Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Προτάσεις ασφάλειας για το πρωτόκολλο MQTT** **Security Enhancements for the MQTT Protocol**
Ονοματεπώνυμο Φοιτητή	**Δημήτριος-Βασίλειος Αγγέλου**
Πατρώνυμο	**Αναστάσιος**
Αριθμός Μητρώου	**ΜΠΣΠ 15098**
Επιβλέπων	**Παναγιώτης Κοτζανικολάου, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Δεκέμβριος 2020**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Παναγιώτης Κοτζανικολάου Αναπληρωτής Καθηγητής	Χρήστος Δουληγέρης Καθηγητής	Δέσποινα Πολέμη Αναπληρώτρια Καθηγήτρια

Περίληψη

Το πρωτόκολλο MQTT χρησιμοποιείται σε εφαρμογές λογισμικού που απαιτείται ασύγχρονη επικοινωνία. Το συγκεκριμένο πρωτόκολλο βασίζεται στο πρότυπο publish/subscribe το οποίο παρέχει αποσύζευξη αποστολέα και παραλήπτη. Οι αποστολείς στέλνουν μηνύματα στον broker ο οποίος τα δρομολογεί στους ενδιαφερόμενους παραλήπτες. Οι παραλήπτες δηλώνουν την επιθυμία τους να λαμβάνουν μηνύματα για κάποιο topic με το να εγγραφούν σε αυτό. Μία από τις προκλήσεις που προκύπτουν είναι η αμοιβαία αυθεντικοποίηση του πελάτη και του broker. Επιπλέον, λόγω της ασύγχρονης φύσης του πρωτοκόλλου είναι αναγκαία η από άκρο σε άκρο κρυπτογράφηση των μηνυμάτων που ανταλλάσσονται μέσω του broker. Ακόμα είναι επιθυμητό το σχήμα κρυπτογράφησης να παρέχει χρονική ασφάλεια (forward secrecy) κάτι το οποίο είναι δύσκολο στην περίπτωση που το άλλο μέρος της επικοινωνίας είναι αποσυνδεδεμένο. Η παρούσα διπλωματική εξετάζει την καταλληλότητα των πρωτοκόλλων PAKE ως μηχανισμό αυθεντικοποίησης για το πρωτόκολλο MQTT. Επιπρόσθετα εξετάζονται σχήματα κρυπτογράφησης από άκρο σε άκρο τα οποία πρέπει να παρέχουν χρονική ασφάλεια. Το προτεινόμενο σχήμα χρησιμοποιεί το πρωτόκολλο SRP για να παρέχει αμοιβαία αυθεντικοποίηση “μεσάζοντα” (broker) και πελάτη με την χρήση κωδικών χωρίς την ανάγκη ύπαρξης υποδομής δημοσίου κλειδιού (PKI). Επιπλέον, συνδυάζει το πρωτόκολλο OTR μαζί με ένα σχήμα κρυπτογράφησης δημοσίου κλειδιού χρονικής ασφάλειας για την κρυπτογράφηση των μηνυμάτων. Τέλος, παρέχεται η υλοποίηση του παραπάνω σχήματος το οποίο χρησιμοποιήθηκε για την κατασκευή μιας ασφαλούς chat εφαρμογής που βασίζεται στο πρωτόκολλο MQTT.

Abstract

The MQTT protocol has been extensively used in software applications where asynchronous communication is needed. The aforementioned protocol follows the publish subscribe pattern which decouples the sender(publisher) from the receiver (subscriber). Publishers send messages to the broker, which is responsible for delivering these messages to the interested subscribers. Message recipients register their interest in a topic by subscribing to this topic. One of the main challenges that this protocol faces is the mutual authentication between the client and the broker. Furthermore, due to the asynchronous nature of the protocol, messages exchanged through the broker should be end-to-end encrypted. A desired property of the encryption scheme is to provide forward secrecy, which is quite challenging when the other party is offline. This master thesis examines the suitability of the PAKE protocols as an authentication mechanism for the MQTT protocol. Moreover, end-to-end encryption schemes are examined with the requirement to provide forward secrecy. The proposed scheme leverages the SRP protocol in order to provide mutual authentication between the client and the broker using a password without the reliance on PKI. Additionally, it combines the OTR protocol with a forward-secure public key encryption scheme for the encryption of the exchanged messages. Finally, the proposed scheme was implemented and it was used to secure the communication of a MQTT based chat application.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα αναπληρωτή καθηγητή Παναγιώτη Κοτζανικολάου για την πολύτιμη βοήθεια του κατά την εκπόνηση της διπλωματικής εργασίας καθώς και τον Δρ. Δημήτρη Γλυνό για την βοήθεια στην επιλογή του θέματος και τις χρήσιμες υποδείξεις του. Επιπλέον, θα ήθελα να ευχαριστήσω τα μέλη της εξεταστικής επιτροπής, καθηγητή Χρήστο Δουληγέρη και αναπληρωτή καθηγητή Δέσποινα Πολέμη για τα εποικοδομητικά σχόλιά τους. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την στήριξη τους κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας.

Η εργασία αυτή υποστηρίχθηκε εν μέρει από το ευρωπαϊκό ερευνητικό έργο CyberSec4Europe (EU H2020-SU-ICT-03-2018, Project No. 830929, cybersec4europe.eu).

Περιεχόμενα

1 Εισαγωγή

Το πρωτόκολλο MQTT ακολουθεί το πρότυπο publish/subscribe το οποίο προσδιορίζει μια μορφή ασύγχρονης επικοινωνίας στην οποία οι αποστολείς (publishers) και οι παραλήπτες(subscribers) ανταλλάσσουν μηνύματα χωρίς να επικοινωνούν απευθείας. Οι αποστολείς στέλνουν τα μηνύματα τους στον broker και ο broker αναλαμβάνει να παραδώσει τα μηνύματα στους παραλήπτες. Οι παραλήπτες εγγράφονται σε ένα topic και μπορεί να μην γνωρίζουν ποιος είναι ο αποστολέας των μηνυμάτων. Επιπλέον μπορεί να μην βρίσκονται όλοι οι παραλήπτες συνδεδεμένοι την ίδια στιγμή και τα μηνύματα να πρέπει να παραδοθούν αργότερα.

Το πρωτόκολλο MQTT χρησιμοποιείται ευρέως σε λύσεις λογισμικού. Το Facebook messenger χρησιμοποιεί το πρωτόκολλο MQTT για την μεταφορά των μηνυμάτων. Επιπλέον χρησιμοποιείται και για την μεταφορά δεδομένων μεταξύ εφαρμογών στον AWS broker καθώς και στον IBM Iot broker. Σε mobile εφαρμογές αρκετές φορές χρησιμοποιείται για την παροχή push notifications.

Όπως παρουσιάστηκε στο συνέδριο DEFCON το MQTT έχει χρησιμοποιηθεί σε πάρα πολλούς τομείς. [1] Αρκετές εφαρμογές το χρησιμοποιούν για να ειδοποιήσουν για την σύνδεση ή αποσύνδεση κάποιας επαφής, για την αποστολή ειδοποιήσεων στον χρήστη σχετικά με φωνητικά μηνύματα που έχει λάβει και την παρακολούθηση της θέσης διαφόρων αντικειμένων και αυτοκινήτων. Ακόμα βρέθηκαν αποτελέσματα τραπεζικών συναλλαγών να μεταφέρονται μέσω MQTT. Σε όλες τις παραπάνω περιπτώσεις τα μηνύματα περιείχαν εμπιστευτικές πληροφορίες όπως αριθμούς τηλεφώνων, ονόματα χρηστών και session ids. Επιπλέον σε πολλές περιπτώσεις ένας μη αυθεντικοποιημένος χρήστης είχε την δυνατότητα να κάνει publish στο σύστημα. Οι εφαρμογές δεν έλεγχαν τα δεδομένα που λάμβαναν μέσω του MQTT με αποτέλεσμα να είναι ευάλωτες σε επιθέσεις τύπου XSS και sql injections.

Μια από τις προκλήσεις που αντιμετωπίζει το πρωτόκολλο είναι η αμοιβαία αυθεντικοποίηση πελάτη και broker καθώς και η κρυπτογράφηση της επικοινωνίας τους. Οι υπάρχουσες λύσεις βασίζονται στην χρήση της υποδομής δημοσίου κλειδιού(PKI). Σε κάποιες περιπτώσεις μπορεί να δυσκολέψει την υλοποίηση καθώς απαιτούνται διαδικασίες για την ανανέωση πιστοποιητικών καθώς και την ανάκληση αυτών σε περίπτωση που κάποιο κλειδί διαρρεύσει.

Ακόμα ένα ζήτημα που πρέπει να αντιμετωπιστεί είναι η κρυπτογράφηση από άκρο σε άκρο (end to end encryption) των μηνυμάτων που ανταλλάσσονται ώστε να εγγυηθούμε την εμπιστευτικότητα των μηνυμάτων στην περίπτωση που ο broker δεν είναι έμπιστος. Αρκετές από τις υπάρχουσες τεχνικές δεν προσδιορίζουν τον τρόπο με τον οποίο θα ανταλλαχθούν τα κλειδιά ή απαιτούν την ύπαρξη μιας έμπιστης αρχής η οποία θα διανέμει τα κλειδιά. Αν η έμπιστη αρχή εκτεθεί, όλο το σύστημα είναι ευάλωτο καθώς αυτή μπορεί να αποκρυπτογραφήσει όλα τα μηνύματα.

Στην περίπτωση του MQTT τα μηνύματα μπορεί να διατηρηθούν από τον broker για αρκετά μεγάλο χρονικό διάστημα λόγω της ασύγχρονης φύσης του πρωτοκόλλου. Το MQTT παρέχει την δυνατότητα να παραδοθούν τα μηνύματα που έχασε ένας χρήστης την επόμενη φορά που αυτός θα συνδεθεί. Επομένως, μια επιθυμητή ιδιότητα του αλγορίθμου κρυπτογράφησης είναι να εγγυάται την εμπιστευτικότητα των προηγούμενων μηνυμάτων ακόμα και αν τα κλειδιά του χρήστη αποκαλυφθούν. Αυτή η ιδιότητα ονομάζεται forward secrecy.

Αν και είναι σχετικά απλό να δημιουργηθεί ένας μηχανισμός που να παρέχει forward secrecy στην περίπτωση σύγχρονων πρωτοκόλλων, είναι πιο δύσκολο να κατασκευασθεί για την περίπτωση ασύγχρονων πρωτοκόλλων όπως το MQTT καθώς οι συμμετέχοντες δεν μπορούν να ανταλλάξουν κλειδιά. Η πιο συνηθισμένη τεχνική είναι η χρήση pre keys η οποία ακολουθείται από το Signal. Ο κάθε πελάτης δημοσιεύει στον εξυπηρετητή εφήμερα κλειδιά τα οποία θα χρησιμοποιηθούν από τον αποστολέα όταν ο παραλήπτης είναι αποσυνδεδεμένος. Ωστόσο η συγκεκριμένη προσέγγιση απαιτεί μεγάλο αποθηκευτικό χώρο.

1.1 Σκοπός διπλωματικής

Σκοπός της παρούσας εργασίας είναι να εξεταστούν τα προβλήματα ασφάλειας που προκύπτουν κατά την χρήση Publish/Subscribe πρωτοκόλλων όπως το MQTT και να προταθούν μηχανισμοί που να επιλύουν τα παραπάνω προβλήματα. Πιο συγκεκριμένα η μελέτη εστιάζει στα παρακάτω:

- Στην εξέταση των απαιτήσεων ασφαλείας Publish/Subscribe πρωτοκόλλων όπως του MQTT.
- Να ερευνηθούν οι ήδη υπάρχουσες τεχνικές για αυθεντικοποίηση καθώς και η καταλληλόλητα αυτών στην περίπτωση mobile εφαρμογών με έμφαση στην ευκολία του deployment. Θα εξετασθούν σε μεγαλύτερο βαθμό τα πρωτόκολλα PAKE καθώς για την εκτέλεση τους απαιτείται από τον χρήστη μόνο η γνώση ενός κωδικού.
- Να εξεταστεί ένας αποδοτικός τρόπος για την από άκρο σε άκρο κρυπτογράφηση των δεδομένων που μεταφέρονται μέσω MQTT. Λόγω της μεγάλης διάρκειας που διατηρούνται τα μηνύματα από τον broker ο μηχανισμός αυτός θα πρέπει να παρέχει forward secrecy. Επιπρόσθετα θα πρέπει να παρέχει forward secrecy ακόμα και στην περίπτωση που κάποια συσκευή είναι αποσυνδεδεμένη.

Για να επιτύχει τους παραπάνω στόχους το προτεινόμενο σχήμα παρέχει αμοιβαία αυθεντικοποίηση χρησιμοποιώντας ένα PAKE αλγόριθμο. Επιπρόσθετα για την κρυπτογράφηση των δεδομένων χρησιμοποιείται ένα ratcheting σχήμα στην περίπτωση που ο παραλήπτης είναι συνδεδεμένος σε συνδυασμό με ένα forward secure σχήμα κρυπτογράφησης δημοσίου κλειδιού για την περίπτωση που ο παραλήπτης δεν είναι συνδεδεμένος.

1.2 Δομή διπλωματικής

Στην δεύτερη ενότητα εξετάζεται το συναφές επιστημονικό έργο. Πιο συγκεκριμένα στην ενότητα 2.1 εξετάζεται το πρότυπο publish/subscribe. Το πρωτόκολλο MQTT το οποίο βασίζεται στην αρχιτεκτονική publish/subscribe αποτελεί το αντικείμενο της ενότητας 2.2. Στην ενότητα 2.3 αναφέρονται τεχνικές που έχουν προταθεί ή χρησιμοποιούνται για το πρωτόκολλο MQTT. Η ενότητα 2.4 περιγράφει την χρήση TLS μαζί με προσυμφωνημένα κλειδιά (pre shared keys). Στην ενότητα 2.5 εξετάζονται τα προβλήματα που αντιμετωπίζουν τα πρωτόκολλα PAKE καθώς και τα γενικότερα χαρακτηριστικά τους. Τέλος αναλύονται γνωστά πρωτόκολλα PAKE και εξετάζεται η ασφάλειά τους. Στην ενότητα 2.6 αναλύονται τεχνικές για την από άκρο σε άκρο κρυπτογράφηση των δεδομένων. Στην ενότητα 3 εξετάζεται η ανάλυση και η σχεδίαση του προτεινόμενου σχήματος. Πιο συγκεκριμένα στην ενότητα 3.1 αναλύονται οι επιθέσεις που μπορούν να πραγματοποιηθούν και εξετάζονται οι απαιτήσεις που πρέπει να ικανοποιεί το προτεινόμενο σχήμα. Η σύγκριση και επιλογή των τεχνικών που θα χρησιμοποιηθούν αποτελεί αντικείμενο της ενότητας 3.2. Η ενότητα 3.3 περιγράφει αναλυτικά το προτεινόμενο σχήμα. Αναφέρεται στην προστασία της επικοινωνίας πελάτη broker καθώς και την κρυπτογράφηση από άκρο σε άκρο των μηνυμάτων. Στην ενότητα 4 περιγράφεται η υλοποίηση του προτεινόμενου σχήματος. Η ενότητα 4.1 περιγράφει την τροποποίηση του Mosquitto broker ώστε να υποστηρίζει το πρωτόκολλο TLS-SRP. Η ενότητα 4.2 περιγράφει την υλοποίηση του πρωτοκόλλου OTR και του forward secure σχήματος για την δημιουργία του MQTT client. Η ενότητα 5 περιέχει ένα τεχνικό εγχειρίδιο στο οποίο παρέχονται οδηγίες για την μεταγλώττιση και την εκτέλεση του προγράμματος. Τέλος στο κεφάλαιο 6 γίνεται επισκόπηση της εργασίας και αναδεικνύονται τα κύρια συμπεράσματα. Επιπλέον αναφέρονται τρόποι βελτίωσης και επέκτασης του σχήματος.

2 Συναφές επιστημονικό έργο

2.1 Πρότυπο publish/subscribe

Όπως αναφέρθηκε προηγουμένως το MQTT [2] βασίζεται στο πρότυπο publish/subscribe. Στο συγκεκριμένο πρότυπο οι αποστολείς των μηνυμάτων (publishers) δεν επικοινωνούν απευθείας με τους παραλήπτες (subscribers). Οι publishers δεν γνωρίζουν για την ύπαρξη subscribers. Επιπλέον δεν προσδιορίζουν ρητά ποιοί είναι οι παραλήπτες των μηνυμάτων καθώς δεν γνωρίζουν την ύπαρξη παραληπτών. Παρόμοια οι subscribers δεν έχουν γνώση για την ύπαρξη κάποιου publisher ή άλλων subscribers και λαμβάνουν μόνο τα μηνύματα στα οποία έχουν εγγραφεί.

Για την μετάδοση των μηνυμάτων υπεύθυνος είναι ο broker ο οποίος είναι γνωστός και από τους publishers και από τους subscribers. Ο broker λαμβάνει τα μηνύματα και τα διαμοιράζει κατάλληλα.

Συνεπώς ο broker είναι αναγκαίο να διαθέτει κάποιο μηχανισμό που θα του επιτρέπει να επιλέγει ποια μηνύματα θα αποστείλει στον κάθε subscriber. Ανάλογα την αρχιτεκτονική του συστήματος αυτός ο μηχανισμός διαφέρει ελαφρώς.

Σε ένα σύστημα βασισμένο σε θέματα (topic-based system) οι publishers κοινοποιούν τα μηνύματα σε κάποια topics. Οι subscribers λαμβάνουν όλα τα μηνύματα που κοινοποιήθηκαν στα topics στα οποία έχουν εγγραφεί. Όλοι οι subscribers που έχουν εγγραφεί στο ίδιο topic θα λάβουν τα ίδια μηνύματα.

Σε ένα σύστημα βασισμένο στο περιεχόμενο (content-based system) τα μηνύματα αποστέλλονται στον subscriber αν το περιεχόμενο ή κάποια χαρακτηριστικά του μηνύματος ταιριάζει με τους περιορισμούς τους οποίους έχει θέσει.

Κάποια συστήματα χρησιμοποιούν συνδυασμό των παραπάνω. Οι publishers κοινοποιούν τα μηνύματα σε topics και οι subscribers εγγράφονται σε ένα ή παραπάνω topics εφαρμόζοντας περιορισμούς και στο περιεχόμενο.

Το πρωτόκολλο MQTT είναι ένα ελαφρύ πρωτόκολλο το οποίο βασίζεται σε μια αρχιτεκτονική publish/subscribe για την ανταλλαγή μηνυμάτων. Δημιουργήθηκε το 1999 από τον Andy Stanford-Clark της IBM και τον Arlen Nipper της Arcom. Σχεδιάστηκε ώστε να είναι κατάλληλο για χρήση από συσκευές με περιορισμένους υπολογιστικούς πόρους καθώς και σε δίκτυα τα οποία παρουσιάζουν μεγάλη καθυστέρηση ή είναι αναξιόπιστα. Ο σκοπός του πρωτοκόλλου είναι να παρέχει έναν κανάλι το οποίο εγγυάται την μετάδοση των δεδομένων και ταυτόχρονα να ελαχιστοποιείται το απαιτούμενο εύρος ζώνης του δικτύου.

Οι συγκεκριμένες αρχές λειτουργίας το καθιστούν ελκυστικό για εφαρμογές “Internet of things” και κινητές συσκευές όπου η κατανάλωση ισχύος και των πόρων του δικτύου πρέπει να είναι μικρή. Χρησιμοποιείται κυρίως σε περιπτώσεις που απαιτείται μετάδοση ασύγχρονων μηνυμάτων από δίκτυα αισθητήρων καθώς και μετάδοση push notifications σε mobile συσκευές.

Πιο συγκεκριμένα το πρωτόκολλο διαθέτει τα παρακάτω χαρακτηριστικά:

- Βασίζεται στην αρχιτεκτονική publish/subscribe με αποτέλεσμα να είναι δυνατή η μετάδοση των μηνυμάτων σε πολλούς παραλήπτες. Επιπλέον δεν υπάρχει η ανάγκη για απευθείας επικοινωνία του αποστολέα του μηνύματος με τον παραλήπτη.
- Χρησιμοποιείται το TCP/IP για την βασική δικτύωση.
- Παρέχει 3 κατηγορίες για να εξασφαλίσει την ποιότητα της επικοινωνίας(quality of service).
    1. «Το πολύ μια μετάδοση (at most once delivery)» : Τα μηνύματα μεταδίδονται σύμφωνα με την δυνατότητες του IP δικτύου. Ο παραλήπτης δεν αποστέλλει καμία επιβεβαίωση και ο αποστολέας δεν επαναλαμβάνει την μετάδοση. Το μήνυμα είναι πιθανό να χαθεί
    2. «Τουλάχιστον μια μετάδοση (at least once delivery)» : Το πρωτόκολλο εγγυάται ότι τα μηνύματα θα φτάσουν στον παραλήπτη αλλά μπορεί να υπάρξουν διπλότυπα.

3. «Ακριβώς μια μετάδοση (exactly once delivery) » : Στην συγκεκριμένη περίπτωση δεν είναι αποδεκτή η απώλεια κάποιου μηνύματος αλλά ούτε και η ύπαρξη διπλότυπων. Ο παραλήπτης θα λάβει το μήνυμα ακριβώς μια φορά. Για παράδειγμα αυτό το επίπεδο μπορεί να χρησιμοποιηθεί σε συστήματα πληρωμών όπου η ύπαρξη διπλότυπων ή η απώλεια μηνυμάτων θα οδηγούσε σε λανθασμένες αλλαγές στο σύστημα.

- Είναι σε θέσει να ανιχνεύσει την μη φυσιολογική αποσύνδεση κάποιας συσκευής και να ειδοποιήσει τις άλλες συσκευές.
- Η ύπαρξη μικρού overhead κατά την μεταφορά. Η σταθερή κεφαλίδα(fixed header) στο πρωτόκολλο είναι μόλις 2 bytes και οι ανταλλαγές του πρωτοκόλλου είναι ελάχιστες ώστε να μειωθεί η κίνηση στο δίκτυο.

2.2 Το πρωτόκολλο MQTT

Θα αναλυθεί η έκδοση 3.1.1 του πρωτοκόλλου [2] καθώς και ο ρόλος του κάθε μέρους καθώς και κάποια ορολογία.

Ο πελάτης είναι αυτός που εγκαθιδρύει την σύνδεση. Μετά από μία επιτυχημένη σύνδεση μπορεί να κοινοποιεί μηνύματα σε κάποιο συγκεκριμένο topic ,να εγγραφεί σε κάποιο topic ώστε να λαμβάνει μηνύματα, να απεγγραφεί από κάποιο topic και τέλος να αποσυνδεθεί από τον broker.

Ο broker δρα ως ενδιάμεσος μεταξύ των publishers και τον subscribers. Δέχεται μηνύματα από τους πελάτες και τα προωθεί ανάλογα με τις εγγραφές των subscribers.

Η εγγραφή (subscription) αποτελείται από ένα φίλτρο θέματος(topic filter) και το μέγιστο QoS που επιθυμεί ο πελάτης. Το φίλτρο θέματος είναι μια έκφραση η οποία προσδιορίζει ένα ή περισσότερα topics. Μπορεί να περιέχει ειδικούς χαρακτήρες. Κάθε εγγραφή αντιστοιχεί με μια συνεδρία(session). Μια συνεδρία μπορεί να περιέχει πολλές εγγραφές. Κάθε εγγραφή σε μία συνεδρία διαθέτει διαφορετικό φίλτρο θέματος. Η συνεδρία στην ουσία διατηρεί τις εγγραφές στα topics καθώς και μηνύματα που δεν έχουν παραδοθεί. Θα αναλυθεί με μεγαλύτερη λεπτομέρεια παρακάτω.

Σε κάθε μήνυμα υπάρχει και το όνομα του θέματος(content name) στο οποίο κοινοποιήθηκε. Το όνομα χρησιμοποιείται για εξακριβωθεί σε ποιόν subscriber πρέπει να σταλεί το μήνυμα. Ο broker θα στείλει το μήνυμα σε κάθε subscriber που η εγγραφή του ταιριάζει με το συγκεκριμένο όνομα θέματος.

Τέλος εκτός από τα πακέτα με το περιεχόμενο του κάθε μηνύματος ανταλλάσσονται και πακέτα ελέγχου (control packets) τα οποία επιτελούν διαφορετικές λειτουργίες του πρωτοκόλλου.

Γενικότερα το πρωτόκολλο MQTT βασίζεται στην στοίβα TCP/IP. Οι συνδέσεις είναι πάντα μεταξύ ενός πελάτη και του broker. Δεν υπάρχει απευθείας σύνδεση μεταξύ των πελατών. Αφού εγκαθιδρυθεί μια σύνδεση tcp μεταξύ του broker και του πελάτη το πρώτο πακέτο που στέλνει ο πελάτης είναι ένα πακέτο CONNECT. Ο broker θα απαντήσει στέλνοντας ένα πακέτο CONNACK και ένα κωδικό κατάστασης (status code). Η συγκεκριμένη σύνδεση θα παραμείνει ανοικτή μέχρι ο πελάτης να αποσυνδεθεί. Μέσω αυτής της σύνδεσης ο πελάτης θα δέχεται ή θα αποστέλλει μηνύματα.

2.2.1 Το πακέτο CONNECT

Όταν ο πελάτης εγκαθιδρύσει μια σύνδεση tcp στον broker το πρώτο πακέτο που πρέπει να στείλει είναι ένα πακέτο CONNECT. Ο πελάτης μπορεί να στείλει το συγκεκριμένο πακέτο μια φορά ανά σύνδεση. Αν ο broker λάβει δεύτερο πακέτο ή το πακέτο είναι λάθος δομημένο πρέπει να τερματίσει την σύνδεση.

Η μεταβλητή κεφαλίδα προσδιορίζει τα πεδία στην ακόλουθη σειρά. Το όνομα του πρωτοκόλλου, την έκδοση του πρωτοκόλλου, κάποιες σημαίες σύνδεσης(connect flags) και Keep Alive. Το όνομα του πρωτοκόλλου είναι μια συμβολοσειρά κωδικοποιημένη κατά UTF-8 και περιέχει την τιμή “MQTT”. Αν το

όνομα δεν είναι σωστό ο broker θα αποσυνδέσει τον πελάτη. Η έκδοση του πρωτοκόλλου περιέχει την έκδοση που χρησιμοποιεί ο πελάτης. Αν δεν υποστηρίζεται ο broker θα απαντήσει με ένα πακέτο CONNACK και μετά θα αποσυνδέσει τον πελάτη. Οι σημαίες σύνδεσης περιέχουν παραμέτρους που προσδιορίζουν την συμπεριφορά της σύνδεσης. Επιπλέον προσδιορίζουν ποια πεδία υπάρχουν στο ωφέλιμο φορτίο.

Bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
byte 8	X	X	X	X	X	X	X	0

Πίνακας 1 Connect Flag bits

Η σημαία clean session περιγράφει πως πρέπει να χειριστεί ο broker την συνεδρία.

Ο πελάτης και ο broker διατηρούν την κατάσταση της συνεδρίας ώστε η εγγυημένη παράδοση ενός μηνύματος να ισχύει και για τις επόμενες συνδέσεις. Αν η τιμή της σημαίας είναι 0 ο broker πρέπει να συνεχίσει την επικοινωνία βασιζόμενος στην προηγούμενη συνεδρία. Αν δεν υπάρχει αποθηκευμένη συνεδρία θα δημιουργηθεί καινούργια. Κατά την αποσύνδεση ο broker και ο πελάτης πρέπει να αποθηκεύσουν την συνεδρία. Αν η τιμή είναι 1 ο broker και ο πελάτης καταστρέφουν όποια προηγούμενη συνεδρία έχει αποθηκευτεί και δημιουργούν καινούργια. Η συνεδρία διατηρείται για όσο διαρκέσει η σύνδεση. Ο broker αντιστοιχεί τις συνεδρίες ανάλογα με το client id.

Αναλυτικότερα την κατάσταση της συνεδρίας για τον πελάτη αποτελούν:

- Τα μηνύματα που στάλθηκαν με QOS 1 και QOS 2 στον broker αλλά δεν έχουν ακόμα επιβεβαιωθεί.
- Μηνύματα που ελήφθησαν από τον broker με QOS 2 και δεν έχουν επιβεβαιωθεί πλήρως.

Για τον broker η κατάσταση της συνεδρίας αποτελείται:

- Ύπαρξη συνεδρίας ακόμα και αν η κατάσταση της συνεδρίας δεν περιέχει τίποτα.
- Οι εγγραφές (subscriptions) του πελάτη.
- Τα μηνύματα που στάλθηκαν με QOS 1 και QOS 2 στον πελάτη αλλά δεν έχουν επιβεβαιωθεί πλήρως.
- Τα μηνύματα με QOS 1 και QOS 2 που αναμένουν να μεταδοθούν στον πελάτη.
- Τα μηνύματα που ελήφθησαν από τον πελάτη με QOS 2 και δεν έχουν ακόμα επιβεβαιωθεί πλήρως.
- Προαιρετικά μηνύματα με QOS 0 που αναμένουν να μεταδοθούν στον πελάτη.

Η σημαία will καθορίζει αν θα σταλεί κάποιο μήνυμα (will message) στους άλλους πελάτες όταν ο κάποιος πελάτης αποσυνδεθεί απρόσμενα. Όταν η τιμή της είναι 1 υποδηλώνει ότι ο broker θα διατηρήσει το will message του πελάτη το οποίο θα συσχετίσει και με την συγκεκριμένη σύνδεση. Το μήνυμα αυτό θα σταλεί στην περίπτωση που η σύνδεση διακοπεί εκτός και αν ληφθεί ένα πακέτο DISCONNECT που δηλώνει ότι η σύνδεση τερματίστηκε κανονικά.

Το μήνυμα θα σταλεί στις παρακάτω περιπτώσεις:

- Ο broker ανιχνεύει κάποια αστοχία στο δίκτυο.
- Ο πελάτης δεν επικοινωνεί μέσα στον χρόνο που υποδηλώνει το πεδίο Keep Alive.
- Ο πελάτης τερματίζει την σύνδεση χωρίς να στείλει ένα πακέτο DISCONNECT.
- Ο broker τερματίζει την σύνδεση λόγω κάποιου σφάλματος στο πρωτόκολλο.

Ο broker αποστέλλει το μήνυμα που προσδιορίζεται στο πεδίο Will Message στο topic που προσδιορίζεται στο Will Topic. Αν η σημαία έχει την τιμή 0 δεν αποστέλλεται τίποτα στην περίπτωση αστοχίας του δικτύου.

Το will QoS προσδιορίζει το QoS που θα σταλεί το will message. Αν η will flag είναι 0 και η συγκεκριμένη σημαία πρέπει να είναι 0. Τέλος η σημαία will retain καθορίζει αν ο broker θα διατηρήσει το μήνυμα μετά την πρώτη αποστολή για να το αποστείλει σε κάποιον που συνδεθεί αργότερα.

Το ωφέλιμο φορτίο (payload) του πακέτου περιέχει ένα μοναδικό id για τον πελάτη, το θέμα στο οποίο θα σταλεί η επιθυμία (will) του πελάτη, το περιεχόμενο της επιθυμίας, το όνομα χρήστη και τον κωδικό. Από αυτά μόνο το id του πελάτη πρέπει να υπάρχει υποχρεωτικά στο πακέτο. Ποια από τα πεδία υπάρχουν προσδιορίζονται στην μεταβλητή κεφαλίδα (variable header). Το client id χρησιμοποιείται από τον broker για τον προσδιορισμό του πελάτη. Το id χρησιμοποιείται ώστε να συσχετίζεται η αποθηκευμένη συνεδρία με τον πελάτη.

Ο broker ακολουθεί τα παρακάτω βήματα πριν εγκαθιδρύσει την σύνδεση.

1. Αν δεν λάβει ένα πακέτο CONNECT μέσα σε ένα λογικό χρονικό περιθώριο ο broker θα κλείσει την σύνδεση.
2. Ο broker ελέγχει αν το πακέτο connect είναι έγκυρο. Αν δεν είναι κλείνει την σύνδεση χωρίς να στείλει ένα πακέτο CONNECT.
3. Μπορεί να κάνει κάποιον επιπλέον έλεγχο στο πακέτο CONNECT ή κάποιον έλεγχο για αυθεντικοποίηση. Αν κάποιος έλεγχος αποτύχει ο broker πρέπει να στείλει ένα πακέτο CONNACK με τον κατάλληλο κωδικό.

Αν τα παραπάνω βήματα είναι επιτυχή ο broker συνεχίζει

1. Αν το client id που δοθεί αναπαριστά ένα ήδη συνδεδεμένο χρήστη τότε ο broker πρέπει να αποσυνδέσει τον ήδη συνδεδεμένο χρήστη.
2. Ελέγχει την τιμή της Clean Session και χειρίζεται την συνεδρία όπως αναφέρθηκε παραπάνω.
3. Ο broker επιβεβαιώνει το πακέτο CONNECT στέλνοντας ένα πακέτο CONNACK με κωδικό 0.

Συνήθως οι πελάτες περιμένουν ένα λάβουν ένα πακέτο CONNACK πριν αρχίσουν να αποστέλλουν άλλα πακέτα. Ωστόσο δεν είναι υποχρεωμένοι να το κάνουν. Αν ο broker απορρίψει το πακέτο CONNECT πρέπει να απορρίψει και όλα τα υπόλοιπα πακέτα που θα έχει στείλει ο πελάτης.

2.2.2 Το πακέτο CONNACK

Όπως έχει αναφερθεί το πακέτο CONNACK αποστέλλεται από τον broker σαν απάντηση στο πακέτο CONNECT. Επομένως είναι το πρώτο πακέτο που θα στείλει ο broker. Αν ο πελάτης δεν λάβει ένα πακέτο CONNACK μέσα σε ένα εύλογο χρονικό διάστημα πρέπει να κλείσει την σύνδεση.

Το σημαντικότερο πεδίο στην μεταβλητή κεφαλίδα είναι το session present. Αν ο broker λάβει μια σύνδεση με Clean Session να έχει την τιμή 1 τότε θέτει το πεδίο session present ίσο με 0 και τον κωδικό επιστροφής στο πακέτο CONNACK ίσο με 0.

Στην περίπτωση που η Clean Session είναι 0 τότε η τιμή του πεδίου session present εξαρτάται από το αν ο broker έχει ήδη μια αποθηκευμένη συνεδρία για το συγκεκριμένο id. Αν υπάρχει αποθηκευμένη συνεδρία η τιμή είναι 1 αλλιώς 0.

Το πεδίο session present δίνει την δυνατότητα στον πελάτη να καταλάβει αν υπάρχει μια αποθηκευμένη συνεδρία την οποία γνωρίζει και ο broker. Στην περίπτωση που η τιμή που λάβει ο πελάτης δεν είναι αυτή που αναμένει τότε μπορεί να συνεχίσει με την υπάρχουσα συνεδρία ή να αποσυνδεθεί. Ο πελάτης μπορεί να ακυρώσει την κατάσταση της συνεδρίας με τον εξής τρόπο: πρέπει να αποσυνδεθεί, να επανασυνδεθεί με το πεδίο Clean Session να είναι 1 και μετά να αποσυνδεθεί ξανά. [2]

Value	Return Code Response	Description
0	0x00 Connection Accepted	Connection accepted
1	0x01 Connection Refused, unacceptable protocol version	The Server does not support the level of the MQTT protocol requested by the Client
2	0x02 Connection Refused, identifier rejected	The Client identifier is correct UTF-8 but not allowed by the Server
3	0x03 Connection Refused, Server unavailable	The Network Connection has been made but the MQTT service is unavailable
4	0x04 Connection Refused, bad user name or password	The data in the user name or password is malformed
5	0x05 Connection Refused, not authorized	The Client is not authorized to connect
6-255		Reserved for future use

Πίνακας 2 Κωδικοί επιστροφής του πακέτου CONNACK

Αν ο broker στείλει το πακέτο CONNACK με οποιονδήποτε κωδικό εκτός του 0 πρέπει να κλείσει την σύνδεση Επιπλέον αν κανένας από τους κωδικούς δεν είναι κατάλληλος για τη περίσταση ο broker πρέπει να κλείσει την σύνδεση χωρίς να στείλει CONNACK. Τέλος το συγκεκριμένο πακέτο δεν έχει ωφέλιμο φορτίο.

2.2.3 Το πακέτο PUBLISH

Το πακέτο PUBLISH αποστέλλεται από τον πελάτη στον broker ή το ανάποδο και μεταφέρει το μήνυμα της εκάστοτε εφαρμογής. Στον πίνακα 3 φαίνεται το περιεχόμενο της σταθερής κεφαλίδας.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

Πίνακας 3 Σταθερή κεφαλίδα του πακέτου PUBLISH

Η τιμή της σημαίας DUP (duplicate) δείχνει αν το συγκεκριμένο πακέτο αποστέλλεται για πρώτη φορά ή αν γίνεται προσπάθεια να ξανασταλεί. Ο πελάτης ή ο broker θέτουν την σημαία ίση με 1 όταν προσπαθούν να στείλουν ξανά ένα πακέτο PUBLISH. Προφανώς στα πακέτα με QoS 0 η σημαία θα πρέπει να είναι πάντα 0 καθώς σε αυτά τα πακέτα δεν γίνεται καμία προσπάθεια να ξανασταλούν. Αξίζει να σημειωθεί ότι η τιμή της σημαίας DUP από ένα εισερχόμενο πακέτο PUBLISH δεν μεταδίδεται όταν ο broker στείλει ένα πακέτο PUBLISH στους subscribers. Η τιμή της σημαίας για ένα εξερχόμενο πακέτο PUBLISH είναι ανεξάρτητη από το εισερχόμενο PUBLISH πακέτο και καθορίζεται από το αν το εξερχόμενο πακέτο αποστέλλεται ξανά.

Αυτό το πεδίο QoS καθορίζει το επίπεδο της εγγυημένης μετάδοσης ενός μηνύματος. Αν ο broker ή ο πελάτης λάβουν πακέτο PUBISH το οποίο έχει και τα 2 bit του QoS 1 θα κλείσουν την σύνδεση.

QoS value	Bit 2	Bit 1	Description
0	0	0	At most once delivery
1	0	1	At least once delivery
2	1	0	Exactly once delivery
-	1	1	Reserved – must not be used

Πίνακας 4 Επίπεδα QoS

Αυτή η σημαία retain καθορίζει αν ο broker θα διατηρήσει το μήνυμα ώστε να σταλεί σε subscribers που θα συνδεθούν αργότερα. Αν η τιμή είναι 1 ο broker πρέπει να διατηρήσει το μήνυμα και το QoS του ώστε να μπορεί να σταλεί σε μελλοντικούς subscribers των οποίων οι εγγραφές ταιριάζουν με το topic του μηνύματος. Όταν πραγματοποιείται μια καινούργια εγγραφή το τελευταίο retain μήνυμα πρέπει να σταλεί στον subscriber.

Όταν ο broker στείλει ένα πακέτο PUBLISH στον πελάτη πρέπει να θέσει την σημαία RETAIN ίση με 1 αν το μήνυμα στέλνεται λόγω ότι η εγγραφή του πελάτη είναι καινούργια. Αντίθετα θα θέσει την σημαία ίση με 0 αν το μήνυμα στέλνεται για μια ήδη υπάρχουσα εγγραφή.

Αν σταλεί ένα πακέτο με την σημαία RETAIN ίση με 1 και με το ωφέλιμο φορτίο να είναι 0 bytes ο broker θα το αποστείλει κανονικά στους subscribers. Επιπλέον οποιοδήποτε retained μήνυμα με το ίδιο topic θα αφαιρεθεί και οι καινούργιοι subscribers δεν θα λάβουν κάποιο retain μήνυμα για αυτό το topic. Άρα ο broker ποτέ δεν διατηρεί ένα μήνυμα 0 bytes ως retained μήνυμα. Τέλος αν η σημαία έχει τιμή 0 ο broker δεν θα αποθηκεύσει αυτό το μήνυμα ούτε θα αντικαταστήσει ένα ήδη retained μήνυμα.

Στην μεταβλητή κεφαλίδα προέχονται τα παρακάτω πεδία: το όνομα του θέματος (topic name) και το προσδιοριστικό πακέτου (packet identifier). Το όνομα του θέματος προσδιορίζει το κανάλι στο οποίο κοινοποιούνται τα δεδομένα. Στο πακέτο PUBLISH το όνομα του θέματος δεν μπορεί να περιέχει ειδικούς χαρακτήρες. Το όνομα του θέματος στο πακέτο που στέλνεται από τον broker πρέπει να ταιριάζει με το topic filter της εγγραφής. Ωστόσο επιτρέπεται στον broker να τροποποιήσει το όνομα του θέματος επομένως μπορεί να μην είναι ίδιο όπως στο αρχικό πακέτο. Το αναγνωριστικό πακέτου υπάρχει μόνο στα πακέτα PUBLISH με QoS 1 ή 2.

Το ωφέλιμο φορτίο περιέχει το μήνυμα που πρέπει να κοινοποιηθεί. Το περιεχόμενο του μηνύματος εξαρτάται από την εφαρμογή. Το μέγεθος του φορτίου μπορεί να υπολογιστεί αφαιρώντας το μέγεθος της μεταβλητής κεφαλίδας από το πεδίο Remaining Length της σταθερής κεφαλίδας. Ένα πακέτο PUBLISH είναι δυνατό να μην περιέχει δεδομένα στο ωφέλιμο φορτίο.

Στον επόμενο πίνακα φαίνεται το πακέτο το οποίο αποστέλλει ο broker ως απάντηση σε ένα PUBLISH πακέτο ανάλογα με το QoS που χρησιμοποιείται.

QoS Level	Expected Response
QoS 0	None
QoS 1	PUBACK Packet
QoS 2	PUBREC Packet

Πίνακας 5 Απάντηση του broker σε ένα πακέτο PUBLISH

Σε περίπτωση που ο πελάτης δεν έχει το δικαίωμα να κάνει publish κάποιο μήνυμα (για παράδειγμα στο topic που επέλεξε να κάνει publish επιτρέπεται μόνο subscribe) ο broker δεν διαθέτει κάποιο τρόπο να τον ενημερώσει. Πρέπει είτε να επιβεβαιώσει το πακέτο PUBLISH σύμφωνα με το QoS που αυτό φέρει είτε να κλείσει την σύνδεση.

2.2.4 Το πακέτο PUBACK

Το συγκεκριμένο πακέτο στέλνεται ως απάντηση σε ένα πακέτο PUBLISH QoS με ίσο με 1. Η μεταβλητή κεφαλίδα περιέχει το αναγνωριστικό του πακέτου PUBLISH που επιβεβαιώνεται. Τέλος το πακέτο PUBACK δεν διαθέτει ωφέλιμο φορτίο.

2.2.5 Το πακέτο PUBREC – Publish received

Το πακέτο PUBREC στέλνεται ως απάντηση σε ένα πακέτο PUBLISH με QoS 2. Είναι το δεύτερο πακέτο που ανταλλάσσεται όταν χρησιμοποιείται QoS 2. Η μεταβλητή κεφαλίδα περιέχει το ίδιο αναγνωριστικό πακέτου με το πακέτο PUBLISH που επιβεβαιώνεται. Παρόμοια το πακέτο PUBREC δεν διαθέτει ωφέλιμο φορτίο.

2.2.6 Το πακέτο PUBREL – Publish release

Το πακέτο PUBREL στέλνεται ως απάντηση σε ένα πακέτο PUBREC. Είναι το τρίτο πακέτο που ανταλλάσσεται για την επιβεβαίωση ενός πακέτου PUBLISH με QoS 2. Η μεταβλητή κεφαλίδα περιέχει το ίδιο αναγνωριστικό πακέτου με το πακέτο PUBREC που επιβεβαιώνεται. Το πακέτο PUBREL δεν διαθέτει ωφέλιμο φορτίο.

2.2.7 Το πακέτο PUBCOMP – Publish complete

Το πακέτο PUBCOMP στέλνεται ως απάντηση σε ένα πακέτο PUBREL. Είναι το τέταρτο και τελευταίο πακέτο που ανταλλάσσεται στην περίπτωση QoS 2. Η μεταβλητή κεφαλίδα περιέχει το ίδιο αναγνωριστικό πακέτου με το πακέτο PUBREL που επιβεβαιώνεται. Το πακέτο PUBCOMP δεν διαθέτει ωφέλιμο φορτίο.

2.2.8 Το πακέτο SUBSCRIBE - Subscribe to topics

Το πακέτο SUBSCRIBE στέλνεται από τον πελάτη στον broker για να δημιουργηθεί μια ή περισσότερες συνδρομές. Με κάθε συνδρομή ο πελάτης εγγράφεται σε ένα ή περισσότερα θέματα (topics). Ο broker στέλνει πακέτα PUBLISH στο πελάτη για τα topics που ταιριάζουν με αυτές τις συνδρομές. Επιπλέον το πακέτο SUBSCRIBE προσδιορίζει το μέγιστο QoS με το οποίο ο broker μπορεί να στείλει μηνύματα στον πελάτη.

Το ωφέλιμο φορτίο του πακέτου περιλαμβάνει την λίστα με τα φίλτρα που προσδιορίζουν σε ποια θέματα θέλει να εγγραφεί ο πελάτης. Ο broker συνήθως υποστηρίζει ειδικούς χαρακτήρες στα φίλτρα. Αν δεν υποστηρίζει ειδικούς χαρακτήρες πρέπει να απορρίψει ένα πακέτο SUBSCRIBE που περιέχει τέτοιους χαρακτήρες. Κάθε φίλτρο ακολουθείται από ένα byte που προσδιορίζει το μέγιστο QoS. Το ωφέλιμο φορτίο του πακέτου πρέπει να περιέχει τουλάχιστον ένα ζεύγος φίλτρο/QoS. Δεν επιτρέπεται πακέτο SUBSCRIBE χωρίς ωφέλιμο φορτίο.

Όταν ο broker λάβει ένα πακέτο SUBSCRIBE πρέπει να απαντήσει με ένα πακέτο SUBACK. Το πακέτο SUBACK πρέπει να έχει το ίδιο αναγνωριστικό πακέτου με το πακέτο που επιβεβαιώνεται. Επιτρέπεται στον broker να στείλει τα πακέτα PUBLISH που αντιστοιχούν στην συνδρομή του πελάτη πριν στείλει το πακέτο SUBACK. Σε περίπτωση που ο broker λάβει SUBSCRIBE που περιέχει φίλτρο που είναι παρόμοιο με κάποιο φίλτρο υπάρχουσας συνδρομής πρέπει να αντικαταστήσει την υπάρχουσα συνδρομή με καινούργια. Το φίλτρο της καινούργιας συνδρομής θα είναι ίδιο με της παλιάς ωστόσο το μέγιστο QoS μπορεί να διαφέρει. Τυχόν retained μηνύματα που υπάρχουν ήδη και ταιριάζουν με το φίλτρο πρέπει να ξανασταλούν. Αν το φίλτρο δεν είναι παρόμοιο τότε δημιουργείται μια καινούργια συνδρομή. Αν λάβει ένα πακέτο SUBSCRIBE που περιέχει πολλά φίλτρα ο broker το χειρίζεται σαν να έλαβε πολλά πακέτα SUBSCRIBE. Ωστόσο θα σταλεί μόνο ένα πακέτο SUBACK για επιβεβαίωση.

Το SUBACK που θα σταλεί από τον broker περιέχει έναν κωδικό επιστροφής για κάθε ζεύγος φίλτρου/QoS. Ο κωδικός επιστροφής δείχνει το μέγιστο QoS που δόθηκε για την συνδρομή ή αν απέτυχε. Εδώ ο broker μπορεί να ειδοποιήσει τον πελάτη ότι δεν επιτρέπεται να εγγραφεί σε κάποιο topic. Επιπλέον το μέγιστο QoS μπορεί να είναι μικρότερο από αυτό που ζήτησε ο πελάτης. Το QoS των μηνυμάτων που στέλνονται για κάποια συνδρομή πρέπει να είναι το ελάχιστο μεταξύ του QoS του αρχικού μηνύματος που έγινε publish και του μέγιστου QoS που δόθηκε από τον broker. Στον broker επιτρέπεται να στείλει διπλότυπα μηνυμάτων στην περίπτωση που το μήνυμα έγινε publish με QoS 1 και το μέγιστο QoS που δόθηκε ήταν 0.

