         sessionHandle0, NULL, sessionAttributes0,
828         sessionHandle1, NULL, sessionAttributes1,
829         sessionHandle2, NULL, sessionAttributes2,
830         TPM_RH_NULL, NULL, 0);
831   clock_gettime(CLOCK_REALTIME, &finish);
832     sub_timespec(start, finish, &delta);
833     printf("[TIMING]\tACTIVATE CREDENTIAL takes:%d.%.9ld\n", (int)delta.tv_sec, delta.
      tv_nsec);
834     if (rc != TPM_RC_SUCCESS)
835     {
836   handle_TPM_error(rc);
837   }
838
839   return out;
840 }
841
```

```
842
843
844 TPMT_SIGNATURE sign(TSS_CONTEXT *tssContext, TPM2B_DIGEST digest, TPM_HANDLE keyHandle,
845   const TPM_HANDLE *policySession, TPMT_TK_HASHCHECK *validation) {
846
847   Sign_In in;
848   Sign_Out out;
849   TPM_RC rc;
850     struct timespec start, finish, delta;
851
852
853   TPMI_SH_AUTH_SESSION sessionHandle0 = policySession == NULL ? TPM_RS_PW : *
        policySession;
854   TPMI_SH_AUTH_SESSION sessionHandle1 = TPM_RH_NULL;
855   TPMI_SH_AUTH_SESSION sessionHandle2 = TPM_RH_NULL;
856   unsigned int sessionAttributes0 = policySession == NULL ? 0 : 1;
857   unsigned int sessionAttributes1 = 0;
858   unsigned int sessionAttributes2 = 0;
859
860   in.keyHandle = keyHandle;
861   in.inScheme.scheme = TPM_ALG_ECDSA;
862   in.inScheme.details.ecdsa.hashAlg = TPM_ALG_SHA256;
863   if (validation != NULL)
864   {
865     in.validation = *validation;
866   }
867   else {
868     in.validation.tag = TPM_ST_HASHCHECK;
869     in.validation.hierarchy = TPM_RH_NULL;
870     in.validation.digest.t.size = 0;
871   }
872
873   in.digest = digest;
874     clock_gettime(CLOCK_REALTIME, &start);
875   rc = TSS_Execute(tssContext,
876       (RESPONSE_PARAMETERS*)&out,
877       (COMMAND_PARAMETERS*)&in,
878       NULL,
879       TPM_CC_Sign,
880       sessionHandle0, NULL, sessionAttributes0,
881       sessionHandle1, NULL, sessionAttributes1,
882       sessionHandle2, NULL, sessionAttributes2,
883       TPM_RH_NULL, NULL, 0);
884     clock_gettime(CLOCK_REALTIME, &finish);
885     sub_timespec(start, finish, &delta);
886     printf("[TIMING]\t SIGN takes:%d.%.9ld\n", (int)delta.tv_sec, delta.tv_nsec);
887
888   if(rc != 0){
889     handle_TPM_error(rc);
890   }
891
892   return out.signature;
```

```
893 }
```