Για παράδειγμα αν ένας πελάτης έχει μια συνδρομή με μέγιστο QoS 1 για ένα θέμα, ένα μήνυμα που θα γίνει publish με QoS 0 θα παραδοθεί με QoS 0 στον πελάτη. Αυτό σημαίνει ότι το πολύ ένα αντίγραφο του μηνύματος θα φτάσει στον πελάτη. Αντίθετα αν γίνει publish ένα μήνυμα με QoS 2 στο ίδιο θέμα, το QoS του υποβαθμίζεται(downgraded) από τον broker ώστε να είναι ίδιο με αυτό του πελάτη. Συνεπώς θα σταλεί με QoS 1 στον πελάτη και μπορεί να υπάρξουν πολλά αντίγραφα του μηνύματος. Αν στον πελάτη έχει δοθεί μέγιστο QoS 0 τότε ένα μήνυμα που γίνεται publish με QoS 2 μπορεί να χαθεί όταν μεταδίδεται στον πελάτη. Ο broker δεν θα ξαναστείλει αυτό το μήνυμα. Ένα μήνυμα που γίνεται publish με QoS 1 στο ίδιο θέμα μπορεί να χαθεί ή να φτάσουν πολλά αντίγραφα του κατά την μετάδοση στον πελάτη.

Από τα παραπάνω προκύπτει ότι όταν ένας πελάτης εγγράφεται με QoS 2 δηλώνει ότι θέλει να λαμβάνει τα μηνύματα με το QoS που στάλθηκαν. Αυτό σημαίνει ότι ο publisher καθορίζει το μέγιστο QoS με το οποίο θα παραδοθεί ένα μήνυμα. Ο subscriber μπορεί να ζητήσει από τον broker να υποβαθμίσει το QoS ώστε να ταιριάζει με αυτό που επιθυμεί.

2.2.9 Το πακέτο SUBACK – Subscribe acknowledgement

Ο broker αποστέλλει ένα πακέτο SUBACK για να επιβεβαιώσει ότι έλαβε ένα πακέτο SUBSCRIBE. Περιέχει τον κωδικό επιστροφής που προσδιορίζει το μέγιστο QoS που δόθηκε για κάθε εγγραφή. Η μεταβλητή κεφαλίδα περιέχει το αναγνωριστικό πακέτου του πακέτου SUBSCRIBE που επιβεβαιώνεται.

Το περιεχόμενο του μηνύματος είναι οι κωδικοί επιστροφής για κάθε φίλτρο θέματος που δόθηκε στο SUBSCRIBE. Η σειρά των κωδικών στο πακέτο SUBACK πρέπει να είναι ίδια με την σειρά των φίλτρων στο πακέτο SUBSCRIBE.

Bit	7	6	5	4	3	2	1	0
	Return Code							
byte 1	X	0	0	0	0	0	X	X

Πίνακας 6 Δομή του περιεχομένου του πακέτου SUBACK

Οι επιτρεπόμενοι κωδικοί είναι οι παρακάτω. Δεν επιτρέπεται να χρησιμοποιηθούν διαφορετικοί κωδικοί από αυτούς.

0x00 - Success - Maximum QoS 0
0x01 - Success - Maximum QoS 1
0x02 - Success - Maximum QoS 2
0x80 - Failure

2.2.10 Το πακέτο UNSUBSCRIBE – Unsubscribe from topics

Το συγκεκριμένο πακέτο στέλνεται από τον πελάτη στον broker όταν επιθυμεί να διαγραφεί από κάποια θέματα. Η μεταβλητή κεφαλίδα περιέχει το αναγνωριστικό πακέτου.

Το ωφέλιμο φορτίο περιέχει τα φίλτρα των θεμάτων από τα οποία ο πελάτης δεν θέλει να λαμβάνει πια μηνύματα. Παρόμοια όπως και στα άλλα πακέτα τα φίλτρα πρέπει να είναι

κωδικοποιημένα με UTF-8. Το πακέτο πρέπει να περιέχει τουλάχιστον ένα φίλτρο. Ένα πακέτο UNSUBSCRIBE χωρίς ωφέλιμο φορτίο αποτελεί παραβίαση του πρωτοκόλλου.

Ο broker ελέγχει τα φίλτρα που έλαβε χαρακτήρα προς χαρακτήρα με τα φίλτρα που έχει για τον συγκεκριμένο πελάτη. Αν οποιοδήποτε φίλτρο ταιριάζει ακριβώς τότε η συνδρομή που αντιστοιχεί σε αυτό διαγράφεται. Σε περίπτωση που ο broker διαγράψει μια συνδρομή:

- Πρέπει να σταματήσει να παραδίδει στον πελάτη καινούργια μηνύματα.
- Πρέπει να περατώσει την αποστολή μηνυμάτων με QoS 1 ή QoS 2 αν έχει ξεκινήσει να τα στέλνει ήδη.
- Αν ο broker επιθυμεί μπορεί να συνεχίσει να στέλνει τα ήδη υπάρχοντα μηνύματα (που βρίσκονται ήδη στον buffer του broker) που προορίζονται για τον πελάτη.

Ο broker πρέπει να απαντήσει στο πακέτο UNSUBSCRIBE στέλνοντας ένα πακέτο UNSUBACK. Το πακέτο UNSUBACK πρέπει να έχει το ίδιο αναγνωριστικό πακέτου με το πακέτο που επιβεβαιώνεται. Ακόμα και στην περίπτωση που δεν διαγραφεί καμία συνδρομή ο broker πρέπει να στείλει UNSUBACK. Παρόμοια όπως στην περίπτωση του SUBSCRIBE αν ο broker λάβει πακέτο UNSUBSCRIBE με πολλά φίλτρα το χειρίζεται σαν να έλαβε πολλά πακέτα UNSUBSCRIBE. Ωστόσο αποστέλλει μόνο ένα πακέτο UNSUBACK.

2.2.11 Το πακέτο UNSUBACK – Unsubscribe acknowledgement

Το πακέτο UNSUBACK στέλνεται από τον broker στον πελάτη για να επιβεβαιωθεί η παραλαβή ενός πακέτου UNSUBSCRIBE. Η μεταβλητή κεφαλίδα περιέχει το αναγνωριστικό πακέτου του πακέτου UNSUBSCRIBE που επιβεβαιώνεται. Το συγκεκριμένο πακέτο δεν περιέχει ωφέλιμο φορτίο.

2.2.12 Το πακέτο PINGREQ – PING request

Το πακέτο PINGREQ στέλνεται από τον πελάτη στον broker. Χρησιμοποιείται για να:

1. Υποδείξει στον broker ότι ο πελάτης είναι συνδεδεμένος. Στέλνεται στην περίπτωση που δεν έχουν σταλεί άλλα πακέτα ελέγχου.
2. Ζητηθεί από τον broker να στείλει ένα πακέτο ώστε να επιβεβαιωθεί ότι παραμένει συνδεδεμένος.

Το πακέτο PINGREQ δεν διαθέτει μεταβλητή κεφαλίδα ούτε ωφέλιμο φορτίο. Όταν ο broker λάβει ένα πακέτο PINGREQ πρέπει να αποκριθεί με ένα πακέτο PINGRESP.

2.2.13 Το πακέτο PINGRESP – PING response

Το πακέτο PINGRESP στέλνεται από τον broker στον πελάτη σαν απάντηση σε ένα πακέτο PINGREQ. Υποδεικνύει στον πελάτη ότι ο broker είναι συνδεδεμένος. Το πακέτο PINGRESP δεν διαθέτει μεταβλητή κεφαλίδα ούτε ωφέλιμο φορτίο.

2.2.14 Το πακέτο DISCONNECT – Disconnect notification

Το πακέτο DISCONNECT είναι το τελευταίο πακέτο που στέλνεται από τον πελάτη. Υποδεικνύει στον broker ότι η σύνδεση τερματίζεται επιτυχώς. Το πακέτο DISCONNECT δεν διαθέτει μεταβλητή κεφαλίδα και ωφέλιμο φορτίο.

Όταν ο πελάτης στείλει το πακέτο DISCONNECT πρέπει:

- Να τερματίσει την σύνδεση.
- Δεν πρέπει να στείλει άλλα πακέτα σε αυτήν την σύνδεση.

Όταν το πακέτο DISCONNECT ληφθεί από τον broker:

- Πρέπει να διαγράψει την τελευταία επιθυμία(last will) για αυτήν την σύνδεση χωρίς να την κοινοποιήσει.
- Πρέπει να κλείσει την σύνδεση αν ο πελάτης δεν την έχει κλείσει ήδη.

2.2.15 Συμπεριφορά του πρωτοκόλλου

Το πρωτόκολλο MQTT είναι συμμετρικό με την έννοια με την έννοια ότι ο πελάτης και ο broker συμπεριφέρονται και ως αποστολέας και ως παραλήπτης. Το πρωτόκολλο ασχολείται με την μεταφορά ενός μηνύματος από έναν αποστολέα σε έναν παραλήπτη. Για την μεταφορά του μηνύματος σε πολλούς παραλήπτες ο broker χειρίζεται κάθε αποστολή ξεχωριστά. Παρακάτω αναλύονται με μεγαλύτερη λεπτομέρεια οι τρόποι υλοποίησης του πρωτοκόλλου.

QoS 0: Το πολύ μια παράδοση. Το μήνυμα αποστέλλεται και ο αποστολλέας δεν περιμένει επιβεβαίωση από τον παραλήπτη ούτε προσπαθεί να το ξαναστείλει. Το μήνυμα θα φτάσει ή θα χαθεί.

Sender Action	Control Packet	Receiver Action
PUBLISH QoS 0, DUP=0		
	→	
		Deliver Application Message to appropriate onward recipient(s)

Πίνακας 7 Ανταλλασσόμενα πακέτα με QoS=0

QoS 1: Τουλάχιστον μια παράδοση. Το συγκεκριμένο QoS εξασφαλίζει ότι το μήνυμα θα φτάσει στον παραλήπτη τουλάχιστον μια φορά. Το πακέτο PUBLISH φέρει αναγνωριστικό πακέτου στην μεταβλητή κεφαλίδα και επιβεβαιώνεται από ένα πακέτο PUBACK.

Ο αποστολέας πρέπει:

- Να εκχωρήσει ένα αχρησιμοποίητο αναγνωριστικό πακέτου για κάθε καινούργιο μήνυμα που στέλνει.
- Πρέπει να στείλει το πακέτο PUBLISH με αυτό το αναγνωριστικό πακέτου με QoS=1 και την σημαία DUP=0.
- Θεωρεί το πακέτο ως μη επιβεβαιωμένο μέχρι να λάβει το αντίστοιχο πακέτο PUBACK από τον παραλήπτη.

Όταν ληφθεί το πακέτο PUBACK τότε το αναγνωριστικό πακέτου μπορεί να ξαναχρησιμοποιηθεί. Επιτρέπεται στον αποστολέα να στείλει επιπλέον πακέτα PUBLISH με διαφορετικά αναγνωριστικά πακέτων όσο περιμένει την παραλαβή των επιβεβαιώσεων.

Ο παραλήπτης πρέπει:

- Να απαντήσει στέλνοντας ένα πακέτο PUBACK το οποίο περιέχει τον αναγνωριστικό πακέτου του πακέτου PUBLISH που επιβεβαιώνεται.
- Από την στιγμή που έχει στείλει το πακέτο PUBACK πρέπει να θεωρήσει κάθε εισερχόμενο πακέτο PUBLISH με το ίδιο αναγνωριστικό ως καινούργια αποστολή ασχέτως τι τιμή περιέχει η σημαία DUP.

Sender Action	Control Packet	Receiver action
Store message		
Send PUBLISH QoS 1, DUP 0, <Packet Identifier>	→	
		Initiate onward delivery of the Application Message
	←	Send PUBACK <Packet Identifier>
Discard message		

Πίνακας 8 Ανταλλασσόμενα πακέτα με QoS=1

Ο παραλήπτης δεν είναι υποχρεωμένος να ολοκληρώσει την αποστολή του μηνύματος (για παράδειγμα όταν πρέπει να σταλεί σε κάποιον subscriber) πριν στείλει το πακέτο PUBACK. Όταν ο αποστολέας έχει λάβει το πακέτο PUBACK τότε το μήνυμα υποτίθεται ότι βρίσκεται στην κατοχή του παραλήπτη.

QoS 2: Ακριβώς μια παραλαβή. Σε αυτό το QoS δεν είναι αποδεκτή η απώλεια του μηνύματος αλλά ούτε η ύπαρξη διπλότυπων. Υπάρχει μεγαλύτερο overhead σε αυτό το QoS από ότι στα άλλα.

Ο παραλήπτης πρέπει:

- Να εκχωρήσει ένα αχρησιμοποίητο αναγνωριστικό πακέτου για κάθε καινούργιο μήνυμα που στέλνει.
- Πρέπει να στείλει το πακέτο PUBLISH με αυτό το αναγνωριστικό πακέτου με QoS=2 και την σημαία DUP=0.
- Θεωρεί το πακέτο PUBLISH ως μη επιβεβαιωμένο μέχρι να λάβει το αντίστοιχο πακέτο PUBEC από τον παραλήπτη.
- Όταν λάβει ένα πακέτο PUBREC από τον παραλήπτη πρέπει να απαντήσει στέλνοντας ένα πακέτο PUBREL το οποίο να περιέχει το αναγνωριστικό πακέτου του PUBLISH πακέτου.
- Θεωρεί το πακέτο PUBREL ως μη επιβεβαιωμένο μέχρι να λάβει το αντίστοιχο πακέτο PUBCOMP από τον παραλήπτη.
- Δεν πρέπει να ξαναστείλει το πακέτο PUBLISH από την στιγμή που λάβει το αντίστοιχο πακέτο PUBREL.

Όταν ληφθεί το πακέτο PUBCOMP το υπάρχων αναγνωριστικό πακέτου μπορεί να ξαναχρησιμοποιηθεί. Επιτρέπεται στον αποστολέα να στείλει πακέτα PUBLISH με διαφορετικό αναγνωριστικό πακέτου όσο περιμένει τις επιβεβαιώσεις.

Ο αποστολέας πρέπει:

- Να απαντήσει στέλνοντας ένα πακέτο PUBREC το οποίο περιέχει τον αναγνωριστικό πακέτου του πακέτου PUBLISH που επιβεβαιώνεται. Το μήνυμα βρίσκεται στην κατοχή του παραλήπτη.
- Μέχρι να λάβει το αντίστοιχο πακέτο PUBREL ο παραλήπτης πρέπει να επιβεβαιώνει ένα ληφθέν PUBLISH πακέτο με το ίδιο αναγνωριστικό πακέτου στέλνοντας ένα PUBREC. Δεν πρέπει σε καμία περίπτωση να στείλει διπλότυπα στους άλλους subscribers.
- Πρέπει να απαντήσει σε ένα πακέτο PUBREL στέλνοντας ένα πακέτο PUBCOMP που θα περιέχει το ίδιο αναγνωριστικό πακέτου με το πακέτο PUBREL.
- Από την στιγμή που έχει στείλει το πακέτο PUBCOMP ο παραλήπτης πρέπει να θεωρήσει κάθε πακέτο PUBLISH με το ίδιο αναγνωριστικό πακέτου ως καινούργια κοινοποίηση μηνύματος.

Sender Action	Control Packet	Receiver Action
Store message		
PUBLISH QoS 2, DUP 0 <Packet Identifier>		
	→	
		Method A, Store message or Method B, Store <Packet Identifier> then Initiate onward delivery of the Application Message
		PUBREC <Packet Identifier>
	←	
Discard message, Store PUBREC received <Packet Identifier>		
PUBREL <Packet Identifier>		
	→	
		Method A, Initiate onward delivery of the Application Message then discard message or Method B, Discard <Packet Identifier>
		Send PUBCOMP <Packet Identifier>
	←	
Discard stored state		

Πίνακας 9 Ανταλλασσόμενα πακέτα με QoS=2

Ο παραλήπτης δεν είναι υποχρεωμένος να τελειώσει την παράδοση του μηνύματος πριν στείλει το πακέτο PUBREC ή το πακέτο PUBCOMP. Όταν ο αρχικός αποστολέας λάβει το πακέτο PUBREC γνωρίζει ότι ο παραλήπτης έχει στην κατοχή του το μήνυμα αλλά δεν ξέρει αν το έχει προωθήσει τους άλλους subscribers. Όπως φαίνεται από το παραπάνω σχήμα υπάρχουν δύο τρόποι με τους οποίους μπορεί να χειριστεί ο παραλήπτης τα μηνύματα με QoS 2. Διαφέρουν ως το σε ποιο σημείο το μήνυμα είναι

διαθέσιμο για αποστολή στους άλλους subscribers. Το ποιος τρόπος θα χρησιμοποιηθεί εξαρτάται από την υλοποίηση του πρωτοκόλλου.

Όταν κάποιος πελάτης συνδεθεί με Clean Session ίση με 0 τότε και ο broker και ο πελάτης πρέπει να ξαναστείλουν όλα τα πακέτα PUBLISH με QoS μεγαλύτερο του 0 καθώς και όλα τα PUBREL πακέτα χρησιμοποιώντας το αρχικό αναγνωριστικό πακέτου. Μόνο στην συγκεκριμένη περίπτωση απαιτείται από τον πελάτη ή τον broker να ξαναστείλουν μηνύματα.

Όταν ο broker λάβει ένα μήνυμα πρέπει να το προσθέσει στην συνεδρία των πελατών που έχουν τις κατάλληλες συνδρομές. Σε κανονικές συνθήκες οι πελάτες λαμβάνουν μηνύματα για τα θέματα τα οποία έχουν εγγραφεί. Ωστόσο υπάρχει περίπτωση να λάβει κάποιο μήνυμα που δεν αντιστοιχεί σε κάποια συνδρομή. Αυτό μπορεί να συμβεί αν ο broker αυτόματα αντιστοιχεί στον πελάτη κάποια συνδρομή. Επιπλέον κάποιος πελάτης μπορεί να λάβει μηνύματα κατά την διάρκεια ενός UNSUBSCRIBE. Ο πελάτης πρέπει να επιβεβαιώσει όλα τα πακέτα PUBLISH (ανάλογα με το QoS τους) ασχέτως αν θα επιλέξει να επεξεργαστεί το μήνυμα που έλαβε.

Για να διατηρηθεί η σειρά των μηνυμάτων ο πελάτης πρέπει να ακολουθεί τους παρακάτω κανόνες κατά την αποστολή των μηνυμάτων:

- Όταν αποστέλλει ξανά κάποια PUBLISH πακέτα πρέπει να τα αποστείλει με την σειρά με την οποία στάλθηκαν αρχικά.
- Πρέπει να αποστείλει τα πακέτα SUBACK με την σειρά που έλαβε τα αντίστοιχα PUBLISH πακέτα (για QoS 1).
- Πρέπει να αποστείλει τα πακέτα PUBREC με την σειρά με την οποία έλαβε τα αντίστοιχα PUBLISH πακέτα (για QoS 2).
- Πρέπει να αποστείλει τα πακέτα PUBREL με την σειρά με την οποία έλαβε τα αντίστοιχα PUBLISH πακέτα (για QoS 2).

Ο broker πρέπει να θεωρεί κάθε θέμα ως διατεταγμένο θέμα (ordered topic). Αν είναι επιθυμητό μπορεί να παρέχει και μη διατεταγμένα θέματα. Όταν ο broker λάβει ένα μήνυμα σε ένα διατεταγμένο θέμα πρέπει να ακολουθήσει τους παραπάνω κανόνες όταν αποστέλλει το μήνυμα στους subscribers. Επιπλέον πρέπει να αποστείλει τα PUBLISH πακέτα (με ίδιο θέμα και QoS) με την σειρά που τα έλαβε.

Οι παραπάνω κανόνες εγγυώνται ότι μηνύματα με QoS 1 θα ληφθούν με την σειρά με την οποία έγιναν publish. Υπάρχει όμως πιθανότητα ένα μήνυμα που αποστέλλεται ξανά να φτάσει πριν από το κανονικό μήνυμα παραβιάζοντας την σειρά.(στο QoS 1 επιτρέπονται διπλότυπα) Για παράδειγμα ένας publisher μπορεί να στείλει τα μηνύματα 1,2,3,4 και ο subscriber να τα λάβει με την σειρά 1,2,3,2,3,4.

Αν ο broker και ο πελάτης αποφασίσουν να στέλνουν μόνο ένα μήνυμα και να μην αποστέλλουν άλλο πριν επιβεβαιωθεί το υπάρχον τότε και τα μηνύματα με QoS 1 θα λαμβάνονται στην σωστή σειρά (για παράδειγμα 1,2,3,3,4 αλλά όχι 1,2,3,2,3,4). Επιπλέον με αυτό τον τρόπο διατηρείται και η σειρά των μηνυμάτων που στάλθηκαν στο ίδιο θέμα με διαφορετικό QoS.

Η σειρά των μηνυμάτων που στέλνονται με QoS 0 ή QoS 2 διατηρείται. Για μηνύματα που στέλνονται με διαφορετικό QoS το καθένα δεν υπάρχει εγγύηση ότι θα φτάσουν με την σειρά που στάλθηκαν.

2.2.16 Ασφάλεια του πρωτοκόλλου

Αρκετές φορές το MQTT χρησιμοποιείται σε περιβάλλοντα τα οποία δεν είναι ασφαλή. Είναι δυνατό τα πακέτα που μεταδίδονται να αναγνωστούν ή να τροποποιηθούν από κάποιον κακόβουλο χρήστη. Επιπλέον όταν ο broker δεν αυθεντικοποιεί τις συνδέσεις είναι εύκολο να γίνει μια επίθεση DoS. Ο επιτιθέμενος μπορεί να συνδεθεί με το id κάποιου πελάτη κάνοντας τον broker να αποσυνδέσει τον νόμιμο πελάτη. Επομένως είναι απαραίτητη η ύπαρξη κάποιου μηχανισμού για:

- Αυθεντικοποίηση(authentication) των χρηστών

- Έλεγχο αν ο χρήστης είναι εξουσιοδοτημένος (authorization) να έχει πρόσβαση στους πόρους του broker.
- Την ακεραιότητα (integrity) των MQTT πακέτων.
- Την εμπιστευτικότητα (confidentiality) των MQTT πακέτων.

Το πρωτόκολλο MQTT ασχολείται μόνο με την μεταφορά των δεδομένων συνεπώς το είδος των μηχανισμών ασφάλειας που χρησιμοποιούνται εξαρτώνται από την υλοποίηση. Συνήθως χρησιμοποιείται TLS.

Ως προς την αυθεντικοποίηση των πελατών το πακέτο CONNECT περιέχει τα πεδία username και password τα οποία μπορούν να χρησιμοποιηθούν για την αυθεντικοποίηση. Εξαρτάται από την υλοποίηση το πως ακριβώς θα χρησιμοποιηθούν αυτά τα πεδία. Για παράδειγμα μπορεί να χρησιμοποιηθεί κάποιος εξωτερικός μηχανισμός αυθεντικοποίησης όπως LDAP. Ωστόσο τα στοιχεία αυτά μεταφέρονται χωρίς κάπου είδους κρυπτογράφηση με αποτέλεσμα κάποιος επιτιθέμενος να μπορεί να τα χρησιμοποιήσει ή να τα καταγράψει. Όταν χρησιμοποιείται TLS η αυθεντικοποίηση μπορεί να γίνει χρησιμοποιώντας πιστοποιητικά (SSL certificates). Ένας άλλος τρόπος είναι η χρήση tls με προσυμφωνημένα κλειδιά (pre-shared keys). Η χρήση TLS-PSK εκτός το ότι αποφεύγει την χρήση PKI που σε κάποιες περιπτώσεις ίσως να μην είναι επιθυμητή, ενδείκνυται και για συσκευές που δεν διαθέτουν μεγάλη επεξεργαστική ισχύ καθώς δεν χρησιμοποιούνται πράξεις δημοσίου κλειδιού που είναι σχετικά δαπανηρές. Τέλος ένας άλλος τρόπος θα ήταν τα στοιχεία της αυθεντικοποίησης να στέλνονται ως μήνυμα(με χρήση MQTT). Ωστόσο αυτός ο τρόπος απαιτεί κάποια προσυμφωνημένη μέθοδο για το τι θα λάβει ο πελάτης από τον broker και πως θα το επαληθεύσει. Επιπλέον δεν παρέχεται κάποια προστασία σε επίπεδο μεταφοράς με αποτέλεσμα ένας επιτιθέμενος να μπορεί να κάνει replay attacks.

Το πρωτόκολλο MQTT δεν παρέχει κάποιο μηχανισμό στον πελάτη για να αυθεντικοποιήσει τον broker. Όταν χρησιμοποιείται TLS ο πελάτης μπορεί να χρησιμοποιήσει το SSL πιστοποιητικό που αποστέλλει ο server για την αυθεντικοποίηση. Στην περίπτωση που χρησιμοποιείται TLS-PSK αν τα δυο μέρη δεν χρησιμοποιούν το ίδιο κλειδί η χειραψία θα αποτύχει. Παρόμοια ένας άλλος τρόπος θα ήταν τα στοιχεία της αυθεντικοποίησης να στέλνονται ως μήνυμα(με χρήση MQTT) από τον server στον πελάτη.

Ο server μπορεί να χρησιμοποιήσει το username, το αναγνωριστικό πελάτη ή το client certificate για να αποφασίσει αν θα επιτρέψει την πρόσβαση σε κάποιον πόρο. Συνήθως αυτό γίνεται με την χρήση access control lists. Οι συγκεκριμένες λίστες προσδιορίζουν σε ποια topics μπορούν να κάνουν subscribe ή publish οι χρήστες. Είναι εμφανές πως για την σωστή λειτουργία του μηχανισμού εξουσιοδότησης θα πρέπει να χρησιμοποιείται ένας εύρωστος μηχανισμός αυθεντικοποίησης ώστε να μπορούμε να είμαστε σίγουροι ποιος είναι ο χρήστης που έχει συνδεθεί. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε τα στοιχεία που περιέχονται στο πακέτο CONNECT για να αυθεντικοποιήσουμε τον χρήστη αλλά με κάποιον τρόπο πρέπει να βεβαιώσουμε ότι και τα υπόλοιπα πακέτα (πχ PUBLISH) έχουν προέλθει από τον συγκεκριμένο χρήστη.

Για να διασφαλιστεί η ακεραιότητα των μηνυμάτων οι εφαρμογές μπορούν να χρησιμοποιήσουν κάποια μορφή ελέγχου ακεραιότητας όπως HMAC. Κάθε μήνυμα θα συνοδεύεται και από ένα hash που προέκυψε από την εφαρμογή κάποιας συνάρτησης HMAC. Ωστόσο έτσι προστατεύεται μόνο το περιεχόμενο των πακέτων PUBLISH και όχι όλων των πακέτων. Στην περίπτωση που χρησιμοποιείται TLS το TLS εγγυάται την ακεραιότητα όλων των δεδομένων που μεταφέρονται.

Παρόμοια κάθε εφαρμογή μπορεί να επιλέξει κρυπτογραφήσει το περιεχόμενο ενός PUBLISH πακέτου. Αυτό αποτρέπει κάποιο επιτιθέμενο να ανακαλύψει το περιεχόμενο του μηνύματος. Έτσι όμως δεν προστατεύονται όλα τα πεδία του πακέτου όπως το όνομα του topic που θα γίνει publish το πακέτο. Αρκετές φορές το όνομα του topic μπορεί να δώσει αρκετή πληροφορία για την φύση της επικοινωνίας. Τέλος δεν μπορούμε να προστατεύσουμε το περιεχόμενο άλλων πακέτων με αυτό τον τρόπο. Ο πιο συνηθισμένος τρόπος να λυθεί αυτό το πρόβλημα είναι η χρήση TLS για την επικοινωνία του πελάτη με τον broker.

2.3 Χρησιμοποιούμενες τεχνικές

Στην συγκεκριμένη ενότητα θα αναλυθούν τεχνικές που έχουν προταθεί για το πρωτόκολλο MQTT.

Στην μελέτη [3] για να γίνει η αποστολή των μηνυμάτων πιο ευέλικτη χωρίς να χρειάζεται ο αποστολέας να γνωρίζει τις ταυτότητες των παραληπτών για την κρυπτογράφηση των μηνυμάτων χρησιμοποιείται ένα υβριδικό σχήμα. Τα δεδομένα κρυπτογραφούνται με τον αλγόριθμο AES και το κλειδί του AES κρυπτογραφείται χρησιμοποιώντας ένα ABE σχήμα. Το ABE σχήμα επιτρέπει να εκφραστεί η πολιτική πρόσβασης των δεδομένων εύκολα και αποδοτικά. Αν και απαιτείται μια έμπιστη αρχή για την διανομή κλειδιών η διαχείριση των κλειδιών είναι ευκολότερη καθώς όταν ένας νέος χρήστης εγγραφεί στο σύστημα δεν χρειάζεται να ανανεωθεί κάποιο κλειδί όπως θα γινόταν στην περίπτωση που χρησιμοποιούνται κοινά κλειδιά.

Στο σχήμα [4] η εμπιστευτικότητα των μηνυμάτων προστατεύεται με την χρήση ενός CP-ABE σχήματος και predicate based encryption(PBE). PBE είναι ένα σχήμα κρυπτογράφησης στο οποίο η κρυπτογράφηση βασίζεται στα χαρακτηριστικά που παρέχει αυτός που κρυπτογραφεί και τα κλειδιά αποκρυπτογράφησης βασίζονται σε κατηγορήματα(predicates). Η αποκρυπτογράφηση είναι επιτυχής αν τα χαρακτηριστικά που προσδιορίστηκαν κατά την κρυπτογράφηση ικανοποιούν τα κατηγορήματα του κλειδιού. Το CP-ABE σχήμα επιτρέπει στον αποστολέα να καθορίσει ποιος μπορεί να δει το μήνυμα χωρίς να ξέρει απαραίτητα ποιος θα το λάβει συγκεκριμένα. Η PBE κρυπτογράφηση επιτρέπει στον εγγεγραμμένο να προσδιορίσει ποια μηνύματα τον ενδιαφέρουν χωρίς το σύστημα να αποκαλύψει τα μεταδεδομένα.

Στο [5]οι συγγραφείς κατασκεύασαν ένα σχήμα για αρχιτεκτονικές publish/subscribe χρησιμοποιώντας KP-ABE, CP-ABE σχήματα. Κάθε εγγραφόμενος ορίζει το φίλτρο των μηνυμάτων ως μια πολιτική πρόσβασης για το σχήμα KP-ABE. Ο broker φιλτράρει τα μηνύματα πραγματοποιώντας κρυπτογραφημένη αναζήτηση (encrypted search) στα κρυπτογραφημένα χαρακτηριστικά και τα προωθεί στους αντίστοιχους χρήστες. Ο αποστολέας κρυπτογραφεί το μήνυμα χρησιμοποιώντας το CP-ABE σχήμα.

Στο [6] χρησιμοποιήθηκε ένα IBE και ένα ABE σχήμα ώστε να εγγυηθεί την εμπιστευτικότητα και αυθεντικότητα των μηνυμάτων. Το σχήμα επιτρέπει στους αποστολείς να υπογράψουν και να κρυπτογραφήσουν μηνύματα χρησιμοποιώντας το IBE σχήμα. Τα μηνύματα δρομολογούνται από τον broker με την χρήση κρυπτογραφημένης αναζήτησης. Τέλος οι παραλήπτες επαληθεύουν τις ψηφιακές υπογραφές των χαρακτηριστικών ενός μηνύματος με την χρήση του CP/KP-ABE σχήματος.

Οι συγγραφείς εισήγαγαν μια τροποποίηση στο πρωτόκολλο MQTT [7]. Προτάθηκε η εισαγωγή ενός νέου PUBLISH μηνύματος που ονομάζεται Spublish. Στην περίπτωση αυτή το περιεχόμενο του μηνύματος κρυπτογραφείται με την χρήση ABE αλγορίθμου [8] χρησιμοποιώντας ελλειπτικές καμπύλες για καλύτερη απόδοση. Αυτού του είδους η κρυπτογράφηση δουλεύει ως εξής: ο αποστολέας κρυπτογραφεί τα δεδομένα σύμφωνα με μια πολιτική πρόσβασης και ο παραλήπτης μπορεί να αποκρυπτογραφήσει τα δεδομένα μόνο αν ικανοποιεί την συγκεκριμένη πολιτική.

Πιο συγκεκριμένα ο broker αναλαμβάνει τον ρόλο ενός PKG (Public Key Generator) ο οποίος δημιουργεί το master secret key και την πολιτική πρόσβασης. Κάθε συσκευή που θέλει να επικοινωνήσει πρέπει να εγγραφεί με μια ταυτότητα και κάποια χαρακτηριστικά (attributes). Ο PKG στέλνει τις δημόσιες παραμέτρους στον αποστολέας και το ιδιωτικό κλειδί για κάθε ιδιότητα στον παραλήπτη. Ο αποστολέας κρυπτογραφεί τα δεδομένα σύμφωνα με τις δημόσιες παραμέτρους και την πολιτική πρόσβασης. Ο παραλήπτης αποκρυπτογραφεί τα δεδομένα χρησιμοποιώντας το κλειδί που του υποδεικνύουν τα χαρακτηριστικά που βρίσκονται στην πολιτική πρόσβασης. Αξίζει να σημειωθεί ότι στην περίπτωση CP-ABE μετά την αρχική φάση εγγραφής και την παραλαβή του κλειδιού δεν απαιτείται η ύπαρξη PKG σε αντίθεση με την χρήση KP-ABE. Η συγκεκριμένη τροποποίηση καταφέρνει να προστατεύσει την εμπιστευτικότητα των δεδομένων από άκρο σε άκρο αν ακόμη αν και ο broker δεν είναι έμπιστος με την προϋπόθεση ότι δεν αναλαμβάνει και τον ρόλο του PKG. Τέλος δεν προσδιορίζεται ο τρόπος με τον οποίο ο PKG θα διανέμει με ασφάλεια τα κλειδιά στις συσκευές.

Στο [9]οι συγγραφείς για να επιτρέψουν την επικοινωνία στην περίπτωση μη έμπιστων brokers εισάγουν μια τροποποίηση στο TLS πρωτόκολλο κάνοντας το να λαμβάνει υπόψη του το MQTT περιεχόμενο που θα μεταφερθεί. Το πρωτόκολλο αυτό ονομάζεται MQTLS. Κατά εγκαθίδρυση της συνόδου με τον broker κάθε publisher και subscriber συμφωνούν σε ένα κλειδί που θα χρησιμοποιηθεί για την κρυπτογράφηση του θέματος, ενώ ο publisher και ο subscriber συμφωνούν σε ένα κλειδί μιας χρήσης (one-time encryption key) και έναν μετρητή για το περιεχόμενο του μηνύματος. Το κλειδί θέματος χρησιμοποιείται για την κρυπτογράφηση του θέματος από τον πελάτη στον broker και το κλειδί μιας χρήσης για την κρυπτογράφηση του κλειδιού του περιεχομένου. Το κρυπτογραφημένο κλειδί περιεχομένου αποστέλλεται στον κάθε subscriber. Έπειτα κάθε φορά που ένας publisher θέλει να στείλει ένα μήνυμα κρυπτογραφεί το περιεχόμενο μαζί με τον μετρητή (χρησιμοποιώντας το κλειδί περιεχομένου) με τον αλγόριθμο AES-GCM.

Ωστόσο δεν παρέχεται forward secrecy καθώς το κλειδί περιεχομένου δεν αλλάζει μετά από κάθε μήνυμα. Επιπλέον δεν είναι δυνατό να σταλεί μήνυμα σε έναν offline χρήστη καθώς το πρωτόκολλο απαιτεί αλληλεπίδραση.

Σε αυτήν την προσέγγιση κινούνται και η περισσότεροι brokers δηλαδή στην χρήση TLS. Ο HiveMQ είναι ένας MQTT broker που παρέχει αρκετές επιλογές για την αυθεντικοποίηση των χρηστών και την κρυπτογράφηση των δεδομένων που μεταφέρονται μέσω του MQTT πρωτοκόλλου. Ως προς την αυθεντικοποίση των χρηστών ο broker επιτρέπει την αυθεντικοποίηση με όνομα χρήστη και κωδικού πρόσβασης. Ωστόσο αυτός ο τρόπος παρέχει ασφάλεια μόνο αν χρησιμοποιείται κάποιος μηχανισμός κρυπτογράφησης κατά την μεταφορά όπως TLS.

Επιπλέον παρέχεται η δυνατότητα χρήσης x509 πιστοποιητικών για τον πελάτη. Σε αυτήν την περίπτωση ο πελάτης αυθεντικοποιεί τον broker ελέγχοντας αν το πιστοποιητικό που αποστέλλει ο broker είναι έμπιστο αλλά και ο broker αυθεντικοποιεί τον πελάτη χρησιμοποιώντας το πιστοποιητικό που απέστειλε ο πελάτης. Αν η αυθεντικοποίηση αποτύχει ο broken ακυρώνει την σύνδεση. Το πλεονέκτημα που παρέχει αυτός ο τρόπος είναι ότι ο broker αυθεντικοποιεί τον πελάτη πριν σταλεί το μήνυμα CONNECT συνεπώς πριν γίνει η εγκαθίδρυση της σύνδεσης. Αυτό μπορεί να βοηθήσει τον broker να εξοικονομήσει πόρους στην περίπτωση που χρησιμοποιεί κάποιον σύνθετο μηχανισμό αυθεντικοποίησης. Για παράδειγμα αν χρησιμοποιεί το όνομα χρήστη που βρίσκεται στο πεδίο CONNECT είναι πιθανό να πρέπει να γίνει κάποιο ερώτημα στην βάση δεδομένων.

Κάποιες άλλες μελέτες εστιάζουν στην αυθεντικοποίηση μεταξύ του broker και της συσκευής. Οι συγγραφείς [10] προτείνουν ένα σχήμα για αυθεντικοποίηση το οποίο δεν χρησιμοποιεί ψηφιακά πιστοποιητικά και βασίζεται στην χρήση κωδικών. Χρησιμοποιείται ένα PAKE πρωτόκολλο το AugPAKE. Σε αυτό το πρωτόκολλο ο χρήστης χρησιμοποιεί έναν κωδικό για να αυθεντικοποιηθεί από το broker ενώ ο broker διατηρεί έναν verifier και όχι τον ίδιο τον κωδικό. Μετά την εκτέλεση του πρωτοκόλλου το κοινό κλειδί χρησιμοποιείται για την κρυπτογράφηση των MQTT μηνυμάτων.

Οι περισσότερες υλοποιήσεις εστιάζουν στην χρήση TLS και PKI για την κρυπτογράφηση επικοινωνίας πελάτη και broker. Η αυθεντικοποίηση του πελάτη γίνεται είτε με πιστοποιητικά είτε με τον κωδικό που βρίσκεται στο πακέτο CONNECT. Ενδιαφέρον παρουσιάζει το σχήμα [10] το οποίο επιτρέπει αμοιβαία αυθεντικοποίηση χωρίς να είναι αναγκαία η χρήση PKI.

Οι τεχνικές [7] [3] [5] [6] βασίζονται σε ABE σχήματα τα οποία σε γενικές γραμμές είναι κατάλληλα για αρχιτεκτονικές publish/subscribe. Παρέχουν την δυνατότητα να κρυπτογραφηθούν τα μηνύματα χωρίς να είναι αναγκαίο αποστολείς και οι παραλήπτες να έχουν ανταλλάξει κλειδιά μεταξύ τους. Αυτό είναι ένα επιθυμητό χαρακτηριστικό καθώς διαφορετικά στην περίπτωση ομαδικής αποστολής μηνυμάτων όταν προστίθεται ένα μέλος θα έπρεπε να ανανεώνεται το κοινό κλειδί. Στην περίπτωση των ABE τα χαρακτηριστικά κλειδιού και του κρυπτοκειμένου καθορίζουν ποιος θα μπορεί να αποκρυπτογραφήσει. Το μειονέκτημα που παρουσιάζουν είναι ότι απαιτείται μια έμπιστη αρχή η οποία θα διανέμει τα κλειδιά. Πρέπει να υπάρχει ένας ασφαλής τρόπος ώστε ο χρήστης να αποδείξει την ταυτότητα του και να παραλάβει το κλειδί του. Τέλος η έμπιστη αρχή είναι σε θέση να αποκρυπτογραφήσει όλα τα μηνύματα καθιστώντας την κρίσιμο σημείο του συστήματος. Το πρόβλημα

αυτό μπορεί να λυθεί χρησιμοποιώντας σχήματα τα οποία επιμερίζουν τα χαρακτηριστικά σε περισσότερες έμπιστες αρχές [11] [12].

2.4 Tls based

2.4.1 Tls-psk

Όπως έχει αναφερθεί μια συνηθισμένη τεχνική είναι η χρήση preshared keys το οποίο παράγεται από τον κωδικό. Την πρώτη φορά που ο χρήστης θα ξεκινήσει την εφαρμογή και θα εισάγει τον κωδικό του η εφαρμογή παράγει ένα κρυπτογραφικό κλειδί με την χρήση κάποιας password key derivation function όπως η PBKDF2. Αν ο κωδικός είναι χαμηλής εντροπίας ο επιτιθέμενος μπορεί να ξεκινήσει μια επίθεση λεξικού χρησιμοποιώντας τα πακέτα της χειραψίας του TLS. Αν χρησιμοποιούνται DHE_PSK ciphersuites τότε ο επιτιθέμενος θα πρέπει να υποδυθεί τον εξυπηρετητή και να εξαπατήσει κάποιον πελάτη να συνδεθεί μαζί του για να αποκτήσει την απαραίτητη πληροφορία. Σε κάθε περίπτωση ο επιτιθέμενος μπορεί να πραγματοποιήσει μια επίθεση λεξικού για να ανακαλύψει τον κωδικό.

Θα εξετάσουμε λεπτομερέστερα πως χρησιμοποιείται ο κάθε αλγόριθμος στην περίπτωση του TLS με pre-shared keys [13]. Στον πίνακα παρουσιάζονται τα βήματα που ακολουθούνται κατά την χειραψία του TLS. Τα στοιχεία που εμφανίζονται στην παρένθεση δεν συμπεριλαμβάνονται στην χειραψία όταν χρησιμοποιείται PSK και το * υποδηλώνει ότι το μήνυμα δεν αποστέλλεται πάντα αλλά σε κάποιες συγκεκριμένες περιπτώσεις.

Client		**Server**
ClientHello	→	
	←	ServerHello (Certificate) ServerKeyExchange* (CertificateRequest) ServerHelloDone
(Certificate) ClientKeyExhcange (CertificateVerify) ChangedCipherSpec Finished	→	
	←	ChangeCipherSpec Finished
Application Data	← →	Application Data

Πίνακας 10 Μηνύματα που ανταλλάσσονται κατά την εκτέλεση του TLS-PSK

Ο πελάτης για να υποδείξει στον εξυπηρετητή ότι επιθυμεί να χρησιμοποιήσει PSK εισάγει κάποιες PSK ciphersuites στον μήνυμα ClientHello. Ο εξυπηρετητής διαλέγει κάποια από αυτές και την τοποθετεί στο μήνυμα ServerHello. Είναι πιθανό ο πελάτης καθώς και ο εξυπηρετητής να διαθέτουν pre-shared keys για επικοινωνία με άλλα μέρη. Για να λυθεί αυτό το πρόβλημα ο πελάτης υποδεικνύει το κλειδί που θα χρησιμοποιήσει εισάγοντας ένα αναγνωριστικό για το κλειδί (PSK identity) στο μήνυμα ClientKeyExchange. Για να βοηθήσει τον πελάτη να επιλέξει το κατάλληλο κλειδί ο εξυπηρετητής εισάγει μια υπόδειξη (PSK identity hint) για το αναγνωριστικό στο μήνυμα ServerKeyExchange. Αν δεν υπάρχει η υπόδειξη το μήνυμα ServerKeyExchange παραλείπεται.

Το premaster secret προκύπτει ως εξής: Έστω ότι το μήκος του PSK είναι N bytes συνενώνονται 2 bytes με τιμή N, N μηδενικά bytes, άλλα 2 bytes με τιμή N και τέλος το PSK. Υπάρχει και η τιμή other_secret αλλά δεν χρησιμοποιείται στον απλό PSK.

Οι ciphersuites DHE_PSK χρησιμοποιούν το PSK για να αυθεντικοποιήσουν μια ανταλλαγή Diffie-Hellman. Ένα πλεονέκτημα που διαθέτουν αυτές οι ciphersuites είναι ότι είναι πιο ανθεκτικές σε επιθέσεις με λεξικό και παρέχουν και Perfect Forward Secrecy. Σε αυτήν την περίπτωση το μήνυμα ServerKeyExchange καθώς και το μήνυμα ClientKeyExchange περιέχουν τις παραμέτρους Diffie-Hellman. Σε αυτήν την περίπτωση το μήνυμα ServerKeyExchange πρέπει να σταλεί ακόμα και αν δεν υπάρχει η υπόδειξη για το PSK. Το premaster secret υπολογίζεται όπως στην περίπτωση του απλού PSK αλλά με κάποιες διαφορές. Πρώτα πραγματοποιείται η ανταλλαγή Diffie-Hellman όπως γίνεται κανονικά στο TLS. Έστω Z η τιμή που προέκυψε από την ανταλλαγή. Έπειτα συνενώνονται 2 bytes που περιέχουν το μήκος του Z(σε bytes), η τιμή Z, άλλα 2 bytes που περιέχουν το μήκος του Z και το PSK.

Τέλος οι ciphersuites RSA_PSK χρησιμοποιούν τον αλγόριθμο RSA και πιστοποιητικά για να αυθεντικοποιήσουν τον εξυπηρετητή μαζί με το PSK. Σε αυτήν την περίπτωση ο εξυπηρετητής στέλνει και ένα μήνυμα Certificate που περιέχει το πιστοποιητικό του. Το μήνυμα ClientKeyExchange που στέλνεται από τον πελάτη εκτός από το αναγνωριστικό για το PSK περιέχει και το EncryptedPreMasterSecret. Το συγκεκριμένο πεδίο περιέχει τον αριθμό έκδοσης των 2 byte και μια τυχαία τιμή των 46 byte κρυπτογραφημένη με το δημόσιο κλειδί του εξυπηρετητή. Το premaster secret δημιουργείται ως εξής: συνενώνονται 2 byte με την τιμή 48, ο αριθμός έκδοσης (2 byte) και η τυχαία τιμή (46 byte), το μέγεθος του PSK και το ίδιο το PSK. Η συγκεκριμένη ciphersuite δεν έχει ευρεία εφαρμογή. Συνήθως χρησιμοποιούνται πιστοποιητικά ή pre-shared keys αλλά σπάνια και τα δυο ταυτόχρονα.

Οι ciphersuites PSK και RSA_PSK δεν παρέχουν perfect forward secrecy. Αυτό σημαίνει ότι αν αποκαλυφθεί το κοινό μυστικό κλειδί στην περίπτωση του απλού PSK ή το κοινό κλειδί και το ιδιωτικό RSA κλειδί του εξυπηρετητή τότε ο επιτιθέμενος μπορεί να αποκρυπτογραφήσει παλαιότερες επικοινωνίες. Οι ciphersuites DHE_PSK παρέχουν perfect forward secrecy με την προϋπόθεση ότι παράγεται ένα καινούργιο κλειδί Diffie-Hellman για κάθε συνεδρία. Επιπλέον αν το κλειδί δεν είναι υψηλής εντροπίας (πχ έχει προκύψει από κάποιον κωδικό) κάποιος επιτιθέμενος μπορεί να πραγματοποιήσει εξαντλητική αναζήτηση ή μια επίθεση με λεξικό για να ανακαλύψει το κλειδί [13].

Στην περίπτωση των PSK ciphersuites ο επιτιθέμενος μπορεί να αποκτήσει την απαραίτητη πληροφορία απλά παρακολουθώντας την χειραψία του TLS. Μπορεί να χρησιμοποιήσει τα μηνύματα Finished για να εξακριβώσει αν το κλειδί είναι σωστό.

Στην περίπτωση της DHE_PSK δεν επαρκεί η απλή παρακολούθηση του καναλιού. Για να αποκτήσει την απαραίτητη πληροφορία ο επιτιθέμενος πρέπει να αναγκάσει κάποιον πελάτη να συνδεθεί μαζί του ώστε να μπορεί να αποκτήσει το κλειδί Z που δημιουργήθηκε από την ανταλλαγή Diffie-Hellman. Έπειτα χρησιμοποιώντας το κλειδί Z και το PSK' (το κλειδί που θέλει να δοκιμάσει) μπορεί να παράγει το premaster secret και από τα μηνύματα Finished να ελέγξει αν είναι σωστό. Στην περίπτωση του RSA_PSK μόνο ο εξυπηρετητής μπορεί να αποκτήσει την απαραίτητη πληροφορία για μια off-line επίθεση.

Το παραπάνω πρόβλημα λύνουν τα πρωτόκολλα PAKE. Μια σημαντική ιδιότητα που διαθέτουν αυτά τα πρωτόκολλα είναι ότι κάποιος επιτιθέμενος (είτε ενεργητικός είτε παθητικός) δεν μπορεί να αποκτήσει αρκετή πληροφορία ώστε να είναι σε θέση να μαντέψει τον κωδικό.

2.5 PAKES

2.5.1 Προβλήματα των πρωτοκόλλων PAKE

Τα πρωτόκολλα που χρησιμοποιούν κωδικούς για την αυθεντικοποίηση των χρηστών καλούνται να αντιμετωπίσουν 3 σημαντικά προβλήματα, τις επιθέσεις λεξικού, το plaintext equivalence του κωδικού και τέλος η παροχή forward secrecy.

Στην συγκεκριμένη περίπτωση επίθεσης με χρήση λεξικού ο επιτιθέμενος προσπαθεί να μαντέψει τον κωδικό χρησιμοποιώντας μια λίστα με τους συχνότερα χρησιμοποιούμενους κωδικούς. Για παράδειγμα ας υποθέσουμε ότι το σύστημα χρησιμοποιεί τον κωδικό του χρήστη για να παράγει ένα κλειδί κρυπτογράφησης (έστω 128 bit) με το οποίο κρυπτογραφεί τα δεδομένα του. Θεωρητικά ο επιτιθέμενος πρέπει να δοκιμάσει 2^128 κλειδιά για να βρει το σωστό κλειδί. Ωστόσο το κλειδί εξαρτάται από τον κωδικό. Σε αυτήν την περίπτωση είναι προτιμότερο να χρησιμοποιηθεί ένα λεξικό με τους συχνά χρησιμοποιούμενους κωδικούς για την δοκιμή των κλειδιών. Ακόμα και αν το λεξικό αποτελείται από εκατομμύρια εγγραφές, η αναζήτηση αυτή θα είναι πολλές φορές γρηγορότερη από την εξαντλητική αναζήτηση.

Παρόμοια η ίδια επίθεση μπορεί να εφαρμοστεί και σε ένα πρωτόκολλο αυθεντικοποίησης. Ένα απλό πρωτόκολλο πρόκλησης-απάντησης (challenge-response) λειτουργεί όπως περιγράφεται παρακάτω.

1. Η Alice συνδέεται στον server.
2. Ο server αποστέλλει στην Alice μια πρόκληση R.
3. Η Alice υπολογίζει την τιμή H(R,P) και την αποστέλλει πίσω στον server. Με P αναπαρίσταται ο κωδικός και με H() μια συνάρτηση κατακερματισμού.
4. Ο server συγκρίνει την τιμή που έλαβε με την τιμή H(R,P) που υπολόγισε χρησιμοποιώντας τον κωδικό που έχει αποθηκευμένο.

Ας υποθέσουμε ότι ένας τρίτος παρακολουθεί και καταγράφει τα μηνύματα που έχουν περάσει πάνω από το δίκτυο. Χρησιμοποιώντας τα μηνύματα μιας επιτυχούς αυθεντικοποίησης μπορεί να εξακριβώσει αν ένας κωδικός P’ είναι σωστός υπολογίζοντας την τιμή H(R,P’) και συγκρίνοντάς την με την τιμή H(R,P). Αν είναι ίδιες έχει βρει τον σωστό κωδικό. Σε πολλές περιπτώσεις ο κωδικός είναι μικρός πχ PIN και ο επιτιθέμενος μπορεί να δοκιμάσει όλους τους πιθανούς συνδυασμούς σε σχετικά μικρό χρονικό διάστημα.

Με τον όρο plaintext-equivalence εννοούμε ότι κάποια δεδομένα παρέχουν την ίδια πρόσβαση που θα παρείχε η γνώση του ίδιου του κωδικού. Τα δεδομένα αυτά είναι ισοδύναμα με τον κωδικό. Ας θεωρήσουμε το πρωτόκολλο που αναφέρθηκε πριν με την διαφορά ότι ο server διατηρεί τον κατακερματισμό(hash) του κωδικού. Συνεπώς στο τρίτο βήμα η Alice υπολογίζει την τιμή H(R,H(P)). Παρόμοια στο τέταρτο βήμα ο server συγκρίνει την τιμή που έλαβε με την τιμή H(R,H(P)) που υπολόγισε χρησιμοποιώντας την αποθηκευμένη τιμή H(P) (τον κατακερματισμό του κωδικού). Το πρωτόκολλο έχει θεωρητικά βελτιωθεί καθώς ο server δεν διατηρεί τον ίδιο τον κωδικό αλλά ένα μετασχηματισμό αυτού. Ωστόσο ο επιτιθέμενος δεν χρειάζεται τον κωδικό καθώς μπορεί να εκμεταλλευτεί το plaintext-equivalence και να χρησιμοποιήσει το hash του κωδικού. Αυτό συμβαίνει γιατί ο υπολογισμός που χρησιμοποιείται για την αυθεντικοποίηση βασίζεται πάντα στο hash του κωδικού και όχι στον ίδιο τον κωδικό. Το ίδιο πρόβλημα παρουσιάζεται και στην αυθεντικοποίηση με τον domain controller με την χρήση NTLM (pass the hash). Από τα παραπάνω προκύπτει ότι ένα πρωτόκολλο θα πρέπει να διαθέτει μηχανισμούς που να το καθιστούν ανθεκτικό ακόμα και στην περίπτωση που το αρχείο με τους κωδικούς έχει διαρρεύσει.

Forward secrecy είναι μια ιδιότητα των πρωτοκόλλων σύμφωνα με την οποία η αποκάλυψη του αρχικού κλειδιού ή κωδικού δεν επιτρέπει την εύρεση των προηγούμενων κλειδιών (session keys). Είναι μια αρκετά επιθυμητή ιδιότητα για τα πρωτόκολλα αυθεντικοποίησης καθώς πολλές φορές

χρησιμοποιούνται για την ανταλλαγή κλειδιών τα οποία χρησιμοποιούνται για κρυπτογράφηση καθώς και αυθεντικοποίηση.

Στο παρακάτω παράδειγμα γίνεται εμφανές πως η έλλειψη της συγκεκριμένης ιδιότητας δίνει την ικανότητα στον επιτιθέμενο να αποκρυπτογραφήσει όλα τα μηνύματα που έχουν ανταλλαχθεί. Το συγκεκριμένο πρωτόκολλο εκτός από την αυθεντικοποίηση πραγματοποιεί και μια ανταλλαγή κλειδιού(key exchange).

1. Η Alice και ο Bob ανταλλάσσουν 2 τυχαίες τιμές, η Alice στέλνει την τιμή R και ο Bob στέλνει την τιμή S.
2. Και οι δυο υπολογίζουν την ποσότητα K= H(R,S,P), όπου P ο κωδικός που γνωρίζουν και οι δυο.
3. Αφού εξακριβώσουν ότι και οι δυο έχουν το ίδιο κλειδί K το χρησιμοποιούν για να κρυπτογραφήσουν τα μηνύματά τους.

Αν ένας επιτιθέμενος καταφέρει να ανακαλύψει τον κωδικό της Alice τότε μπορεί να αποκρυπτογραφήσει όλα τα μηνύματα που στάλθηκαν με την χρήση του κωδικού P. Παρακολουθώντας τα μηνύματα μπορεί να ανακτήσει το R,S και με τον κωδικό που ανακάλυψε να υπολογίσει το κλειδί για την τρέχουσα συνεδρία.

Τέλος το παραπάνω πρωτόκολλο είναι ευάλωτο και σε ένα άλλο είδος επίθεσης. Αν ο επιτιθέμενος ανακαλύψει το κλειδί K κάποιας συνεδρίας τότε μπορεί να ξεκινήσει μια επίθεση με λεξικό δοκιμάζοντας διάφορους κωδικούς P'. Συγκρίνοντας την ποσότητα H(R,S,P') με το κλειδί K μπορεί να εξακριβώσει ποιος είναι ο σωστός κωδικός. Η συγκεκριμένη επίθεση με λεξικό ονομάζεται επίθεση Denning-Sacco [14]. Συνεπώς είναι προφανές ότι η forward secrecy είναι μια απαραίτητη ιδιότητα στα πρωτόκολλα αυθεντικοποίησης με χρήση κωδικών.

2.5.2 Χαρακτηριστικά πρωτοκόλλων PAKE

Η χρήση κωδικών πρόσβασης είναι ο πιο συνηθισμένος τρόπος αυθεντικοποίησης που χρησιμοποιείται λόγω της ευκολίας που παρέχει. Συνήθως δεν έχουν μεγάλη τυχαιότητα καθώς οι χρήστες επιλέγουν κωδικούς που θα μπορούν εύκολα να απομνημονεύσουν. Αυτό καθιστά τους καθιστά ευάλωτους σε επιθέσεις εξαντλητικής αναζήτησης.

Ένα πρωτόκολλο που βασίζεται σε PAKE προσπαθεί να αντιμετωπίσει αυτό το πρόβλημα ανταλλάζοντας δεδομένα με τέτοιο τρόπο ώστε να μην μεταδίδεται ο κωδικός πάνω από το κανάλι. Τα εμπλεκόμενα μέρη αποδεικνύουν την κατοχή του κωδικού χωρίς να τον αποκαλύψουν. Αυτού του είδους η ανταλλαγή είναι ανθεκτική στα dictionaries attacks. Με την λήξη της χειραψίας και τα δύο μέρη έχουν αυθεντικοποιήσει ο ένας τον άλλον και έχουν ανταλλάξει και ένα κλειδί (το οποίο μπορούν και να χρησιμοποιήσουν για να κρυπτογραφήσουν την μετέπειτα επικοινωνία τους). Ένα ενδιαφέρον χαρακτηριστικό των PAKE είναι ότι παρέχουν αμοιβαία αυθεντικοποίηση (mutual authentication) χωρίς να απαιτείται η χρήση υποδομής δημοσίων κλειδιών (PKI).

Τα πρωτόκολλα PAKE χωρίζονται σε δύο κατηγορίες ανάλογα με τον τρόπο που διατηρούν τον κωδικό [15]. Όταν και οι τα δύο μέρη διατηρούν την ίδια αναπαράσταση για τον κωδικό τότε το πρωτόκολλο είναι ισορροπημένη (balanced) PAKE. Στην συγκεκριμένη περίπτωση ο κωδικός μπορεί να διατηρείται με την μορφή hash με κάποιο salt ή και σε plaintext μορφή. Αυτό το είδος PAKE έχει το πλεονέκτημα ότι μπορεί να χρησιμοποιηθεί σε περιπτώσεις που χρειάζεται και τα δυο μέρη να είναι σε θέση να ξεκινήσουν την ανταλλαγή.

Όταν το ένα μέρος διατηρεί ένα μετασχηματισμό του κωδικού και το άλλο μέρος διατηρεί τον κωδικό τότε λέγεται augmented PAKE. Τυπικά ο πελάτης διατηρεί τον κωδικό και ο server διατηρεί έναν verifier που έχει προκύψει από τον κωδικό. Το πλεονέκτημα αυτής της μεθόδου είναι ότι ο server δεν διαθέτει τον κωδικό με αποτέλεσμα αν κάποιος καταφέρει να κλέψει τον verifier δεν θα μπορεί να υποδυθεί κάποιον χρήστη. Θα πρέπει πρώτα να ανακαλύψει τον κωδικό με κάποιου είδους dictionary attack. Αυτό συμβαίνει γιατί ο υπολογισμός που κάνει ο πελάτης για να αποδείξει ότι γνωρίζει τον κωδικό

βασίζεται στον ίδιο τον κωδικό και όχι στον verifier (ή όποια αναπαράσταση χρησιμοποιείται). Αυτό δεν ισχύει στην balanced PAKE.

Τα πρωτόκολλα που βασίζονται σε PAKE βρίσκουν μεγάλη εφαρμογή στο περιβάλλον των κινητών συσκευών. Παρέχουν ευκολία στο deployment της εφαρμογής καθώς δεν απαιτούν την ρύθμιση κάποιας αρχής πιστοποίησης (certificate authority) ή την εγκατάσταση client certificate στο κινητό. Ο χρήστης μπορεί απλώς να εισάγει τον κωδικό του και να εγκαθιδρύσει ένα ασφαλές κανάλι. Επιπλέον μπορεί εύκολά να αλλάξει συσκευή χωρίς να απαιτείται να ρυθμιστεί ξανά.

Όλα τα γνωστά PAKE πρωτόκολλα βασίζονται σε κρυπτογραφία δημοσίου κλειδιού. Ωστόσο μια χαρακτηριστική διαφορά στο κάθε πρωτόκολλο είναι το πως μεταδίδεται το δημόσιο κλειδί [15].

Στην πρώτη κατηγορία PAKE χρησιμοποιείται συμμετρική κρυπτογραφία για να κρυπτογραφηθεί το δημόσιο κλειδί πριν την μετάδοση του. Το συμμετρικό κλειδί προκύπτει από τον κωδικό. Η ικανότητα του άλλου χρήστη να αποκρυπτογραφήσει το δημόσιο κλειδί επιτυχώς αποδεικνύει και την γνώση του κοινού κωδικού. Ένα τέτοιο παράδειγμα αποτελεί το πρωτόκολλο EKE (Encrypted Key Exchange).

Σε μια άλλη κατηγορία PAKE τα δημόσια κλειδιά μεταφέρονται μη κρυπτογραφημένα. Κατά την ανταλλαγή κλειδιών ανταλλάσσονται εφήμερα δημόσια κλειδιά τα οποία έχουν παραχθεί από τον κωδικό. Αν οι κωδικοί που χρησιμοποίησαν οι δύο πλευρές είναι ίδιοι τότε μπορούν να παράγουν ένα κοινό μυστικό συνδυάζοντας τον κωδικό, τα δημόσια κλειδιά και τα ιδιωτικά κλειδιά.

2.5.3 Ασφάλεια των πρωτοκόλλων PAKE

Όλα τα πρωτόκολλα που βασίζονται σε PAKE διαθέτουν εγγενώς έναν περιορισμό. Αν κάποιος επιτιθέμενος μαντέψει τον κωδικό μπορεί να λάβει μέρος στη ανταλλαγή. Επομένως είναι απαραίτητο να υπάρχει κάποιος τρόπος να μοντελοποιηθεί η πιθανότητα κάποιου επιτιθέμενου να μαντέψει τον κωδικό ώστε να μπορούμε να εξάγουμε συμπεράσματα για την ασφάλεια του πρωτοκόλλου. Αν η διαδικασία να μαντέψει τον κωδικό απαιτεί αλληλεπίδραση με τους συμμετέχοντες του πρωτοκόλλου και δεν βασίζεται σε κάποιου είδους υπολογισμό τότε η PAKE είναι ασφαλής. Αυτό σημαίνει ότι κάποιος επιτιθέμενος δεν μπορεί να λάβει κάποιο είδος πληροφορίας παρακολουθώντας το πρωτόκολλο (passive attack). Δεν μπορεί να χρησιμοποιήσει κάποιο λεξικό για να μαντέψει τον κωδικό χωρίς να αλληλοεπιδράσει με τους συμμετέχοντες για κάθε δοκιμή. Συνεπώς κάθε φορά που προσπαθεί να μαντέψει τον κωδικό πρέπει να εκτελεί το πρωτόκολλο. Με κάθε δοκιμή μαθαίνει μόνο αν ο κωδικός που χρησιμοποίησε είναι σωστός ή λάθος.

Η ασφάλεια της PAKE βασίζεται στο γεγονός ότι ο επιτιθέμενος δεν μπορεί να εξακριβώσει αν έχει μαντέψει σωστά τον κωδικό χωρίς να αλληλοεπιδράσει με τους συμμετέχοντες. Επομένως είναι εφικτό να εμποδίσουμε τον επιτιθέμενο μειώνοντας τον αριθμό των επιτρεπόμενων προσπαθειών αυθεντικοποίησης.

Τέλος τα σχήματα PAKE βασίζουν την ασφάλεια τους σε κάποιες υποθέσεις. Συνήθως βασίζονται σε κάποιο δύσκολο πρόβλημα. Για να μπορέσει ο επιτιθέμενος να σπάσει το πρωτόκολλο θα πρέπει να είναι σε θέση να λύσει κάποιο πρόβλημα το οποίο θεωρείται δύσκολο όπως να υπολογίσει τον διακριτό λογάριθμο.

2.5.4 Κατασκευή πρωτοκόλλων αυθεντικοποίησης

Όπως έχει αναφερθεί οι απλές τεχνικές challenge-response είναι ευάλωτες σε επιθέσεις με την χρήση λεξικού. Επιπλέον δεν παρέχουν forward secrecy δίνοντας στον επιτιθέμενο την δυνατότητα να αποκρυπτογραφήσει παλαιότερες συνεδρίες. Παρακάτω θα αναφερθούν κάποιες τεχνικές που λύνουν τα παραπάνω προβλήματα.

Το 1992 παρουσιάστηκε ένα πρωτόκολλο γνωστό ως EKE(Encrypted Key Exchange) [16]. Το πρωτόκολλο αυτό χρησιμοποιεί έναν συνδυασμό συμμετρικής και ασύμμετρης κρυπτογραφίας. Αυτό

έχει ως αποτέλεσμα να είναι ανθεκτικό σε επιθέσεις λεξικού καθώς δεν δίνει επαρκή πληροφορία σε έναν παθητικό επιτιθέμενο ώστε να εξακριβώσει τον κωδικό. Επιπλέον μετά την αυθεντικοποίηση γίνεται μια ανταλλαγή κλειδιού ώστε και οι δυο συμμετέχοντες να μπορούν να κρυπτογραφήσουν την επικοινωνία τους. Στο EKE τα δυο μέρη κρυπτογραφούν εφήμερα δημόσια κλειδιά με ένα συμμετρικό αλγόριθμο χρησιμοποιώντας ως κλειδί τον κοινό κωδικό. Από αυτό το πρωτόκολλο έχουν προκύψει διάφορες οικογένειες πρωτοκόλλων που προσθέτουν επιπλέον ιδιότητες.

Για παράδειγμα το DH-EKE [16] και το SPEKE [17] προσθέτουν την ιδιότητα του forward secrecy. Αυτό σημαίνει ότι αν ο κωδικός αποκαλυφθεί στον επιτιθέμενο αυτός δεν μπορεί να αποκτήσει τα κλειδιά των προηγούμενων συνόδων. Ακόμη αν ένα κλειδί κάποιας συνόδου είναι γνωστό ο επιτιθέμενος δεν μπορεί να το χρησιμοποιήσει για να ανακαλύψει τον κωδικό.

Το σημαντικότερο πρόβλημα που αντιμετωπίζει η συγκεκριμένη οικογένεια πρωτοκόλλων είναι ότι απαιτούν και από τα δυο μέρη να έχουν στην κατοχή τους των κωδικό ή κάποιο hash. Συνεπώς αν η βάση με τους κωδικούς κλαπεί από τον server ο επιτιθέμενος μπορεί να αυθεντικοποιηθεί παριστάνοντας κάποιον χρήστη. Υπάρχει μια τροποποίηση του EKE που ονομάζεται A-EKE (Augmented EKE) η οποία κάνει την συγκεκριμένη οικογένεια πρωτοκόλλων να βασίζεται σε verifier ωστόσο η συγκεκριμένη τροποποίηση καταστρέφει την ιδιότητα του forward secrecy. Τέλος το πρωτόκολλο B-SPEKE [18](είναι Augmented EKE) προσπαθεί να λύσει το πρόβλημα προσθέτοντας μια επιπλέον ανταλλαγή κλειδιού με σκοπό να εξακριβώσει ότι ο πελάτης διαθέτει τον κωδικό. Αυτή η προσθήκη λύνει το πρόβλημα αλλά αυξάνει σημαντικά τον χρόνο εκτέλεσης του πρωτοκόλλου.

2.5.5 J-PAKE

Το πρωτόκολλο J-PAKE [19] ανήκει στην κατηγορία των balanced PAKES. Αυτό σημαίνει ότι και τα δυο μέρη χρησιμοποιούν την ίδια αναπαράσταση για τον κωδικό. Σκοπός του πρωτοκόλλου είναι τα δυο μέρη να παράγουν ένα κοινό κλειδί από ένα μυστικό χαμηλής εντροπίας που στην συγκεκριμένη περίπτωση είναι ο κοινός κωδικός.

Το πρωτόκολλο λειτουργεί ως εξής: Ας θεωρήσουμε ως G μια υποομάδα του Z^* , με τάξη q και g ένας γεννήτορας της G. Ως s αναπαριστούμε τον κοινό κωδικό. Ανάλογα την εφαρμογή μπορεί να χρησιμοποιείται το hash του κωδικού με κάποιο salt. Επιπλέον γίνεται η υπόθεση ότι ο κωδικός βρίσκεται στο διάστημα $[1, q-1]$.

Η Alice επιλέγει τυχαία δυο τιμές x1,x2 με $x1 \in [0, q-1]$ και $x2 \in [1, q-1]$. Παρόμοια ο Bob επιλέγει x3,x4 με $x3 \in [0, q-1]$ και $x4 \in [1, q-1]$.

Έπειτα η Alice αποστέλλει τις τιμές g^{x1} και g^{x2} καθώς και κάποια απόδειξη ότι γνωρίζει τις τιμές x1,x2. Παρόμοια ο Bob στέλνει τις τιμές g^{x3} και g^{x4} και μια απόδειξη ότι γνωρίζει τις τιμές x3,x4. Η παραπάνω διαδικασία μπορεί να γίνει σε ένα βήμα καθώς κανένας υπολογισμός δεν εξαρτάται από κάποια τιμή του άλλου μέλους.

Για την δημιουργία των αποδείξεων οι συγγραφείς προτείνουν την χρήση της υπογραφής Schnorr (Schnorr's signature). Ο συγκεκριμένος τρόπος δεν απαιτεί αλληλεπίδραση μεταξύ των δυο μερών και επιτρέπει στο κάθε μέρος να αποδείξει ότι γνωρίζει τον διακριτό λογάριθμο. Για να αποδείξει κάποιος ότι γνωρίζει τον εκθέτη για την τιμή $X = g^x$ στέλνει τις τιμές {SignerID,V=g^u,r=u-xh} με SignerID να είναι το αναγνωριστικό χρήστη, $u \in Z$ και h=H(g,V,X,SignerID). Ο παραλήπτης ελέγχει αν το X είναι στοιχείο της υποομάδας τάξης q και αν το g^u ισούται με $g^r X^h$. Ο λόγος για τον οποίο η τιμή SignerID χρησιμοποιείται στον υπολογισμό του hash h είναι για να μην είναι εφικτό στην Alice να απαντήσει χρησιμοποιώντας την υπογραφή του Bob και το αντίστροφο.

Στο επόμενο βήμα η Alice αποστέλλει στην τιμή $A = g^{(x1+x3+x4)x2 \cdot s}$ και μια απόδειξη ότι γνωρίζει την τιμή $x2 \cdot s$. Παρόμοια ο Bob αποστέλλει την τιμή $B = g^{(x1+x2+x3) \cdot x4 \cdot s}$και μια απόδειξη ότι γνωρίζει την τιμή$x4 \cdot s$.Έπειτα η Alice υπολογίζει την τιμή $K = (B/g^{x2 \cdot x4 \cdot s})^{x2} = g^{(x1+x3) \cdot x2 \cdot x4 \cdot s}$ και ο Bob ως $K =$

$(A/g^{x2\cdot x4\cdot s})^{x4} = g^{(x1+x3)\cdot x2\cdot x4\cdot s}$. Αν και οι δύο χρησιμοποιούν τον σωστό κωδικό s θα καταλήξουν στην ίδια τιμή K. Έπειτα το κλειδί συνόδου μπορεί να παραχθεί ως k=H(K).

Το πρωτόκολλο ικανοποιεί όλες τις απαιτήσεις που πρέπει να ικανοποιούν τα πρωτόκολλα PAKE. Πιο συγκεκριμένα έχει τις παρακάτω ιδιότητες:

- Είναι ανθεκτικό σε offline επιθέσεις με λεξικό. Η τιμή xa= x1+x2+x3 είναι τυχαία και δεν μπορεί να υπολογιστεί από τον επιτιθέμενο. Επιπλέον σύμφωνα με την υπόθεση DDH (Decision Diffie-Hellman) ένας επιτιθέμενος ή ο Bob δεν μπορεί να ξεχωρίσει την τιμή A=…. από ένα τυχαίο στοιχείο της ομάδας G. Για αυτό τον λόγο δεν μπορεί και να συσχετίσει την τιμή A με την τιμή B ώστε να εξάγει κάποια πληροφορία. Συνεπώς δεν αποκαλύπτεται πληροφορία για τον κωδικό στον επιτιθέμενο.
- Forward secrecy. Ακόμα και αν ο κωδικός αποκαλυφθεί δεν είναι δυνατό να υπολογιστούν τα προηγούμενα κλειδιά συνόδου. Για να υπολογίσει τα κλειδιά συνόδου ο επιτιθέμενος θα πρέπει να υπολογίσει την τιμήg^{x^2} χρησιμοποιώντας την τιμέςg, g^x. Αυτό σημαίνει ότι θα πρέπει να λύσει το πρόβλημα SCDH (Square Computational Diffie-Hellman) το οποίο είναι υπολογιστικά αδύνατο. Επιπλέον αν αποκαλυφθεί κάποιο κλειδί συνόδου δεν δίνεται η δυνατότητα στον επιτιθέμενο να υπολογίσει άλλα κλειδιά συνόδου καθώς η παραγωγή του κλειδιού εξαρτάται από εφήμερες τιμές που προέρχονται και από τα δυο μέλη.
- Ανθεκτικότητα σε on-line dictionary επιθέσεις. Στην συγκεκριμένη περίπτωση ένας επιτιθέμενος εκτελεί κανονικά τα βήματα του πρωτόκολλου χρησιμοποιώντας κάποιον κωδικό. Σε αυτή την περίπτωση αν ο κωδικός είναι λάθος δεν μπορεί να παράγει το σωστό κλειδί. Κατά την διαδικασία εξακρίβωσης του κλειδιού ο επιτιθέμενος θα διαπιστώσει ότι το κλειδί δεν είναι ίδιο άρα ο κωδικός που χρησιμοποιήθηκε είναι λανθασμένος. Το πρωτόκολλο δεν παρέχει κάποια άλλη πληροφορία για τον κωδικό στον επιτιθέμενο.

2.5.6 AMP

Οι στόχοι του πρωτοκόλλου [20] είναι να προστατευτεί η μεταφορά του κωδικού μεταξύ των δύο μελών καθώς και το αρχείο με του κωδικούς που διατηρεί ο εξυπηρετητής. Η βασική ιδέα είναι η «ενίσχυση» του κωδικού(ο οποίος έχει χαμηλή εντροπία) με μια πηγή υψηλής εντροπίας με σκοπό να αποφευχθούν οι επιθέσεις λεξικού.

Η μέθοδος που χρησιμοποιείται για την ενίσχυση (amplification) του κωδικού είναι ότι το ένα μέλος αποδεικνύει την γνώση του κωδικού π στέλνοντας την ποσότητα x+π mod q αντί για το π μόνο. Η ποσότητα x επιλέγεται τυχαία. Επομένως ο ενισχυμένος κωδικός (amplified password) x+π δεν μπορεί να προβλεφθεί αν το x παραμείνει μυστικό. Έστω A(π;x) η συνάρτηση που παράγει τον ενισχυμένο κωδικό.

Η απόδειξη της γνώσης του κωδικού αποτελείται από τρία βήματα. Η Alice γνωρίζει τον κωδικό π και ο Bob την τιμή g^π . Επιπλέον ορίζονται τρείς συναρτήσεις G_1 , G_2 , και H οι οποίες αντιστοιχούν σε κάθε βήμα αντίστοιχα.

1. Η Alice δεσμεύεται στην τιμή του x που επέλεξε(commitment). Η τιμή που θα αποστείλει πρέπει να μην δίνει την δυνατότητα ούτε στον Bob να υπολογίσει το x. Επομένως $G_1 = g^x$
2. Στο δεύτερο βήμα αποστέλλεται μια τυχαία πρόκληση από τον Bob. Αν έχει οριστεί ως $A(\pi; x) = (x + \pi)^{-1}$ mod q =W τότε μόνο κάποιος που γνωρίζει το x, π μπορεί να υπολογίσει το αποτέλεσμα της συνάρτησης. Για αυτό τον λόγο επιλέγεται ως $G_2 = g^{(x+\pi)\cdot y}$. Το y επιλέγεται τυχαία από τον Bob. Ο Bob μπορεί να υπολογίσει την τιμή G_2 καθώς γνωρίζει το g^π και το g^x.
3. Στο τρίτο βήμα αποστέλλεται η απάντηση της Alice που αποδεικνύει την γνώση του ενισχυμένου κωδικού. Ως απόδειξη ορίζεται η τιμή $H = g^y$. Η Alice μπορεί να την υπολογίσει ως $\left(g^{(x+\pi)*y}\right)^W = g^y$

Η παραπάνω μέθοδος μπορεί να χρησιμοποιηθεί για την κατασκευή ενός σχήματος που παρέχει και ανταλλαγή κλειδιού μεταξύ των δύο μερών. Επιπλέον αξίζει να σημειωθεί ότι δεν παρέχει τις

εγγυήσεις των augmented πρωτοκόλλων καθώς η τιμή που διατηρεί ο εξυπηρετητής μπορεί να χρησιμοποιηθεί για να υποδυθεί κάποιος τον πελάτη.

Για να καταστεί αδύνατο σε έναν επιτιθέμενο να ανακαλύψει τον κωδικό με την χρήση λεξικού στην περίπτωση που καταφέρει να πάρει το αρχείο των κωδικών, οι συγγραφείς εισάγουν την έννοια του ενισχυμένου αρχείου κωδικών (amplified password file). Το συγκεκριμένο αρχείο αποτελείται από εγγραφές της μορφής (id,t,v). Η τιμή v είναι $v = g^{(s+r)^{-1}\cdot u}$ με το u=h(id,π). Η τιμή s διαλέγεται τυχαία στο Z_q και η τιμή t διαλέγεται τυχαία στο $\{0,1\}^\kappa$. Το αρχείο αυτό μπορεί να αποθηκευτεί κανονικά στον server αλλά η τιμή s πρέπει να παραμείνει κρυφή και να αποθηκευτεί κάπου με ασφάλεια(σε HSM αν είναι δυνατόν). Επομένως το αρχείο των κωδικών είναι ασφαλές από επιθέσεις λεξικού και από την χρήση του για να υποδυθεί κάποιος τον server υπό την προϋπόθεση ότι το s παραμένει μυστικό.

Το πρωτόκολλο AMP κατασκευάζεται συνδυάζοντας το amplified password file και την ιδέα του amplified password proof. Θεωρούμε $A(\pi; x) = (x+u)^{-1} \cdot (x+e)$ με u=h_1(id,π) και e=h_2=(G_1,G_2,id,Alice,Bob). Η τιμή e χρησιμοποιείται για να αποφευχθεί client impersonation αν το αρχείο κωδικών και το s διαρρεύσουν. Τ βήματα του πρωτοκόλλου είναι τα παρακάτω:

1. Η Alice υπολογίζει την τιμή $G_1 = g^x$επιλέγοντας τυχαία από το Z_q και αποστέλλει το ζεύγος (id, G_1).
2. Ο Bob ανασύρει το t,v και υπολογίζει την τιμή $G_2 = (G_1)^y v^{(s+t)y} = (g^x g^u)^y$ διαλέγοντας ένα τυχαίο y στο Z_q . Υψώνοντας το v εις την (s+t) παράγεται το u καθώς αφαιρούμε το amplification.
3. Η Alice υπολογίζει την τιμή u=h_1=(id,π) και την τιμή $X = (x+u)^{-1} \mathrm{mod} q$. Όταν λάβει το δεύτερο μήνυμα υπολογίζει τα e=h_2=(G_1,G_2,id,Alice,Bob) , $W = X(x+e) mod q$ και $a = (G_2)^W = \left(g^{(x+u)y}\right)^W = g^{y(x+e)}$. Τέλος υπολογίζει την τιμή K_1=h_3(a) και H_{11}=h_4(id, G_1,K_1). Στον Bob αποστέλλει την τιμή H_{11}.
4. Ο Bob υπολογίζει την τιμή e=h_2 (G_1, G_2 ,id,Alice,Bob), b $(G_1)^y g^e y = (g^x g^e)^y = g^{(x+e)y}$, K_2 =h_3(b), H_{12}=h_4(id,G_1,K_2). Όταν λάβει το τρίτο μήνυμα συγκρίνει το H_{12} με το H_{11}. Αν είναι ίδια ο Bob έχει αυθεντικοποιήσει την Alice η οποία έχει αποδείξει ότι γνωρίζει τον ενισχυμένο κωδικό. Έπειτα υπολογίζει H_{22}=h_5(id,G_2,K_2) και το αποστέλλει στην Alice. Σε αυτό το σημείο ο Bob έχει συμφωνήσει στο κλειδί K(=K_1=K_2).
5. Η Alice υπολογίζει την τιμή H_{21}=h_5(id,G_2,K_1). Όταν λάβει το τέταρτο μήνυμα συγκρίνει το H_{21} με το H_{22}. Αν είναι ίδια η Alice έχει αυθεντικοποιήσει τον Bob ο οποίος γνωρίζει το v και συμφωνεί στο κλειδί K(=K_1=K_2).

Το παραπάνω πρωτόκολλο έχει χρησιμοποιηθεί και με κάποιες διαφοροποιήσεις.

- AMP[a] Το συγκεκριμμένο πρωτόκολλο δεν χρησιμοποιεί την τιμή e για την παραγωγή του amplified password. Συνεπώς κατά την εκτέλεση του πρωτοκόλλου τα δυο μέρη δεν χρησιμοποιούν την τιμή e για τον υπολογισμό του a,b και υπολογίζουν κοινό κλειδί g^{xy}. Αυτή η παραλλαγή του πρωτοκόλλου παραμένει ασφαλής από client impersonation ακόμα και το αρχείο κωδικών γνωστοποιηθεί με την προϋπόθεση ότι το ιδιωτικό κλειδί του εξυπηρετητή παραμένει ασφαλές(παράμετρος s). Η συνάρτηση παραγωγής του ενισχυμένου κωδικού είναι $A(\pi; x) = (x+u)^{-1} x mod q$.
- AMP[e] Σε αυτήν την περίπτωση δεν χρησιμοποιείται η παράμετρος s. Η ασφάλεια του αρχείου κωδικών εξαρτάται μόνο από την παράμετρο e (e-protection). Αυτό σημαίνει ότι αν γνωστοποιηθεί το αρχείο κωδικών δεν είναι δυνατό να υποδυθεί κάποιος τον πελάτη. Αντίθετα μπορεί να υποδυθεί τον εξυπηρετητή καθώς και να προσπαθήσει να ανακαλύψει τον κωδικό με την χρήση επίθεσης λεξικού. Η συνάρτηση παραγωγής του ενισχυμένου κωδικού είναι $A(\pi; x) = (x+u)^{-1}(x+e) mod q$.
- AMP[i] Η συγκεκριμένη έκδοση παρέχει υπονοούμενη αυθεντικοποίηση καθώς τα τελευταία μηνύματα που χρησιμοποιούνται για την επαλήθευση του κλειδιού δεν ανταλλάσσονται . Ωστόσο η εμπιστευτικότητα της συνόδου που δημιουργείται είναι εγγυημένη καθώς κάποιος που δεν γνωρίζει τον κωδικό δεν θα παράγει το σωστό κλειδί.

- AMP^{+H} Η διαφοροποίηση αυτού του πρωτοκόλλου είναι ότι στον υπολογισμό του G_2 εισάγεται μια επιπλέον παράμετρος $e1=h_2(G_1,id,Alice,Bob)$ ώστε να τροποποιηθεί η ομοιότητα μεταξύ του G_2 και του a,b. Ωστόσο αυτή η προσθήκη δεν παρέχει κάτι στην ασφάλεια του πρωτοκόλλου.
- AMP^{++}
- AMP^{n+} αποτελεί μια επέκταση του AMP^n η οποία γίνεται για μεγαλύτερη αποδοτικότητα σε σχέση με το AMP. Το AMP^n δεν παρέχει καμία ασφάλεια στην περίπτωση που το αρχείο κωδικών γίνει γνωστό. Η βασική διαφορά είναι ότι οι τιμές γίνονται $v=h_2(t,u)$ με $u=h_1(id,\pi)$. Το πρωτόκολλο αποκτά ανθεκτικότητα σε επιτιθέμενους που χρησιμοποιούν το αρχείο κωδικών για να υποδυθούν τον πελάτη. Ωστόσο το πρωτόκολλο χάνει την ιδιότητα της μηδενικής γνώσης καθώς ο εξυπηρετητής μπορεί να διαβάσει το u κατά την εκτέλεση του πρωτοκόλλου.

2.5.7 SPEKE

Το πρωτόκολλο SPEKE [17] αποτελείται από δυο στάδια. Το πρώτο στάδιο χρησιμοποιεί μια ανταλλαγή Diffie-Hellman για να ανταλλάξει ένα κοινό κλειδί K με την διαφορά ότι δεν χρησιμοποιείται η συνηθισμένη βάση g αλλά μια βάση που προκύπτει από τον κοινό κωδικό που μοιράζονται οι χρήστες. Το δεύτερο στάδιο ακολουθεί κανονικά τα βήματα της ανταλλαγής Diffie-Helman. Για την ανάλυση του πρωτοκόλλου χρησιμοποιούνται τα παρακάτω σύμβολα:

S	Ο κοινός κωδικός
p	Ένας μεγάλος πρώτος αριθμός
q	Ένας μεγάλος πρώτος παράγοντας του αριθμού p-1
g	Η βάση του DH
G_x	Μια υποομάδα του Z_p^* τάξης x. Το x είναι παράγοντας του p-1.
f(S)	Η συνάρτηση που μετατρέπει τον κωδικό S σε μια κατάλληλη βάση για DH
R_A, R_B	Τυχαίοι αριθμοί οι οποίοι διαλέγονται από το κάθε μέρος.
Q_A, Q_B	Οι εκθέτες που προκύπτουν
$E_k(m)$	Μια συνάρτηση κρυπτογράφησης με κλειδί το k.
h(m)	Μια συνάρτηση κατακερματισμού.
A→B:m	Το βέλος υποδεικνύει αποστολή δεδομένων. Η Alice στέλνει την τιμή m στον Bob.
K	Το κλειδί συνόδου.

Πίνακας 11 Επέξηγηση συμβόλων του πρωτοκόλλου Speke

Τα βήματα του πρωτοκόλλου έχουν ως εξής:

1. Η Alice υπολογίζει την τιμή $Q_A = f(S)^{R_A} mod p$, A→B: Q_A
2. Ο Bob υπολογίζει την τιμή $Q_B = f(S)^{R_B} mod p$, B→A: Q_B

Η Alice υπολογίζει την τιμή $K = h\left(Q_B^{R_A} mod p\right)$ και ο Bob υπολογίζει την τιμή $K = h\left(Q_A^{R_B} mod p\right)$.Τα επόμενα βήματα χρησιμοποιούνται ώστε να εξακριβωθεί ότι και τα δύο μέρη γνωρίζουν το κλειδί συνόδου K.

3. Η Alice επιλέγει μια τυχαία τιμή C_A, A→B: $E_K(C_A)$
4. Ο Bob επιλέγει μια τυχαία τιμή C_B, B→A: $E_K(C_B, C_A)$
5. Η Alice εξακριβώνει ότι η τιμή C_A είναι σωστή, A→B: $E_K(C_B)$
6. Ο Bob εξακριβώνει ότι η τιμή C_B είναι σωστή.

Για την φάση του ελέγχου του κλειδιού μπορεί να χρησιμοποιηθούν και άλλοι τρόποι. Για παράδειγμα η Alice μπορεί να στείλει ως απόδειξη την τιμή h(h(K)). Ο Bob θα ελέγξει αν η τιμή είναι σωστή και θα αποστείλει την τιμή h(K) στην Alice.

2.5.8 DH-EKE (Diffie-Hellman Encrypted Key Exchange)

Παρόμοια με το SPEKE αυτό το πρωτόκολλο [16] μπορεί να διαιρεθεί σε δυο φάσεις. Κατά την πρώτη φάση γίνεται μια ανταλλαγή DH για να παραχθεί το κοινό κλειδί κατά την οποία και τα δυο μέλη κρυπτογραφούν τον εκθέτη χρησιμοποιώντας τον κωδικό S ως κλειδί. Η μια κρυπτογράφηση μπορεί να παραλειφθεί αλλά αυτό πιθανώς μειώνει την ασφάλεια του πρωτοκόλλου [16] [21]. Προφανώς οι τιμές p και g καθώς και ο αλγόριθμος κρυπτογράφησης πρέπει να επιλεγούν κατάλληλα.

Στην δεύτερη φάση του πρωτοκόλλου και τα δυο μέρη επιβεβαιώνουν την γνώση του κλειδιού K. Τα βήματα του πρωτοκόλλου είναι τα παρακάτω:

1. Η Alice υπολογίζει την τιμή $Q_A = g^{R_A} mod p$, A→B: $E_S(Q_A)$
2. Ο Bob υπολογίζει την τιμή $Q_B = g^{R_B} mod p$, B→A: $E_S(Q_B)$

Η Alice υπολογίζει την τιμή $K = h\left(Q_B^{R_A} od\right)$ Bob υπολογίζει την τιμή $K = h\left(Q_A^{R_B} od\right)$

2.5.9 Ανάλυση SPEKE DH-EKE

Παρακάτω θα αναλυθούν τρεις κατηγορίες επιθέσεων που μπορούν να εφαρμοστούν σε αυτά τα πρωτόκολλα.

Υπολογισμός διακριτού λογαρίθμου. Η συγκεκριμένη επίθεση έχει ως σκοπό να υπολογιστεί ο διακριτός λογάριθμος ώστε να αποκαλυφθεί ο εκθέτης και τελικά ο κοινός κωδικός. Η δυσκολία του υπολογισμού εξαρτάται από το μέγεθος και κάποια χαρακτηριστικά του p. Το μέγεθος του p πρέπει να είναι τουλάχιστον 2048 bits. Αρκετά μεγάλα modulus καθυστερούν το πρωτόκολλο σημαντικά. Επιπλέον η τιμή p πρέπει να επιλεγεί κατάλληλα ώστε να αποφευχθούν κάποιες επιθέσεις. Η τιμή p-1 πρέπει να έχει ένα μεγάλο πρώτο παράγοντα q ώστε να το πρωτόκολλο να μην είναι εύκολο να υπολογιστεί ο διακριτός λογάριθμος με την μέθοδο Pohlig-Hellman [22]. Μπορεί να χρησιμοποιηθεί ένας ασφαλής πρώτος αριθμός(safe prime) της μορφής p=2q+1.

Information leak. Αν οι παράμετροι δεν επιλεγούν σωστά οι τιμές Q_A και Q_B μπορεί να αποκαλύψουν πληροφορία για τον κωδικό S επιτρέποντας μια επίθεση διαμέρισης (partition attack).

Στην περίπτωση του DH-EKE η τιμή p-1 πρέπει να έχει ένα μεγάλο πρώτο παράγοντα q καθώς και η τιμή g να είναι γεννήτορας της πολλαπλασιαστικής ομάδας mod p. Η τιμή g πρέπει να είναι γεννήτορας ώστε να αποφευχθεί η επίθεση διαμέρισης [17]. Η συγκεκριμένη επίθεση δουλεύει ως εξής: Ο επιτιθέμενος δοκιμάζει να αποκρυπτογραφήσει την τιμή $E_S\left(g^{(R_X)} mod p\right)$ χρησιμοποιώντας ένα λεξικό με κωδικούς. Αν η βάση g δεν είναι γεννήτορας τότε μια λανθασμένη πρόβλεψη μπορεί να επιβεβαιωθεί αν το αποτέλεσμα είναι πρωταρχική ρίζα. Συνεπώς οι τιμές Q_X πρέπει να μην έχουν κάποια συγκεκριμένη δομή ώστε να αποφευχθεί αυτή η επίθεση. Αρκεί η βάση g να είναι γεννήτορας ώστε να έχουμε τυχαία κατανομή των στοιχείων.

Στην περίπτωση του πρωτοκόλλου SPEKE τα αποτελέσματα δεν είναι κρυπτογραφημένα και ο επιτιθέμενος μπορεί να τα εξετάσει απευθείας. Αν το αποτέλεσμα είναι πρωταρχική ρίζα του p τότε γνωρίζει ότι και η βάση είναι πρωταρχική ρίζα. Για ένα ασφαλή πρώτο p αυτό αποκαλύπτει 1 bit πληροφορίας για τον κωδικό S. Αν για κάθε κωδικό S η βάση είναι γεννήτορας μια μεγάλης πρώτης υποομάδας τότε δεν αποκαλύπτεται καθόλου πληροφορία. Γενικότερα η συνάρτηση f(S) θα πρέπει να παράγει ένα αποτέλεσμα ίδιας τάξης για όλους τους κωδικούς S ώστε εξετάζοντας την τάξη του αποτελέσματος $f(S)^x mod p$να μην είναι δυνατό να εξάγουμε πληροφορία για τον κωδικό S.

Έχουν δημοσιευτεί δύο επιθέσεις για το πρωτόκολλο SPEKE οι οποίες θα αναλυθούν εκτενέστερα παρακάτω [23]. Η πρώτη επίθεση επιτρέπει στον επιτιθέμενο να αυθεντικοποιηθεί χωρίς να γνωρίζει τον κωδικό ξεκινώντας δύο παράλληλες συνεδρίες με το θύμα. Στην δεύτερη επίθεση ένας επιτιθέμενος που παρακολουθεί το κανάλι μπορεί να τροποποιήσει το κλειδί της συνεδρίας χωρίς να γίνει αντιληπτός (key malleability).

Impersonation attack. Η συγκεκριμένη επίθεση πραγματοποιείται όταν ο χρήστης χρησιμοποιεί πολλές συνεδρίες ταυτόχρονα με έναν άλλον χρήστη. Αυτό μπορεί να συμβεί αν ο χρήστης τρέχει διαφορετικές εφαρμογές που η κάθε μια χρησιμοποιεί το δικό της κανάλι επικοινωνίας.

Αρχικά η Alice εκκινεί μια συνεδρία με τον Bob στέλνοντας την τιμή g^x. Αυτή είναι η πρώτη συνεδρία. Ο Mallory που παρακολουθεί το κανάλι αναχαιτίζει το μήνυμα. Έπειτα επιλέγει ένα z από το διάστημα [2,p-2] και υψώνει την τιμή g^x εις την z (δηλαδή g^{xz}). Έπειτα εκκινεί μια καινούργια συνεδρία με την Alice και σε αυτήν στέλνει την τιμή g^{xz} υποδυόμενος τον Bob. Αυτή είναι η δεύτερη συνεδρία. Το z χρησιμοποιείται για να κάνει τα μηνύματα διαφορετικά για κάθε συνεδρία. Στην δεύτερη συνεδρία η Alice αποστέλλει την δικιά της τιμή g^y. Ο Mallory υψώνει αυτήν τιμή εις την z παράγοντας την τιμή g^{yz} την οποία αποστέλλει στην Alice ως απάντηση στην πρώτη συνεδρία. Σε αυτήν την φάση η Alice πρέπει να αποδείξει ότι γνωρίζει το κοινό κλειδί οπότε αποστέλλει την τιμή H(H(K)). Ο Mallory στέλνει αυτό το μήνυμα στην δεύτερη συνεδρία ως την απόδειξη του Bob. Έπειτα στην δεύτερη συνεδρία η Alice απαντάει στην απόδειξη του Bob στέλνοντας την τιμή H(K) την οποία ο Mallory αποστέλλει στην πρώτη συνεδρία. Ο Mallory έχει καταφέρει να αυθεντικοποιηθεί ως ο Bob και στις 2 συνεδρίες χωρίς να γνωρίζει τον κωδικό. Επιπλέον το κλειδί K που έχει προκύψει για κάθε συνεδρία είναι ίδιο.

Στην ουσία ο επιτιθέμενος τροποποιεί ένα μήνυμα του χρήστη και το αποστέλλει πίσω. Στο τέλος του πρωτοκόλλου η Alice πιστεύει ότι μοιράζεται ένα κλειδί με τον Bob ενώ στην πραγματικότητα έχει ανοίξει μια καινούργια συνεδρία με τον εαυτό της. Η σύγχυση της ταυτότητας του άλλου άκρου μπορεί να οδηγήσει σε προβλήματα σε κάποιες περιπτώσεις. Για παράδειγμα η Alice μπορεί να στείλει ένα μήνυμα στον Bob (πρώτη συνεδρία) ζητώντας του να εκτελέσει μια ενέργεια. Η Alice εγγυάται την αυθεντικότητα του μηνύματος πχ με την χρήση HMAC χρησιμοποιώντας το κλειδί K που έχει παραχθεί από την εκτέλεση του πρωτοκόλλου SPEKE. Ο Mallory μπορεί να στείλει αυτό το μήνυμα στην δεύτερη συνεδρία (καθώς το κλειδί K είναι ίδιο). Η Alice θα δεχθεί το μήνυμα και θα εκτελέσει την ενέργεια καθώς φαίνεται ότι προέρχεται από τον Bob.

Key-malleability attack. Στην συγκεκριμένη επίθεση ο επιτιθέμενος παρεμβάλλεται στην επικοινωνία δύο χρηστών και αλλάζει τα κλειδιά που ανταλλάσσονται υψώνοντας τα εις την z. Και τα δύο μέρη θα παράγουν το ίδιο κλειδί $k = H(g^{xyz})$ χωρίς να καταλάβουν ότι το κλειδί έχει τροποποιηθεί.

Αυτό φαίνεται να μην έχει κάποιο αντίκτυπο στην πρακτική εφαρμογή του πρωτοκόλλου. Ωστόσο το ότι ο επιτιθέμενος έχει καταφέρει να τροποποιήσει το κλειδί της συνεδρίας χωρίς να γίνει αντιληπτός παραβιάζει την θεωρητική ασφάλεια του πρωτοκόλλου [23]

2.5.10 Asymmetric Key Exchange

Όπως το EKE η βασική λειτουργία του AKE είναι να ανταλλάξει κλειδιά μεταξύ των δυο μερών και χρησιμοποιώντας αυτό το κλειδί να εξακριβώσει ότι και τα δυο μέρη γνωρίζουν τον κωδικό. Το AKE δεν χρησιμοποιεί κρυπτογράφηση όπως το EKE. Αντίθετα χρησιμοποιεί κάποιες προκαθορισμένες

μαθηματικές σχέσεις για να συνδυάσει τις εφήμερες τιμές που θα ανταλλαχθούν με τον κωδικό. Η αποφυγή της κρυπτογράφησης παρέχει κάποια πλεονεκτήματα:

- Το πρωτόκολλο είναι απλούστερο καθώς παραλείπονται τα βήματα για την επιλογή του αλγορίθμου. Επιπλέον αν ο αλγόριθμος καθορίζεται από το πρωτόκολλο τότε το πρωτόκολλο εξαρτάται από τον συγκεκριμένο αλγόριθμο.
- Η χρήση κρυπτογραφικών αλγορίθμων πολλές φορές υπόκειται σε περιορισμούς από την εκάστοτε νομοθεσία.
- Η ύπαρξη κάποια αδυναμίας στον αλγόριθμο κρυπτογράφησης λογικά θα έχει ως αποτέλεσμα την εισαγωγή της αδυναμίας και στο πρωτόκολλο αυθεντικοποίησης. Επιπλέον η χρήση του κωδικού ως κλειδί δεν είναι τόσο ασφαλής τεχνική.

Μια ακόμα διαφορά του AKE από το EKE είναι ότι EKE χρησιμοποιεί κάποιο κοινό μυστικό για την αυθεντικοποίηση. Και τα δυο μέρη διατηρούν ακριβώς το ίδιο μυστικό το οποίο χρησιμοποιούν για την αυθεντικοποίηση. Η γνώση του μυστικού αυτού αρκεί ώστε κάποιος να μπορεί να παριστάνει οποιοδήποτε από τα δυο μέρη. Συνεπώς και τα δυο μέρη πρέπει να διατηρούν το κοινό μυστικό ασφαλές.

Το AKE ακολουθεί μια άλλη προσέγγιση σύμφωνα με την οποία το κάθε μέλος υπολογίζει ένα μυστικό και χρησιμοποιεί μια συνάρτηση κατακερματισμού για να παράγει ένα verifier τον οποίο δίνει στο άλλο μέρος. Τώρα ο verifier δεν μπορεί να χρησιμοποιηθεί απευθείας από κάποιον επιτιθέμενο πρέπει πρώτα να ανακαλύψει τον κωδικό. Οι συμβολισμοί του AKE φαίνονται στον παρακάτω πίνακα

w,x,y,z	Τυχαίες παράμετροι
P(x)	Μονόδρομη συνάρτηση για την δημιουργία του verifier
Q(x,y), R(x,y)	Συναρτήσεις μίξης για τις ιδιωτικές και τις δημόσιες παραμέτρους αντίστοιχα
S(x,y)	Συνάρτηση παραγωγής του κλειδιού
K	Κλειδί της συνόδου

Πίνακας 12 Επεξήγηση συμβόλων AKE

Για την λειτουργία του AKE απαιτείται να ισχύει η παρακάτω σχέση

$$(\forall w, x, y, z) S\left(R\big(P(w), P(x)\big), Q(y, z)\right) = S\left(R\big(P(y), P(z)\big), Q(y, z)\right)$$

Η σχέση αυτή δεν παρέχει κάποια εγγύηση για την ασφάλεια του πρωτοκόλλου. Η ασφάλεια εξαρτάται από την επιλογή των συναρτήσεων P(), Q(), R() και S().

Κατά την αρχή του πρωτοκόλλου η Carol και ο Steve επιλέγουν τις παραμέτρους x και z αντίστοιχα. Οι τιμές αυτές αποτελούν τους κωδικούς των χρηστών. Η Carol υπολογίζει την τιμή P(x) την οποία αποστέλλει στον Steve. Αντίστοιχα ο Steve υπολογίζει την τιμή P(z) και την στέλνει στην Carol. Τώρα μπορεί να χρησιμοποιηθεί το AKE για να γίνει η ανταλλαγή κλειδιών. Τα βασικά βήματα φαίνονται στον παρακάτω πίνακα

Carol		Steve
Δημιουργία τυχαίου w	P(w) →	K = S(R(P(w),P(x)),Q(y,z))
K = S(R(P(y),P(z)),Q(w,x))	←P(y)	Δημιουργία τυχαίου y

Πίνακας 13 Εκτέλεση AKE

Αν οι τιμές x και z αντιστοιχούν στις τιμές P(x) και P(z) που συμφωνήθηκαν πριν τότε από την προηγούμενη εξίσωση οι δύο τιμές του K θα είναι ίδιες. Για την αυθεντικοποίηση τα δυο μέλη μπορούν

να χρησιμοποιήσουν κάποια προσυμφωνημένη μέθοδο για να εξακριβώσουν αν τα κλειδιά είναι ίδια. Η ασφάλεια του πρωτοκόλλου εξαρτάται από αυτήν την επιλογή.

Μπορούμε να συμπεράνουμε ότι οι τιμές x και z έχουν τον ρόλο των μυστικών για κάθε μέλος ενώ οι τιμές w και y είναι εφήμερες παράμετροι με σκοπό το κλειδί που προκύπτει να είναι διαφορετικό για κάθε συνεδρία. Γενικότερα η ασφάλεια του AKE βασίζεται στην επιλογή των συναρτήσεων. Η συνάρτηση P() θα πρέπει να είναι δύσκολο να αντιστραφεί και να μην αποκαλύπτει κάποια πληροφορία για την είσοδο. Το ίδιο ισχύει και για την S() και ειδικότερα για την δεύτερη παράμετρο. Επιπλέον θα πρέπει να είναι υπολογιστικά αδύνατο να υπολογιστεί η τιμή K χρησιμοποιώντας τις τιμές P(w), P(x), P(y), P(z).

2.5.11 Το πρωτόκολλο SRP

Το SRP ακολουθεί μια συγκεκριμένη κατασκευή που ονομάζεται AKE(asymmetric key exchange). Σε αυτήν την περίπτωση δεν κρυπτογραφούνται τα μηνύματα του πρωτοκόλλου (σε αντίθεση με την EKE) αλλά χρησιμοποιούνται κάποιες μαθηματικές σχέσεις για τον συνδυασμό των εφήμερων τιμών με τον κωδικό.

Το πρωτόκολλο SRP αποτελεί μια υλοποίηση της AKE. Στο SRP όλοι οι υπολογισμοί πραγματοποιούνται σε ένα πεπερασμένο σώμα GF(n). Αρχικά διαλέγεται ένας μεγάλος πρώτος αριθμός n και όλες οι πράξεις (προσθέσεις, πολλαπλασιασμοί και ύψωση σε δύναμη) εκτελούνται modulo n. Συνεπώς η είσοδος και η έξοδος των συναρτήσεων P(), Q(), R(), S() είναι ακέραιοι αριθμοί μεταξύ 0 και n-1.

Η μονόδρομη συνάρτηση P() για την δημιουργία του verifier είναι μια ύψωση σε δύναμη όπου g είναι ένας γεννήτορας στο GF(n).

$$P(x) = g^x$$

Οι συναρτήσεις Q(), R() και S() είναι οι παρακάτω:

$$Q(w, x) = w + ux$$
$$R(w, x) = wx^u$$
$$S(w, x) = w^x$$

Η u είναι μια τυχαία παράμετρος η οποία είναι δημόσια.

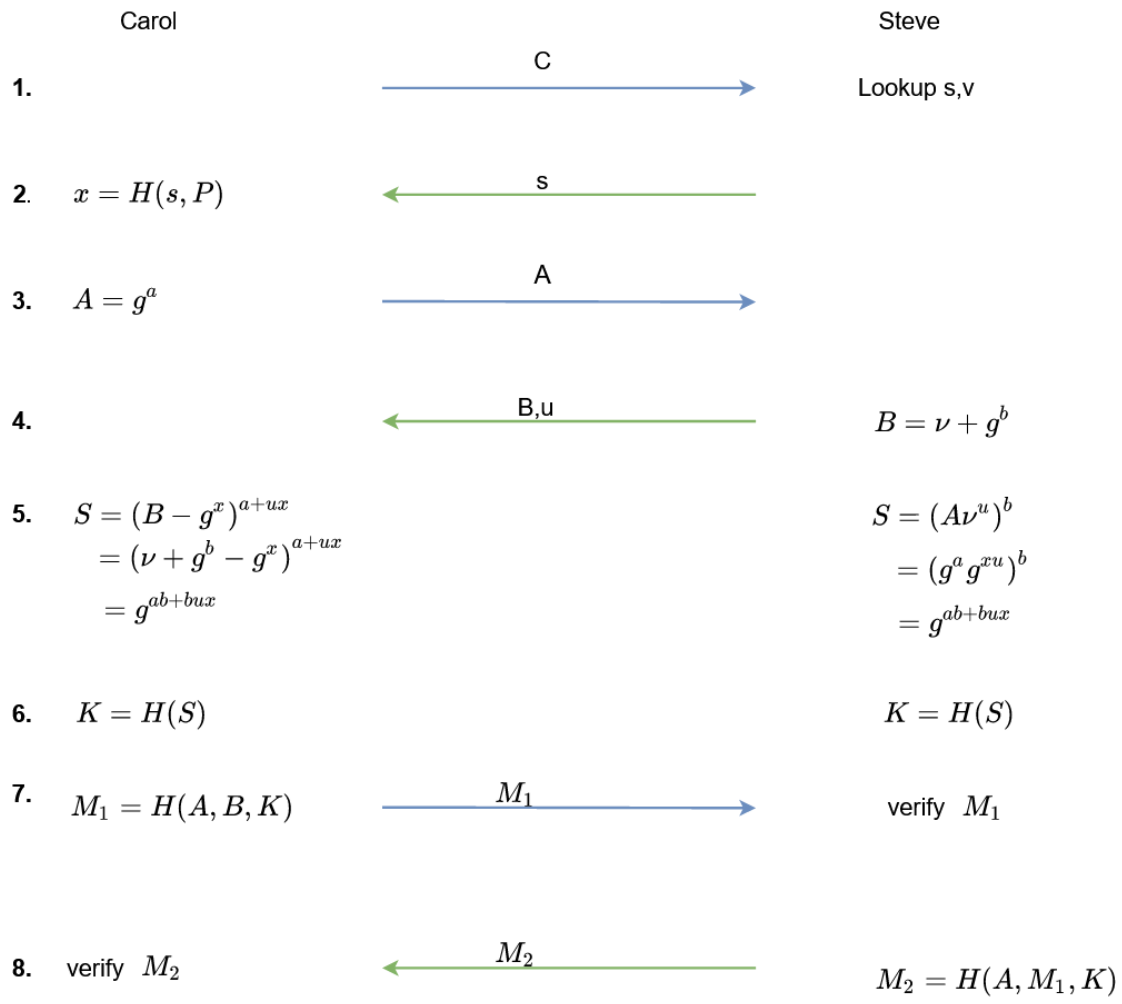
Ο ρόλος της θα αναλυθεί παρακάτω. Παρακάτω ακολουθεί μια αναλυτική περιγραφή του πρωτοκόλλου [24]. Αρχικά για την δημιουργία του verifier η Carol διαλέγει ένα τυχαίο salt s και υπολογίζει τις παρακάτω τιμές. Με P συμβολίζουμε τον κωδικό του χρήστη. Οι τιμές των g, n έχουν συμφωνηθεί εκ των προτέρων.

$$x = H(s, P)$$

$$v = g^x$$

Ο Steve αποθηκεύει τον verifier(v) και το salt(s). Η τιμή χ καταστρέφεται καθώς είναι ισοδύναμη με τον κωδικό. Όπως έχει αναφερθεί το AKE επιτρέπει και στο άλλο μέρος (στην περίπτωση μας ο Steve) να έχει κάποιον κωδικό z με το αντίστοιχο δημόσιο κλειδί g^z (δηλαδή ο verifier) να βρίσκεται στην κατοχή της Carol. Το SRP θέτει το z=0. Αφού το ιδιωτικό κλειδί είναι 0 το αντίστοιχο δημόσιο κλειδί θα είναι 1. Αυτό απλοποιεί το πρωτόκολλο καθώς ο Steve χρειάζεται να προστατεύει μόνο τον verifier της Carol αντί να προστατεύει και τον δικό του κωδικό. Παρόμοια η Carol δεν χρειάζεται να διατηρεί το δημόσιο κλειδί του Steve.

Για την αυθεντικοποίηση των δυο μερών ακολουθούνται τα βήματα που εικονίζονται στο παρακάτω σχήμα:



Εικόνα 1 Το πρωτόκολλο SRP

1. Η Carol στέλνει στον Steve το όνομα χρήστη
2. Ο Steve βρίσκει τον verifier v της Carol και το salt s. Στέλνει το s στην Carol. Η Carol υπολογίζει το ιδιωτικό κλειδί x χρησιμοποιώντας το salt και τον κωδικό της P.
3. Η Carol παράγει ένα τυχαίο αριθμό α τέτοιον ώστε 1<α<n και υπολογίζει το εφήμερο κλειδί $A = g^a$ το οποίο στέλνει στον Steve.
4. Ο Steve δημιουργεί ένα τυχαίο αριθμό b τέτοιον ώστε 1<b<n και υπολογίζει το εφήμερο κλειδί $B = v + g^b$ το οποίο αποστέλλει στην Carol μαζί με την τυχαία παράμετρο u.
5. Και τα δυο μέρη υπολογίζουν την κοινή τιμή $S = g^{(ab+bux)}$ χρησιμοποιώντας τιμές που είναι διαθέσιμες στον καθένα. Αν ο κωδικός P της Carol είναι ο ίδιος που χρησιμοποίησε για την δημιουργία του verifier τότε οι δυο τιμές του S που θα παράγουν θα είναι ίδιες.
6. Και τα δύο μέρη χρησιμοποιούν μια συνάρτηση κατακερματισμού για να παράγουν το τελικό κλειδί.
7. Η Carol στέλνει στον Steve την τιμή M_1 ως απόδειξη ότι διαθέτει το σωστό κλειδί για την συγκεκριμένη σύνοδο. Ο Steve υπολογίζει το M_1 και ελέγχει αν ταιριάζει με την τιμή που έλαβε.
8. Ο Steve στέλνει στην Carol την τιμή M_2 ως απόδειξη ότι και αυτό διαθέτει το σωστό κλειδί. Παρόμοια η Carol υπολογίζει το M_2 και ελέγχει αν ταιριάζει με αυτό που έλαβε από τον Steve.

Το παραπάνω πρωτόκολλο προκύπτει αν αντικαταστήσουμε τις παραπάνω εξισώσεις στο πρωτόκολλο AKE προσθέτοντας κάποια βήματα για την ανταλλαγή πληροφοριών όπως η ταυτότητα του χρήστη και το salt. Επιπλέον το srp προσθέτει δυο επιπλέον μηνύματα στο τέλος για να εξακριβωθεί αν το κλειδί που ανταλλάχθηκε είναι σωστό. Μετά την επιτυχή εκτέλεση του πρωτοκόλλου και τα δυο μέρη θα συμφωνήσουν στο κλειδί συνόδου $S = g^{(ab+bux)}$.

Υπολογισμός της τιμής B. Η τιμή B προκύπτει ως άθροισμα του verifier και του g^b. Για απλοποίηση του πρωτοκόλλου θα μπορούσαμε να χρησιμοποιήσουμε $B = g^b$. Ωστόσο αυτή η μικρή αλλαγή καθιστά το πρωτόκολλο ευάλωτο σε επιθέσεις με την χρήση λεξικού. Ο επιτιθέμενος στην συγκεκριμένη περίπτωση υποδύεται τον εξυπηρετητή και πείθει την Carol να αυθεντικοποιηθεί. Για την αυθεντικοποίηση ακολουθούνται τα παρακάτω βήματα:

1. Η Carol αποστέλλει το όνομα χρήστη.
2. Ο επιτιθέμενος αποστέλλει στην Carol το salt που βρήκε από κάποια προηγούμενη εκτέλεση του πρωτοκόλλου.
3. Η Carol αποστέλλει το εφήμερο δημόσιο κλειδί της A.
4. Ο επιτιθέμενος διαλέγει την δική του τυχαία τιμή b και u και υπολογίζει το δημόσιο κλειδί B το οποίο αποστέλλει στην Carol μαζί με το u.
5. Η Carol υπολογίζει το κλειδί συνόδου $S = B^{(a+ux)}$, υπολογίζει το K από το S και στέλνει την απόδειξη ότι γνωρίζει το K.
6. Ο επιτιθέμενος διακόπτει την σύνδεση ή ειδοποιεί την Carol ότι κ κωδικός είναι λανθασμένος.

Μετά από την εκτέλεση του πρωτοκόλλου ο επιτιθέμενος διαθέτει το A το δικό του b μαζί με την απόδειξη του K. Τώρα μπορεί να δοκιμάσει κάποιο κωδικό p' ως εξής: υπολογίζει το x' από τον κωδικό και μετά το v', κατασκευάζει το $S' = (Av'^u)^b$ και τέλος υπολογίζει K'=H(S') το οποίο και συγκρίνει με την απόδειξη της Carol για το πραγματικό K. Αν ταιριάζουν ο κωδικός που μάντεψε είναι σωστός.

Δεδομένου ότι η επίθεση αυτή πραγματοποιείται από κάποιον που δεν γνωρίζει το v ένας τρόπος να αποφευχθεί είναι να αναγκάσουμε τον εξυπηρετητή να δεσμευτεί στην τιμή του v στο βήμα 4. Ωστόσο ο τρόπος με τον οποίο θα συνδυαστούν τα δύο υπόλοιπα g^b και v πρέπει να επιλεχθεί κατάλληλα. Πιο συγκεκριμένα αν $f(v, g^b)$ η συνάρτηση που συνδυάζει τα δύο υπόλοιπα πρέπει να αποφύγουμε συναρτήσεις f οι οποίες έχουν την ιδιότητα $f(g^x, g^y) = g^{\left(f'(x,y)\right)}$ για κάποια συνάρτηση f'. Η παραπάνω επίθεση μπορεί να χρησιμοποιηθεί όταν η f έχει την παραπάνω ιδιότητα [24].

Επιπλέον η τιμή B θα πρέπει να αποκαλύπτει όσο το δυνατόν λιγότερη πληροφορία για τον verifier. Αυτό αποκλείει συναρτήσεις όπως $f(x, y) = x \oplus y$ ή $f(x, y) = E_y(x)$ όπου $E_k(x)$ ένας συμμετρικός αλγόριθμος κρυπτογράφησης. Σε αυτές τις περιπτώσεις ο επιτιθέμενος μπορεί να δοκιμάζει κωδικούς ποιο αποδοτικά καθώς έχει την δυνατότητα να εξαλείψει κωδικούς ως αδύνατους(ονομάζεται partition attack). Για παράδειγμα αν χρησιμοποιηθεί η $B = v \oplus g^b$ ο επιτιθέμενος θα δημιουργήσει έναν verifier v' για κάθε κωδικό και αν $B \oplus v' > n$ τότε ο κωδικός αυτός μπορεί να απορριφθεί. Αν αυτό γίνει για πολλές συνόδους πιθανώς ο επιτιθέμενος να καταφέρει να αποκλείσει αρκετούς κωδικούς.

Το SRP χρησιμοποιεί την πρόσθεση για να συνδυάσει τον verifier και το g^b καθώς δεν αποκαλύπτει κάποια πληροφορία για τον verifier και επιτρέπει στο πρωτόκολλο να αντιμετωπίσει τις επιθέσεις όταν κάποιος προσποιείται τον εξυπηρετητή. Τέλος η παράμετρος g πρέπει να είναι μια πρωταρχική ρίζα (primitive root) του GF(n) ώστε όλες οι τιμές του B να είναι ισοπίθανες για κάθε verifier. Αν αυτό δεν ισχύει και πάλι μπορεί να πραγματοποιηθεί η παραπάνω επίθεση (partition attack)

Η παράμετρος u συμβάλει στην ασφάλεια του πρωτοκόλλου παρόλο που αποστέλλεται κανονικά. Ας υποθέσουμε ότι ο επιτιθέμενος έχει ανακαλύψει τον verifier και προσπαθεί να λάβει πρόσβαση στον εξυπηρετητή. Επιπλέον έχει μάθει και την τιμή του u στο βήμα 3. Η επίθεση γίνεται ως εξής:

1. Ο επιτιθέμενος στέλνει το όνομα χρήστη της Carol στον Steve.
2. Ο Steve αποστέλλει κανονικά το salt της Carol στον επιτιθέμενο.

3. Ο επιτιθέμενος υπολογίζει το: $A = g^a v^{(-u)}$ και το στέλνει στον Steve αντί του κανονικού Α.
4. Ο Steve αποστέλλει κανονικά το $B = v + g^b$.
5. Ο επιτιθέμενος υπολογίζει το κλειδί για την σύνοδο ως: $K = H((B - v)^a mod n)$
6. Ο επιτιθέμενος στέλνει στον Steve την απόδειξη ότι γνωρίζει το κλειδί Κ και συνδέεται σαν να είναι η Carol.

Η επίθεση λειτουργεί γιατί ο Steve υπολογίζει το κλειδί ως :

$$S = (Av^u)^b = \left(g^a v^{(-u)} v^u\right)^b = g^{(ab)}$$

Η τιμή αυτή είναι ανεξάρτητη από τον κωδικό και μπορεί να υπολογιστεί από τον επιτιθέμενο. Από την στιγμή που έχει το ίδιο κλειδί με τον Steve μπορεί να υποδυθεί την Carol. Τέλος η επίθεση λειτουργεί αν το u έχει μια εκ των προτέρων γνωστή τιμή. Για να αποτρέψουμε κάποιον επιτιθέμενο να ακυρώσει τον verifier με τον παραπάνω τρόπο ο Steve πρέπει να μην αποκαλύψει την τιμή του u μέχρι να λάβει το Α από τον άλλο χρήστη.

Είναι σχετικά εύκολο να αποδειχθεί ότι το AKE και το SRP λειτουργούν σωστά με την έννοια ότι και οι δύο συμμετέχοντες θα συμφωνήσουν στο ίδιο κλειδί αν οι κωδικοί ήταν σωστοί. Ωστόσο είναι δυσκολότερο να αποδείξουμε ότι τα πρωτόκολλα είναι ασφαλή. Γενικότερα θα μπορούσαμε να πούμε ότι ένας επιτιθέμενος δεν πρέπει να έχει την δυνατότητα να λάβει πρόσβαση στον εξυπηρετητή απλά παρακολουθώντας τα βήματα του πρωτοκόλλου. Το SRP εισάγει επιπλέον περιορισμούς σε αυτόν τον ορισμό.

1. Κατά την εκτέλεση του πρωτοκόλλου δεν αποκαλύπτεται κάποια πληροφορία για τον κωδικό ή το ιδιωτικό κλειδί x. Ο επιτιθέμενος δεν έχει την δυνατότητα να δοκιμάσει κωδικούς χρησιμοποιώντας τα μηνύματα που ανταλλάχθηκαν.
2. Δεν αποκαλύπτεται κάποια χρήσιμη πληροφορία για το κλειδί συνόδου Κ. Από την στιγμή που το Κ είναι ένα κλειδί και όχι ένας κωδικός χαμηλής εντροπίας δεν μας απασχολούν επιθέσεις εξαντλητικής αναζήτησης δεδομένου ότι ο επιτιθέμενος δεν μπορεί να υπολογίσει το Κ απευθείας.
3. Ακόμα και αν επιτιθέμενος έχει την δυνατότητα να δημιουργεί ή να τροποποιεί τα μηνύματα που αποστέλλονται, το πρωτόκολλο πρέπει να αποτρέψει στον επιτιθέμενο να συνδεθεί στον εξυπηρετητή. Επιπλέον δεν πρέπει να λάβει καμία πληροφορία για τον κωδικό ή το κλειδί της συνόδου.
4. Αν ο επιτιθέμενος μάθει την τιμή του verifier δεν θα πρέπει να είναι σε θέση να υποδυθεί τον χρήστη χωρίς να δοκιμάσει κωδικούς (dictionary search).
5. Αν το κλειδί κάποιας προηγούμενης συνόδου αποκαλυφθεί ο επιτιθέμενος δεν θα πρέπει να είναι σε θέση να ανακαλύψει τον κωδικό.
6. Αν ο κωδικός του χρήστη αποκαλυφθεί το πρωτόκολλο δεν πρέπει να επιτρέψει στον επιτιθέμενο να ανακαλύψει το κλειδί των προηγούμενων συνόδων. Ακόμα και οι τωρινές σύνοδοι θα πρέπει να είναι προστατευμένες από παθητικές επιθέσεις.

Ένα πρωτόκολλο με αυτές τις ιδιότητες θεωρείται εύρωστο(robust) δηλαδή είναι ανθεκτικό σε επιθέσεις ακόμα και αν οι συμμετέχοντες δεν είναι εντελώς αξιόπιστοι ή ασφαλείς.

Το πρωτόκολλο SRP έχει σχεδιαστεί να είναι ανθεκτικό στην επίθεση Denning-Sacco [14]. Η συγκεκριμένη επίθεση πραγματοποιείται όταν ο επιτιθέμενος ανακαλύψει το κλειδί κάποιας συνόδου και το χρησιμοποιεί για να υποδυθεί τον χρήστη ή για εκτελέσει μια επίθεση λεξικού με σκοπό να ανακαλύψει τον κωδικό. Αν το κλειδί αποκαλυφθεί ο επιτιθέμενος δεν λαμβάνει κάποια πληροφορία συνδυάζοντας το κλειδί με τα μηνύματα επιβεβαίωσης M_1 ή M_2. Αυτό συμβαίνει γιατί το M_1 και το M_2 υπολογίζονται απευθείας από γνωστά δεδομένα και το κλειδί. Δεδομένου ότι δεν μπορεί να χρησιμοποιήσει το κλειδί για να εξακριβώσει αν ο κωδικός είναι σωστός δεν υπάρχει κάποιος τρόπος για να ανακαλύψει τον κωδικό.

Οι ενεργητικές επιθέσεις μπορούν να πραγματοποιηθούν με διαφορετικούς τρόπους ανάλογα με το τι πληροφορία είναι διαθέσιμη στον επιτιθέμενο. Αν ο επιτιθέμενος γνωρίζει το ιδιωτικό κλειδί

κάποιου χρήστη τότε μπορεί και να υποδυθεί τον συγκεκριμένο χρήστη. Αν γνωρίζει τον verifier τότε μπορεί να υποδυθεί τον server στην επικοινωνία με τον παραπάνω χρήστη. Για αυτόν τον λόγο ο verifier θα πρέπει να φυλάσσεται και δεν θα πρέπει να θεωρείται δημόσια γνωστή παράμετρος. Τέλος για μια επίθεση man in the middle απαιτείται ο επιτιθέμενος να μπορεί να ξεγελάσει και τα δύο μέρη της επικοινωνίας. Αυτό δεν μπορεί να γίνει αν δεν γνωρίζει τον κωδικό του χρήστη καθώς δεν μπορεί να εξαπατήσει τον server ότι επικοινωνεί με τον σωστό χρήστη. Αν δεν γνωρίζει ούτε τον verifier τότε δεν μπορεί να κοροϊδέψει ούτε τον χρήστη ότι επικοινωνεί με τον σωστό server.

Για την ασφαλή εκτέλεση του πρωτοκόλλου πρέπει να τηρούνται κάποιες προϋποθέσεις σχετικά με την επιλογή των παραμέτρων και των κρυπτογραφικών συναρτήσεων που χρησιμοποιούνται.

Η ασφάλεια του SRP βασίζεται στο ότι οι τιμές w,y παραμένουν μυστικές και οι τιμές P(w) και P(y) είναι δημόσια γνωστές. Άρα η συνάρτηση πρέπει να μην αντιστρέφεται εύκολα. Το SRP χρησιμοποιεί ως συνάρτηση P την $P(x) = g^x mod n$. Για να υπολογίσει κάποιος την τιμή x πρέπει να υπολογίσει τον διακριτό λογάριθμο. Ο υπολογισμός του διακριτού λογαρίθμου θεωρείται ένα δύσκολο πρόβλημα για μεγάλες τιμές του n. Στο ίδιο πρόβλημα στηρίζεται και η ασφάλεια του πρωτοκόλλου Diffie-Hellman. Συνεπώς η ασφάλεια του SRP είναι συνδεδεμένη με την ασφάλεια του Diffie-Hellman.

Στην περίπτωση του SRP επιθυμούμε η τιμή n να μην είναι smooth prime το οποίο σημαίνει η τιμή n-1 να μην αποτελείται αποκλειστικά από μικρούς παράγοντες. [24]

Επιπλέον τα πρωτόκολλα που βασίζονται σε διακριτούς λογαρίθμους είναι ευάλωτα σε μια τύπου επίθεση που ονομάζεται subgroup confinement στην οποία ο επιτιθέμενος αναγκάζει τα δυο μέλη να χρησιμοποιήσουν ένα κλειδί το οποίο είναι περιορισμένο σε μια μικρή υποομάδα του GF(n)[9 από srp]. Ο επιτιθέμενος μετά μπορεί να εξετάσει όλα τα στοιχεία της υποομάδας. Λόγω του τρόπου που υπολογίζονται τα κλειδιά συνόδου το SRP δεν είναι ευάλωτο σε αυτήν την επίθεση.

Για μεγαλύτερη ασφάλεια προτείνεται η τιμή n να είναι ένας ασφαλής πρώτος αριθμός της μορφής n=2p+1 όπου p είναι πρώτος αριθμός. Με την χρήση αυτών των αριθμών ο αριθμός των πιθανών υποομάδων είναι δυο επομένως μια επίθεση τύπου subgroup confinement μπορεί να ανιχνευτεί εύκολα.

Τέλος οι παράμετροι n,g μπορούν να αποστέλλονται στην αρχή του πρωτοκόλλου ή να είναι προσυμφωνημένες. Στην περίπτωση που οι τιμές δεν είναι προσυμφωνημένες και αποστέλλονται κατά την εκτέλεση του πρωτοκόλλου θα πρέπει να ελέγχονται και από τα δυο μέρη ότι ακολουθούν κάποιους συγκεκριμένους περιορισμούς. Οι έλεγχοι που πρέπει να γίνουν είναι οι ακόλουθοι (στην παρένθεση φαίνεται ποιος πρέπει να πραγματοποιήσει τον έλεγχο):

- Η τιμή n πρέπει να είναι ασφαλής πρώτος: Ο πελάτης πρέπει να ελέγξει αν το n είναι αρκετά μεγάλο καθώς και αν n και (n-1)/2 είναι πρώτοι αριθμοί.
- Η τιμή g πρέπει να είναι μια πρωταρχική ρίζα του GF(n)
- A ≠0 (server): Έχει ως σκοπό να αποτρέψει το κλειδί συνόδου του server να πάρει την τιμή 0.
- B≠0 (client): Αυτός ο έλεγχος γίνεται για να αποφευχθεί μια επίθεση λεξικού από κάποιον που υποδύεται τον χρήστη.
- a, b>$log_g n$: Οι τιμές g^a και g^bπρέπει να είναι μεγάλες ώστε να εφαρμοστεί η ακέραια διαίρεση και να αποτραπεί η χρήση του αλγεβρικού λογαρίθμου για να υπολογιστεί το a,b. Ωστόσο η πιθανότητα να συμβεί αυτό είναι απειροελάχιστη.

2.5.12 Βελτιώσεις πρωτοκόλλου SRP. Το πρωτόκολλο SRP-6

Η έκδοση του SRP που αναλύθηκε παραπάνω είναι ευάλωτη σε μια επίθεση που ονομάζεται two-for-one-guess [25]. Αυτή η επίθεση επιτρέπει σε έναν επιτιθέμενο που υποδύεται τον server να δοκιμάσει δυο κωδικούς σε μια σύνδεση με κάποιον πελάτη. Αυτό μειώνει την ασφάλεια του πρωτοκόλλου καθώς το επιθυμητό είναι ο επιτιθέμενος να μπορεί να δοκιμάσει μόνο ένα κωδικό με κάθε σύνδεση.

Η επίθεση λειτουργεί με τον εξής τρόπο. Ο εξυπηρετητής στην κανονική εκτέλεση στέλνει την τιμή $v + g^b$ η οποία ισούται με την τιμή $g^x + g^b$. Ο επιτιθέμενος αφού δεν γνωρίζει τον κωδικό ή τον verifier το μόνο που μπορεί να κάνει είναι να διαλέξει έναν verifier και να εκτελέσει το πρωτόκολλο υποδυόμενος τον εξυπηρετητή. Ωστόσο λόγω της συμμετρίας που διαθέτει η εξίσωση ο επιτιθέμενος μπορεί να εισάγει μια επιπλέον δοκιμή για τον κωδικό στην θέση της τυχαίας τιμής b και να στείλει εν τέλει την τιμή $B = g^x + g^y$ με x,y τους κωδικούς που θέλει να δοκιμάσει. Αν ο x είναι ο σωστός κωδικός τότε ο πελάτης θα αφαιρέσει το g^x και θα χρησιμοποιήσει το g^y ως βάση για τον υπολογισμό του κλειδιού. Παρόμοια αν ο y ήταν ο σωστός κωδικός τότε ως βάση θα χρησιμοποιηθεί το g^x στον υπολογισμό του κλειδιού. Έπειτα ο επιτιθέμενος παράγει δύο κλειδιά συνόδου χρησιμοποιώντας ως κωδικό το x για το πρώτο κλειδί και ως κωδικό το y για το δεύτερο κλειδί. Όταν λάβει το verification του κλειδιού από τον πελάτη μπορεί να το συγκρίνει με το verification των κλειδιών του και να διαπιστώσει αν κάποιος από τους δυο κωδικούς είναι σωστός. Επομένως ο επιτιθέμενος κατάφερε με μια συνεδρία να δοκιμάσει δύο κωδικούς το οποίο μειώνει και την πιθανότητα να γίνει αντιληπτός.

Ένας τρόπος για να αποφευχθεί η επίθεση είναι να πολλαπλασιάσουμε την τιμή v με μια τιμή προσυμφωνημένη τιμή k ώστε να αφαιρεθεί αυτή η συμμετρία από την εξίσωση. Άρα η τιμή που αποστέλλει ο server γίνεται $B = kv + g^b$. Σύμφωνα με τους δημιουργούς του πρωτοκόλλου η τιμή k=3 εξαλείφει την επίθεση και διατηρεί το πρωτόκολλο ασφαλές [25].

2.5.13 OPAQUE

Το πρωτόκολλο OPAQUE [26] σχεδιάστηκε με σκοπό να λύσει το πρόβλημα των pre-computation attacks που εμφανίζεται στην περίπτωση άλλων πρωτοκόλλων. Στην περίπτωση αυτών των επιθέσεων ο επιτιθέμενος μπορεί να προϋπολογίσει ένα λεξικό με κωδικούς το οποίο και να χρησιμοποιήσει αργότερα όταν καταφέρει να πάρει το αρχείο των κωδικών από τον εξυπηρετητή. Στην περίπτωση που χρησιμοποιείται salt για την παραγωγή του αρχείου των κωδικών, ο επιτιθέμενος είναι αναγκασμένος να δημιουργήσει ξεχωριστό λεξικό για κάθε χρήστη. Ωστόσο στις περισσότερες περιπτώσεις η κατασκευή του λεξικού μπορεί να γίνει πριν την αποκάλυψη του αρχείου κωδικών καθώς το salt δεν θεωρείται μυστικό και στα περισσότερα πρωτόκολλα αποστέλλεται plaintext στο άλλο μέρος. Το OPAQE προσπαθεί να λύσει αυτό το πρόβλημα κρατώντας το salt μυστικό.

Το συγκεκριμένο πρωτόκολλο εισάγει την έννοια των ισχυρών(strong) aPAKE (SaPAKE) πρωτοκόλλων τα οποία είναι ανθεκτικά σε pre-computation attacks. Το OPAQUE χρησιμοποιεί Oblivious PRF (OPRF) συναρτήσεις για αυτόν τον σκοπό. Αυτές οι συναρτήσεις είναι ένα πρωτόκολλο που στην περίπτωση μας εκτελείται μεταξύ του εξυπηρετητή ο οποίος έχει στην κατοχή του ένα PRF κλειδί k(το οποίο αποτελεί τα δεδομένα που παρέχει ο εξυπηρετητής στο πρωτόκολλο) και το χρήστη που διαθέτει τον κωδικό pw. Στο τέλος της διαδικασίας ο χρήστης μαθαίνει το αποτέλεσμα της συνάρτησης $F_k(pw)$ ενώ ο εξυπηρετητής δεν μαθαίνει τίποτα σχετικά με τον κωδικό. Επιπλέον είναι δυνατόν να κατασκευάσουμε ένα SaPAKE πρωτόκολλο για οπουδήποτε aPAKE πρωτόκολλο αν ο χρήστης υπολογίσει την τιμή $rw=F_k(pw)$ με τον server και χρησιμοποιήσει την τιμή rw ως κωδικό στο aPAKE πρωτόκολλο [26]. Αξίζει να σημειωθεί ότι το κλειδί k που χρησιμοποιείται στην OPRF συνάρτηση έχει τον ρόλο του salt. Σε αυτήν την περίπτωση το salt παραμένει μυστικό για να αποφευχθούν τα pre-computation attacks.

Η OPRF που χρησιμοποιείται από το πρωτόκολλο ονομάζεται DH-OPRF [26]. Όπως αναφέρθηκε ο χρήστης συμμετέχει στο πρωτόκολλο με τον κωδικό και ο εξυπηρετητής με το κλειδί k. Τα βήματα είναι τα παρακάτω:

- Ο χρήστης επιλέγει μια τυχαία τιμή r στο διάστημα [0..q-1] και στέλνει την τιμή $a = H'(x)g^r$ στον εξυπηρετητή. Ως H' θεωρούμε μια συνάρτηση κατακερματισμού που απεικονίζει τυχαίες συμβολοσειρές σε στοιχεία της κυκλικής ομάδας G. Η τιμή g^r μέσω του πολλαπλασιασμού (multiplicative blinding) εμποδίζει τον server να μάθει τον κωδικό.
- Ο εξυπηρετητής αποστέλλει τις τιμές $v = g^k$ και $b = a^k$

- Ο χρήστης αφού λάβει τις αυτές τις τιμές υπολογίζει το αποτέλεσμα της συνάρτησης ως $H(x, v, b \cdot v^{-r})$

Σε κάθε βήμα του πρωτοκόλλου οι τιμές a,b,v ελέγχονται ώστε να μην είναι μοναδιαία στοιχεία της ομάδας G. Αν ο έλεγχος αποτύχει η εκτέλεση του πρωτοκόλλου διακόπτεται.

Η βασική ιδέα είναι η ταυτόχρονη χρήση μιας OPRF συνάρτησης μαζί με ένα AKE(authenticated key exchange) πρωτόκολλο. Ωστόσο απαιτείται από το πρωτόκολλο να είναι ανθεκτικό σε επιθέσεις KCI(Key Compromise Impersonation) [27] ή αλλιώς reverse impersonation. Αυτή η αδυναμία επιτρέπει σε ένα επιτιθέμενο που έχει στην κατοχή του τα διαπιστευτήρια του πελάτη (πχ το ιδιωτικό κλειδί ενός client certificate) όχι απλά να υποδυθεί τον πελάτη στον εξυπηρετητή αλλά και να υποδυθεί οποιονδήποτε εξυπηρετητή στον πελάτη. Αυτό σημαίνει ότι πρωτόκολλα ανταλλαγής κλειδιών που βασίζονται σε προσυμφωνημένα κλειδιά δεν μπορούν να χρησιμοποιηθούν σε αυτήν την περίπτωση. Το TLS περιέχει κάποια πρωτόκολλα που είναι ευάλωτα σε αυτήν την επίθεση όπως η χρήση μη εφήμερου Diffie Helman για την ανταλλαγή κλειδιών (non-ephemeral Diffie-Hellman key exchange) όταν χρησιμοποιούνται client certificates με στατικά κλειδιά Diffie-Helman [28].Το OPAQUE χρησιμοποιεί ως πρωτόκολλο ανταλλαγής κλειδιών το πρωτόκολλο HQMV [29].

Τα βήματα του πρωτοκόλλου είναι τα παρακάτω:

- Ο χρήστης εγγράφεται στο σύστημα
- Ο χρήστης διαλέγει έναν κωδικό PwdU και δημιουργεί ένα ζεύγος κλειδιών κατάλληλα για το πρωτόκολλο ανταλλαγής κλειδιών που θα χρησιμοποιηθεί.
- Ο εξυπηρετητής επιλέγει το κλειδί kU που θα χρησιμοποιηθεί στην OPRF το οποίο είναι διαφορετικό και τυχαίο για κάθε χρήστη και θέτει $vU = g^k U$. Επιπλέον επιλέγει το δικό του ζεύγος ιδιωτικού και δημόσιου κλειδιού PrivS και PubS αντίστοιχά ανάλογα το πρωτόκολλο ανταλλαγής κλειδιών που επιλέχθηκε. Τέλος στέλνει το PubS στον χρήστη. Αξίζει να σημειωθεί πως ο εξυπηρετητής μπορεί να χρησιμοποιήσει το ίδιο ζεύγος κλειδιών και με άλλους χρήστες.
- Ο χρήστης U και ο εξυπηρετητής S εκτελούν την OPRF(kU;PwdU) αλλά μόνο ο χρήστης μαθαίνει το αποτέλεσμα RwdU.
- Ο χρήστης παράγει έναν «φάκελο» (envelope) ο οποίος ορίζεται ως EnvU = AuthEnc(RwdU; PrivU, PubU, PubS, vU) με AuthEnc να είναι μια συνάρτηση κρυπτογράφησης με αυθεντικοποίηση (authenticated encryption) και πρέπει να διαθέτει μια ιδιότητα που ονομάζεται “key committing”. Η ιδιότητα αυτή εξασφαλίζει ότι αν έχουμε δύο ζεύγη AuthEnc κλειδιών είναι υπολογιστικά αδύνατο να κατασκευάσουμε κρυπτοκείμενο που θα αποκρυπτογραφείται επιτυχώς και με τα δύο κλειδιά. Από τις ποσότητες που περιέχονται στον φάκελο μόνο η ποσότητα PrivU χρειάζεται να κρυπτογραφηθεί ενώ όλες οι άλλες ποσότητες εκτός της vU χρειάζεται να αυθεντικοποιηθούν.
- Τέλος ο χρήστης αποστέλλει τον φάκελο EnvU και το δημόσιο κλειδί PubU στον εξυπηρετητή και διαγράφει τον κωδικό PwdU, την τιμή RwdU (randomized password) και όλα τα κλειδιά. Ο εξυπηρετητής αποθηκεύει τις τιμές (EnvU, PubS, PrivS, PubU, kU, vU) οι οποίες αποτελούν την εγγραφή που θα χρησιμοποιείται για την αυθεντικοποίηση του χρήστη.

Ένας χρήστης που έχει ήδη εγγραφεί στο σύστημα εκτελεί τα παρακάτω βήματα για να αυθεντικοποιηθεί από τον εξυπηρετητή.

- Αποστέλλει στον εξυπηρετητή το αναγνωριστικό του (πχ το username)
- Ο χρήστης και ο εξυπηρετητής εκτελούν την διαδραστικά την OPRF και ο χρήστης αποκτά την τιμή RwdU.
- Έπειτα ο εξυπηρετητής αποστέλλει τον φάκελο EnvU στον χρήστη
- Ο χρήστης αποκρυπτογραφεί τον φάκελο χρησιμοποιώντας την τιμή RwdU και ανακτά το ιδιωτικό και δημόσιο κλειδί του καθώς και το δημόσιο κλειδί του εξυπηρετητή PubS.
- Τέλος και ο χρήστης και εξυπηρετητής εκτελούν το επιλεγμένο πρωτόκολλο ανταλλαγής κλειδιών χρησιμοποιώντας ο καθένας το ζεύγος κλειδιών του.

Στην περίπτωση του HMQV η ανταλλαγή γίνεται ως εξής:

- Ο χρήστης αποστέλλει την τιμή g^x
- Ο εξυπηρετητής αποστέλλει την τιμή g^y καθώς και την τιμή $Mac(K_{M1}; g^x, g^y)$
- Έπειτα ο χρήστης αποστέλλει την τιμή $Mac(K_{M2}; g^y, g^x)$

Το κοινό κλειδί K υπολογίζεται από τον χρήστη ως $K = H((g^y PubS^e)^{x+d\cdot PrivU})$ και από τον εξυπηρετητή ως $K = H((g^x PubU^d)^{y+e\cdot PrivS})$. Η τιμή d είναι ίση με $d = H(g^x, IdS)$ και η τιμή ε είναι ίση με $e = H(g^y, IdU)$. Οι τιμές IdU και IdS αναπαριστούν την ταυτότητα του χρήστη και του εξυπηρετητή αντίστοιχα.

Ένα πλεονέκτημα του συγκεκριμένου πρωτοκόλλου όπως παρουσιάστηκε είναι ότι ο εξυπηρετητής δεν μαθαίνει τον κωδικό του χρήστη κατά την εγγραφή. Ωστόσο η εγγραφή του χρήστη μπορεί να εκτελεστεί εξολοκλήρου από το εξυπηρετητή αν αυτό είναι αναγκαίο. Η αμοιβαία αυθεντικοποίηση επιτυγχάνεται με την αποστολή τριών συνολικά μηνυμάτων. Και τα δύο μέλη μπορούν να εξακριβώσουν αν έχουν υπολογίσει το κοινό κλειδί ελέγχοντας τις τιμές των Mac μηνυμάτων. Αν το τρίτο βήμα παραλειφθεί το πρωτόκολλο επιτυγχάνει υπονοούμενη αυθεντικοποίηση (implicit authentication). Αυτό σημαίνει ότι το ένα μέλος δεν γνωρίζει αν το άλλο μέλος έχει παράγει το σωστό κλειδί.

Παρόμοια με τα άλλα πρωτόκολλα αν ο επιτιθέμενος καταφέρει να πάρει πρόσβαση στο αρχείο των κωδικών μπορεί να υποδυθεί τον εξυπηρετητή αφού κάθε εγγραφή διατηρεί την τιμή kU και το ιδιωτικό κλειδί του server. Αν ο επιτιθέμενος δεν καταφέρει να αποκτήσει το ιδιωτικό κλειδί δεν θα καταφέρει να υποδυθεί τον server. Σε κάθε περίπτωση θα πρέπει να κάνει επίθεση λεξικού για να ανακαλύψει τον κωδικό του χρήστη.

Επιπλέον η χρήση του HMQV ως πρωτόκολλο ανταλλαγής κλειδιών παρέχει στο OPAQUE forward secrecy [29]. Αυτό έχει ως αποτέλεσμα ακόμα και αν ο κωδικός αποκαλυφθεί να οι προηγούμενες σύνοδοι να παραμένουν ασφαλείς.

Η δομή του OPAQUE αποτρέπει τα pre-computation attacks και επιπλέον επιτρέπει να ρυθμιστεί το κόστος που απαιτείται για να πραγματοποιηθεί μια επίθεση λεξικού όταν ο επιτιθέμενος αποκτήσει πρόσβαση στο αρχείο κωδικών. Αυτό μπορεί να πραγματοποιηθεί αλλάζοντας τον υπολογισμό της τιμής rw σε $rw = H^n\big(Fk(pw)\big)$ που σημαίνει ότι ο χρήστης θα εφαρμόσει n φορές την συνάρτηση H στο αποτέλεσμα της OPRF. Πρακτικά η συνάρτηση H θα είναι κάποια PBKDF όπως η PBKDF2 ή argon2. Ο επιτιθέμενος είναι αναγκασμένος για κάθε δοκιμή να καλέσει n φορές την επιλεγμένη συνάρτηση.

Τέλος η ασφάλεια του πρωτοκόλλου έχει αποδειχθεί και μαθηματικά χρησιμοποιώντας το random oracle model [26].

2.5.14 Spake2

Το πρωτόκολλο Spake [30] αποτελεί μια παραλλαγή του πρωτοκόλλου που προτάθηκε από τους Bellovin και Merritt [16]. Στο Spake η συνάρτηση κρυπτογράφησης έχει αντικατασταθεί με μια one-time pad function. Αυτό σημαίνει ότι κάθε φορά που ο χρήστης A θέλει να στείλει μια τιμή X κρυπτογραφημένη αποστέλλει την αποστέλλει ως $X \cdot M^{pw}$ με ένα στοιχείο της ομάδας G συσχετισμένο με τον χρήστη και pw ο κωδικός που μοιράζονται τα δύο μέρη.

Οι γνωστές πληροφορίες του πρωτοκόλλου είναι η πολλαπλασιαστική ομάδα G, γεννήτορας της ομάδας που συμβολίζεται με g, καθώς και τα common reference strings M,N τα οποία αποτελούν συμβολοσειρές

Το πρωτόκολλο εκτελείται ως εξής:

- Ο χρήστης A επιλέγει τυχαία την τιμή x και υπολογίζει την τιμή g^x. Έπειτα αποστέλλει την τιμή $X^* = X \cdot M^{pw}$.

- Παρόμοια ο χρήστης B επιλέγει τυχαία y και αποστέλλει την ποσότητα $Y^* = Y \cdot N^{pw}$
- Ο χρήστης A υπολογίζει την κοινή ποσότητα $K_A = (Y^*/N^{pw})^x = g^{xy}$
- Παρόμοια ο χρήστης B υπολογίζει την κοινή ποσότητα $K_B = (X^*/M^{pw})^y = g^{xy}$
- Τέλος και τα δύο μέρη υπολογίζουν το κλειδί της συνόδου χρησιμοποιώντας μια συνάρτηση κατακερματισμού στην οποία συμμετέχουν οι ταυτότητες των χρηστών A,B οι τιμές που ανταλλάχθηκαν, ο κωδικός και το κλειδί Diffie-Helman δηλαδή $SK = H(A, B, X^*, Y^*, pw, K)$ με $K=K_A=K_B$

Στην ουσία η διαφορά που παρουσιάζει το Spake1 είναι στον υπολογισμό του κλειδιού συνόδου στον οποίο δεν συμμετέχει ο κωδικός δηλαδή $SK = H(A, B, X^*, Y^*, K)$

Το πρωτόκολλο αυτό αν και είναι απλό εκτός από την γνώση του κωδικού απαιτεί και τα δυο μέρη να μοιράζονται δυο common reference strings(CRS). Η χρήση CRS παρέχει κάποια πλεονεκτήματα αλλά σε κάποιες περιπτώσεις μπορεί να περιορίζει την εφαρμογή του πρωτοκόλλου σε κάποιες περιπτώσεις. Για παράδειγμα σε ένα σύστημα με στο οποίο αυθεντικοποιούνται υπάλληλοι της εταιρίας θα ήταν δυνατό να επιλέξουν το CRS και να μεταδοθεί ασφαλώς σε όλους τους χρήστες. Σε άλλες περιπτώσεις μπορεί να μην είναι δυνατό να υπάρξει αυτού του είδους η εμπιστοσύνη. Επιπλέον αν ο επιτιθέμενος διαλέξει το CRS είναι δυνατόν να καταφέρει να ανακαλύψει τον κωδικό τον άλλων χρηστών [31]. Τέλος το Spake προσφέρει weak forward secrecy [32].

2.6 End-to-end encryption

Σε αυτήν την ενότητα θα εξεταστούν πρωτόκολλα για την κρυπτογράφηση των μηνυμάτων. Επιπρόσθετα θα διερευνηθεί αν μπορούν να παρέχουν forward secrecy.

2.6.1 SCIMP

Το πρωτόκολλο SCIMP [33] είναι ένα σύγχρονο πρωτόκολλο το οποίο χρησιμοποιείται από την εταιρία Silent Circle. Για να δημιουργήσει καινούρια κλειδιά κρυπτογράφησης ώστε να παρέχει forward secrecy χρησιμοποιεί τον παρακάτω μηχανισμό.

- Όταν η Alice στείλει ένα μήνυμα στον Bob χρησιμοποιεί μια συνάρτηση παραγωγής κλειδιών (KDF) δίνοντας ως είσοδο το τωρινό κλειδί κρυπτογράφησης ώστε να παράγει το επόμενο κλειδί κρυπτογράφησης.
- Η Alice καταστρέφει το τωρινό κλειδί και χρησιμοποιεί το καινούριο για να στείλει το επόμενο μήνυμα.
- Ο Bob ακολουθεί την ίδια διαδικασία

Η συγκεκριμένη τεχνική παρέχει forward secrecy καθώς τα προηγούμενα κλειδιά καταστρέφονται και τα επόμενα δεν μπορούν να χρησιμοποιηθούν για να παραχθούν τα προηγούμενα. Ωστόσο υπάρχουν κάποια μειονεκτήματα όπως τι γίνεται σε περιπτώσεις που τα μηνύματα φτάσουν εκτός σειράς ή χαθούν. Για είναι δυνατή η αποκρυπτογράφηση πρέπει ο παραλήπτης να διατηρεί κάποια από τα προηγούμενα κλειδιά δημιουργώντας έτσι ένα μεγαλύτερο χρονικό περιθώριο στο οποίο τα κλειδιά παραμένουν εκτεθειμένα. Επιπλέον δεν παρέχει future secrecy καθώς στην περίπτωση που ένα κλειδί γίνει γνωστό ο επιτιθέμενος μπορεί να παράγει όλα τα επόμενα κλειδιά. Σε αντίθεση το πρωτόκολλο OTR παρέχει self healing καθώς το κλειδί θα τροποποιηθεί στο επόμενο μήνυμα και δεν μπορεί να προβλεφθεί.

Η πρώτη έκδοση του πρωτοκόλλου δεν υποστηρίζει την αποστολή μηνυμάτων όταν ο άλλος χρήστης είναι offline. Στην δεύτερη έκδοση ακολουθείται μια τεχνική που ονομάζεται «Progressive Encryption». Ο κάθε χρήστης αποστέλλει ένα δημόσιο κλειδί στον server το οποίο ο server διατηρεί για κάποιο χρονικό διάστημα. Όταν η Alice θέλει να στείλει ένα μήνυμα στον Bob (ενώ είναι αποσυνδεδεμένος) κατεβάζει το κλειδί του Bob και το χρησιμοποιεί για την παραγωγή ενός συμμετρικού κλειδιού όπως θα γινόταν σε μια κανονική ανταλλαγή Diffie-Helman. Με το συμμετρικό κλειδί

κρυπτογραφεί το μήνυμα, παράγει ένα καινούργιο εφήμερο ζεύγος κλειδιών και αποστέλλει στον Bob το μήνυμα και το εφήμερο κλειδί.

Αυτή η τεχνική επιτρέπει την ανταλλαγή offline μηνυμάτων αλλά δημιουργεί προβλήματα στην αυθεντικοποίηση των δύο μερών επιτρέποντας σε κάποιον τρίτο να παρεμβληθεί στην επικοινωνία [34].

2.6.2 OTR

Το πρωτόκολλο OTR κατασκευάστηκε για να παρέχει perfect forward secrecy και deniability. Με τον όρο deniability εννοούμε ότι τα εμπλεκόμενα μέρη μπορούν να εξακριβώσουν την αυθεντικότητα των μηνυμάτων αλλά δεν μπορούν να το αποδείξουν σε έναν τρίτο.

Θα αναλυθεί το πρωτόκολλο OTR v3. [35] Αρχικά το OTR υποθέτει ότι το δίκτυο θα παραδώσει τα πακέτα με την σωστή σειρά αλλά ότι και κάποια πακέτα θα χαθούν. Για να ξεκινήσει η επικοινωνία της Alice με τον Bob πρέπει να τον ειδοποιήσει ότι επιθυμεί να χρησιμοποιήσει το OTR. Αυτό μπορεί να γίνει είτε στέλνοντας ένα OTR Query Message στον Bob είτε συμπεριλαμβάνοντας μια ειδική ετικέτα που αποτελείται από whitespace χαρακτήρες σε ένα από τα μηνύματα που θα στείλει στον Bob. Όταν ο Bob λάβει το μήνυμα θα εκκινήσει την ανταλλαγή κλειδιού (Authenticated Key Exchange). Ως AKE χρησιμοποιείται μια παραλλαγή του πρωτοκόλλου SIGMA.

Παρακάτω περιγράφεται η λειτουργεία της συγκεκριμένη έκδοση του πρωτοκόλλου SIGMA. Παρακάτω όλες οι πράξεις γίνονται modulo p όπου p ένας πρώτος αριθμός των 1536 bit. Επιπλέον τα long-term κλειδιά της Alice και του Bob συμβολίζονται με pub_A και pub_B αντίστοιχα.

Η βασική ιδέα είναι ότι τα δύο μέρη πραγματοποιούν μια ανταλλαγή κλειδιού Diffie Hellman χωρίς αυθεντικοποίηση ώστε να δημιουργήσουν ένα κρυπτογραφημένο κανάλι και μετά να αυθεντικοποιήσουν ο ένας τον άλλον εντός αυτού του καναλιού.

Τα βήματα είναι :

Bob:

1. Επιλέγει μια τυχαία τιμή r (128 bits). Η τιμή αποτελεί το κλειδί που θα χρησιμοποιηθεί για τον αλγόριθμο AES.
2. Επιλέγει μια τυχαία τιμή x (τουλάχιστον 320 bits)
3. Στέλνει στην Alice τις τιμές $AES_r(g^x), HASH(g^x)$.

Alice:

1. Επιλέγει μια τυχαία τιμή y (τουλάχιστον 320 bits)
2. Στέλνει στον Bob την τιμή g^x

Bob:

1. Εξακριβώνει ότι η τιμή g^x είναι έγκυρη δηλαδή $2 \leq g^y \leq modulus - 2$. Αυτό γίνεται για να αποκλείσουμε στοιχεία που ανήκουν στην υποομάδα με τάξη 2.
2. Υπολογίζει την τιμή $s = (g^y)^x$
3. Παράγει δύο AES κλειδιά c,c' και τέσσερα MAC κλειδιά m1, m1',m2,m2' κατακερματίζοντας(hashing) την τιμή s με διάφορους τρόπους.
4. Επιλέγει ένα σειριακό αριθμό $keyid_B$ για το κλειδί g^x
5. Υπολογίζει την τιμή $M_B = MAC_{m1}(g^x, g^y, pub_B, keyid_B)$
6. Υπολογίζει την τιμή $X_B = pub_B, keyid_B, sig_B(M_B)$
7. Στέλνει στην Alice το κλειδί r καθώς και $AES_c(X_B), MAC_{m2}\big(AES_c(X_B)\big)$

Alice:

1. Χρησιμοποιεί το κλειδί r που έλαβε για να αποκρυπτογραφήσει την τιμή g^x
2. Υπολογίζει την τιμή $HASH(g^x)$ και εξακριβώνει ότι είναι ίδια με την τιμή που έλαβε πριν

3. Ελέγχει ότι το κλειδί g^x του Bob είναι έγκυρο δηλαδή $2 \leq g^x \leq modulus - 2$
4. Υπολογίζει την τιμή $s = (g^x)^y$ η οποία πρέπει να είναι ίδια με την τιμή που υπολόγισε ο Bob
5. Υπολογίζει δύο AES κλειδιά c, c' και τέσσερα MAC κλειδιά m1, m1',m2,m2' κατακερματίζοντας(hashing) την τιμή s με τον ίδιο τρόπο που το έκανε και ο Bob
6. Με το κλειδί m2 εξακριβώνει ότι η τιμή $MAC_{m2}\big(AES_c(X_B)\big)$ είναι σωστή
7. Χρησιμοποιεί το κλειδί c για να αποκρυπτογραφήσει την τιμή $AES_c(X_B)$ και να αποκτήσει τις τιμές $X_B = pub_B, keyid_B, sig_B(M_B)$
8. Υπολογίζει την τιμή $M_B = MAC_{m1}(g^x, g^y, pub_B, keyid_B)$
9. Χρησιμοποιεί το δημόσιο κλειδί pub_B για να επαληθεύσει την ψηφιακή υπογραφή $sig_B(M_B)$
10. Επιλέγει ένα σειριακό αριθμό $keyid_A$ για το κλειδί της g^y
11. Υπολογίζει την τιμή $M_A = MAC_{m1'}(g^y, g^x, pub_A, keyid_A)$
12. Υπολογίζει $X_A = pub_A, keyid_A, sig_A(M_A)$
13. Στέλνει στον Bob τις τιμές $AES_{c'}(X_A), MAC_{m2'}\big(AES_{c'}(X_A)\big)$

Bob:

1. Χρησιμοποιεί το κλειδί m2' για να επαληθεύσει την τιμή $MAC_{m2'}\big(AES_{c'}(X_A)\big)$
2. Χρησιμοποιεί το κλειδί c' για να αποκρυπτογραφήσει την τιμή $AES_{c'}(X_A)$ για να λάβει την τιμή $X_A = pub_A, keyid_A, sig_A(M_A)$
3. Υπολογίζει την τιμή $M_A = MAC_{m1'}(g^y, g^x, pub_A, keyid_A)$
4. Χρησιμοποιεί το δημόσιο κλειδί pub_A για να επαληθεύσει την ψηφιακή υπογραφή $sig_A(M_A)$

Αν όλοι οι έλεγχοι είναι επιτυχείς τότε τα δύο μέρη μοιράζονται την σωστή τιμή s. Θα αναλύσουμε την λογική των βημάτων του πρωτοκόλλου.

Στο πρώτο βήμα ο Bob αποστέλλει την τιμή $HASH(g^x)$ αντί της τιμής g^x την οποία αποκαλύπτει παρακάτω. Αυτή η τεχνική ονομάζεται hash commitment και γίνεται ώστε κανένα να είναι αδύνατο σε κάποιο από τα δύο μέρη να επιλέξει την τιμή g^x λαμβάνοντας υπόψη του την τιμή του άλλου μέρους.

Η επιλογή της κρυπτογράφησης της τιμής g^x και η αποστολή του r σε επόμενο βήμα γίνεται λόγω του περιορισμού που θέτουν κάποια πρωτόκολλα στο μέγιστο μέγεθος των μηνυμάτων [36]. Θα μπορούσε η τιμή να μην κρυπτογραφηθεί και να αποσταλεί κανονικά στην θέση του r.

Η χρήση της MAC συνάρτησης έχει ως σκοπό να αποτρέψει μια επίθεση που ονομάζεται identity misbinding attack. Συνοπτικά στην επίθεση αυτήν ένας ενδιάμεσος ξεκινά ταυτόχρονη συνομιλία και με την Alice και με τον Bob. Στα μηνύματα της Alice προς τον Bob αφαιρεί την ψηφιακή υπογραφή της Alice τοποθετώντας την δική του, ενώ τα μηνύματα από τον Bob στην Alice δεν τροποποιούνται. Ο παρακάτω πίνακας δείχνει την ανταλλαγή των μηνυμάτων.

Alice → Eve	$g^x, Sign_A(g^x), I_A$
Eve →Bob	$g^x, Sign_E(g^x), I_E$
Bob → Eve	$g^y, Sign_B(g^y), I_B$
Eve → Alice	$g^y, Sign_B(g^y), I_B$

Πίνακας 14 Εκτέλεση της επίθεσης identity misbinding

Αυτό έχει ως αποτέλεσμα η Alice να λάβει την τιμή g^y υπογεγραμμένη από τον Bob και να υποθέσει ότι επικοινωνεί με αυτόν. Ο Bob όμως λαμβάνει την τιμή g^x υπογεγραμμένη από την Eve με αποτέλεσμα να νομίζει ότι επικοινωνεί με την Eve ενώ επικοινωνεί με την Alice. Αξίζει να σημειωθεί πως η Alice και ο Bob έχουν συμφωνήσει στο ίδιο κλειδί s. Το κλειδί που χρησιμοποιείται στην MAC συνάρτηση παράγεται ως κατακερματισμός της τιμής $s = g^{xy}$ το οποίο δεν γνωρίζει η Eve. Έτσι η χρήση της MAC εμποδίζει την παραπάνω επίθεση.

Αφού έχει ολοκληρωθεί η αρχική ανταλλαγή του κλειδιού η Alice για να στείλει ένα μήνυμα ακολουθεί τα παρακάτω βήματα

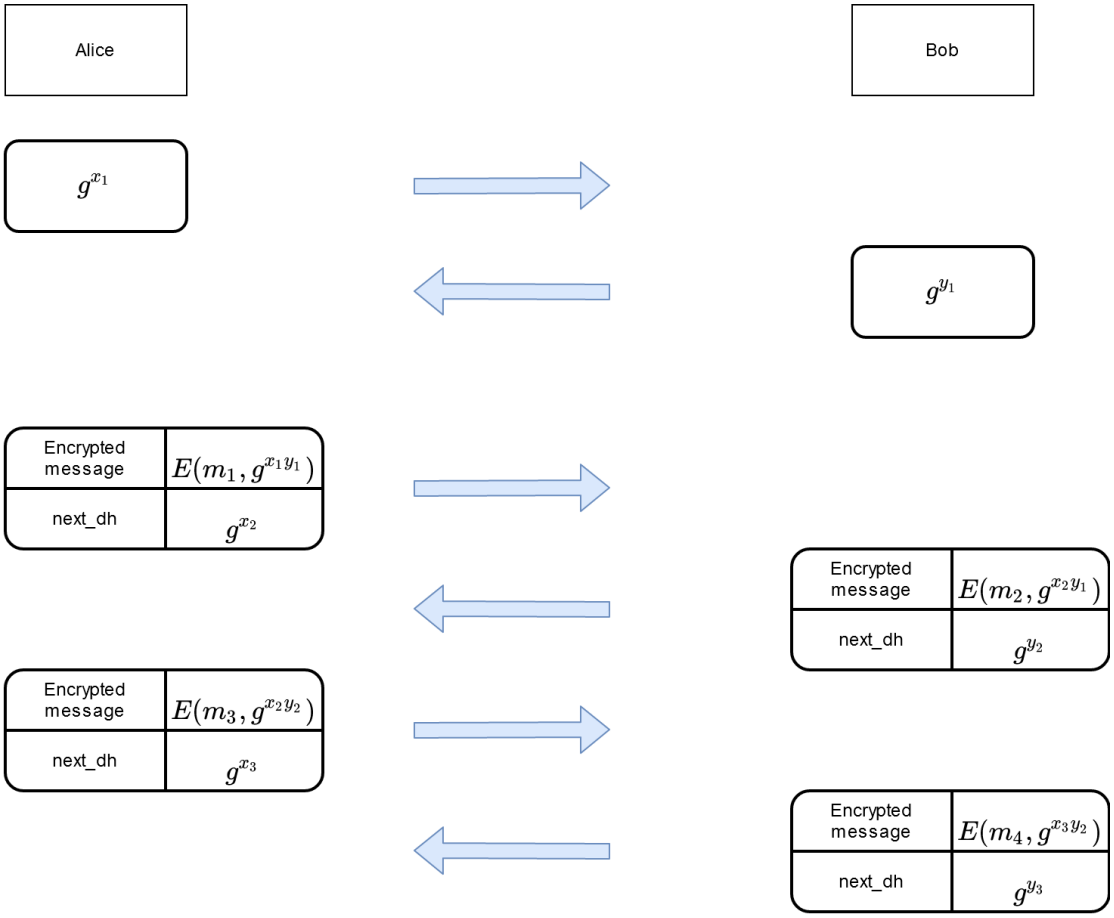
Alice

1. Επιλέγει το πιο πρόσφατο κλειδί από τα δικά της DH κλειδιά που έχει λάβει ο Bob. Γνωρίζουμε ότι ο Bob έχει λάβει το κλειδί αν το έχει χρησιμοποιήσει για να στείλει ένα μήνυμα. Αν δεν έχει στείλει κάποιο μήνυμα το κλειδί είναι το κλειδί της AKE. Έστω key_A το κλειδί και $keyid_A$ ο σειριακός αριθμός του.
2. Αν το παραπάνω κλειδί είναι το πιο πρόσφατο κλειδί της Alice τότε παράγει ένα καινούριο DH κλειδί next_dh με σειριακό αριθμό $keyid_A + 1$.
3. Επιλέγει το πιο πρόσφατο DH κλειδί που έχει λάβει από τον Bob (από κάποιο μήνυμα ή από την AKE). Έστω key_B το κλειδί και $keyid_B$ ο σειριακός αριθμός του.
4. Υπολογίζει το κοινό μυστικό που χρησιμοποιώντας τα κλειδιά key_A, key_B (όπως σε μια ανταλλαγή Diffie Helman) και παράγει το AES κλειδί ek το οποίο θα χρησιμοποιηθεί για την κρυπτογράφηση του μηνύματος. Τέλος υπολογίζει και το κλειδί της MAC συνάρτησης mk.
5. Συγκεντρώνει τα MAC κλειδιά τα οποία χρησιμοποιήθηκαν σε παλαιότερα μηνύματα σε μια λίστα oldmackeys. Αυτά τα κλειδιά δεν θα ξαναχρησιμοποιηθούν καθώς τα αντίστοιχα DH κλειδιά δεν θα είναι τα πιο πρόσφατα.
6. Επιλέγει έναν μετρητή ctr έτσι ώστε η τριάδα (key_A, key_B, ctr) να μην είναι ίδια σε κανένα άλλο μήνυμα που θα στείλει η Alice στον Bob.
7. Κατασκευάζει το μήνυμα προς αποστολή $T_A = \left(keyid_A, keyid_B, next_dh, ctr, AES - CTR_{ek,ctr}(msg)\right)$
8. Στέλνει στον Bob: $T_A, MAC_mk(T_A), oldmackeys$

Bob

1. Υπολογίζει το κοινό μυστικό χρησιμοποιώντας τα κλειδιά με σειριακό αριθμό $keyid_A, keyid_B$ και παράγει τα κλειδιά ek, mk.
2. Επαληθεύει την τιμή $MAC_mk(T_A)$ χρησιμοποιώντας το κλειδί mk.
3. Αποκρυπτογραφεί το κρυπτοκείμενο $AES - CTR_{ek,ctr}(msg)$

Όπως φαίνεται και στο παρακάτω σχήμα το κλειδί κρυπτογράφησης ανανεώνεται σε κάθε μήνυμα. Κάθε μήνυμα εκτός από το κρυπτοκείμενο φέρει και το δημόσιο DH κλειδί που θα χρησιμοποιηθεί για να παραχθεί το κλειδί κρυπτογράφησης για τα επόμενα μηνύματα.



Εικόνα 2 Απεικόνιση του otr ratchet. Θεωρούμε ότι τα δυο μέρη έχουν ανταλλάξει τα κλειδιά g^{x_1}, g^{x_2}

Ωστόσο για να επιτύχουμε forward secrecy τα δύο μέρη πρέπει να διαγράφουν τα παλιά κλειδιά μετά από κάθε ανταλλαγή. Ιδανικά αν η Alice στείλει στον Bob το κλειδί x_n θα μπορούσε να διαγράψει το κλειδί x_{n-1} . Μπορεί όμως να υπάρχουν μηνύματα κρυπτογραφημένα με το κλειδί x_{n-1}τα οποία δεν έχουν φτάσει. Επομένως η Alice πρέπει να κρατήσει το προηγούμενο κλειδί μέχρι να λάβει ένα μήνυμα από τον Bob που να χρησιμοποιεί το καινούριο κλειδί x^n με την προϋπόθεση ότι τα μηνύματα φτάνουν στην σωστή σειρά. Αν υπάρχει περίπτωση τα μηνύματα να φτάσουν με λανθασμένη σειρά τότε η Alice μπορεί να διατηρεί το κλειδί x_{n-1} για κάποιο χρονικό διάστημα μετά την λήψη μηνύματος από τον Bob ώστε να μπορεί να αποκρυπτογραφήσει τυχόν καθυστερημένα μηνύματα.

Αξίζει να σημειωθεί ότι η Alice δεν παράγει καινούριο κλειδί αν δεν λάβει απάντηση από τον Bob. Όταν λάβει απάντηση από τον Bob ότι χρησιμοποιεί το κλειδί x_n διαγράφει το κλειδί x_{n-1} και παράγει το κλειδί x_{n+1} το οποίο θα επισυνάψει στο επόμενο μήνυμα. Για μειωθεί το χρονικό διάστημα που διατηρούνται τα κλειδιά λόγω του ότι ο Bob δεν έχει στείλει απάντηση, ο Bob ανά διαστήματα να στέλνει ένα κενό μήνυμα ώστε να επιβεβαιώσει ότι γνωρίζει το καινούριο κλειδί της Alice.

Η συνεχής ανταλλαγή κλειδιών δίνει στο OTR την ικανότητα της αυτοΐασης (self-healing). Αυτό σημαίνει ότι αν κάποιο εφήμερο κλειδί αποκαλυφθεί δεν θα επηρεάσει την εμπιστευτικότητα των επόμενων μηνυμάτων καθώς κοινό κλειδί κρυπτογράφησης θα έχει αλλάξει.

Τέλος το πρωτόκολλο OTR παρέχει την δυνατότητα της χρήσης του πρωτοκόλλου SMP (Socialist Millionaires Protocol) [36] για την αυθεντικοποίηση των δύο χρηστών. Μια σημαντική ιδιότητα του SMP είναι ότι δεν αποκαλύπτει καμία πληροφορία για τα μυστικά των χρηστών πάρα μόνο αν αυτά είναι ίδια.

Αυτό επιτρέπει να χρησιμοποιηθούν μυστικά με μικρή εντροπία. Στόχος του πρωτοκόλλου είναι να εξακριβωθεί ότι τα μυστικά είναι ίδια.

Το πρωτόκολλο SMP εκτελείται ως εξής. Όλοι οι υπολογισμοί εκτελούνται σε μια ομάδα τάξης q με q έναν πρώτο αριθμό. Ο γεννήτορας της ομάδας είναι γνωστός και ισούται με $g_1 = 2$. Με x συμβολίζεται το μυστικό της Alice και με y το μυστικό του Bob. Για την εκτέλεση του πρωτοκόλλου απαιτούνται δύο επιπλέον γεννήτορες g_2, g_3 .

- Η Alice επιλέγει μια τυχαία τιμή a_2 και ο Bob μια τυχαία τιμή b_2. Ανταλλάσσουν τις τιμές ${g_1}^{a_2}, {g_1}^{b_2}$ και υπολογίζουν την τιμή $g_2 = {g_1}^{a_2 b_2}$
- Η ίδια διαδικασία επαναλαμβάνεται για τις τιμές a_3, b_3. Και οι δύο υπολογίζουν την τιμή $g_3 = {g_1}^{a_3 b_3}$

Για να μην είναι δυνατή η εύρεση των τιμών x,y ακολουθείται μια διαδικασία τύφλωσης (blinding).

- Η Alice επιλέγει μια τυχαία τιμή a και υπολογίζει $(P_a, Q_a) = ({g_3}^a, {g_1}^a {g_2}^x)$
- Παρόμοια ο Bob επιλέγει μια τυχαία τιμή b και υπολογίζει $(P_b, Q_b) = ({g_3}^b, {g_1}^b {g_2}^y)$
- Οι τιμές P_a, Q_a, P_b, Q_b ανταλλάσσονται.

Ο έλεγχος αν x=y γίνεται ως εξής:

- Η Alice χρησιμοποιεί την τιμή a_3 και υπολογίζει την τιμή $R_a = \left(\frac{Q_a}{Q_b}\right)^{a_3}$
- Αντίστοιχα ο Bob υπολογίζει την τιμή $R_b = \left(\frac{Q_a}{Q_b}\right)^{b_3}$
- Οι τιμές ανταλλάσσονται
- Και οι δύο υπολογίζουν την τιμή $R_{ab} = R_a^{b_3} = R_b^{a_3}$
- Ελέγχουν αν $R_{ab} = \left(\frac{P_a}{P_b}\right)$

Ο παραπάνω έλεγχος αρκεί καθώς $R_{ab} = \left(\frac{Q_a}{Q_b}\right)^{a_3 b_3} = \left(g_1^{a-b} g_2^{x-y}\right)^{a_3 b_3} = g_3^{a-b} g_2^{(x-y) a_3 b_3} = \left(\frac{P_a}{P_b}\right) \left(g_2^{a_3 b_3}\right)^{(x-y)}$

Το πρωτόκολλο OTR ελέγχει το hash του αναγνωριστικού της συνόδου (session id), τα fingerprints των δύο χρηστών και ένα κοινό μυστικό που μοιράζονται οι δύο χρήστες. Αν κάποιος παρεμβάλλεται στην επικοινωνία τα fingerprints θα είναι διαφορετικά. Αν η εκτέλεση του SMP είναι επιτυχής τότε δεν υπάρχει κάποιος ενδιάμεσος στην επικοινωνία. Ακόμα επειδή το πρωτόκολλο SMP δεν παρέχει πληροφορία για το κοινό μυστικό, κάθε δοκιμή του μυστικού από τον επιτιθέμενο δεν μπορεί να γίνει offline. Αναγκαστικά θα πρέπει να γίνεται αλληλοεπιδρώντας με τον χρήστη.

2.6.3 Signal

Το πρωτόκολλο Signal μπορεί να διαιρεθεί σε τρία στάδια.

- Την αρχική ανταλλαγή κλειδιών που ονομάζεται X3DH [37] (extended triple Diffie-Hellman) η οποία συνδυάζει το long-tern medium-term και τα εφήμερα Diffie-Hellman κλειδιά για να παράγει ένα κοινό root κλειδί.
- Ένα ratchet στάδιο στο οποίο οι χρήστης στέλνουν εναλλάξ εφήμερα κλειδιά για να δημιουργήσουν chain keys τα οποία παρέχουν forward-secrecy στο πρωτόκολλο.
- Ένα ratchet στάδιο στο οποίο χρησιμοποιούνται συναρτήσεις παραγωγής κλειδιών για να παραχθούν συμμετρικά κλειδιά από τα chain keys.

Το πρωτόκολλο X3DH χρησιμοποιεί την έννοια των prekeys για να επιτρέψει την αποστολή ασύγχρονων μηνυμάτων. Αρχικά ο Bob αποστέλλει στον εξυπηρετητή μια ομάδα κλειδιών που έχει παράγει που περιέχει:

- Το κλειδί που προσδιορίζει την ταυτότητα του Bob (identity key IK_B)
- Ένα υπογεγραμμένο prekey SPK_B
- Την υπογραφή για το παραπάνω κλειδί (prekey signature) $Sig(IK_B, Encode(SPK_B))$
- Prekeys μιας χρήσης (OPK_B^1, OPK_B^2, OPK_B^3, ...)

Ο Bob αποστέλλει το identity key μια φορά αλλά μπορεί να χρειαστεί να αποστείλει prekeys μιας χρήσης όταν αρχίσουν να εξαντλούνται. Το SPK_B αλλάζει ανά τακτά διαστήματα και ο Bob πρέπει να διαγράψει τα προηγούμενα για forward secrecy. Τα prekeys μιας χρήσης διαγράφονται με την λήψη του κάθε μηνύματος.

Για να ξεκινήσει η ανταλλαγή η Alice ζητά από τον εξυπηρετητή prekey bundle που περιέχει αυτά που περιέχει το IK_B , SPK_B την υπογραφή του prekey και αν είναι διαθέσιμο ένα prekey μιας χρήσης.

Η Alice ελέγχει αν η υπογραφή του prekey είναι έγκυρη μετά παράγει ένα εφήμερο ζεύγος κλειδιών με EK_A να συμβολίζει το δημόσιο κλειδί. Στην περίπτωση που το bundle δεν περιέχει κάποιο prekey ο υπολογισμός του κοινού κλειδιού γίνεται ως εξής:

$$DH1 = DH(IK_A, SPK_B)$$
$$DH2 = DH(EK_A, IK_B)$$
$$DH3 = DH(EK_A, SPK_B)$$
$$SK = KDF(DH1 \,||\, DH2 \,||\, DH3)$$

Αν περιέχει κάποιο prekey μιας χρήσης τότε γίνεται μια επιπλέον ανταλλαγή Diffie-Helman.

$$DH4 = DH(EK_A, OPK_B)$$
$$SK = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, DH4)$$

Οι ποσότητες DH1, DH2 παρέχουν αμοιβαία αυθεντικοποίηση στα 2 μέρη ενώ οι ποσότητες DH3, DH4 παρέχουν forward secrecy στο πρωτόκολλο.

Αφού υπολογίσει το κλειδί SK διαγράφει το εφήμερο ιδιωτικό κλειδί της και τις τιμές DH1, DH2, DH3, DH4. Έπειτα υπολογίζει την ακολουθία AD = Encode(IK_A) || Encode(IK_B). Τέλος αποστέλλει ένα μήνυμα στον Bob το οποίο περιέχει το identity key IK_A της Alice, το εφήμερο κλειδί EK_A , ένα αναγνωριστικό που περιλαμβάνει ποιο prekey χρησιμοποιήθηκε και ένα κρυπτοκείμενο κρυπτογραφημένο χρησιμοποιώντας κάποιο AEAD σχήμα κρυπτογράφησης με κλειδί το SK και την τιμή AD ως associated data.

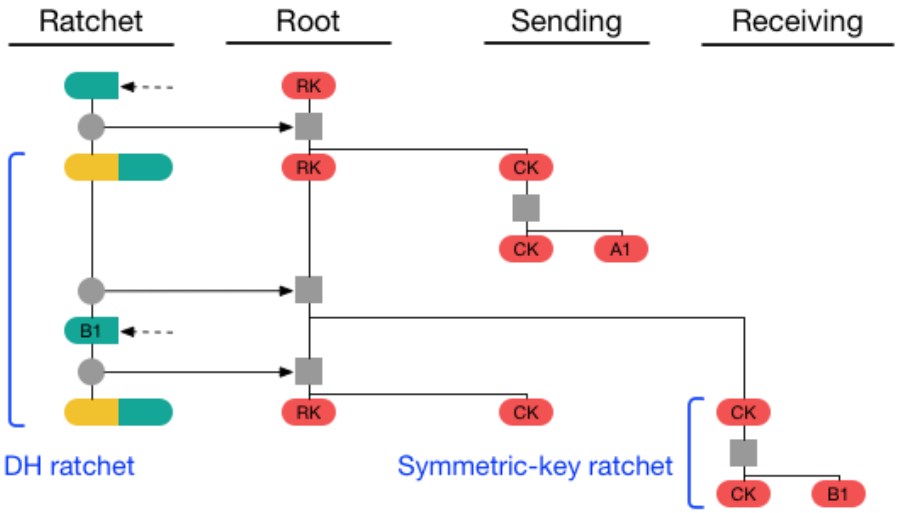
Ο Bob αφού λάβει το μήνυμα χρησιμοποιεί τα κλειδιά που υποδεικνύονται στο μήνυμα για να παράξει το κοινό κλειδί SK, κατασκευάζει την ακολουθία AD και αποκρυπτογραφεί το μήνυμα. Αν η αποκρυπτογράφηση είναι επιτυχής το κλειδί το κοινό κλειδί SK θα χρησιμοποιηθεί στην εκτέλεση του πρωτόκολλου Double Ratchet.

Το πρωτόκολλο Double Ratchet [38] συνδυάζει και τις δύο μεθόδους με το να χρησιμοποιεί dh ratchets ως είσοδο σε KDF ratchets. Υπάρχουν τρεις αλυσίδες η ριζική αλυσίδα (root chain), η αλυσίδα αποστολής(sending chain) και η αλυσίδα παραλαβής(receiving chain). Καθώς τα δυο μέρη ανταλλάσσουν μηνύματα ανταλλάσσουν και δημόσια κλειδιά Diffie-Helman. Το αποτέλεσμα της ανταλλαγής αποτελεί είσοδο στην ριζική αλυσίδα. Τα αποτελέσματα της ριζικής αλυσίδας αποτελούν τα καινούρια κλειδιά της KDF που χρησιμοποιείται από την αλυσίδας αποστολής και της αλυσίδας παραλαβής. Αυτό αποτελεί το Diffie-Helman ratchet.

Κάθε φορά που αποστέλλεται ή λαμβάνεται ένα μήνυμα η αλυσίδα προχωρά ένα βήμα και το αποτέλεσμα της χρησιμοποιείται για την κρυπτογράφηση του μηνύματος. Αυτή η διαδικασία αποτελεί το symmetric-key ratchet.

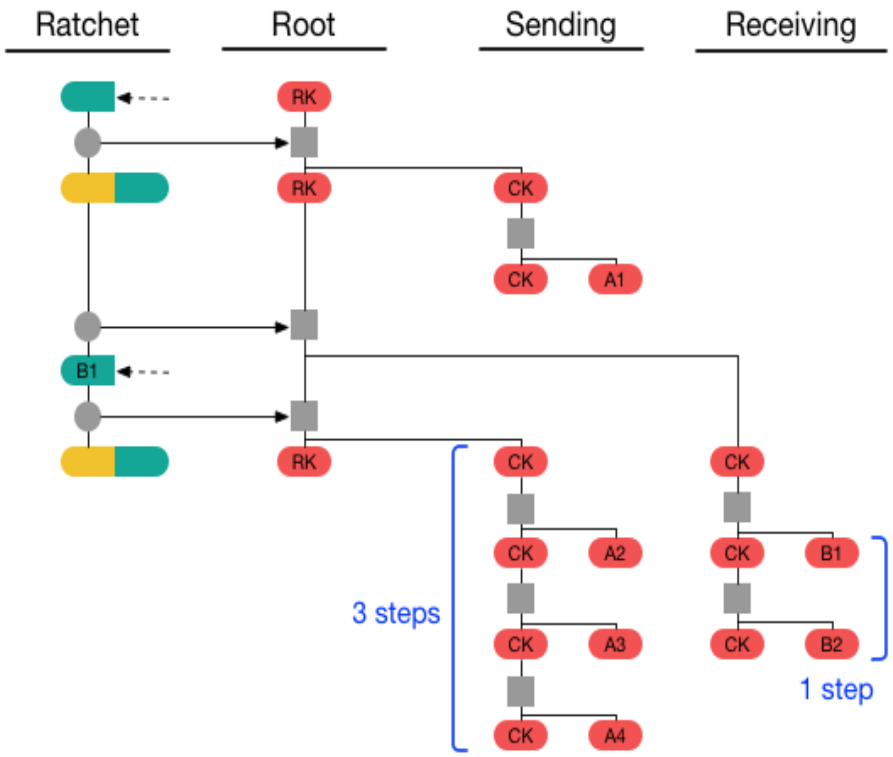
Επομένως όταν αποστέλλεται ή λαμβάνεται ένα μήνυμα χρησιμοποιείται ένα συμμετρικό βήμα στο ratchet για να παραχθεί το κλειδί. Όταν ληφθεί ένα δημόσιο κλειδί χρησιμοποιείται ένα βήμα Diffie-Helman ratchet πριν το συμμετρικό ratchet.

Για παράδειγμα ας υποθέσουμε ότι η Alice και ο Bob έχουν παράγει το κοινό κλειδί που αποτελεί το αρχικό root κλειδί και έχει λάβει και το DH κλειδί του Bob. Η Alice αρχικά θα δημιουργήσει ένα καινούριο ratchet ζεύγος κλειδιών (χρησιμοποιώντας το κλειδί του Bob) και θα το δώσει ως είσοδο στην root KDF για να παράγει το καινούριο root key και το κλειδί της αλυσίδας αποστολής. Αν η Alice θέλει να στείλει το μήνυμα A1 θα εφαρμόσει ένα symmetric-key ratchet βήμα στην αλυσίδα αποστολής για να δημιουργήσει το κλειδί κρυπτογράφησης. Το προηγούμενο κλειδί μπορεί να διαγραφεί. Στην περίπτωση που η Alice λάβει ένα μήνυμα από τον Bob (έστω B1) αυτό θα περιέχει ένα καινούριο DH κλειδί. Η Alice θα εφαρμόσει ένα DH ratchet βήμα για να παράγει τα καινούρια κλειδιά της αλυσίδας αποστολής και της αλυσίδας λήψης. Έπειτα εφαρμόζει ένα βήμα symmetric-key ratchet στη αλυσίδα παραλαβής για να παράγει το κλειδί για το μήνυμα B1.



Εικόνα 3 Παραγωγή κλειδιών από την Alice

Έστω ότι η Alice στείλει ένα νέο μήνυμα A2 και λάβει ένα μήνυμα B2 με το παλιό DH κλειδί του Bob. Έπειτα η Alice στέλνει τα μηνύματα A3, A4. Η παραπάνω ακολουθία μηνυμάτων θα έχει ως αποτέλεσμα η αλυσίδα αποστολής της Alice να προχωρήσει 3 βήματα και η αλυσίδα λήψης να προχωρήσει ένα βήμα όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 4 Ή αλυσίδα κλειδιών μετά από 3 βήματα

Για να είναι δυνατό να χειριστεί καθυστερημένα μηνύματα κάθε μήνυμα φέρει σαν κεφαλίδα τον αριθμό του μηνύματος στην αλυσίδα και το μήκος (δηλαδή τον αριθμό των κλειδιών των μηνύματα) της προηγούμενης αλυσίδας αποστολής. Αυτό επιτρέπει στον παραλήπτη να υπολογίσει το κλειδί που απαιτείται για το τωρινό μήνυμα αλλά και να αποκρυπτογραφήσει καθυστερημένα μηνύματα διατηρώντας τα κλειδιά που απαιτούνται.

2.6.4 Identity based encryption

Ο αρχικός σκοπός αυτού του είδους κρυπτογράφησης όπως προτάθηκε από τον Shamir [39] ήταν να απλοποιήσει την διαχείριση των πιστοποιητικών στα συστήματα ηλεκτρονικού ταχυδρομείου. Η βασική ιδέα είναι ότι το δημόσιο κλειδί κάποιου χρήστη μπορεί να είναι μια αυθαίρετη συμβολοσειρά όπως για παράδειγμα η διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη.

Το σύστημα δουλεύει ως εξής. Όταν η Alice θέλει να στείλει ένα μήνυμα στον Bob κρυπτογραφεί τον μήνυμα χρησιμοποιώντας ως δημόσιο κλειδί την διεύθυνση του Bob πχ bob@example.com χρησιμοποιώντας κάποιες δημόσιες παραμέτρους που χαρακτηρίζουν το σύστημα. Δεν χρειάζεται να αποκτήσει το δημόσιο κλειδί του Bob. Ο Bob για να αποκρυπτογραφήσει το μήνυμα επικοινωνεί με το PKG(private key generator) αποδεικνύει την ταυτότητά του και παραλαμβάνει το ιδιωτικό κλειδί του και έτσι αποκρυπτογραφεί το μήνυμα. Η Alice μπορεί να αποστείλει ένα μήνυμα ακόμα και αν ο Bob δεν έχει δημιουργήσει το ζεύγος κλειδιών του κάτι που απαιτείται με την υπάρχουσα υποδομή του ηλεκτρονικού ταχυδρομείου.

Το συγκεκριμένο σχήμα [40] ορίζεται από 4 αλγορίθμους:

- Setup: Δέχεται ως είσοδο μια παράμετρο k και δημιουργεί τις δημόσιες παραμέτρους του συστήματος και το master κλειδί το οποίο πρέπει να παραμείνει κρυφό και το γνωρίζει μόνο το

PKG. Οι δημόσιες παράμετροι οι οποίες είναι γνωστές από όλους περιλαμβάνουν το πως αναπαρίσταται ο σύνολο των μηνυμάτων και πως το αναπαρίσταται ο χώρος των κρυπτοκειμένων.

- Extract: Δέχεται ως είσοδο τις δημόσιες παραμέτρους και το master κλειδί και μια οποιαδήποτε συμβολοσειρά ID και επιστρέφει ένα ιδιωτικό κλειδί d. Η παραπάνω συμβολοσειρά αποτελεί το δημόσιο κλειδί και το ιδιωτικό κλειδί που δημιουργήθηκε είναι το αντίστοιχο ιδιωτικό κλειδί.
- Encrypt: Δέχεται ως είσοδο το μήνυμα, τις δημόσιες παραμέτρους, την συμβολοσειρά ID και παράγει το κρυπτοκείμενο.
- Decrypt: Δέχεται ως είσοδο τις δημόσιες παραμέτρους, το κρυπτοκείμενο, και το ιδιωτικό κλειδί d και εξάγει το αποκρυπτογραφημένο μήνυμα.

Το συγκεκριμένο σχήμα βασίζεται στην έννοια των bilinear maps. Έστω G_1 G_2 δυο ομάδες τάξης q και μια απεικόνιση $e: G_1 \times G_1 \to G_2$ για την οποία πρέπει να ισχύουν οι ιδιότητες:

- Να είναι διγραμμική δηλαδή $e(aP, bQ) = e(P, Q)^{ab}$ για όλα τα P,Q που ανήκουν στο G_1 και όλα τα a,b που ανήκουν στο G.
- Η συνάρτηση θα πρέπει να μην απεικονίζει όλα τα ζεύγη G_1 X G_1 στο ταυτοτικό στοιχείο της G_2
- Να υπάρχει ένας αποδοτικός αλγόριθμος για να υπολογιστεί η απεικόνιση e(P,Q) για οποιοδήποτε P,Q που ανήκει στο G_1

Οι αλγόριθμοι γίνονται:

- Setup: Έστω k (ορίζει πόσα bit θα είναι η τάξη q) μια παράμετρος ασφάλειας στο Z^+

1. Σύμφωνα με την παράμετρο k παράγουμε ένα πρώτο q και δυο ομάδες G_1 , G_2 τάξης q και μια διγραμμική απεικόνιση e που ικανοποιεί τις παραπάνω ιδιότητες. Έπειτα επιλέγουμε ένα τυχαίο γεννήτορα P του G_1 .
2. Διαλέγουμε ένα τυχαίο s στο Z_q^* και θέτουμε ως P_{pub} =sP
3. Επιλέγουμε δυο συναρτήσεις κατακερματισμού $H_1: \to G_1^*$ και $H_2: \to \{0,1\}^n$ για κάποιο n. Ο χώρος των μηνυμάτων M ορίζεται ως $M = \{0,1\}^n$. Ο χώρος των κρυπτογραφημένων μηνυμάτων ορίζεται ως $C = G_1^* \times \{0,1\}^n$. Τέλος οι δημόσιες παράμετροι είναι params=(q, G_1 , G_2, e, n, P, P_{pub},H_1, H_2) και το master κλειδί είναι το $s \in Z_q^*$

- Extract: Για κάθε συμβολοσειρά $ID \in \{0,1\}^*$ ο αλγόριθμος υπολογίζει $Q_{ID} = H_1(ID) \in G_1^*$, και θέτει το ιδιωτικό κλειδί ίσο με $d_{id} = sQ_{id}$
- Encrypt: Για να κρυπτογραφηθεί ένα μήνυμα υπολογίζεται η τιμή Q_{ID} =H_1(ID) και μια τυχαία τιμή $r \in z_q^*$. Το κρυπτοκείμενο υπολογίζεται ως $C = \left(rP, M \oplus H2(g_{ID}^r)\right)$ με $g_{ID} = e\left(Q_{ID}, P_{pub}\right) \in G_2^*$
- Decrypt: Έστω C=(U,V) το κρυπτογράφημα που προέκυψε. Για την αποκρυπτογράφηση υπολογίζουμε $V \oplus H_2\left(e(d_{ID}, U)\right) = M$. Η αποκρυπτογράφηση λειτουργεί καθώς $e(d_{ID}, U) = e(sQ_{ID}, rP) = e(Q_{ID}, P)^{sr} = e\left(Q_{ID}, P_{pub}\right)^r = g_{id}^r$

Η ασφάλεια της κρυπτογράφησης βασίζεται στο Bilinear Diffie-Hellman Assumption. Δηλαδή δοθέντος των P, aP, bP, cP για $a, b, c \in Z_q^*$ είναι δύσκολο να υπολογιστεί η τιμή $W = e(P, P)^{abc} \in G_2$

Αξίζει να σημειωθεί ότι το παραπάνω σχήμα δεν είναι ασφαλές σε chosen ciphertext επιθέσεις. Μπορεί να γίνει ασφαλές χρησιμοποιώντας την παρακάτω τεχνική των Fujisaki-Okamoto [41]. Ως E_{pk}=(M;r) ορίζουμε την κρυπτογράφηση του μηνύματος M με το κλειδί pk χρησιμοποιώντας μια τυχαία ποσότητα r. Οι Fujisaki-Okamoto χρησιμοποιούν ένα υβριδικό σύστημα E^{hy} που ορίζεται ως: $E_{pk}^{hy}(M) = E_p k\left(\sigma; H_3(\sigma, M)\right), H_4(\sigma) \oplus M >$. Η τιμή σ επιλέγεται τυχαία και H_3,H_4 είναι συναρτήσεις κατακερματισμού. Το παραπάνω σχήμα είναι ασφαλές από chosen ciphertext επιθέσεις. Με τις παραπάνω προσθήκες οι αλγόριθμοι γίνονται ως:

- Setup: Εκτός από ότι αναφέρθηκε παραπάνω απαιτούνται δύο επιπλέον συναρτήσεις οι οποίες ορίζονται ως: H_3:$\{0,1\}^n \times \{0,1\}^n \to Z_q^*$ και H_4:$\{0,1\}^n \to \{0,1\}^n$
- Extract: Δεν απαιτείται κάποια τροποποίηση
- Encrypt: Επιλέγουμε ένα τυχαίο $\sigma \in \{0,1\}^n$ και $r = H_3(\sigma, M)$, $C = \left(rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma)\right)$ με $g_l D = E(Q_{ID}, P_{PUB}) \in G_2$
- Decrypt: Έστω $C = (U, V, W)$. Αν $U \notin G_1^*$ τότε το μήνυμα απορρίπτεται. Για την αποκρυπτογράφηση ακολουθούνται τα παρακάτω βήματα:
    - Υπολογίζεται η τυχαία τιμή $\sigma = V \oplus H_2\left(e(d_{ID}, U)\right)$
    - Αποκρυπτογραφείται το μήνυμα $M = W \oplus H_4(\sigma)$
    - Θέτουμε $r = H_3(\sigma, M)$. Αν δεν ισχύει $U = rP$ το μήνυμα απορρίπτεται.
    - Το M είναι το αποκρυπτογραφημένο μήνυμα

2.6.5 Hierarchical identity based encryption

Η ιεραρχική IBE (HIBE) αποτελεί μια γενίκευση της IBE στην οποία παρατηρείται μια ιεραρχική οργάνωση των κλειδιών. Ένα PKG στο επίπεδο κ μπορεί να δημιουργήσει κλειδιά για τους απογόνους του αλλά δεν μπορεί για άλλους κόμβους .

Μια ενδιαφέρουσα εφαρμογή της HIBE είναι η forward secure κρυπτογράφηση. Το συγκεκριμένο σχήμα [42] επιτρέπει στους χρήστες να δημοσιεύσουν ένα δημόσιο κλειδί το οποίο δεν χρειάζεται να αλλάξουν και να ανανεώνουν τα ιδιωτικά κλειδιά ώστε ένα μήνυμα κρυπτογραφημένο την περίοδο ν να μην μπορεί να αποκρυπτογραφηθεί από ένα κλειδί κάποιας περιόδου ν'>ν. Για να παρέχει 2^t χρονικές περιόδους χρησιμοποιείται HIBE με βάθος t με τις ταυτότητες (identities) να είναι δυαδικά διανύσματα μήκους t. Για να κρυπτογραφήσει κάποιος ένα μήνυμα την χρονική στιγμή n πρέπει να χρησιμοποιήσει ως ταυτότητα την ταυτότητα που αντιστοιχεί στον n-οστό κόμβο ενός δυαδικού δένδρου βάθους t.

Ένα μειονέκτημα που έχουν τα HIBE συστήματα όταν χρησιμοποιούνται με αυτόν τον τρόπο ότι το μέγεθος των κρυπτοκειμένων είναι O(t) και το μέγεθος των ιδιωτικών κλειδιών είναι O(t^2). Έχει προταθεί ένα σχήμα [43] το οποίο επιτρέπει σταθερός μέγεθος κρυπτοκειμένων και σταθερό κόστος αποκρυπτογράφησης για οποιανδήποτε αριθμό χρονικών στιγμών.

Θα εξετάσουμε πως δομείται ένα HIBE σύστημα. Παρόμοια με ένα IBE σύστημα ένα HIBE σύστημα αποτελείται και αυτό από τέσσερις αλγορίθμους. Ωστόσο στην περίπτωση των HIBE συστημάτων οι ταυτότητες των χρηστών είναι διανύσματα με ένα διάνυσμα διάστασης k να αναπαριστά μια ταυτότητα σε βάθος κ. Ο Setup αλγόριθμος δημιουργεί τις παραμέτρους του συστήματος και το master κλειδί. Το master κλειδί μπορεί να θεωρηθεί ως το κλειδί με βάθος 0 όπως κατά αναλογία ένα IBE σύστημα είναι ένα HIBE σύστημα με όλες τις ταυτότητες να είναι σε βάθος 1. Ο αλγόριθμος Keygen δέχεται ως είσοδο μια ταυτότητα $ID = (I_1, \dots, I_k)$ σε βάθος k και το ιδιωτικό κλειδί $d_{ID|k-1}$ της γονικής ταυτότητας $ID_{|k-1} = (I_1, \dots, I_{k-1})$ για βάθος k-1 και παράγει το ιδιωτικό κλειδί d_{ID} για την ταυτότητα ID. Ο αλγόριθμος κρυπτογράφησης κρυπτογραφεί το μήνυμα για μια ταυτότητα χρησιμοποιώντας τις παραμέτρους του συστήματος. Ο αλγόριθμος αποκρυπτογράφησης αποκρυπτογραφεί χρησιμοποιώντας το αντίστοιχο ιδιωτικό κλειδί.

Αναλυτικότερα ας εξετάσουμε πως λειτουργεί το [43]. Έστω G μια ομάδα πρώτης τάξης p και $e: G \times G \to G_1$ μια διγραμμική απεικόνιση. Τα δημόσια κλειδιά (ταυτότητες) στο βάθος k είναι διανύσματα στοιχείων του $\left(Z_p^*\right)^k$ και $ID = (I_1, \dots, I_k) \in \left(Z_p^*\right)^k$

- Setup: Για την δημιουργία των παραμέτρων του συστήματος με μέγιστο βάθος l επιλέγεται ένας γεννήτορας $g \in G$ μια τυχαία ποσότητα $a \in Z_p$ και θέτουμε $g_1 = g^a$ Έπειτα επιλέγουμε τυχαία στοιχεία $g_2, g_3, h_1, \dots, h_l \in G$. Οι παράμετροι του συστήματος είναι οι τιμές $g, g_1, g_2, , g_3, h_1, \dots, h_l$ και το master κλειδί η τιμή g_2^a

- Keygen ή extract: Για την δημιουργία ενός ιδιωτικού κλειδιού d_{ID} για μια ταυτότητα ID σε βάθος $k \leq l$ επιλέγουμε μια τυχαία τιμή $r \in Z_p$ και το ιδιωτικό κλειδί ορίζεται ως $d_{ID} = \left(g_2^a \cdot \left(h_1^{I_1} \dots h_k^{I_k} \cdot g_3\right)^r, g^r, h_{k+1}^r, \dots, h_l^r\right) \in G^{2+l-k}$

  Αξίζει να σημειωθεί πως το μέγεθος του ιδιωτικού κλειδιού μειώνεται όσο αυξάνει το βάθος.

  Το ιδιωτικό κλειδί για μια ταυτότητα μπορεί να παραχθεί από το κλειδί της γονικής ταυτότητας $d_{ID|k-1} = \left(g_2^a \cdot \left(h_1^{I_1} \dots h_{k-1}^{I_{k-1}} \cdot g_3\right)^{r'}, g^{r'}, h_k^{r'}, \dots, h_l^{r'}\right) = (a_0, a_1, b_k, \dots, b_l)$ με τον παρακάτω τρόπο: επιλέγουμε μια τυχαία τιμή $t \in Z_p$ και παράγουμε το ιδιωτικό κλειδί ως $d_{ID} = \left(a_0 \cdot b_k^{I_k} \cdot \left(h_1^{I_1} \dots h_k^{I_k} \cdot g_3\right)^t, a_1 \cdot g^t, b_{k+1} \cdot h_{k+1}^t, \dots, b_l \cdot h_l^t\right)$ με $r = r' + t \in Z_p$
- Encrypt: Για να κρυπτογραφήσουμε ένα μήνυμα $M \in G_1$ χρησιμοποιώντας το κλειδί $ID = (I_1, \dots, I_k) \in \left(Z_p\right)^k$ επιλέγουμε μια τυχαία τιμή $s \in Z_p$ και ως έξοδο έχουμε το κρυπτοκείμενο $CT = \left(e(g_1, g_2)^s \cdot M, g^s, \left(h_1^{I_1} \dots h_k^{I_k} \cdot g_3\right)^s\right) \in G_1 \times G^2$
- Decrypt: Για να αποκρυπτογραφήσουμε ένα κρυπτογραφημένο μήνυμα CT=(A,B,C) χρησιμοποιώντας το κλειδί $d_{ID} = (a_0, a_1, b_{k+1}, \dots, b_l)$ έχουμε:

$$M = \frac{A \cdot e(a_1, C)}{e(B, a_0)}$$

Το παραπάνω ισχύει καθώς

$$\frac{e(a_1, C)}{e(B, a_0)} = \frac{e\left(g^r, \left(h_1^{I_1} \dots h_k^{I_k} \cdot g_3\right)^s\right)}{e\left(g^s, g_2^a \left(h_1^{I_1} \dots h_k^{I_k} \cdot g_3\right)^r\right)} = \frac{1}{e(g, g_2)^{sa}} = \frac{1}{e(g_1, g_2)^s}$$

Μπορούμε να παρατηρήσουμε ότι για οποιοδήποτε βάθος των ταυτοτήτων το κρυπτοκείμενο περιέχει τρία στοιχεία και η αποκρυπτογράφηση απαιτεί δύο κλήσεις της διγραμμικής απεικόνισης. Επιπλέον στην περίπτωση της κρυπτογράφησης το $e(g_1, g_2)$ μπορεί να προϋπολογιστεί ώστε να μην χρειάζεται η διγραμμική απεικόνιση.

Η ασφάλεια αυτού του σχήματος βασίζεται στην Bilinear Diffie-Helman inversion assumption. Έστω Γ μια διγραμμική ομάδα τάξης p, $w \in G$ ένας γεννήτορας και $\beta \in Z_p^*$. Το πρόβλημα l-BDHI ορίζει ότι δοθέντος των $w, w^\beta, w^{(\beta^2)}, \dots, w^{(\beta^l)}$ δεν υπάρχει αποδοτικός αλγόριθμος που να υπολογίζει την τιμή $e(w, w)^{1/\beta}$

2.6.6 Attribute based encryption

Σε πολλές περιπτώσεις τα δεδομένα των χρηστών αποθηκεύονται σε κάποιον εξυπηρετητή χωρίς κρυπτογράφηση και ο εξυπηρετητής ελέγχει ποιος μπορεί να αποκτήσει πρόσβαση στα δεδομένα. Ωστόσο αυτή περίπτωση είναι προβληματική καθώς αν κάποιος μη εξουσιοδοτημένος αποκτήσει πρόσβαση στον εξυπηρετητή θα καταφέρει να αποκτήσει πρόσβαση και στα δεδομένα των χρηστών. Ακόμα και αν τα δεδομένα είναι κρυπτογραφημένα σε κάποιες περιπτώσεις είναι επιθυμητό να υπάρχει κάποιος τρόπος να δώσουμε πρόσβαση σε κάποιον χρήστη ενώ σε κάποιον άλλον όχι.

Τα ABE σχήματα προσπαθούν να λύσουν αυτό το πρόβλημα επιτρέποντας την αποκρυπτογράφηση των δεδομένων από κάποιον χρήστη ανάλογα με κάποια πολιτική ασφαλείας. Πιο συγκεκριμένα τα κρυπτογραφημένα δεδομένα συσχετίζονται με μια ομάδα χαρακτηριστικών και το κλειδί του κάθε χρήστη είναι συσχετισμένο με κάποια δομή πρόσβασης που συνδέεται με αυτά τα χαρακτηριστικά. Ο χρήστης μπορεί να αποκρυπτογραφήσει τα δεδομένα αν τα χαρακτηριστικά των δεδομένων ικανοποιούν την δομή πρόσβασης με την οποία είναι συσχετισμένο το κλειδί.

Η βασική ιδέα της ABE εισάγεται στην μελέτη των Sahai και Waters [8] η οποία βασίζεται σε διάφορες ιδέες από IBE σχήματα. Στο συγκεκριμένο σχήμα που ονομάζεται FIBE(Fuzzy Identity-Based

Encryption) μια ταυτότητα αναπαρίσταται ως ένα σύνολο χαρακτηριστικών και επιτρέπει το ιδιωτικό κλειδί μιας ταυτότητας id να αποκρυπτογραφήσει ένα κρυπτοκείμενο της ταυτότητας id' αν και μόνο αν οι δύο ταυτότητες (id και id') είναι κοντά η μια στην άλλη σύμφωνα με το πόσα κοινά στοιχεία έχουν τα σύνολά τους. Δηλαδή ένα μήνυμα κρυπτογραφημένο με ένα σύνολο χαρακτηριστικών id' μπορεί να αποκρυπτογραφηθεί από ένα κλειδί για το σύνολο χαρακτηριστικών id αν $|id \cap id'| \geq d$ για μια τιμή d που επιλέγεται κατά το setup. Λόγω ενός κατωφλίου από το οποίο και μετά μπορεί να γίνει αποκρυπτογράφηση το συγκεκριμένο σχήμα είναι κατάλληλο να χρησιμοποιηθεί με βιομετρικές ταυτότητες. Ωστόσο δεν παρέχει κάποιον τρόπο για να εκφραστούν σύνθετες σχέσεις. Για παράδειγμα σε ένα οργανισμό μπορεί να επιθυμούμε να είναι δυνατή η αποκρυπτογράφηση των οικονομικών δεδομένων μόνο από τον διευθυντή και το οικονομικό τμήμα.

Θα εξετάσουμε το σχήμα που βασίζεται σε KP-ABE (key-policy ABE) [44]. Παρόμοια αποτελείται από τέσσερις αλγορίθμους.

- Setup: Δέχεται την παράμετρο ασφαλείας και παράγει τις δημόσιες παραμέτρους PK και το master κλειδί MK.
- Encryption: Δέχεται ως είσοδο το μήνυμα M, ένα σύνολο χαρακτηριστικών γ και τις δημόσιες παραμέτρους PK και ως έξοδο λαμβάνουμε το κρυπτοκείμενο E.
- Key Generation: Δέχεται ως είσοδο την δομή πρόσβασης (access structure) A το master κλειδί MK και τις δημόσιες παραμέτρους. Έξοδος αυτού είναι το κλειδί αποκρυπτογράφησης D.
- Decryption: Δέχεται ως είσοδο το κρυπτοκείμενο το οποίο κρυπτογραφήθηκε με μια ομάδα χαρακτηριστικών γ, το κλειδί αποκρυπτογράφησης D (για την δομή πρόσβασης A) και τις δημόσιες παραμέτρους PK. Η έξοδος είναι το μήνυμα M αν το σύνολο χαρακτηριστικών ανήκει στην δομή πρόσβασης.

Για την υλοποίηση του σχήματος χρησιμοποιούνται τεχνικές διαμοίρασης μυστικών (secret sharing schemes). Γενικότερα αυτές οι τεχνικές δίνουν την δυνατότητα σε κάποιον να διαμοιράσει ένα μυστικό σε κάποια ομάδα έτσι ώστε μόνο κάποια εξουσιοδοτημένα σύνολα ατόμων να είναι σε θέση να ανακτήσουν το μυστικό. Η συλλογές των συνόλων αυτών ονομάζονται δομές πρόσβασης(access structures). Αξίζει να σημειωθεί πως ένα σχήμα διαμοίρασης μπορεί να δημιουργηθεί μόνο για μονότονες δομές πρόσβασης [45]. Αυτό σημαίνει αν ένα υποσύνολο B της δομής πρόσβασης A μπορεί να ανακτήσει το μυστικό τότε οποιοδήποτε υπερσύνολο του B μπορεί επίσης να ανακτήσει το μυστικό. Χρησιμοποιώντας τις παραπάνω τεχνικές οι αλγόριθμοι γίνονται:

- Setup: Αρχικά επιλέγεται μια παράμετρος d η οποία προσδιορίζει πόσα χαρακτηριστικά έχει κάθε κρυπτογράφημα. Επιλέγονται τυχαία δύο μυστικά α, β από το Z_p. Ορίζουμε $g_1 = g^{\alpha}$ και $g_2 = g^{\beta}$. Επιπλέον επιλέγονται τυχαία 2 πολυώνυμα h(x) και q(x) d βαθμού με τον περιορισμό ότι $q(0) = \beta$. Οι δημόσιες παράμετροι PK είναι $PK = \left(g, g_1; g_2 = g^{q(0)}, g^{q(1)}, g^{q(2)}, \dots, g^{q(d)}; g^{h(0)}, g^{h(1)}, \dots, g^{h(d)}\right)$. Το master είναι ίσο με α. Οι δημόσιες παράμετροι ορίζουν επιπλέον δύο συναρτήσεις $T, V: Z_p \to G$ με $T(x) = g_2^{x^d} \cdot g^{h(x)}$ και $V(x) = g^{q(x)}$.
- Encryption: Έστω $e: G \times G \to G_T$ μια διγραμμική απεικόνιση. Για να κρυπτογραφηθεί ένα μήνυμα σύμφωνα με ένα σύνολο d χαρακτηριστικών $\gamma \subset Z_p$, επιλέγουμε μια τυχαία τιμή $s \in Z_p$ και υπολογίζουμε το κρυπτοκείμενο ως
$$E = \left(\gamma, E^{(1)} = Me(g_1, g_2)^s, E^{(2)} = g^s, \{E_x^{(3)} = T(x)^s\}_{x\in\gamma}, \{E_x^{(4)} = V(x)^s\}_{x\in\gamma}\right)$$
- Key generation: Έχει ως έξοδο ένα κλειδί το οποίο επιτρέπει στον χρήστη να αποκρυπτογραφήσει ένα μήνυμα αν και μόνο αν τα χαρακτηριστικά του μηνύματος ικανοποιούν την δομή πρόσβασης. Η δομή πρόσβασης που χρησιμοποιείται $\mathcal{A}$ είναι μη μονότονη για κάποια μονότονη δομή πρόσβασης σε ένα σύνολο χαρακτηριστικών P. Εφαρμόζουμε ένα γραμμικό μηχανισμό διαμοίρασης Π ώστε να αποκτήσουμε $\{\lambda_i\}$ μέρη του μυστικού α. Συμβολίζουμε με $\vec{x_\iota} \in P$το μέλος που κατέχει το λ_i μέρος του μυστικού ενώ ως x_i

συμβολίζουμε το χαρακτηριστικό που αντιστοιχεί σε αυτό το μέλος. Για κάθε i επιλέγεται μια τυχαία τιμή $r_i \in Z_p$. Το ιδιωτικό κλειδί D αποτελείται από τέσσερα στοιχεία της ομάδας. Για κάθε i τέτοιο ώστε το $\check{x}_\iota$ να μην είναι negated χαρακτηριστικό έχουμε

$$D_i = \left(D_i^{(1)} = g_2^{\lambda_i} \cdot T(x_i)^{r_i}, D_i^{(2)} = g_i^r\right)$$

Για κάθε i τέτοιο ώστε το $\check{x}_\iota$ να είναι ένα negated χαρακτηριστικό έχουμε

$$D_i = \left(D_i^{(3)} = g_2^{\lambda+r_i}, D_i^{(4)} = V(x_i)^{r_i}, D_i^{(5)} = g_i^r\right)$$

Το κλειδί D αποτελείται από όλα τα D_i για όλα τα μέρη του μυστικού.

- Decryption: Για την αποκρυπτογράφηση ενός κρυπτογραφήματος E με ένα κλειδί D ακολουθείται η παρακάτω διαδικασία. Αρχικά ελέγχεται αν $\gamma \in A$. Αν ο έλεγχος αποτύχει τότε η έξοδος είναι κενή. Υπολογίζονται οι τιμές $Z_i = e(g_2, g)^{(s\lambda_i)}$ και η αποκρυπτογράφηση γίνεται υπολογίζοντας

$$\frac{E^{(1)}}{\Pi_{i\in I} Z_i^{\omega_i}} = \frac{Me(g_2, g)^{s\alpha}}{e(g_2, g)^{s\alpha}}$$

Γενικότερα κάθε negative χαρακτηριστικό σε ένα κλειδί συνδέεται με ένα πολυώνυμο d βαθμού(το d είναι ο μέγιστος αριθμός χαρακτηριστικών που χρησιμοποιούνται για την περιγραφή του κρυπτογραφήματος). Για να ανακτήσει κάποιος το μυστικό για αυτόν τον κόμβο πρέπει να χρησιμοποιήσει τουλάχιστον d+1 σημεία ώστε να καταφέρει μέσω παρεμβολής(interpolation) να υπολογίσει το πολυώνυμο. Το σχήμα αντιστοιχεί τα χαρακτηριστικά σε σημεία του πολυωνύμου. Ο αλγόριθμος κρυπτογράφησης μπορεί να συγκεντρώσει d σημεία του πολυωνύμου από τα χαρακτηριστικά του πολυωνύμου. Για να αποκτήσει το τελευταίο σημείο πρέπει να εξετάσει το σημείο που αντιστοιχεί στο negative χαρακτηριστικό σε αυτόν τον κόμβο πρόσβασης. Αν αυτό το χαρακτηριστικό δεν υπάρχει στο κρυπτοκείμενο τότε θα έχει d+1 σημεία του πολυωνύμου και έτσι θα μπορεί να αποκρυπτογραφήσει. Σε διαφορετική περίπτωση αν το χαρακτηριστικό του κλειδιού υπάρχει στο κρυπτοκείμενο ο αλγόριθμος κρυπτογράφησης θα έχει στην διάθεσή του d σημεία καθώς κάποιο σημείο θα είναι διπλό με αποτέλεσμα να μην μπορεί να υπολογίζει το πολυώνυμο με παρεμβολή(interpolation) και συνεπώς να μην μπορέσει να ανακτήσει το μέρος του μυστικού που αντιστοιχεί σε αυτόν τον κόμβο.

2.6.7 Forward Secure Public Key Encryption

Οι τεχνικές που αναφέρθηκαν παραπάνω μπορούν να χρησιμοποιηθούν για να κατασκευαστεί ένα forward secure σχήμα δημοσίου κλειδιού. Όπως έχει ειπωθεί ένα HIBE σχήμα [42] μπορεί να χρησιμοποιηθεί ώστε οι χρήστες να διανέμουν ένα δημόσιο κλειδί το οποίο δεν μεταβάλλεται και να τροποποιούν το ιδιωτικό κλειδί ώστε να μην μπορεί να αποκρυπτογραφήσει μηνύματα μιας παλαιότερης χρονικής περιόδου.

Ωστόσο ένα μειονέκτημα είναι ότι το παραπάνω σχήμα μας αναγκάζει να απωλέσουμε την δυνατότητα αποκρυπτογράφησης όλων των μηνυμάτων των προηγούμενων χρονικών περιόδων. Αποτέλεσμα αυτού είναι να πρέπει να διατηρήσουμε το κλειδί μέχρι να είμαστε σίγουροι ότι έχουμε παραλάβει όλα τα μηνύματα αλλά ταυτόχρονα αυξάνουμε την χρονική περίοδο που τα μηνύματα μένουν εκτεθειμένα.

Το σχήμα που περιγράφεται [46] παρέχει την δυνατότητα να ανακαλούμε την δυνατότητα αποκρυπτογράφησης ενός κλειδιού για συγκεκριμένα μηνύματα διατηρώντας ταυτόχρονα την δυνατότητα να αποκρυπτογραφούμε τα υπόλοιπα μηνύματα. Για να επιτύχουν αυτό το αποτέλεσμα οι συγγραφείς εισάγουν ένα puncture αλγόριθμο ο οποίος βασίζεται στο σχήμα [44]**.** Ο αλγόριθμος δέχεται ένα μυστικό κλειδί SK και μια ετικέτα t και παράγει ένα κλειδί SK’ το οποίο μπορεί να αποκρυπτογραφεί όσα μηνύματα δεν κρυπτογραφήθηκαν με την ετικέτα t. Η διαδικασία αυτή μπορεί να επαναληφθεί. Μετά από n punctures το μέγεθος του δημοσίου κλειδιού και του κρυπτοκειμένου καθώς και το κόστος

της κρυπτογράφησης είναι σταθερό ενώ το μέγεθος του ιδιωτικού κλειδιού και το κόστος αποκρυπτογράφησης είναι O(n).

Επιπλέον συνδυάζουν το puncture αλγόριθμο με το HIBE σχήμα [43] ώστε να είναι δυνατή η διαγραφή κλειδιών για μια καθορισμένη χρονική περίοδο χωρίς να απαιτείται η διαγραφή των προηγούμενων χρονικών περιόδων. Αυτή η προσέγγιση επιτρέπει το κόστος της αποκρυπτογράφησης αλλά και το μέγεθος που απαιτείται για την αποθήκευση του κλειδιού να αυξάνουν γραμμικά σε σχέση με των μέγιστο αριθμό μηνυμάτων εντός μιας χρονικής περιόδου και λογαριθμικά σε σχέση με τον αριθμό των χρονικών περιόδων.

Το συνδυασμένο σχήμα για κάθε χρονική περίοδο T χρησιμοποιεί ένα ζεύγος κλειδιών (A_T, B_T) με A_T αναπαριστά το κλειδί για το FS-PKE την χρονική περίοδο T και B_T αναπαριστά το κλειδί για τον puncturable αλγόριθμο. Αρχικά το κλειδί B_T δεν έχει punctures για κάποιο tag και θα τροποποιηθεί με κάθε puncture. Πριν πραγματοποιήσουμε κάποιο puncture παράγουμε και το ζεύγος κλειδιών (A_{T+1}, B_{T+1}) για την επόμενη χρονική περίοδο. Το κλειδί B_{T+1} αρχικά είναι κενό.

Η παραγωγή του κλειδιού της χρονικής περιόδου T+1 από την χρονική περίοδο T έγκειται στην διαγραφή του ζεύγους (A_T, B_T) και την χρήση του (A_{T+1}, B_{T+1}) ώστε να παραχθεί το ζεύγος (A_{T+2}, B_{T+2}) για την χρονική περίοδο T+2. Τα punctures εφαρμόζονται στο κλειδί B_{T+1}.

Και το HIBE σχήμα χρησιμοποιεί ως master κλειδί την τιμή α και την δημόσια παράμετρο $g_1 = g^{\alpha}$. Ο puncturable αλγόριθμος ακολουθεί παρόμοια κατασκευή. Οπότε αρχικά διαμοιράζεται το κλειδί $\alpha = \alpha_1 + \alpha_2$. Το α_1 χρησιμοποιείται ως το master κλειδί για το HIBE σχήμα και το α_2 για τον puncture αλγόριθμο.

Το σχήμα αποτελείται από τους παρακάτω αλγορίθμους. Με BBG.algorithm συμβολίζουμε το κάθε αλγόριθμο που περιγράφεται στο HIBE σχήμα των Boneh, Boyen and Goh [43]

- PFSE.Keygen: Ως είσοδο δέχεται την παράμετρο k, τον αριθμό των ετικετών για το κρυπτοκείμενο (το συμβολίζουμε με d) και το βάθος l του δένδρου. Επιλέγουμε δύο ομάδες G, G_T τάξης p. Έπειτα διαλέγουμε τυχαία $\alpha, \beta \in Z_p$ και $g_3, h_1, \dots, h_l \in G$ και θέτουμε $g_1 = g^{\alpha}, g_2^{\beta}$. Έστω μια συνάρτηση κατακερματισμού $H: 0,1 \to Z_p$ και t_0 μια ετικέτα η οποία δεν ανήκει στις ετικέτες που θα χρησιμοποιηθούν κανονικά κατά την εκτέλεση. Ορίζουμε $PK = (PK_{PPKE}, PK_{PFSE})$ με

$$PK_P PKE = \left(g, g_1, g_2, g_{q(1)}, \dots, g^{q(d)}\right) \quad \text{και} \quad PK_{PFSE} = (g, g_1, g_2, g_3, h_1, \dots, h_l) \quad .$$

Διαλέγουμε τυχαία $r_1, r_2, r_3 \in Z_p$ καθώς και δύο τιμές $\alpha_1, \alpha_2 \in Z_p$ με τον περιορισμό $\alpha_1 + \alpha_2 = \alpha$. Χρησιμοποιώντας την τιμή α_1 ως το master κλειδί για το HIBE σχήμα υπολογίζουμε τα HIBE κλειδιά που αντιστοιχούν στις ταυτότητες «0» και «1» δηλαδή τις ταυτότητες αριστερά και δεξιά της ρίζας του δένδρου.

$$hsk_L = BBG.Keygen\left(PK_{PFSE}, \alpha_{1,}0\right), hsk_R = BBG.Keygen\left(PK_{PFSE}, \alpha_{1,}1\right)$$

Έπειτα υπολογίζουμε το αρχικό PPKE μέρος του μυστικού κλειδιού χρησιμοποιώντας το master κλειδί α_2 και την ετικέτα t_0 :

$$Ppkesk_{\emptyset} = \left[\left(g_2^{\alpha_2 + r_3}, V\left(H(t_0)\right)^{r_3}, g^{r_3}, t_0\right)\right]$$

Θέτουμε $D_0 = (0, hsk_L, hsk_R, ppkesk_{\emptyset})$. Αυτή η πλειάδα αποτελεί το seed για να υπολογιστεί το κλειδί για την πρώτη χρονική περίοδο όπως φαίνεται παρακάτω

$$\left(tsk_{\emptyset}^1, D_1\right) \leftarrow PFSE.NextInterval(D_0)$$

Η έξοδος του αλγορίθμου είναι το δημόσιο κλειδί PK και το αρχικό ιδιωτικό κλειδί $SK = \left(tsk_{\emptyset}^{1}, D_1\right)$.

- PFSE.Encrypt: Δέχεται ως είσοδο το κλειδί PK, το μήνυμα M, την τωρινή χρονική περίοδο T_{cur}, ένα σύνολο από ετικέτες $t_1, \dots, t_d \in 0,1$. Επιλέγουμε μια τυχαία τιμή $s \in z_p$ και υπολογίζουμε την HIBE ταυτότητα $T_1, \dots, T_k = IndexToPath(T_{cur}, l)$. Οι χρονικές περίοδοι απεικονίζονται σε ένα δυαδικό δένδρο χρησιμοποιώντας pre-order traversal. Με την χρήση ενός map γίνεται η αντιστοίχιση της χρονικής περιόδου στο κλειδί. Υπολογίζουμε

$$ct^{(1)} = e(g_1, g_2)^s \cdot M, ct^{(2)} = g^s, ct^{(3,1)} = V\left(H(t_1)\right)^s, \dots, ct^{3,d} = V\left(H(t_d)\right)^s, ct^{(4)} = \left(h_1^{T_1} \dots h_k^{T_k} \cdot g_3\right)^s$$

Το τελικό κρυπτοκείμενο $ct = \left(ct^{(1)}, ct^{(2)}, ct^{(3,1)}, \dots, ct^{(3,d)}, ct^{(4)}\right)$ και $T_{cur}, (t_1, \dots, t_d)$
Όπως παρατηρούμε το μέγεθος του κρυπτοκειμένου είναι σταθερό και αποτελείται από τα στοιχεία $ct^{(1)}, ct^{(2)}, ct^{(3,1)}, \dots, ct^{(3,d)}$ και $ct^{(4)}$

- PFSE.Puncture: Ο αλγόριθμος δέχεται το τωρινό μυστικό κλειδί $SK = \left(tsk_T^i, D_i\right)$ με το T να αναπαριστά το σύνολο των punctures την τωρινή περίοδο. Το ιδιωτικό κλειδί αποτελείται από ένα κλειδί για κάθε σχήμα $tsk_T^i = (hsk_i, ppkesk_T)$ οπότε υπολογίζουμε

$$ppkesk_{T\cup\{t\}} \leftarrow PPKE.Puncture(PK_{PPKE}, ppkesk_T, t)$$

Η έξοδος του αλγορίθμου είναι ένα καινούριο μυστικό κλειδί $tsk_{T\cup\{t\}}^i = \left(hsk_i, ppkesk_{T\cup\{t\}}\right)$ το οποίο δεν μπορεί να αποκρυπτογραφήσει μηνύματα που περιέχουν ετικέτες στο σύνολο $T \cup \{t\}$.

- PFSE.NextInterval: Έστω το κλειδί $D_i = (i, HSKs, ppkesk_{\emptyset})$. Εξάγουμε το HIBE κλειδί hsk_P που αντιστοιχεί στην χρονική περίοδο i από την τιμή $HSKs$ και παράγουμε τα αριστερά και δεξιά κλειδιά ως

$$hsk_L = BBG.Keygen(hsk_P, 0), hsk_R = BBG.Keygen(hsk_P, 1)$$

Έπειτα υπολογίζουμε ένα νέο $HSKs'$ χρησιμοποιώντας τα δύο νέα κλειδιά χωρίς να συμπεριλάβουμε τον γονέα δηλαδή $HSKs' = HSKsk_P \cup \{hsk_L, hsk_R\}$ και θέτουμε $D' = (i + 1, HSKs', ppkesk_{\emptyset})$.
Τέλος παράγουμε το κλειδί $(tsk_{\emptyset}^i = \left(hsk_i', ppkesk_{\emptyset}'\right))$ αποκρυπτογράφησης των μηνυμάτων για την χρονική περίοδο i συνδυάζοντας το HIBE κλειδί hsk_P με μια τυχαιοποιημένη(randomized) έκδοση του P-PKE κλειδιού $ppkesk_{\emptyset}$. Η έξοδος του αλγορίθμου είναι το κλειδί $\left(tsk_{\emptyset}^i, D'\right)$.

- PFSE.Decrypt: θεωρούμε την τιμή tsk_T^i ως $(hsk_i, ppkesk_T)$ και ct ως $ct^{(1)}, ct^{(2)}, ct^{(3,1)}, \dots, ct^{(3,d)}, ct^{(4)}$

$$A \leftarrow PPKE.Decrypt\left(ppkesk_T, \left(1, ct^{(2)}, ct^{(3,1)}, \dots\right), t_1, \dots, t_d\right)B \leftarrow HIBE.Decrypt\left(hsk_i, \left(1, ct^{(2)}, ct^{(4)}\right)\right)$$

Το αποκρυπτογραφημένο μήνυμα είναι $M = \frac{ct^{(1)}}{A \cdot B}$.

3 Ανάλυση και σχεδίαση

3.1 Ανάλυση

3.1.1 Ανάλυση αρχιτεκτονικής

Το πρωτόκολλο MQTT τυγχάνει ευρείας χρήσης σε εφαρμογές που απαιτούν την ασύγχρονη ενημέρωση των clients για κάποιο συμβάν. Αυτό συμβαίνει λόγω συγκεκριμένων ιδιοτήτων που διαθέτει το πρωτόκολλο. Αρχικά παρέχει το πλεονέκτημα της αποσύζευξης (decoupling) μεταξύ του αποστολέα του μηνύματος(publisher) και του παραλήπτη(subscriber). Τα διάφορα μέρη δεν απαραίτητο να βρίσκονται στο ίδιο δίκτυο ή να έχουν γνώση ο ένας του άλλου. Ακόμα δεν απαιτείται τα διάφορα μέρη να είναι συνδεδεμένα ταυτόχρονα(decoupled in time). Ο broker διατηρεί τα μηνύματα και τα προωθεί στους παραλήπτες όταν συνδεθούν. Ένα παράδειγμα που απαιτείται αυτού του είδους η ασύγχρονη επικοινωνία είναι η αποστολή και παραλαβή άμεσων μηνυμάτων. Το Facebook χρησιμοποίησε αρχικά το συγκεκριμένο πρωτόκολλο για την μεταφορά μηνυμάτων στον Messenger. [47]

Για να γίνει κατανοητή η αρχιτεκτονική εφαρμογών που βασίζονται στο MQTT θα αναλυθεί μια πιθανή περίπτωση χρήσης του.

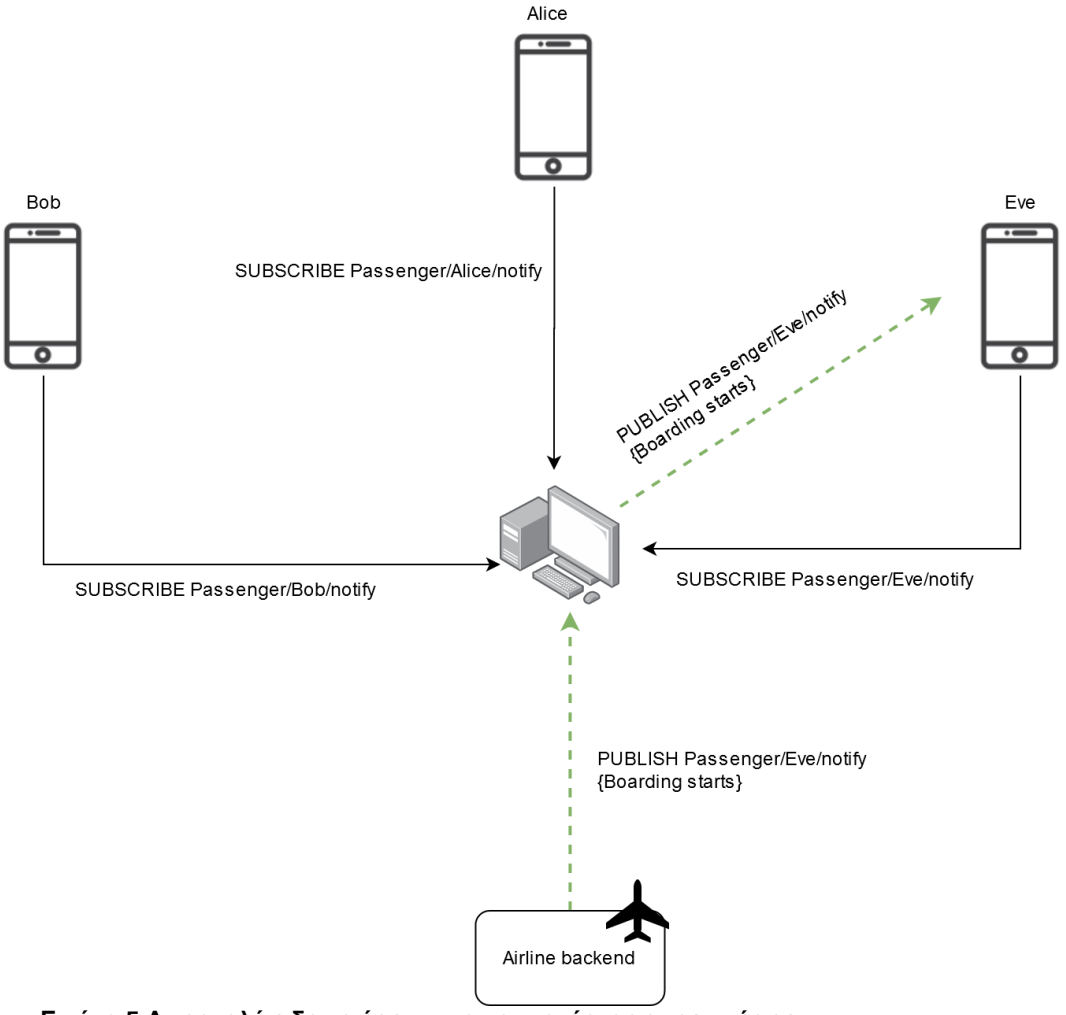
Μια αεροπορική εταιρεία επιθυμεί να αποστέλλει ειδοποιήσεις στους πελάτες της μέσω της εφαρμογής που έχουν εγκατεστημένη στο κινητό τους. Όταν η πτήση καθυστερήσει ή όταν αρχίσει το boarding οι πελάτες της αεροπορικής εταιρείας θα πρέπει να λάβουν μια ειδοποίηση. Ας υποθέσουμε ότι κάθε πελάτης είναι εγγεγραμμένος στο σύστημα με το όνομα του.

Το σύστημα θα μπορούσε να υλοποιηθεί με τον ακόλουθο τρόπο. Η εφαρμογή του κάθε πελάτη εγγράφεται σε ένα μοναδικό topic. Το topic μπορεί να κατασκευασθεί ως «Passenger/<passenger_name>/Notify» . Για παράδειγμα αν έχουμε τρείς πελάτες τα topic τα οποία εγγράφονται φαίνονται στον επόμενο πίνακα.

ΕΠΙΒΑΤΗΣ	**TOPIC**
Bob	Passenger/Bob/Notify
Alice	Passenger/Alice/Notify
Eve	Passenger/Eve/Notify

Πίνακας 15 Τα topics του συστήματος της αεροπορικής εταιρείας

Στην περίπτωση που η αεροπορική εταιρεία θέλει να ειδοποιήσει τον Bob ότι η πτήση του θα έχει καθυστέρηση θα στείλει ένα μήνυμα στο topic Passenger/Bob/Notify. Αντίστοιχα αν θέλει να ενημερώσει την Eve το μήνυμα θα σταλεί στο topic Passenger/Eve/Notify. Το σύστημα φαίνεται στο παρακάτω σχήμα.



Εικόνα 5 Αποστολή ειδοποιήσεων για την κατάσταση της πτήσης

Ένα πιο σύνθετο παράδειγμα θα ήταν μια εφαρμογή μέσω της οποίας ένας πελάτης θα μπορούσε να στείλει ένα μήνυμα σε κάποιο τμήμα της εταιρείας. Αντίστοιχα μπορεί να λάβει απάντηση από το αντίστοιχο τμήμα. Έστω ότι έχουμε δύο τμήματα το τμήμα πωλήσεων(Sales) και το τμήμα ανθρωπίνου δυναμικού(HRM). Ο κάθε πελάτης διαθέτει ένα topic στο οποίο στέλνει τα μηνύματα και ένα topic στο οποίο εγγράφεται για να λάβει τα μηνύματα.

App	Recipient	Action	Topic
Bob	Sales	Publish	Message/Bob/SalesRequest
	HRM	Publish	Message/Bob/HRMRequest
	--	Subscribe	Message/Bob/SalesResponse
	--	Subscribe	Message/Bob/HRMResponse

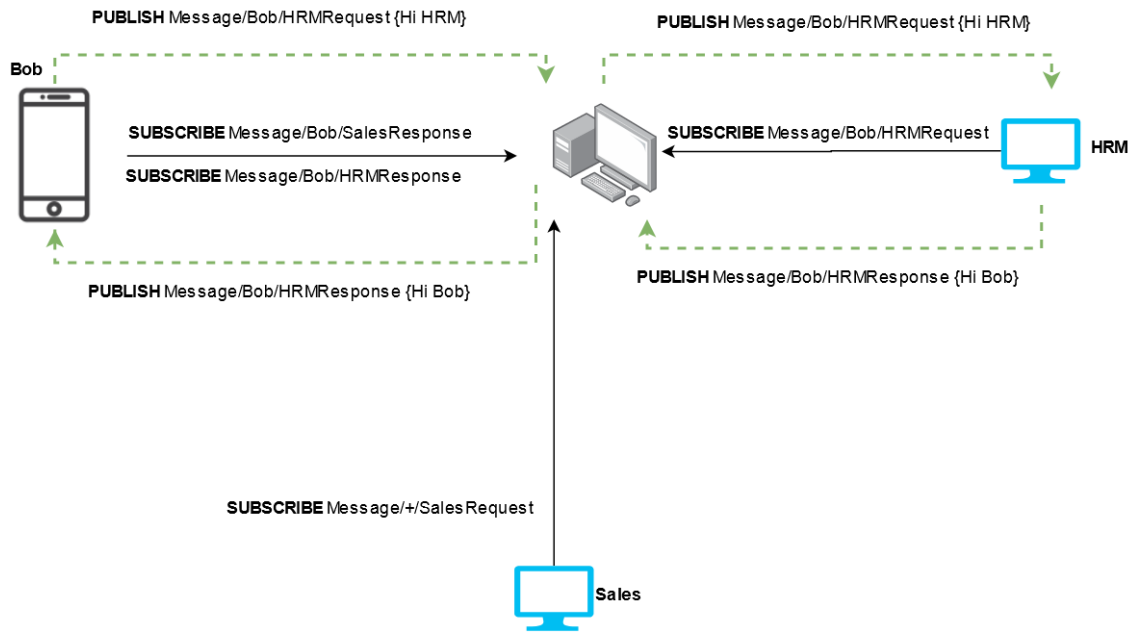
Πίνακας 16 Τα topics στα οποία εγγράφεται ή αποστέλλει μηνύματα ένας πελάτης

Τα topics που χρησιμοποιούνται από το κάθε τμήμα είναι τα ακόλουθα

App	Recipient	Action	Topic
Sales	--	Subscribe	Message/+/SalesRequest
	Bob	Publish	Message/Bob/SalesResponse
HRM	--	Subscribe	Message/+/HRMRequest
	Bob	Publish	Message/Bob/HRMResponse

Πίνακας 17 Τα topics που χρησιμοποιούνται από το κάθε τμήμα

Η συνολική αρχιτεκτονική του συστήματος φαίνεται στο παρακάτω διάγραμμα.



Εικόνα 6 Ένα σύστημα συνομιλιών (chat)

3.1.2 Ανάλυση απαιτήσεων

Για να εξεταστεί καλύτερα το περιβάλλον των mobile εφαρμογών πρέπει να αναλυθούν οι επιθέσεις που μπορεί να πραγματοποιήσει κάποιος επιτιθέμενος καθώς και ποιες επιπτώσεις επιφέρουν.

- Eavesdropping attack: Ο επιτιθέμενος μπορεί και παρακολουθεί την επικοινωνία του χρήστη με τον εξυπηρετητή χωρίς να μπορεί όμως να τροποποιήσει τα δεδομένα που μεταφέρονται. Επιπλέον έχει τη δυνατότητα να αποθηκεύσει τα μηνύματα που έχουν περάσει από το κανάλι ώστε να τα χρησιμοποιήσει για μετέπειτα επιθέσεις.
- Impersonation attack: Ο επιτιθέμενος μπορεί να μπορεί να υποδυθεί κάποιο άλλον νόμιμο χρήστη και να συνδεθεί στο σύστημα ως αυτός ο χρήστης.
- Man in the middle attack: Ο επιτιθέμενος έχει την δυνατότητα να παρακολουθεί την επικοινωνία μεταξύ δύο μερών. Επιπλέον σε αυτή την επίθεση έχει την δυνατότητα να τροποποιεί, να προσθέτει, να διαγράφει είτε να καθυστερεί τα πακέτα που ανταλλάζονται.

- Denial of service attack: Ο επιτιθέμενος έχει ως σκοπό να διακόψει την παροχή της υπηρεσίας καταναλώνοντας τους πόρους του συστήματος.
- Parallel session attack: Ο επιτιθέμενος προσπαθεί να δημιουργήσει παράλληλες συνόδους εκμεταλλευόμενος μηνύματα ήδη έχουν περάσει από το δίκτυο ή εξαπατώντας κάποιον χρήστη ώστε να χρησιμοποιήσει μια τιμή που έχει διαλέξει ο επιτιθέμενος. Για παράδειγμα αν ένα πρωτόκολλο επιστρέφει μια τυχαία τιμή και ζητάει από τον χρήστη να την υπογράψει ώστε να αυθεντικοποιηθεί ο επιτιθέμενος θα μπορούσε να παρακάμψει αυτό τον έλεγχο ως εξής. Ο επιτιθέμενος ζητάει έναν τυχαίο αριθμό από τον server και τον στέλνει σε έναν χρήστη που προσπαθεί να αυθεντικοποιηθεί εκείνη την στιγμή. Ο χρήστης υπογράφει αυτήν την τιμή. Τώρα ο επιτιθέμενος μπορεί να απαντήσει στην πρόκληση του server χρησιμοποιώντας αυτήν την τιμή που μόλις υπογράφηκε.
- Offline guessing attack: Σε αυτήν την περίπτωση ο επιτιθέμενος προσπαθεί να μαντέψει τον κωδικό χρησιμοποιώντας λίστες με συχνούς κωδικούς. Για να εξακριβωθεί αν ο κωδικός που δοκιμάστηκε είναι σωστός χρησιμοποιούνται πακέτα που έχουν καταγραφεί από την εκτέλεση του πρωτοκόλλου. Η επίθεση αυτή γίνεται χωρίς να απαιτείται αλληλεπίδραση με τον εξυπηρετητή κάτι που την καθιστά δύσκολο να ανιχνευτεί.

Ένα ακόμα σημείο που πρέπει να διευκρινισθεί είναι ότι η αρχιτεκτονική της εφαρμογής που εξετάζουμε είναι αυτή που αναλύθηκε στην προηγούμενη ενότητα. Αυτό σημαίνει ότι ο χρήστης γνωρίζει τους παραλήπτες και στέλνει ένα διαφορετικό μήνυμα για κάθε παραλήπτη.

Γενικότερα μπορούμε να διακρίνουμε τρία είδη απαιτήσεων. Τις απαιτήσεις που πρέπει να ικανοποιεί το πρωτόκολλο αυθεντικοποίησης, τις απαιτήσεις για την δομή εμπιστοσύνης που θα χρησιμοποιηθεί δηλαδή με ποιο τρόπο θα αποδεικνύονται οι σχέσεις εμπιστοσύνης και τέλος οι απαιτήσεις του σχήματος που θα εγγυάται την εμπιστευτικότητα και την ακεραιότητα των δεδομένων.

3.1.3 Απαιτήσεις αυθεντικοποίησης

Από τα παραπάνω προκύπτουν κάποιες απαιτήσεις ασφαλείας που θα πρέπει να ικανοποιεί το πρωτόκολλο αυθεντικοποίησης που θα χρησιμοποιηθεί.

- Το πρωτόκολλο θα πρέπει να παρέχει αμοιβαία αυθεντικοποίηση. Αυτό σημαίνει ότι ο εξυπηρετητής αλλά και ο πελάτης θα πρέπει να μπορούν να αυθεντικοποιήσουν ο ένας τον άλλον.
- Να παρέχει εμπιστευτικότητα. Όλα τα μηνύματα που ανταλλάσσονται θα πρέπει να είναι κρυπτογραφημένα. Σε αυτήν την περίπτωση το πρωτόκολλό μπορεί απλά να δίνει την δυνατότητα στα δυο μέρη να παράγουν ένα κοινό κλειδί το οποίο μπορεί αργότερα να χρησιμοποιηθεί για κρυπτογράφηση. Αυτό απαιτείται για να αποφευχθούν επιθέσεις(πχ eavesdropping) στην επικοινωνία μετά την αυθεντικοποίηση.
- Το πρωτόκολλο δεν πρέπει να βασίζεται στην χρήση προσυμφωνημένων κλειδιών ή να απαιτεί την αποθήκευση κλειδιών ή διαπιστευτηρίων από την μεριά του πελάτη. Στην περίπτωση εφαρμογών που προορίζονται για κινητές συσκευές είναι πολύ πιθανόν ο χρήστης να χρησιμοποιήσει την εφαρμογή και από άλλη συσκευή. Η ανάγκη αποθήκευσης των κλειδιών στην συσκευή θα καθιστούσε την μεταφορά της εφαρμογής σε άλλη συσκευή προβληματική. Επιπλέον αν η συσκευή κλαπεί είναι πιθανόν κάποιος να ανακτήσει αυτά τα διαπιστευτήρια.
- Το πρωτόκολλο θα πρέπει να προστατεύει την ταυτότητα του χρήστη. Σε κάθε περίπτωση δεν θα πρέπει να είναι σε θέση να εξακριβώσει αν κάποιος χρήστης υπάρχει ήδη στο σύστημα.
- Θα πρέπει να παρέχει forward secrecy. Σε κάθε εκτέλεση του πρωτοκόλλου θα πρέπει να παράγεται ένα τυχαίο καινούργιο κλειδί. Ένας επιτιθέμενος που αποκτήσει το αρχικό κλειδί δεν θα πρέπει να μπορεί να αποκρυπτογραφήσει την επικοινωνία. Αυτή η ιδιότητα είναι ιδιαίτερα σημαντική στην περίπτωση χρήσης κωδικών. Ας υποθέσουμε ότι κάποιος επιτιθέμενος ανακαλύπτει τον κωδικό κάποιου χρήστη. Ο χρήστης αυτός συνδέεται (με τον κωδικό που γνωρίζει ο επιτιθέμενος) στο σύστημα και αλλάζει τον κωδικό του. Αν η ιδιότητα που

αναφέραμε δεν ισχύει, τότε αν ο επιτιθέμενος είχε καταγράψει την επικοινωνία θα μπορούσε να ανακαλύψει τον καινούργιο κωδικό. Αν η ιδιότητα ισχύει τότε η επίθεση σταματάει με το που αλλάξει ο κωδικός.

- Η εκτέλεση του πρωτοκόλλου θα πρέπει να μην αποκαλύπτει κάποια πληροφορία για τον κωδικό του χρήστη ώστε να αποφευχθούν επιθέσεις με λεξικό.

3.1.4 Απαιτήσεις εμπιστοσύνης

Όπως αναφέρθηκε και προηγουμένως οι περισσότερες εφαρμογές χρησιμοποιούν TLS για να μεταφέρουν τα δεδομένα με ασφάλεια. Η αυθεντικοποίηση στο TLS βασίζεται στην χρήση PKI(public key infrastructure).

Η PKI υλοποιείται ιεραρχικά με την έννοια ότι τα πιστοποιητικά υπογράφονται από μια αρχή πιστοποίησης την οποία θα εμπιστεύονται οι χρήστες. Αυτό σημαίνει ότι ένας οργανισμός πρέπει να εμπιστεύεται ότι η αρχή πιστοποίησης δεν θα εκδώσει κάποιο λανθασμένο πιστοποιητικό.

Για να λυθεί το πρόβλημα αυτό ο οργανισμός θα μπορούσε να δημιουργήσει την δικιά του αρχή πιστοποίησης. Ωστόσο χρειάζεται να ρυθμίσει τις άλλες συσκευές να εμπιστεύονται την συγκεκριμένη αρχή πιστοποίησης παρέχοντάς τους το δημόσιο κλειδί της αρχής πιστοποίησης. Στην περίπτωση κινητών συσκευών αυτό δεν μπορεί να γίνει εύκολα. Για παράδειγμα αν η συσκευή «κατεβάζει» το κλειδί από κάποια υπηρεσία δεν μπορούμε να είμαστε σίγουροι ότι δεν έχει τροποποιηθεί κατά την μεταφορά. Η ανάγκη να μεταφέρουμε το κλειδί της αρχής πιστοποίησης ή τα πιστοποιητικά των πελατών καθιστά δύσκολη την χρήση της εφαρμογής σε διαφορετικές συσκευές καθώς ο χρήστης είναι υποχρεωμένος να μεταφέρει τα κλειδιά του σε κάθε συσκευή. Τέλος μπορεί να μην επιθυμούμε την αποθήκευση κρυπτογραφικών κλειδιών στην συσκευή ή η αποθήκευση να μην είναι πρακτική.

Ακόμα η χρήση PKI απαιτεί την ύπαρξη κάποιων διαδικασιών για την διαχείριση των κλειδιών. Παραδείγματος χάρη πότε θα σταματούν να είναι έγκυρα τα πιστοποιητικά και πότε αυτά θα ανανεώνονται. Αν είναι έγκυρα για μεγάλο χρονικό διάστημα τότε είναι αναγκαίο να υπάρχει κάποιος μηχανισμός όπως η δημοσίευση Certificate Revocation Lists ώστε στην περίπτωση που κάποιο ιδιωτικό κλειδί γίνει γνωστό να είναι δυνατό να ακυρωθεί το πιστοποιητικό. Αυτό απαιτεί και κάποια διαδικασία με την οποία μπορούμε να αντιληφθούμε ότι κάποιο κλειδί έχει διαρρεύσει [48], [49].

Για την λειτουργία της PKI απαιτείται ρύθμιση και από την μεριά του πελάτη(δηλαδή του χρήστη της εφαρμογής). Αν χρησιμοποιούνται self-signed certificates ο χρήστης θα πρέπει να εισάγει το πιστοποιητικό της αρχής πιστοποίησης που χρησιμοποιεί η εφαρμογή στα πιστοποιητικά που θεωρεί το σύστημα ως αξιόπιστα. Επιπλέον κάπως θα πρέπει να λάβει και το ζεύγος κλειδιών του.

Από τα παραπάνω γίνεται εμφανές ότι μια ιεραρχική μορφή εμπιστοσύνης δεν είναι κατάλληλη για την περίπτωση που εξετάζουμε. Δεν επιθυμούμε μια κεντρική αρχή που να προσδιορίζει ποιος είναι έμπιστος. Αντίθετα οι σχέσεις εμπιστοσύνης δημιουργούνται απευθείας(direct trust) με το άλλο μέρος. Αυτό μπορεί να πραγματοποιηθεί χρησιμοποιώντας ένα μυστικό που μοιράζονται τα δύο μέρη.

3.1.5 Απαιτήσεις εμπιστευτικότητας, ακεραιότητας

Εκτός από την επικοινωνία broker πελάτη είναι αναγκαίο να εξεταστεί η ασφάλεια του περιεχομένου των μηνυμάτων γενικότερα.

Σύμφωνα με το MQTT ο εξυπηρετητής διατηρεί τα topics στα οποία είναι εγγεγραμμένος ο κάθε πελάτης και ανάλογα στέλνει τα μηνύματα που αντιστοιχούν στο κάθε topic. Όταν ο πελάτης αποσυνδεθεί τα topics στα οποία έχει εγγραφεί χάνονται και πρέπει να εγγραφεί ξανά. Στην περίπτωση που ο πελάτης έχει συνδεθεί ζητώντας μια διαρκούσα συνεδρία (persistent session) τότε ο εξυπηρετητής διατηρεί τα μηνύματα για τα topics που έχει εγγραφεί ο πελάτης όσο αυτός δεν ήταν συνδεδεμένος και

θα τα αποστείλει όταν αυτός συνδεθεί. Σε αρκετές περιπτώσεις το διάστημα που πρέπει να διατηρηθούν τα μηνύματα μπορεί να είναι αρκετά μεγάλο.

Συνεπώς είναι απαραίτητο να υπάρχει κάποιος μηχανισμός ο οποίος θα διασφαλίζει ότι δεν θα είναι δυνατό κάποιος μη εξουσιοδοτημένος να διαβάσει ή να τροποποιήσει τα μηνύματα όσο αυτά θα βρίσκονται αποθηκευμένα στον εξυπηρετητή. Επομένως είναι αναγκαία η χρήση τεχνικών κρυπτογράφησης από άκρο σε άκρο (End-to-end encryption) και κάποιος μηχανισμός που να εγγυάται την ακεραιότητα των δεδομένων.

Επιπρόσθετα λόγω της διάρκειας που διατηρούνται τα μηνύματα αν ένας επιτιθέμενος καταφέρει να αποκτήσει πρόσβαση στα long-term keys κάποιου χρήστη πιθανώς να αποκτήσει πρόσβαση σε όλα τα μηνύματα του χρήστη. Άρα οι τεχνικές που θα χρησιμοποιηθούν θα πρέπει να παρέχουν forward secrecy. Αυτό σημαίνει πως αν τα κλειδιά κάποιας συνόδου ή το ιδιωτικό κλειδί γίνουν γνωστά κάποιος επιτιθέμενος δεν θα είναι σε θέση να ανακτήσει τα κλειδιά για τις προηγούμενες συνόδους. Επιπλέον μια περίπτωση που πρέπει να ληφθεί υπόψη είναι να μην βρίσκονται και τα δύο μέρη ταυτόχρονα συνδεδεμένα επομένως οι τεχνικές αυτές θα πρέπει να παρέχουν forward secrecy χωρίς να απαιτούν αλληλεπίδραση.

Η χρήση μια ενιαίας έμπιστης αρχής η οποία θα διανέμει κλειδιά απαιτεί κάποιο ασφαλές κανάλι για την αυθεντικοποίηση και την διανομή των κλειδιών και σε κάποιες περιπτώσεις δυσκολεύει την υλοποίηση. Ακόμα η έμπιστη αρχή μπορεί να αποκρυπτογραφήσει όλα τα μηνύματα το οποίο δεν είναι επιθυμητό. Στις εφαρμογές για κινητά όπως τα chats ο αποστολέας συνήθως αποστέλλει μηνύματα σε έναν άλλον χρήστη. Ο χρήστης επιθυμεί να είναι ο μόνος που μπορεί να αποκρυπτογραφήσει την συνομιλία επομένως θα πρέπει να διαχειρίζεται τα κλειδιά κρυπτογράφησης.

3.2 Επιλογή τεχνικών

Οι πιο συχνές τεχνικές για την προστασία της επικοινωνίας του πελάτη και του broker βασίζονται στην χρήση προσυμφωνημένων κλειδιών ή την χρήση PKI. Όπως αναφέρθηκε σε κάποιες περιπτώσεις μπορεί να είναι προβληματική η εφαρμογή τους.

Η χρήση PAKE αλγορίθμων είναι μια ελκυστική λύση. Πιο συγκεκριμένα η χρήση augmented PAKE σχημάτων κρίνεται καταλληλότερη καθώς παρέχει ασφάλεια στην περίπτωση που ο επιτιθέμενος αποκτήσει πρόσβαση στο αρχείο κωδικών. Επιπλέον στην περίπτωση του MQTT η συσκευή ξεκινάει την επικοινωνία επομένως δεν χρειαζόμαστε την συμμετρία που παρέχουν οι balanced PAKE αλγόριθμοι ώστε να μπορεί οποιοσδήποτε να αρχίσει την επικοινωνία.

Τέλος με την χρήση PAKES αποφεύγονται κάποιες επιθέσεις που στοχεύουν στην εξαπάτηση του χρήστη για να λειτουργήσουν. Για παράδειγμα ένα είδος επίθεσης που ονομάζεται IDN homograph attack [50] έχει ως σκοπό να εξαπατήσει τον χρήστη εκμεταλλευόμενη το γεγονός ότι κάποιοι χαρακτήρες φαίνονται ίδιοι(για παράδειγμα το λατινικό α με το ελληνικό α). Ένας επιτιθέμενος μπορεί να παράγει ένα έγκυρο πιστοποιητικό για το site example.com χρησιμοποιώντας τον ελληνικό χαρακτήρα για το a. Ένας χρήστης που θα συνδεθεί στο site δεν θα καταλάβει την διαφορά και αφού το πιστοποιητικό είναι έγκυρο (για εκείνο το site) δεν θα ειδοποιηθεί και από τον browser. Επομένως απαιτείται από τον χρήστη να ελέγχει το site στο οποίο βρίσκεται. Στην περίπτωση των PAKES αυτό δεν θα συνέβαινε καθώς δεν υπάρχει κάποια τρίτη αρχή που να εμπιστεύεται ο χρήστης. Ακόμα και στην περίπτωση που ο χρήστης είχε συνδεθεί σε λάθος server αυτού του είδους τα πρωτόκολλα δεν επιτρέπουν σε κάποιο από τα μέρη της επικοινωνίας να μάθει τον κωδικό (zero knowledge protocols).

Αρκετές από τις παραλλαγές του AMP δεν είναι ασφαλείς. Παρόλα αυτά το πρωτόκολλο AMP2 είναι αποδοτικό και έχει γίνει standardized [51] ωστόσο δεν τυγχάνει ευρείας χρήσης. Το πρωτόκολλο OPAQUE παρέχει το πλεονέκτημα ότι δεν είναι δυνατόν να κατασκευαστεί το λεξικό με τους κωδικούς πριν ο επιτιθέμενος αποκτήσει πρόσβαση στο αρχείων των κωδικών καθώς το salt είναι μυστικό σε αντίθεση με τα άλλα πρωτόκολλα. Επιπλέον μπορεί να υλοποιηθεί χρησιμοποιώντας ελλειπτικές καμπύλες με αποτέλεσμα να είναι ένα πιο αποδοτικό πρωτόκολλο. Μέχρι στιγμής δεν έχει

χρησιμοποιηθεί ευρέως. Επιπλέον η εγγραφή του χρήστη στο σύστημα είναι λίγο πιο πολύπλοκη καθώς ο χρήστης και ο εξυπηρετητής πρέπει να εκτελέσουν μια συνάρτηση OPRF. Τα πρωτόκολλα DH-EKE και SPEKE είναι ευάλωτα σε συγκεκριμένες επιθέσεις. Επιπλέον όπως και το πρωτόκολλο J-PAKE είναι balanced PAKE πρωτόκολλα το οποίο δεν παρέχει προστασία στην περίπτωση που κάποιος αποκτήσει πρόσβαση στο αρχείο κωδικών.

Το πρωτόκολλο SRP-6 είναι η τελευταία έκδοση του πρωτοκόλλου SRP το οποίο χρησιμοποιείται σε αρκετά προϊόντα. Δυο γνωστές περιπτώσεις είναι το iCloud Key Vault [52] και ο password manager 1Password [53]. Στην περίπτωση του iCloud Key Vault ο χρήστης μπορεί να ανακτήσει τα κλειδιά της συσκευής του (keychain) χρησιμοποιώντας τον iCloud Security Code που διαθέτει. Η Apple εξακριβώνει ότι ο χρήστης γνωρίζει τον κωδικό χρησιμοποιώντας το πρωτόκολλο SRP. Παρόμοια ο password manager 1Password χρησιμοποιεί το SRP για να αυθεντικοποιήσει τον χρήστη στον server πριν του αποστείλει τα δεδομένα που βρίσκονται στο cloud. Τέλος έχει ενσωματωθεί ως μέθοδος αυθεντικοποίησης στο πρωτόκολλο TLS [54] και βρίσκεται σε αρκετές υλοποιήσεις. Ένα μειονέκτημα που παρουσιάζει είναι ότι δεν είναι προφανές το πως μπορεί να τροποποιηθεί ώστε να χρησιμοποιεί αριθμητική βασισμένη σε ελλειπτικές καμπύλες κάτι που το κάνει λιγότερο αποδοτικό. Ωστόσο η έκδοση SRP-5 χρησιμοποιεί ελλειπτικές καμπύλες.

Λόγω της ευρείας χρήσης πρωτοκόλλου SRP θεωρήθηκε καταλληλότερη η επιλογή του για την αμοιβαία αυθεντικοποίηση του πελάτη με τον εξυπηρετητή.

Ως προς την κρυπτογράφηση από άκρο σε άκρο τα ratchet σχήματα που αναφέρθηκαν καλύπτουν την περίπτωση όταν και τα δυο μέρη μπορούν να ανταλλάξουν μηνύματα την ίδια χρονική στιγμή. Για τα offline μηνύματα και το SCIMP και το Signal ακολουθούν μια παρόμοια τεχνική, ο κάθε χρήστης αποθηκεύει κλειδιά στον server ώστε να χρησιμοποιηθούν για την παραγωγή ενός συμμετρικού κλειδιού ενώ αυτός είναι offline. Το SCIMP χρησιμοποιεί ένα δημόσιο DH κλειδί το οποίο αποθηκεύει ο server για κάποιο διάστημα. Όπως αναφέρθηκε η μέθοδος που χρησιμοποιεί δεν είναι ασφαλής. Για την αποστολή offline μηνυμάτων το Signal χρησιμοποιεί την έννοια των pre-keys. Ο κάθε χρήστης τροφοδοτεί ανά τακτά διαστήματα το server με κλειδιά μιας χρήσης, τα οποία χρησιμοποιεί όποιος θέλει να στείλει μήνυμα σε αυτόν τον χρήστη. Ωστόσο αυτό απαιτεί ο χρήστης να επικοινωνεί με τον εξυπηρετητή που διατηρεί τα pre-keys κάθε φορά που επιθυμεί να εγκαθιδρύσει μια σύνοδο. Ωστόσο στην περίπτωση που δεν απαιτείται η σύνοδος να ανανεώνεται ανά διαστήματα η επικοινωνία με τον εξυπηρετητή γίνεται μόνο την πρώτη φορά που θα σταλεί ένα μήνυμα. Επιπλέον απαιτείται αρκετός αποθηκευτικός χώρο για την αποθήκευση των κλειδιών στον εξυπηρετητή. Τέλος στην περίπτωση που κάποιος καταφέρει να εξαντλήσει τα pre-keys τότε το πρωτόκολλο γίνεται ευάλωτο σε replay attacks [55] [37] αν δεν έχει υλοποιηθεί κάποιος μηχανισμός ώστε να αντικαθίστανται τα παλαιότερα υπογεγραμμένα κλειδιά.

Τα ABE σχήματα που εξετάστηκαν παρέχουν ένα σημαντικό πλεονέκτημα. Δεν απαιτείται η ανταλλαγή κλειδιών μεταξύ των χρηστών. Τα χαρακτηριστικά του κλειδιού που διαθέτει κάθε χρήστης καθώς και του κρυπτοκειμένου καθορίζουν ποιος μπορεί να αποκρυπτογραφήσει. Ωστόσο η κύρια δυσκολία κατά την εφαρμογή είναι ότι απαιτείται μια έμπιστη αρχή η οποία διανέμει τα κλειδιά στους χρήστες ανάλογα με τα χαρακτηριστικά τους. Αυτό σημαίνει πως πρέπει να υπάρχει ένας επιπλέον μηχανισμός για την αυθεντικοποίηση του χρήστη καθώς και ένας ασφαλής τρόπος να παραλάβει το κλειδί του. Τέλος η έμπιστη αρχή είναι σε θέση να αποκρυπτογραφήσει όλα τα μηνύματα καθώς αυτή παράγει και τα κλειδιά. Για τους παραπάνω λόγους η χρήση τους κρίθηκε ακατάλληλη.

Το σχήμα που περιγράφεται εδώ [46] συνδυάζει ένα HIBE σχήμα με ένα ABE σχήμα για να κατασκευάσουμε ένα forward secure σχήμα δημοσίου κλειδιού. Ο τρόπος λειτουργίας του είναι παρόμοιος με ένα ABE σχήμα αλλά αντί να υπάρχει μια κεντρική αρχή διανομής κλειδιών ο κάθε χρήστης αναλαμβάνει τον ρόλο αυτής της αρχής παράγοντας το κάθε κλειδί. Το συγκεκριμένο σχήμα παρέχει forward secrecy επιτρέποντας την τροποποίηση του ιδιωτικού κλειδιού ώστε να μην έχει την δυνατότητα να αποκρυπτογραφήσει συγκεκριμένα μηνύματα ενώ διατηρεί την ικανότητα αποκρυπτογράφησης των άλλων μηνυμάτων.

Επομένως έχουμε παρόμοιο forward secrecy με το Signal για τα offline μηνύματα χωρίς την ανάγκη μιας επιπλέον υποδομής για την διανομή prekeys. Για το online κομμάτι της επικοινωνίας επιλέχθηκε η χρήση του OTR.

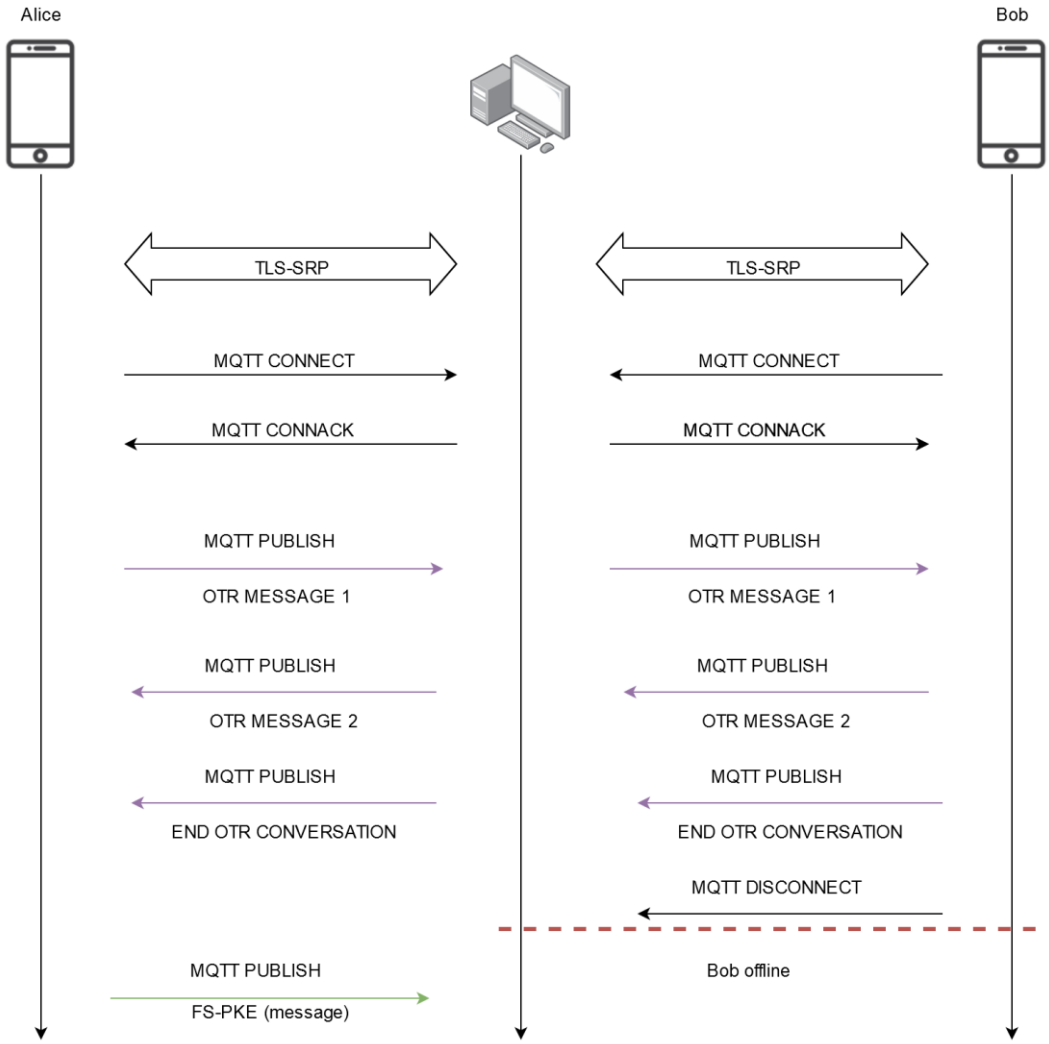
3.3 Σχεδιασμός

3.3.1 Σύντομη περιγραφή

Σε αυτήν την ενότητα θα αναλυθούν σύντομα η προσέγγιση που ακολουθείται καθώς και τα συστατικά στοιχεία του προτεινόμενου σχήματος. Το σχήμα καλείται να λύσει τρία προβλήματα:

- Την προστασία της επικοινωνίας πελάτη broker. Θα χρησιμοποιηθεί το πρωτόκολλο TLS για την προστασία της επικοινωνίας με το πρωτόκολλο SRP να χρησιμοποιείται για την αυθεντικοποίηση. Ο χρήστης συμμετέχει στο πρωτόκολλο με τον κωδικό του και ο broker με τον verifier.
- Την από άκρο σε άκρο κρυπτογράφηση των μηνυμάτων που ανταλλάσσονται όταν και οι δύο χρήστες είναι συνδεδεμένοι. Σε αυτήν την περίπτωση χρησιμοποιείται το πρωτόκολλο OTR.
- Την από άκρο σε άκρο κρυπτογράφηση όταν ο παραλήπτης δεν είναι συνδεδεμένος. Σε αυτή την περίπτωση δε μπορεί να χρησιμοποιηθεί το πρωτόκολλο OTR καθώς δεν μπορούμε να στείλουμε το μήνυμα πριν ανταλλάξουμε κλειδιά. Έχει επιλεγεί η χρήση ενός forward secure σχήματος κρυπτογράφησης δημοσίου κλειδιού(fs-pke) [46]. Το σχήμα αυτό επιτρέπει σε έναν χρήστη να δημοσιεύσει ένα δημόσιο κλειδί και να τροποποιεί το ιδιωτικό κλειδί του ώστε να ανακαλεί την ικανότητα αποκρυπτογράφησης ενός μηνύματος. Το δημόσιο κλειδί δεν αλλάζει. Το μήνυμα κρυπτογραφείται με ένα τυχαίο κλειδί χρησιμοποιώντας τον αλγόριθμο AES. Το συμμετρικό κλειδί κρυπτογραφείται με το fs-pke σχήμα χρησιμοποιώντας το δημόσιο κλειδί του παραλήπτη. Τέλος αποστέλλεται το κρυπτογραφημένο μήνυμα μαζί με το κρυπτογραφημένο κλειδί.

Μια ενδεικτική εκτέλεση του σχήματος φαίνεται στην παρακάτω εικόνα. Με μαύρα βέλη εικονίζονται τα μηνύματα ελέγχου του MQTT (δηλαδή μηνύματα που δεν μεταφέρουν δεδομένα για κάποιον παραλήπτη). Τα μωβ βέλη εικονίζουν μηνύματα που αποστέλλονται με την χρήση του πρωτοκόλλου OTR. Με πράσινο βέλος εικονίζεται το μήνυμα που στέλνεται χρησιμοποιώντας το forward secure σχήμα κρυπτογράφησης δημοσίου κλειδιού(fs-pke).



Εικόνα 7 Ανταλλαγή μηνυμάτων χρησιμοποιώντας το προτεινόμενο σχήμα

3.3.2 Διαχείριση κλειδιών

Αρχικά ο κάθε χρήστης διαθέτει ένα ζεύγος κλειδιών DSA το οποίο χρησιμοποιείται από το πρωτόκολλο OTR ως το long-term κλειδί του χρήστη. Δημιουργεί και ένα ζεύγος κλειδιών για το fs-pke σχήμα. Όπως αναφέρθηκε ο κάθε χρήστης αναλαμβάνει τον ρόλο του PKG,δηλαδή παράγει το master κλειδί και με αυτό το δημόσιο κλειδί καθώς και τα ιδιωτικά κλειδιά για κάθε περίοδο. Στο τέλος καταστρέφει το master κλειδί. Τα ιδιωτικά κλειδιά δεν είναι απαραίτητο να δημιουργηθούν όλα εξαρχής καθώς μπορούν να δημιουργηθούν από το προηγούμενο ιδιωτικό κλειδί. Το δημόσιο κλειδί αποτελείται από τις δημόσιες παραμέτρους του κάθε υποσχήματος δηλαδή $PK = (PK_{PPKE}, PK_{PFSE})$ με PK_{PPKE} να δηλώνει το κλειδί της puncturable κρυπτογράφησης και PK_{PFSE} το κλειδί του HIBE σχήματος. Το δημόσιο κλειδί υπογράφεται από το long-term κλειδί του χρήστη.

Επιπλέον με το DSA κλειδί θα υπογράφονται τα offline μηνύματα ώστε ο παραλήπτης να μπορεί να εξακριβώσει την αυθεντικότητα των μηνυμάτων.

Για την διανομή των δημόσιων κλειδιών μπορούμε να χρησιμοποιήσουμε το ίδιο το MQTT πρωτόκολλο. Όταν συνδέεται μια συσκευή αποστέλλει το δημόσιο DSA κλειδί, καθώς και το δημόσιο

κλειδί του fs-pke σχήματος το οποίο είναι υπογεγραμμένο από το long-term DSA κλειδί. Το topic που χρησιμοποιείται είναι “pk/user_name”.

Για την εξακρίβωση της αυθεντικότητας του κλειδιού χρησιμοποιείται η ακόλουθη TOFU (trust on first use) προσέγγιση: Το λογισμικό αποθηκεύει το πρώτο ζεύγος κλειδιών που θα λάβει από τον server, ελέγχει αν τα υπόλοιπα κλειδιά είναι υπογεγραμμένα από το long-term key. Αν λάβει αργότερα κάποιο διαφορετικό ζεύγος κλειδιών για κάποιον χρήστη τα θεωρεί μη έμπιστα. Ωστόσο ο χρήστης θα πρέπει να εξακριβώσει ότι το DSA κλειδί του άλλου χρήστη είναι σωστό είτε συγκρίνοντας το fingerprint του κλειδιών είτε εκτελώντας το πρωτόκολλο SMP (Socialist Millionaires Protocol) όταν και οι δύο χρήστες είναι συνδεδεμένοι. Τα κλειδιά που χρησιμοποιούνται από τον κάθε χρήστη φαίνονται στον παρακάτω πίνακα.

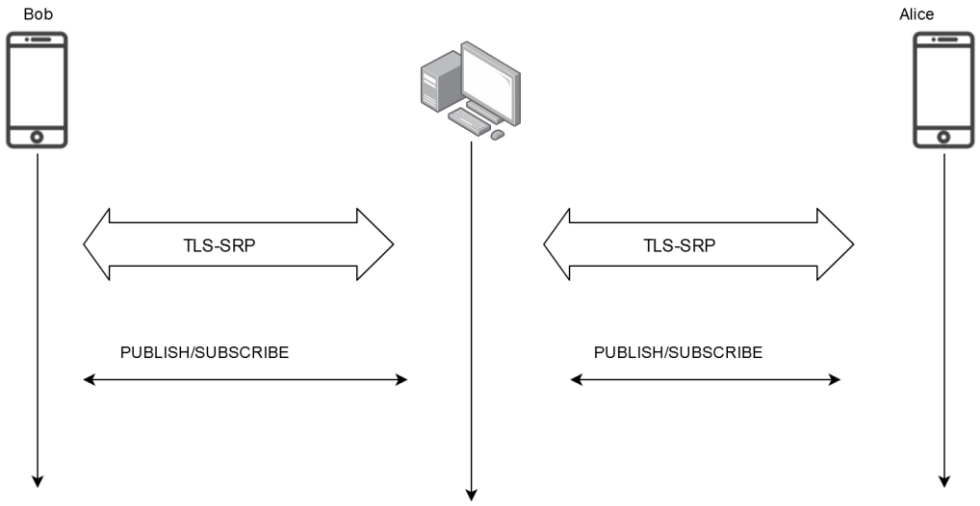
Long-term DSA key	Το ζεύγος κλειδιών χρησιμοποιείται για την αυθεντικοποίηση του χρήστη στο πρωτόκολλο OTR. Επιπλέον χρησιμοποιείται και για την υπογραφή των μηνυμάτων που στέλνονται σε κάποιον παραλήπτη εκτός σύνδεσης.
FS-PKE key	Είναι το ζεύγος κλειδιών που χρησιμοποιείται για την κρυπτογράφηση των offline μηνυμάτων. Το κλειδί αυτό υπογράφεται από το long-term DSA key του χρήστη.

Πίνακας 18 Τα κλειδιά του σχήματος

3.3.3 Αναλυτική περιγραφή σχήματος

Σε αυτήν την ενότητα περιγράφεται η δομή και ο τρόπος λειτουργίας του προτεινόμενου σχήματος. Αρχικά περιγράφεται ο τρόπος με τον οποίο κρυπτογραφείται η επικοινωνία του πελάτη με τον broker και έπειτα πως προστατεύονται τα μηνύματα που ανταλλάσσονται μεταξύ των πελατών.

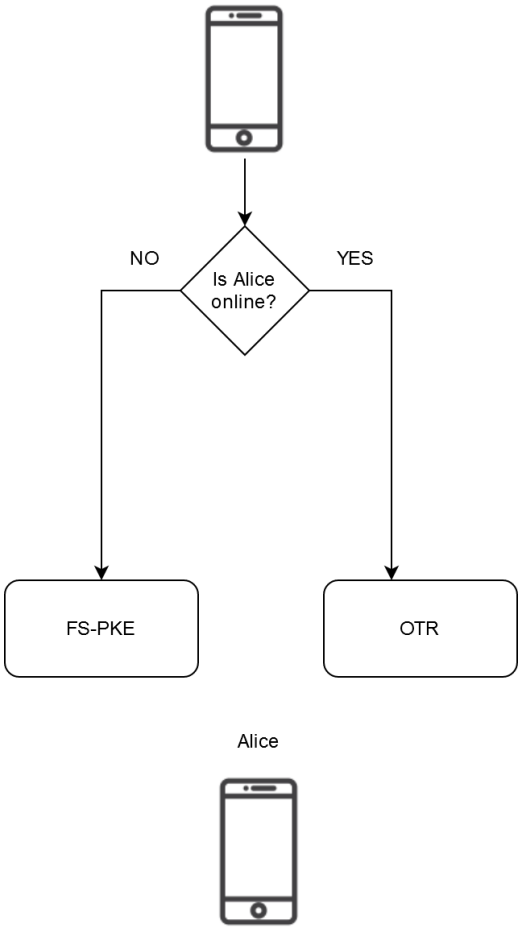
Ειδικότερα η χρήση PAKE πρωτοκόλλων διευκολύνει το αρχικό setup της εφαρμογής καθώς ο χρήστης χρειάζεται να θυμάται μόνο τον κωδικό του. Από την πλευρά του ο broker διατηρεί μόνο τον verifier.Υποθέτουμε ότι ο χρήστης έχει ανταλλάξει έναν κωδικό με τον broker μέσω ενός ασφαλούς καναλιού. Για να αυθεντικοποιηθεί στον broker εκτελείται το πρωτόκολλο SRP. Αν η εκτέλεση είναι επιτυχής ο χρήστης έχει εξακριβώσει ότι επικοινωνεί με τον σωστό broker και ο broker με τον σωστό χρήστη. Έπειτα μπορούν να χρησιμοποιήσουν το κοινό κλειδί για να κρυπτογραφήσουν τα μηνύματα που ανταλλάσσουν. Ωστόσο δεν επαρκεί να κρυπτογραφηθούν μόνο τα PUBLISH μηνύματα πρέπει να κρυπτογραφήσουμε ολόκληρη την επικοινωνία. Μπορούμε να χρησιμοποιήσουμε το SRP μαζί με το πρωτόκολλο TLS. Τα μηνύματα του TLS μπορούν να τροποποιηθούν ώστε να εκτελεστεί το SRP για την αυθεντικοποίηση χρησιμοποιώντας τις επεκτάσεις που παρέχει το TLS (tls extensions) όπως θα αναλυθεί σε παρακάτω ενότητα. Συνεπώς ο χρήστης και ο broker χρησιμοποιούν το TLS-SRP για να εγκαθιδρύσουν ένα ασφαλές κανάλι το οποίο χρησιμοποιείται για την μετέπειτα επικοινωνία τους όπως απεικονίζεται στην παρακάτω εικόνα.



Εικόνα 8 Αυθεντικοποίηση και κρυπτογράφηση επικοινωνίας πελάτη broker

Το πρωτόκολλο MQTT δεν υποθέτει κάποια συγκεκριμένη κατάσταση για τους πελάτες που ανταλλάσσουν μηνύματα. Επιπλέον δεν υποχρεώνει τον αποστολέα και τον παραλήπτη να βρίσκονται συνδεδεμένοι ταυτόχρονα και κάθε συσκευή δεν έχει γνώση αν κάποιος λαμβάνει τα μηνύματα που αυτή στέλνει. Παρόλα αυτά θα μπορούσαμε να διακρίνουμε δύο περιπτώσεις. Στην μια περίπτωση και οι δυο συσκευές είναι συνδεδεμένες στον εξυπηρετητή και να ανταλλάσσουν μηνύματα (μέσω του broker) ενώ στην άλλη περίπτωση μια συσκευή βρίσκεται offline.

Τα περισσότερα πρωτόκολλα που παρέχουν forward secrecy όπως το OTR απαιτούν κάποιο είδος αλληλεπίδρασης μεταξύ των μελών. Όταν και οι δύο συσκευές είναι συνδεδεμένες, υπάρχει η δυνατότητα αλληλεπίδρασης ώστε να μπορούμε να παρέχουμε forward secrecy χρησιμοποιώντας κάποιο ratchet protocol. Στην περίπτωση που κάποια συσκευή είναι offline δεν μπορούμε να ακολουθήσουμε αυτήν την προσέγγιση. Ωστόσο είναι δυνατόν να χρησιμοποιηθεί κάποιο forward secure σχήμα δημοσίου κλειδιού. Η προσέγγιση που ακολουθείται είναι να χρησιμοποιείται το πρωτόκολλο OTR στην περίπτωση που και τα δύο μέρη είναι συνδεδεμένα αλλιώς χρησιμοποιείται forward secure σχήμα δημοσίου κλειδιού όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 9 Επιλογή σχήματος κρυπτογράφησης ανάλογα την κατάσταση του παραλήπτη

Χρησιμοποιώντας το σχήμα [46] μπορούμε να έχουμε forward secrecy για κάθε μήνυμα ανακαλώντας την ικανότητα αποκρυπτογράφησης ενός συγκεκριμένου μηνύματος χωρίς να περιμένουμε να παρέλθει κάποιο χρονικό όριο. Παρόλα αυτά η συνεχής ανανέωση του κλειδιού ώστε να μην μπορεί να αποκρυπτογραφεί κάποια μηνύματα αυξάνει το μέγεθος του ιδιωτικού κλειδιού και το κόστος αποκρυπτογράφησης ενός μηνύματος. Εφόσον το σχήμα χρησιμοποιεί διαφορετικά κλειδιά για κάθε χρονική περίοδο το κόστος αποκρυπτογράφησης λόγω των ανακλήσεων ισχύει μόνο για εκείνη την περίοδο. Αν τα μηνύματα που λαμβάνουμε δεν είναι πολλά(ένα puncture ανά μήνυμα) σε αυτήν την χρονική περίοδο το κόστος αποκρυπτογράφησης θα παραμείνει μικρό.

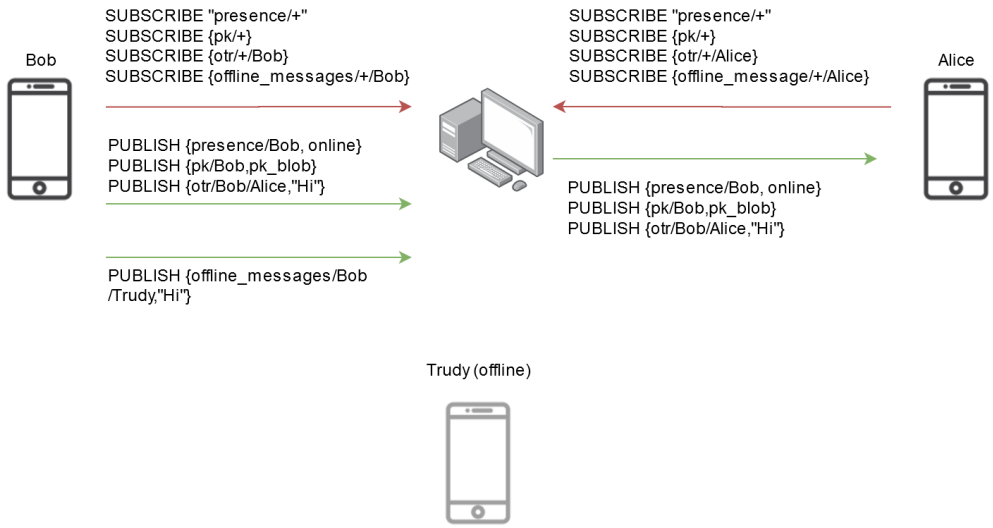
Ωστόσο το πρωτόκολλο MQTT ως ένα publish/subscribe πρωτόκολλο λόγω της αποσύζευξης που παρέχει δεν διαθέτει εγγενώς κάποιο τρόπο ώστε μια συσκευή να γνωρίζει την κατάσταση της άλλης. Όμως μπορεί να υλοποιηθεί εύκολα στέλνοντας ένα μήνυμα σε κάποιο καθορισμένο topic. Για παράδειγμα η Alice μπορεί να στείλει το μήνυμα «online» στο topic “presence/Alice”. Όσες συσκευές επιθυμούν να λάβουν γνώση για την κατάσταση της Alice γράφονται σε αυτό το topic.

Πιο συγκεκριμένα η Alice όταν συνδεθεί στέλνει ένα retained μήνυμα στο topic /presence/Alice με περιεχόμενο το string “online” και όταν αποσυνδεθεί στέλνει ένα retained μήνυμα με περιεχόμενο “offline”. Χρησιμοποιούνται retained μηνύματα ώστε ο broker να διατηρήσει το τελευταίο μήνυμα που έστειλε η Alice και να το στείλει σε κάποιον που εγγράφεται στο topic πρώτη φορά. Επομένως η κάθε συσκευή εγγράφεται στο topic ”presence/+” και αποστέλλει μηνύματα στο topic “presence/user_name”.

Τέλος χρησιμοποιείται η τελευταία επιθυμία(last will) που παρέχει το MQTT για να είναι σίγουρο ότι θα ενημερωθεί η κατάσταση της κάθε συσκευής στην περίπτωση ξαφνικής αποσύνδεσης. Κάθε συσκευή θέτει ως τελευταία επιθυμία την αποστολή ενός μηνύματος με περιεχόμενο «offline». Ο broker όταν εντοπίσει ότι κάποια συσκευή αποσυνδέθηκε απρόσμενα(δεν έλαβε ένα πακέτο PINGREQ μέσα στο προκαθορισμένο χρονικό διάστημα) θα στείλει το μήνυμα στις εγγεγραμμένες συσκευές.

Ο κάθε χρήστης διατηρεί μια λίστα με την κατάσταση των άλλων χρηστών και ανάλογα επιλέγει πως θα στείλει το μήνυμα. Αν και ο άλλος χρήστης βρίσκεται συνδεδεμένος μπορούν να ξεκινήσουν μια OTR σύνοδο. Αν είναι αποσυνδεδεμένος θα χρησιμοποιήσει το fs-pke σχήμα για να στείλει ένα μήνυμα. Στην παρακάτω εικόνα φαίνονται τα μηνύματα που ανταλλάσσονται ανάμεσα στον Bob και την Alice. Επιπλέον ο Bob στέλνει ένα offline μήνυμα στην Trudy. Έχουν παραληφθεί τα μηνύματα που στέλνει η Alice για να ενημερώσει την κατάσταση της και για να γνωστοποιήσει τα δημόσια κλειδιά της.

Τα topics που χρησιμοποιούνται για την αποστολή /παραλαβή μηνυμάτων έχουν την ακόλουθη μορφή: otr/<αποστολέας>/<παραλήπτης> και offline_messages/<αποστολέας>/<παραλήπτης>.



Εικόνα 10 Μηνύματα που ανταλλάσσονται κατά την λειτουργία της εφαρμογής. Τα κόκκινα βέλη αναπαριστούν τα μηνύματα Subscribe ενώ τα πράσινα τα μηνύματα Publish

Τα topics που χρησιμοποιούνται και ο ρόλος τους αναλύονται στον παρακάτω πίνακα.

Πελάτης	**Action**	**Topic**	**Σκοπός**
Bob	Subscribe	presence/+	Το topic ενημερώνει για την κατάσταση των άλλων χρηστών. Πχ presence/Alice αναφέρεται στο αν η Alice είναι συνδεδεμένη ή αποσυνδεδεμένη
		pk/+	Σε αυτό το topic εγγράφεται ο κάθε χρήστης για να λάβει τα δημόσια κλειδιά των άλλων χρηστών.
		otr/+/Bob	Σε αυτό το topic εγγράφεται ο χρήστης για να λάβει τα μηνύματα που απευθύνονται για αυτόν όσο

			είναι συνδεδεμένος. Για την κρυπτογράφηση των μηνυμάτων χρησιμοποιείται το πρωτόκολλο otr.
		offline_messages/+/Bob	Σε αυτό το topic ο χρήστης λαμβάνει τα μηνύματα που εστάλησαν όσο αυτός ήταν αποσυνδεδεμένος. Για την κρυπτογράφηση των μηνυμάτων χρησιμοποιείται το fs-pke σχήμα.
	Publish	presence/Bob	Κάνοντας publish σε αυτό το topic ο χρήστης δημοσιεύει την κατάσταση του αν είναι συνδεδεμένος(στέλνει το μήνυμα online) ή αποσυνδεδεμένος(στέλνει το μήνυμα offline).
		pk/Bob	Σε αυτό το topic δημοσιεύει ο χρήστης τα δημόσια κλειδιά του.
		Otr/Bob/<recipient>	Σε αυτό το topic δημοσιεύει μηνύματα τα οποία κρυπτογραφούνται με την χρήση του πρωτοκόλλου OTR. Για παράδειγμα αν ο Bob θέλει να στείλει μήνυμα στην Alice θα χρησιμοποιήσει το topic otr/Bob/Alice.
		Offline_messages/Bob/<recipient>	Σε αυτό το topic δημοσιεύει τα μηνύματα όταν ο παραλήπτης δεν είναι συνδεδεμένος. Για παράδειγμα αν ο Bob ξέρει ότι η Alice δεν είναι συνδεδεμένη και θέλει να στείλει ένα μήνυμα θα κρυπτογραφήσει το μήνυμα με το fs-pke σχήμα και θα στείλει το μήνυμα στο topic offline_messages/Bob/Alice.

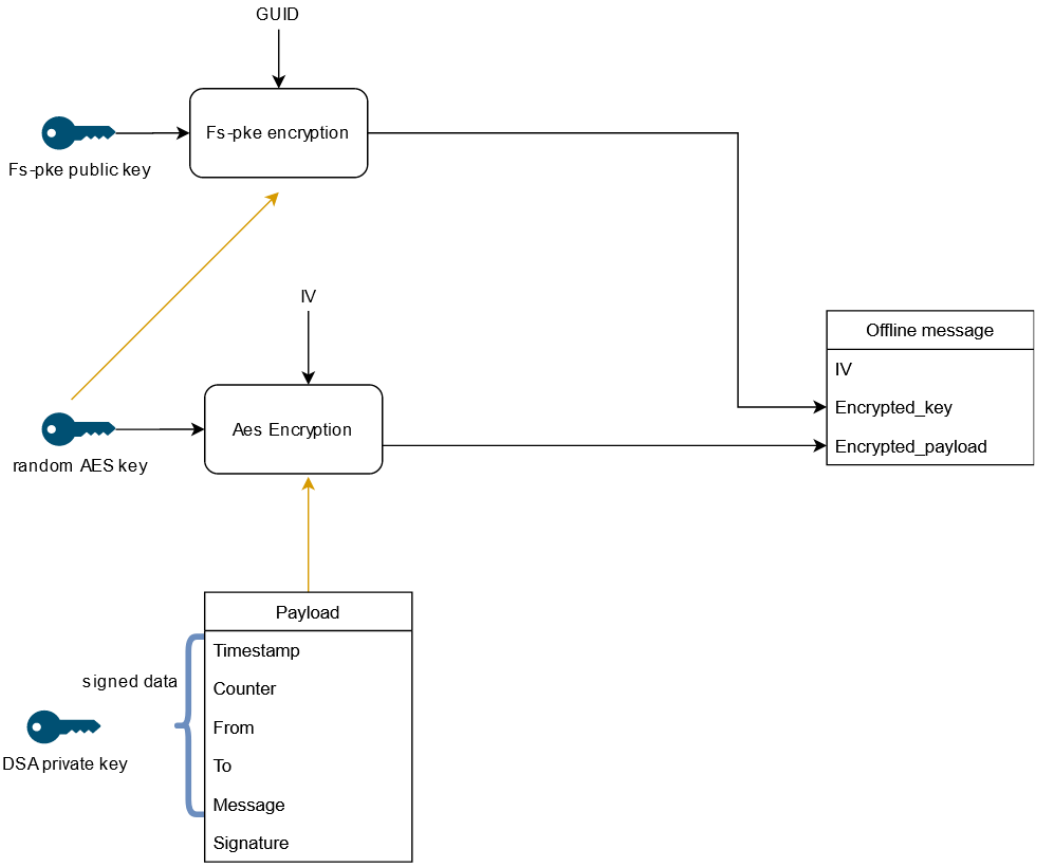
Πίνακας 19 Τα topics του προτεινόμενου συστήματος

3.3.4 Διαχείριση offline μηνυμάτων

Η βασική ιδέα που χρησιμοποιούμε για την αποστολή μηνυμάτων σε αποσυνδεδεμένους χρήστες είναι ο χρήστης κρυπτογραφεί το μήνυμα χρησιμοποιώντας έναν συμμετρικό αλγόριθμο όπως τον AES και να κρυπτογραφεί το συμμετρικό κλειδί με το fs-pke σχήμα. Ο άλλος χρήστης όταν λάβει το μήνυμα αποκρυπτογραφεί το κρυπτογραφημένο συμμετρικό κλειδί χρησιμοποιώντας το ιδιωτικό κλειδί του και έπειτα αποκρυπτογραφεί το κρυπτογραφημένο μήνυμα. Τέλος για να εγγυηθεί το forward secrecy του μηνύματος τροποποιεί το ιδιωτικό του κλειδί ώστε να μην μπορεί πια να αποκρυπτογραφήσει το συγκεκριμένο μήνυμα.

Για να είναι δυνατόν να αφαιρέσουμε την δυνατότητα αποκρυπτογράφησης ενός συγκεκριμένου μηνύματος το κάθε μήνυμα φέρει μια μοναδική ετικέτα. Ως ετικέτα χρησιμοποιούμε ένα GUID το οποίο παράγεται τυχαία. Απαιτούμε να είναι τυχαίο γιατί στην περίπτωση που οι ετικέτες είναι προβλέψιμες

ένας επιτιθέμενος θα μπορούσε να στείλει μηνύματα τα οποία θα φέρουν τις ετικέτες που θα χρησιμοποιηθούν αργότερα και έτσι να εμποδίσει τα μηνύματα ενός χρήστη να αποκρυπτογραφηθούν. Αυτό θα συμβεί διότι ο παραλήπτης θα έχει αφαιρέσει την ικανότητα του κλειδιού να αποκρυπτογραφεί μηνύματα με αυτές τις ετικέτες. Μια άλλη μέθοδος για να παράγουμε την ετικέτα θα ήταν να χρησιμοποιούμε το id του αποστολέα μαζί με τον αύξων αριθμό του μηνύματος(πχ bob_4 για το τέταρτο μήνυμα που θα στείλει ο Bob). Σε αυτήν την περίπτωση θα πρέπει να ελέγχεται ότι το μήνυμα προέρχεται από τον αποστολέα. Η λειτουργία του σχήματος φαίνεται στην παρακάτω εικόνα.



Εικόνα 11 κρυπτογράφηση offline μηνύματος. Τα κίτρινα βέλη ορίζουν τα δεδομένα προς κρυπτογράφηση

Το μήνυμα που αποστέλλεται φέρει τα παρακάτω πεδία:

- **iv:** Είναι το διάνυσμα αρχικοποίησης (initialization vector) που χρησιμοποιήθηκε κατά την κρυπτογράφηση με τον συμμετρικό αλγόριθμο
- **Encrypted_symmetric_key:** Είναι το κρυπτογραφημένο συμμετρικό κλειδί το οποίο κρυπτογραφήθηκε με το pk-fse σχήμα
- **Encrypted_payload:** Περιέχει το κρυπτογραφημένο περιεχόμενο
- **Guid:** Περιέχει την ετικέτα του μηνύματος. Δεν είναι ξεχωριστό πεδίο αλλά βρίσκεται στο encrypted_key.

Το κρυπτογραφημένο περιεχόμενο εκτός από το μήνυμα φέρει και κάποια επιπλέον στοιχεία και έχει την παρακάτω δομή:

- **Timestamp:** Η χρονοσφραγίδα την στιγμή που δημιουργήθηκε το μήνυμα

- **Counter:** Δείχνει την σειρά αυτού του μηνύματος πχ αν είναι 3 αυτό είναι το τρίτο μήνυμα. Το πεδίο χρησιμοποιείται ώστε να μπορούμε να αναδιατάξουμε μηνύματα που έφτασαν με λάθος σειρά.
- **From:** Ποιος είναι ο αποστολέας του μηνύματος
- **To:** Ποιος είναι ο αποδέκτης του μηνύματος. Το πεδίο αυτό είναι αναγκαίο ώστε να αποφευχθεί η δρομολόγηση των μηνυμάτων από έναν επιτιθέμενο σε άλλον παραλήπτη.
- **Message:** Περιέχει το μήνυμα που θέλει να στείλει ο χρήστης.
- **Signature:** Περιέχει την ψηφιακή υπογραφή όλων των παραπάνω πεδίων με το DSA κλειδί.

Όταν ο παραλήπτης λάβει ένα μήνυμα πρέπει να αποκρυπτογραφήσει το συμμετρικό κλειδί χρησιμοποιώντας το fs-pke σχήμα. Αν η δυνατότητα αποκρυπτογράφησης έχει ήδη αφαιρεθεί για την συγκεκριμένη ετικέτα η αποκρυπτογράφηση θα αποτύχει(Το κρυπτοκείμενο περιέχει την ετικέτα). Τότε το μήνυμα απορρίπτεται. Σε περίπτωση που η αποκρυπτογράφηση του συμμετρικού κλειδιού είναι επιτυχής αποκρυπτογραφείται κρυπτογραφημένο φορτίο (encrypted_payload) και ελέγχεται αν ψηφιακή υπογραφή είναι έγκυρη χρησιμοποιώντας το δημόσιο κλειδί το αποστολέα. Αν η επαλήθευση δεν είναι επιτυχής το μήνυμα απορρίπτεται. Έπειτα ελέγχεται αν είμαστε ο σωστός παραλήπτης του μηνύματος. Τέλος χρησιμοποιείται η τιμή counter για να τοποθετηθεί το μήνυμα στην σωστή σειρά. Επιπλέον εφόσον τα μηνύματα μας είναι αριθμημένα, μπορούμε να αποφύγουμε τις επιθέσεις επανάληψης (replay attacks) ελέγχοντας αν έχουμε λάβει μήνυμα με παλαιότερο αύξοντα αριθμό.

Ως προς τον αλγόριθμο συμμετρικής κρυπτογράφησης καλό είναι να χρησιμοποιηθεί ένας AEAD(Authenticated Encryption with Associated Data) αλγόριθμος όπως ο Aes gcm. Οι συγκεκριμένοι αλγόριθμοι εγγυώνται την εμπιστευτικότητα αλλά και την αυθεντικότητα των δεδομένων. Αν το κρυπτοκείμενο τροποποιηθεί κατά την μεταφορά αυτό θα γίνει αντιληπτό καθώς οι αποκρυπτογράφηση θα αποτύχει. Αν δεν χρησιμοποιηθεί ένας AEAD αλγόριθμος η τροποποίηση θα γίνει και πάλι αντιληπτή μετά την αποκρυπτογράφηση καθώς η επαλήθευση της ψηφιακής υπογραφής θα αποτύχει. Ωστόσο η χρήση ενός AEAD αλγορίθμου βοηθά να αποφευχθούν επιθέσεις που βασίζονται στην ύπαρξη παράπλευρων καναλιών (side-channels) λόγω της υλοποίησης της εφαρμογής. Για παράδειγμα η εφαρμογή μπορεί να επιστρέφει ένα συγκεκριμένο σφάλμα αν η αποκρυπτογράφηση του μηνύματος αποτύχει λόγω λανθασμένου padding στην περίπτωση που χρησιμοποιείται cbc mode (padding oracle attack) είτε στην περίπτωση που η μορφή των αποκρυπτογραφημένων δεδομένων δεν είναι αυτή που αναμένει η εφαρμογή(format oracle attack).

Επειδή η επικοινωνία δεν είναι σύγχρονη οι μετρητές που κρατά το κάθε μέλος είναι πιθανόν να αποσυγχρονιστούν. Αν η Alice στείλει ένα μήνυμα και ο Bob δεν το λάβει ποτέ, ο μετρητής της Alice θα βρίσκεται ένα μήνυμα μπροστά από τον μετρητή του Bob. Όταν ο Bob λάβει το επόμενο μήνυμα θα καταλάβει ότι έχει χαθεί ένα μήνυμα. Ωστόσο δεν μπορούμε να ζητήσουμε από την Alice να ξαναστείλει καθώς μπορεί να είναι offline. Αν αυτό επαναληφθεί πολλές φορές οι μετρητές θα διαφέρουν κατά πολύ.

Στην περίπτωση που τα μηνύματα φτάσουν με άλλη σειρά (για παράδειγμα 6,5) αν θεωρήσουμε ως τελευταίο μήνυμα το 6 όταν φτάσει το πέμπτο μήνυμα θα απορριφθεί. Επομένως θα πρέπει να αποθηκεύουμε και ποια νούμερα έχουμε λάβει τα οποία είναι μεγαλύτερα από την τελευταία σωστή τιμή του μετρητή.

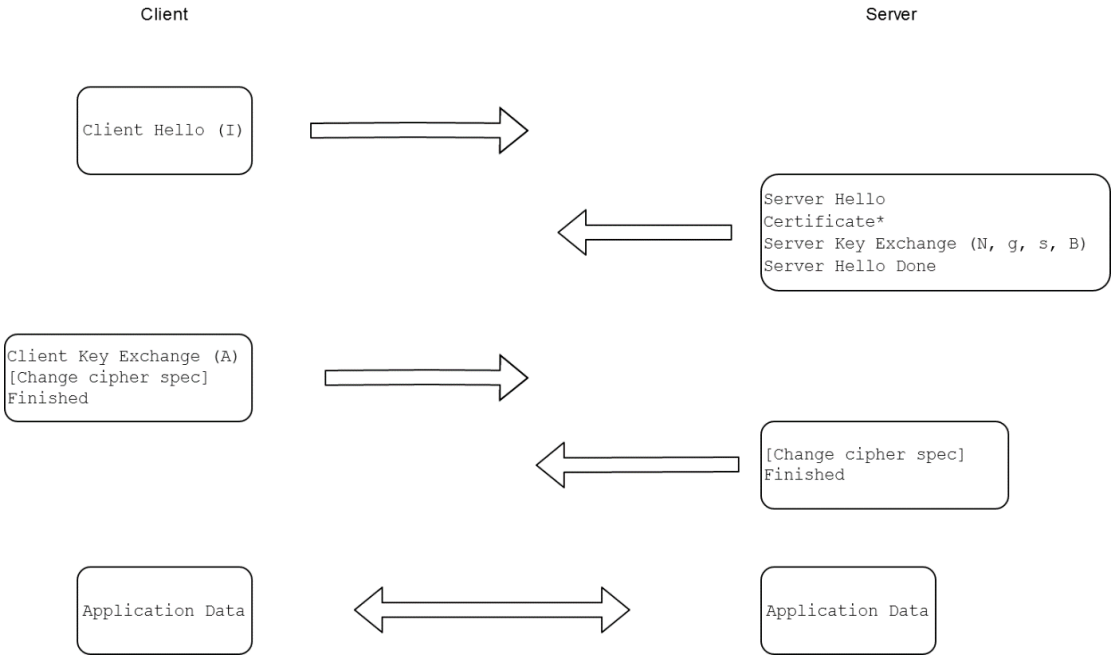
Η τιμή counter μπορεί να μηδενίζεται ανά χρονική περίοδο αν δεν επιθυμούμε να μετράμε τα μηνύματα από την αρχή, και να χρησιμοποιείται συνδυασμός της χρονικής περιόδου και της τιμής counter για την αναδιάταξη περιορίζοντας το παραπάνω πρόβλημα στην τρέχουσα χρονική περίοδο. Επιπλέον σε κάθε χρονική περίοδο χρησιμοποιείται διαφορετικό κλειδί ένα μήνυμα προηγούμενης χρονικής περιόδου δεν μπορεί να τοποθετηθεί νωρίτερα ή αργότερα. Αν τοποθετηθεί σκοπίμως σε άλλη χρονική περίοδο θα αποκρυπτογραφηθεί με λάθος κλειδί με αποτέλεσμα η αποκρυπτογράφηση να μην γίνει σωστά και ο έλεγχος της ψηφιακής υπογραφής να αποτύχει.

Αξίζει να σημειωθεί ότι υπάρχει η περίπτωση κάποιος κακόβουλος χρήστης να επιχειρήσει να αποστείλει πολλά μηνύματα ώστε να δημιουργήσει μια DOS κατάσταση σε κάποιον χρήστη

εκμεταλλευόμενος το γεγονός κάθε φορά που θα λαμβάνει ένα μήνυμα θα ανακαλεί την ικανότητα αποκρυπτογράφησης αυτού του μηνύματος. Κάθε φορά που τροποποιείται το μυστικό κλειδί ώστε να ανακληθεί η δυνατότητα αποκρυπτογράφησης ενός μηνύματος το κόστος αποκρυπτογράφησης για αυτήν την χρονική περίοδο αυξάνει. Είναι πολύ πιθανό κάποια στιγμή ο χρόνος αποκρυπτογράφησης να γίνει ασύμφορος. Για να αποφευχθούν τέτοιες περιπτώσεις, όταν λάβουμε πολλά μηνύματα μπορούμε να σταματήσουμε να τροποποιούμε το ιδιωτικό κλειδί (για τα μηνύματα που λαμβάνουμε από αυτόν τον χρήστη) για να διατηρήσουμε την απόδοση σε ένα επιθυμητό σημείο. Όταν το χρονικό διάστημα του κλειδιού παρέλθει διαγράφουμε το κλειδί. Μέσα σε αυτό το χρονικό διάστημα τα μηνύματα αυτού του χρήστη μπορούν να αποκρυπτογραφηθούν αν κάποιος αποκτήσει πρόσβαση στο ιδιωτικό κλειδί.

3.3.5 TLS-SRP

Το πρωτόκολλο SRP μπορεί να υλοποιηθεί χρησιμοποιώντας τα μηνύματα που ανταλλάσσονται κατά την χειραψία του TLS όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 12 Μηνύματα του TLS όταν χρησιμοποιείται το SRP ως πρωτόκολλο αυθεντικοποίησης

Θα αναλυθούν οι αλλαγές στα μηνύματα που ανταλλάσσονται όταν χρησιμοποιείται το SRP. Το όνομα χρήστη τοποθετείται στο μήνυμα client hello χρησιμοποιώντας τον μηχανισμό επέκτασης που παρέχει το TLS. Στην περίπτωση που ο εξυπηρετητής δεν έχει τον verifier για αυτόν τον χρήστη θα απορρίψει την χειραψία με σφάλμα ότι δεν γνωρίζει τον συγκεκριμένο χρήστη(unknown_psk_identity). Αν είναι επιθυμητό να μην αποκαλυφθεί η ύπαρξη του χρήστη ο εξυπηρετητής μπορεί να συνεχίσει την εκτέλεση του πρωτοκόλλου όπως στην περίπτωση που είχε τον verifier και να απορρίψει το μήνυμα finished του πελάτη με σφάλμα bad_record_mac όπως θα συνέβαινε και στην περίπτωση που ο κωδικός ήταν λανθασμένος.

Για να προσομοιώσουμε την λειτουργία του πρωτοκόλλου χωρίς τον verifier πρέπει ο εξυπηρετητής να επιστρέφει το ίδιο salt s και τις ίδιες παραμέτρους N, g για το ίδιο όνομα χρήστη. Μια περίπτωση είναι ο εξυπηρετητής να διατηρεί ένα μυστικό κλειδί seed_key και να χρησιμοποιεί μια HMAC για να παράγει τα salts. Η τιμή B μπορεί να παράγεται τυχαία. Ωστόσο αξίζει να σημειωθεί ότι ο εξυπηρετητής θα πρέπει να προσομοιώσει και τυχόν χρονικές καθυστερήσεις. Παρόμοια ένας τρόπος να πραγματοποιηθεί αυτό είναι να χρησιμοποιηθεί η HMAC για να παραχθεί ο verifier και έπειτα να εκτελεστεί το πρωτόκολλο κανονικά [54].

Το μήνυμα server certificateχρησιμοποιείται στην περίπτωση που ο εξυπηρετητής επιλέξει κάποιο αλγόριθμο που χρησιμοποιεί ψηφιακή υπογραφή για την αυθεντικοποίηση πρέπει να συμπεριλάβει και ένα πιστοποιητικό. Αν όχι το μήνυμα παραλείπεται.

Το μήνυμα server key exchange περιέχει τον πρώτο αριθμό N τον γεννήτορα της ομάδας g και το salt s. Επιπλέον περιέχει την τιμή B που υπολογίζεται ως $B = k * v + g^b mod N$. Η τιμή k ισούται με k = SHA1(N | PAD(g)). Για την επιλογή των τιμών g,N,b ισχύουν οι ίδιες απαιτήσεις που ισχύουν κατά την εκτέλεση του SRP.

Τέλος το μήνυμα client key exchange περιέχει την τιμή $A = g^a mod N$.

Ο πελάτης υπολογίζει το premaster secret ως εξής:

- Ο χρήστης εισάγει τις τιμές I,P
- Ο server στέλνει τις τιμές N,g,s,B
- a= random()
- $A = g^a mod N$
- u=SHA1(PAD(A) | PAD(B))
- k=SHA1(N | PAD(g))
- x = SHA1(s | SHA1(I | ":" | P))
- $\text{premastersecret} = \left(B - (k \cdot g^x)\right)^{(a+(u \cdot x))} \text{modN}$

Ο εξυπηρετητής υπολογίζει το premaster secret ως εξής:

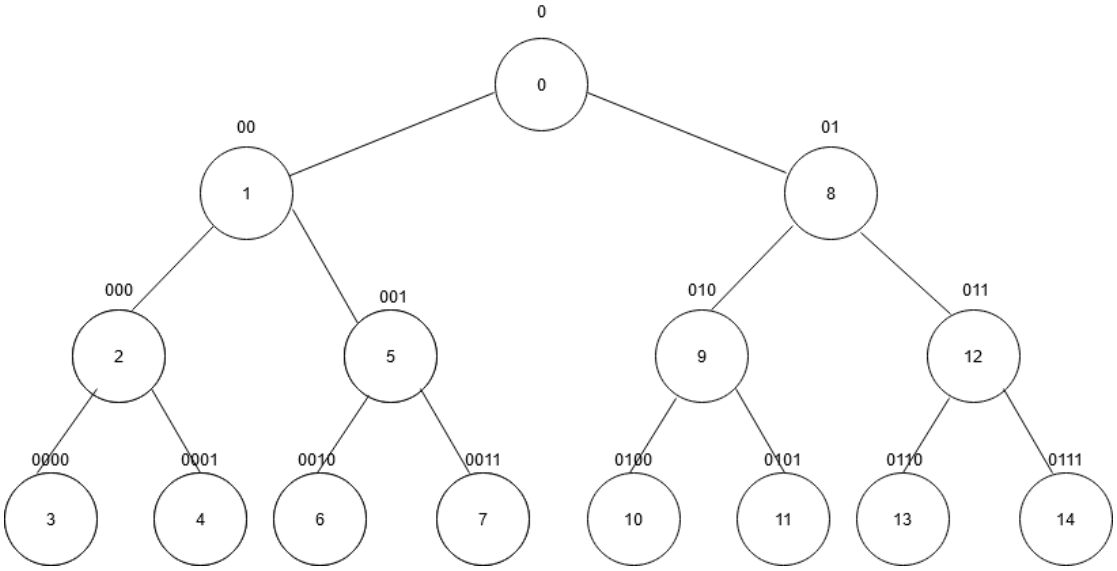
- Ανακτά τις τιμές N, g, s, v από το αρχείο κωδικών
- b = random()
- k = SHA1(N | PAD(g))
- $B = k * v + g^b mod N$
- Λαμβάνει την τιμή A από τον πελάτη
- u = SHA1(PAD(A) | PAD(B))
- $\text{premastersecret} = (A \cdot v^u)^b \text{modN}$

Τα μηνύματα finished επιτελούν την ίδια λειτουργία που επιτελούν τα μηνύματα M1, M2 δηλαδή να εξακριβώσουν αν και τα δυο μέρη έχουν παράγει το ίδιο κλειδί. Αν ο πελάτης ή ο εξυπηρετητής υπολογίσουν λάθος το premaster secret τότε τα finished μηνύματα θα αποκρυπτογραφηθούν λάθος και η εκτέλεση του πρωτοκόλλου θα τερματιστεί με σφάλμα bad_record_mac.

3.3.6 Ανάλυση pk-fse σχήματος

Θα εξετάσουμε πως οι διάφορες παράμετροι επηρεάζουν τους αλγόριθμους του σχήματος. Αρχικά πρέπει να αναλυθεί πως απεικονίζονται οι χρονικές περίοδοι σε ένα δυαδικό δένδρο. Η ρίζα του δένδρου αναπαριστά την χρονική περίοδο 0 και οι υπόλοιπες αντιστοιχούνται στους κόμβους του δένδρου ακολουθώντας pre order traversal.

Η δυαδική αναπαράσταση των ταυτοτήτων των κόμβων προκύπτει ως εξής: έστω w^i ο κόμβος που αντιστοιχεί στην χρονικής περίοδο i και έχουμε N-1 χρονικές περιόδους. Αν ο κόμβος w^i δεν είναι φύλλο του δένδρου τότε $w^{i+1} = w^i 0$. Αν είναι φύλλο τότε έχουμε $w^{i+1} = w'1$ με w' να είναι η μεγαλύτερη ακολουθία χαρακτήρων έτσι ώστε το w'0 να είναι πρόθεμα του w^i.



Εικόνα 13 Αναπαράσταση των χρονικών στιγμών σε δυαδικό δένδρο

Όταν παράγουμε το κλειδί για την επόμενη χρονική περίοδο παράγουμε τα κλειδιά των κόμβων παιδιών της τωρινής χρονικής περιόδου. Για παράδειγμα αν είμαστε στην χρονική περίοδο 1 και θέλουμε να προχωρήσουμε στην χρονική περίοδο 2 παράγουμε το κλειδί της χρονικής περιόδου 2 και της χρονικής περιόδου 5. Το κλειδί της χρονικής περιόδου 1 μπορεί να διαγραφεί. Συνεπώς κάθε φορά στην χρονική περίοδο i+1 διατηρούμε το κλειδί του κόμβου w^{i+1} καθώς και όλα τα κλειδιά των δεξιών παιδιών των κόμβων που βρίσκονται στην μονοπάτι από την ρίζα έως στον κόμβο w^{i+1}.

Οι αλγόριθμοι keygen, encrypt, puncture δεν εξαρτώνται από των αριθμό των punctures του μυστικού κλειδιού ωστόσο εξαρτώνται από τον αριθμό των χρονικών περιόδων που επιτρέπονται.

Η απόδοση του αλγορίθμου next_interval είναι ανεξάρτητη του αριθμού των punctures που έχουν πραγματοποιηθεί, αλλά εξαρτάται γραμμικά από το μέγεθος του κλειδιού της εκάστοτε χρονικής περιόδου. Τα κλειδιά στο HIBE σχήμα γίνονται μικρότερα καθώς αυξάνει το μέγεθος της ταυτότητας επομένως η παραγωγή του επόμενου κλειδιού γίνεται ταχύτερη με την πάροδο του χρόνου.

Ο αλγόριθμος αποκρυπτογράφησης (decrypt) εξαρτάται μόνο από τον αριθμό των punctures και των αριθμό των ετικετών που υποστηρίζονται και δεν επηρεάζεται καθόλου από την τωρινή χρονική περίοδο ή από τον αριθμό των χρονικών περιόδων.

Έχουμε επιλέξει να χρησιμοποιήσουμε μόνο μια ετικέτα για το κάθε μήνυμα, δίνοντας στον παραλήπτη την δυνατότητα να ανακαλέσει την δυνατότητα αποκρυπτογράφησης για το συγκεκριμένο μήνυμα. Ωστόσο το γενικότερο σχήμα επιτρέπει την χρήση πολλών ετικετών. Για παράδειγμα οι υπόλοιπες ετικέτες θα μπορούσαν να περιέχουν και άλλα δεδομένα όπως το id του αποστολέα ή κάποιο θέμα συζήτησης. Σε αυτήν την περίπτωση θα ήταν δυνατό να εμποδίσουμε την αποκρυπτογράφηση όλων των μηνυμάτων του συγκεκριμένου αποστολέα ή των μηνυμάτων που αφορούν κάποια συγκεκριμένη συζήτηση.

Ένα άλλο σημείο που χρειάζεται διερεύνηση είναι το πως θα επιλεγεί το πλήθος των χρονικών περιόδων(απαιτούνται τόσα κλειδιά όσες και οι χρονικές περίοδοι) καθώς και η διάρκεια αυτών.

Η χρονική διάρκεια της περιόδου επηρεάζει την απόδοση της αποκρυπτογράφησης με των παρακάτω τρόπο. Αν η διάρκεια είναι μεγάλη τότε σε αυτό το διάστημα αναμένουμε να λάβουμε αρκετά μηνύματα με αποτέλεσμα να χρειαστεί να κάνουμε αρκετές τροποποιήσεις(punctures) στο κλειδί για να αφαιρέσουμε την ικανότητα αποκρυπτογράφησης για αυτά τα μηνύματα. Όσο αυξάνεται ο αριθμός των punctures αυξάνεται και ο χρόνος που απαιτείται για την αποκρυπτογράφηση. Επιπλέον η διάρκεια

επηρεάζει και το μέγεθος των κλειδιών καθώς για μικρότερα διαστήματα χρειαζόμαστε δένδρο μεγαλύτερου βάθους για να καλύψουμε το ίδιο χρονικό διάστημα. Το βάθους του δένδρου επηρεάζει και τον χρόνο που απαιτείται για την παραγωγή του κάθε κλειδιού. Συνεπώς μικρότερα χρονικά διαστήματα αυξάνουν την γενικότερη δουλειά που απαιτείται για την παράγωγη των κλειδιών.

Συνεπώς η απόδοση του σχήματος εξαρτάται από την διάρκεια του κάθε διαστήματος και το πλήθος αυτών.

Από τις μετρήσεις που πραγματοποιήθηκαν [46] η ιδανική χρονική διάρκεια είναι αυτή κατά την οποία θα λάβουμε ένα μήνυμα. Το κόστος παραγωγής του επόμενου κλειδιού δεν επηρεάζει το κόστος της αποκρυπτογράφησης καθώς το καινούριο κλειδί θα υπολογιστεί στο τέλος της χρονικής περιόδου. Στην περίπτωση μας το συγκεκριμένο σχήμα χρησιμοποιείται για την αποστολή μηνυμάτων την στιγμή που ο παραλήπτης βρίσκεται offline, επομένως το χρονικό διάστημα στο οποίο θα λάβει ένα μήνυμα ακριβώς μπορεί να είναι σχετικά μεγάλο πχ 10 λεπτά καθώς ένας χρήστης είναι πιθανόν να μην στείλει μήνυμα αμέσως αν ο παραλήπτης δεν απαντήσει. Αναλόγως την εφαρμογή η ιδανική χρονική περίοδος θα διαφέρει.

Τέλος μας ενδιαφέρει και το χρονικό παράθυρο κατά το οποίο θα μπορούμε να αποκρυπτογραφήσουμε τυχόν καθυστερημένα μηνύματα. Το μέγεθος του παραθύρου επηρεάζει το πλήθος των κλειδιών που θα χρειαστεί να διατηρήσουμε και όχι την γενικότερη απόδοση του σχήματος. Ο χρήστης πρέπει να διατηρήσει τα κλειδιά τα οποία έληξαν όσο αυτός ήταν offline ή όσο αναμένει ότι είναι η μέγιστη καθυστέρηση του δικτύου, ώστε να μπορεί να αποκρυπτογραφήσει μηνύματα που φτάσουν αργότερα. Αν θεωρήσουμε ότι ο χρήστης θα είναι το πολύ 6 ώρες offline πρέπει να κρατήσουμε τα κλειδιά που αντιστοιχούν σε αυτές τις 6 ώρες.

4 Υλοποίηση

Ο βασικός σκοπός είναι η υλοποίηση ενός MQTT client που θα χρησιμοποιεί τις τεχνικές που αναφέρθηκαν για αυθεντικοποίηση και για την προστασία των μηνυμάτων από άκρο σε άκρο. Συνοψίζοντας επιθυμούμε ο πελάτης να αυθεντικοποιείται στον broker με την χρήση του πρωτοκόλλου SRP, και ανάλογα την περίπτωση να ανταλλάσσει μηνύματα χρησιμοποιώντας το πρωτόκολλο OTR ή το σχήμα δημοσίου κλειδιού που αναφέρθηκε.

4.1 Mosquitto broker

Για την υλοποίηση των παραπάνω είναι απαραίτητος ένας broker που να επιτρέπει την χρήση του TLS-SRP. Αυτή είναι και η μοναδική απαίτηση που έχουμε από τον broker καθώς οι υπόλοιπές λειτουργίες(κρυπτογράφηση μηνυμάτων,forward secrecy) αφορούν την υλοποίηση του πελάτη. Ως broker επιλέχθηκε ο Mosquitto broker καθώς πρόκειται για ένα λογισμικό ανοικτού κώδικα το οποίο είναι δυνατόν να τροποποιηθεί. Ο συγκεκριμένος broker για την υλοποίηση των κρυπτογραφικών πρωτοκόλλων χρησιμοποιεί την βιβλιοθήκη openssl.

Η βιβλιοθήκη openssl υλοποιεί τα πρωτόκολλα TLS και SSL αλλά γενικότερα αποτελεί μια βιβλιοθήκη κρυπτογραφίας γενικού σκοπού. Ωστόσο ενώ η openssl παρέχει την δυνατότητα χρήσης του TLS-SRP ο broker δεν την υποστηρίζει. Ο broker τροποποιήθηκε για να υποστηρίξει την χρήση TLS-SRP.

Για την αρχικοποίηση του SRP πρέπει να κληθούν δύο συναρτήσεις η SSL_CTX_set_srp_cb_arg(γραμμή 581) και η SSL_CTX_set_srp_username_callback(γραμμή 583). Η πρώτη προσδιορίζει τα δεδομένα(η τιμή srp_callback_parm) που θα περαστούν στην κλήση της SSL_CTX_set_srp_username_callback. Η SSL_CTX_set_srp_username_callback θέτει την συνάρτηση που θα κληθεί όταν χρειαστεί να γίνει η αυθεντικοποίηση κάποιου πελάτη.

```
562:         srp_callback_parm.vb = SRP_VBASE_new(srp_seed_key);
563:         srp_callback_parm.user = NULL;
564:         srp_callback_parm.login = NULL;
565:
566:
567:   /*
568:  * this function parses verifier file. Format is:
569:  * string(index):base64(N):base64(g):0
570:  * string(username):base64(v):base64(salt):int(index)
571:  */
572: if ((ret =
573:              SRP_VBASE_init(srp_callback_parm.vb,
574:                             srp_verifier_file)) != SRP_NO_ERROR) {
575:             fprintf(stdout,
576:                       "Cannot initialize SRP verifier file \"%s\":ret=%d\n",
577:                       srp_verifier_file, ret);
578:
579:         }
580:         // Set the argument for the callback
581:         SSL_CTX_set_srp_cb_arg(listener->ssl_ctx, &srp_callback_parm);
582:         // The callback to be invoked for a new connection in order to check the
verifier
583:         SSL_CTX_set_srp_username_callback(listener->ssl_ctx,
ssl_srp_server_param_cb);
[...]
599: static int ssl_srp_server_param_cb(SSL *s, int *ad, void *arg)
600: {
601:
602:     srpsrvparm *p = (srpsrvparm *) arg;
603:     int ret = SSL3_AL_FATAL;
604:
605:     p->login=SSL_get_srp_username(s);
606:
607:    // If SRP_VBASE_get1_by_user returns null then user doen't exist. This is true
only if SRP_VBASE is initialized without seed_key
608:    SRP_user_pwd * user= SRP_VBASE_get1_by_user(p->vb,p->login);//look at
srp_vfy.c for description
609:    p->user=user;
610:
611:
612:     if (p->login == NULL && p->user == NULL) {
613:         p->login = SSL_get_srp_username(s);
614:         fprintf(stdout, "SRP username = \"%s\"\n", p->login);
615:         return (-1);
616:     }
617:
618:     if (p->user == NULL) {
619:         fprintf(stdout, "User %s doesn't exist\n", p->login);
620:         goto err;
621:     }
622:
623:     if (SSL_set_srp_server_param
624:         (s, p->user->N, p->user->g, p->user->s, p->user->v,
625:          p->user->info) < 0) {
626:         *ad = SSL_AD_INTERNAL_ERROR;
627:         goto err;
628:     }
```

Εικόνα 14 Υλοποίηση TLS-SRP

Αξίζει να σημειωθεί είναι ότι αν κατά την κλήση της `SRP_VBASE_new(srp_seed_key)` (γραμμή 562) η τιμή srp_seed_key δεν είναι null τότε η συνάρτηση SRP_VBASE_get1_by_user θα συμπεριφερθεί αλλιώς. Αντί να επιτρέψει null στην περίπτωση που ο χρήστης δεν υπάρχει, θα δημιουργήσει ένα seed και θα επιστρέψει ένα τυχαίο verifier προσομοιώνοντας την περίπτωση που ο χρήστης υπήρχε. Αυτό γίνεται ώστε να μην αποκαλυφθεί αν ο χρήστης υπάρχει στο σύστημα.

Έχει προστεθεί και στο αρχείο παραμετροποίησης του mosquito (mosquito.conf) η παράμετρος «srp_seed_key» στην οποία ο χρήστης προσδιορίζει το κλειδί που θα χρησιμοποιηθεί για την παραγωγή του seed.

4.2 Mqtt client

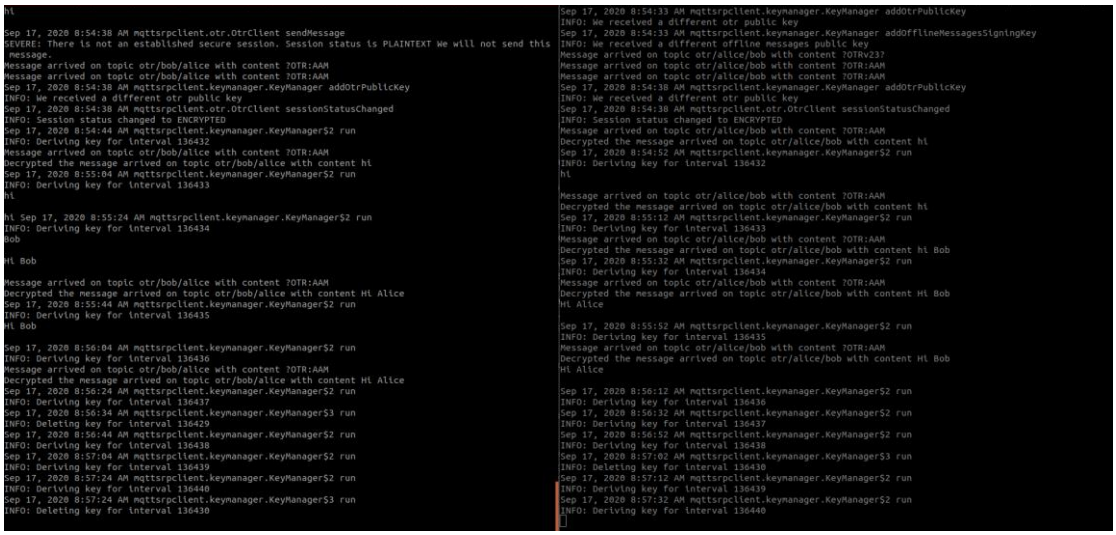
4.2.1 Διεπαφή χρήστη

Ο client που αναπτύχθηκε παρέχει κρυπτογραφημένη επικοινωνία από άκρο σε άκρο ανάμεσα σε δύο χρήστες μέσω του πρωτοκόλλου MQTT. Ο χρήστης με τον οποίο επιθυμούμε να επικοινωνήσουμε προσδιορίζεται κατά την εκκίνηση του προγράμματος μαζί με κάποιες επιπλέον πληροφορίες.

Το πρόγραμμα επιτρέπει την αποστολή μηνύματος είτε ο άλλος χρήστης είναι online είτε offline. Επιλέγει αυτόματα το σχήμα που θα χρησιμοποιήσει ανάλογα αν το άλλο μέρος είναι συνδεδεμένο ή όχι. Δεν χρειάζεται να γίνει από τον χρήστη η επιλογή.

Ο χρήστης έχει δύο επιλογές που μπορεί να χρησιμοποιήσει

- /smp: Εκτελεί το πρωτόκολλο SMP(Socialist Millionaire protocol) ώστε να εξακριβωθεί η ταυτότητα του άλλου μέλους. Ο χρήστης δίνει ως είσοδο την ερώτηση και την απάντηση στην ερώτηση η οποία αποτελεί το μυστικό.
- /s: Εγκαθιδρύει μια OTR σύνοδο. Αυτό το βήμα δεν είναι απαραίτητο, ο χρήστης μπορεί να πληκτρολογήσει το μήνυμα του και ο client θα αναλάβει να εγκαθιδρύσει την σύνοδο πριν στείλει το μήνυμα.



Εικόνα 15 Στιγμιότυπο εκτέλεσης δύο clients

4.2.2 Υλοποίηση TLS-SRP

Ο mqtt client υλοποιήθηκε στην γλώσσα προγραμματισμού Java καθώς λόγω της πληθώρας των υλοποιήσεων του πρωτοκόλλου Mqtt καθώς και των ευκολιών που παρέχει σχετικά με την διαχείριση

μνήμης. Για την υλοποίηση του Mqtt πρωτοκόλλου επιλέχθηκε ο Eclipse Paho Java Client (https://www.eclipse.org/paho/index.php?page=clients/java/index.php). Η βιβλιοθήκη παρέχει μια σύγχρονη και μια ασύγχρονη προγραμματιστική διεπαφή (api) στην οποία η εφαρμογή ειδοποιείται για την ολοκλήρωση της κάθε διαδικασίας μέσω callbacks.

Για την υλοποίηση του πρωτοκόλλου TLS-SRP χρησιμοποιήθηκε η βιβλιοθήκη Bouncy Castle. Καθώς ο Paho client δεν παρέχει κάποιο συγκεκριμένο binding το tls-srp πρέπει να υλοποιηθεί παρέχοντας ένα custom SSLSocketFactory στον client όπως φαίνεται παρακάτω.

```
TlsSrpSocketFactory tlsSrpSocketFactory = new TlsSrpSocketFac-
tory(username,password);
MqttConnectOptions connOpts = new MqttConnectOptions();
connOpts.setCleanSession(false);
connOpts.setConnectionTimeout(30);
connOpts.setSocketFactory(tlsSrpSocketFactory);
```

4.2.3 Υλοποίηση OTR

Όπως αναφέρθηκε και στον σχεδιασμό για την αποστολή μηνυμάτων όταν και τα δύο μέρη είναι διαθέσιμα χρησιμοποιείται το πρωτόκολλο OTR. Για το πρωτόκολλο αυτό χρησιμοποιήθηκε η βιβλιοθήκη otr4j (https://github.com/jitsi/otr4j). Η βιβλιοθήκη είναι υλοποιημένη σε Java και υποστηρίζει και τις τρεις εκδόσεις του πρωτοκόλλου. Στη έκδοση 3 του OTR οι πελάτες πρέπει να τοποθετούν και instance tags το οποίο προσδιορίζει τον αποστολέα και τον παραλήπτη σε κάθε μήνυμα που στέλνουν. Τα instance tags πρέπει να διατηρούνται σε κάθε εκτέλεση του προγράμματος. Η λογική είναι ότι αν πελάτης συνδεθεί με τον ίδιο λογαριασμό από δυο διαφορετικές τοποθεσίες τα instance tags θα διαφέρουν. Αυτό βοηθάει σε περιπτώσεις όπου δίκτυο δρομολογεί τα μηνύματα σε όλες τις ενεργές συνόδους για έναν λογαριασμό με αποτέλεσμα οι πελάτες να προσπαθούν να εγκαθιδρύσουν μια σύνοδο επ' άπειρον. Ένας περιορισμός της προαναφερθείσας βιβλιοθήκης είναι ότι δεν επιτρέπει την διατήρηση των instance tags. Με κάθε εκτέλεση δημιουργείται καινούριο. Επομένως αν η Alice τερματίσει την συνεδρία με τον Bob, επανεκκινήσει τον client, και εγκαθιδρύσει νέα συνεδρία ο Bob θα νομίσει πως η Alice έχει συνδεθεί από άλλη τοποθεσία καθώς έχει ήδη μια σύνοδο με την Alice η οποία είναι στην κατάσταση FINISHED. Επιπλέον όταν ο Bob θελήσει να στείλει ένα μήνυμα θα εγκαθιδρύσει νέα συνεδρία. Αυτό γίνεται καθώς η βιβλιοθήκη θεωρεί ως κύρια σύνοδο (master session) την πρώτη σύνοδο που δημιουργείται και αν υπάρχουν άλλες σύνοδοι (slave sessions) δεν ξέρει ποια να επιλέξει για να στείλει. Η σύνοδος μπορεί να επιλεγεί με την `SessionImpl#setOutgoingInstance`. Αν δεν επιτρέψουμε πολλαπλές συνόδους μπορούμε να διαλέγουμε την τελευταία συνεδρία που βρίσκεται στην κατάσταση ENCRYPTED. Αν επιτρέπονται πολλαπλές σύνοδοι τότε πρέπει ο χρήστης να επιλέξει την σωστή σύνοδο. Η υλοποίηση μας δεν επιτρέπει σύνδεση με τον ίδιο λογαριασμό από πολλές συσκευές.

4.2.4 Υλοποίηση fs-pke σχήματος

Για το forward secure σχήμα δημοσίου κλειδιού χρησιμοποιήθηκε η υλοποίηση που παρέχουν οι δημιουργοί του σχήματος (https://github.com/imichaelmiers/libforwardsec). Η συγκεκριμένη υλοποίηση υποστηρίζει οποιονδήποτε αριθμό ετικετών αλλά περιορίζει το μέγεθος των δεδομένων προς κρυπτογράφηση στα 32 bytes. Ωστόσο αυτό δεν αποτελεί πρόβλημα καθώς το μόνο που απαιτείται να κρυπτογραφηθεί είναι ένα συμμετρικό κλειδί.

Ένα σημείο που αξίζει αναφοράς είναι το πως διαγράφονται τα παλαιότερα κλειδιά. Στην περίπτωση της εφαρμογής μας είναι πιθανό ένας χρήστης να συνδέεται στο σύστημα ανά πολύ αραιά

διαστήματα. Η εφαρμογή θα πρέπει να διαγράψει τα κλειδιά τα οποία έχουν υπερβεί την χρονική τους διάρκεια. Αναλόγως την διάρκεια των κλειδιών ο αριθμός των κλειδιών προς διαγραφή μπορεί να είναι μεγάλος. Για παράδειγμα αν τα κλειδιά διαρκούν ένα λεπτό απαιτούνται 1440 κλειδιά για μια ημέρα. Κατά την εκτέλεση του προγράμματος παρατηρήθηκε ότι απαιτούταν αρκετός χρόνος για την διαγραφή των κλειδιών καθώς η βιβλιοθήκη απαιτεί την δημιουργία των κλειδιών που αντιστοιχούν στα παιδιά πριν επιτραπεί η διαγραφή του γονέα. Αυτό είναι απαραίτητο καθώς αν διαγραφεί ο γονέας δεν μπορούμε να παράγουμε τα υπόλοιπα κλειδιά. Στην περίπτωση όμως που δεν θα χρησιμοποιήσουμε κάποια κλειδιά δεν υπάρχει λόγος να τα παράγουμε. Για τον λόγο αυτό στην βιβλιοθήκη προστέθηκε μια επιπλέον συνάρτηση η οποία διαγράφει το κλειδί που θα ζητηθεί χωρίς να είναι υποχρεωτική η παραγωγή των κλειδιών των απογόνων. Η συνάρτηση ονομάζεται “fastEraseIntervalKeys”. Ως είσοδο δέχεται το δημόσιο και ιδιωτικό κλειδί pk, sk αντίστοιχα, το διάστημα i του οποίου το κλειδί πρέπει να διαγραφεί καθώς και το διάστημα “lastIntervalToPreserve” το οποίο δείχνει το κλειδί το οποίο πρέπει να διατηρηθεί γιατί θα χρησιμοποιηθεί αργότερα. Ο αλγόριθμος δημιουργεί το μονοπάτι από την ρίζα έως το κλειδί που θέλουμε να διατηρηθεί και ελέγχει αν το κλειδί προς διαγραφή βρίσκεται μέσα στο μονοπάτι. Αν βρίσκεται στο μονοπάτι παράγουμε τα κλειδιά των παιδιών και έπειτα το διαγράφουμε. Αν δεν βρίσκεται στο μονοπάτι μπορούμε να το διαγράψουμε αμέσως καθώς δεν θα χρειαστούμε τους απογόνους του. Συνεπώς για κάθε διαγραφή κερδίζουμε τον χρόνο της παραγωγής των κλειδιών τον απογόνων στην περίπτωση που αυτή δεν χρειάζεται.

```
void GMPfse::fastEraseIntervalKeys(const GMPfsePublicKey & pk,
GMPfsePrivateKey &sk,unsigned int i,unsigned int
lastIntervalToPreserve){
    libforwardsec_DBG(cout << "Fast erasing key for interval " << i <<
endl;)
    unsigned int level=0;
    vector<ZR> path = indexToPath(lastIntervalToPreserve,depth);
    unsigned int pathsize = path.size();
    for (level =0 ; level < pathsize ; level++){

        if(pathToIndex(path,depth) == i && sk.needsChildKeys(i) ){
        // The requested node for deletion intersects the path so we
need to generate child keys
            GMPfse::prepareIntervalAfter(pk,sk,i);
            break;
        }
        path.resize(path.size()-1);
     }
     sk.fastErase(i);
}
```

Εικόνα 16 Η συνάρτηση fastEraseIntervalKeys

Για παράδειγμα έστω ότι έχουμε το δένδρο της εικόνας 13 και θέλουμε να διαγράψουμε το κλειδί για το διάστημα 2 αλλά να διατηρήσουμε την ικανότητα να παράγουμε τα κλειδιά για το διάστημα 5 και μετά. Σε αυτήν την περίπτωση μπορούμε να διαγράψουμε το κλειδί για το διάστημα 2. Αντίθετα αν θέλαμε να διαγράψουμε το κλειδί για το διάστημα 1 τότε πρέπει πρώτα να παράγουμε τα κλειδιά για τα διαστήματα 2, 5.

4.2.5 Μορφή μηνυμάτων

Τα μηνύματα που αποστέλλονται σε json μορφή για μεγαλύτερη ευκολία στην διαχείριση των δεδομένων και την ευκολότερη αποσφαλμάτωση όπως φαίνεται στην εικόνα 15. Θα μπορούσε να χρησιμοποιηθεί και μια δυαδική αναπαράσταση καθώς το MQTT δεν θέτει κάποιον περιορισμό στο είδος των δεδομένων.

```
{
  "signature":
"MEQCICWZsEgOtu7hxHkWopRvB/NAOxlw8AfhAnesYQY48pu1AiB+lI4WKZPIZ70O2dqN
SSiTO6ic11w/3YUqm2MDyeGplQ==",
  "message": {
    "from": "alice",
    "to": "bob",
    "text": "test message",
    "counter": 3
  }
}
```

Εικόνα 17 Το περιεχόμενο ενός μη κρυπτογραφημένου μηνύματος που αποστέλλεται σε έναν μη συνδεδεμένο χρήστη

```
{
  "public_keys": {

    "longTermKey":
"MIIBuDCCASwGByqGSM44BAEwggEfAoGBAP1/U4EddRIpUt9KnC7s5Of2EbdSPO9EAMMe
P4C2USZpRV1AIlH7WT2NWPq/xfW6MPbLm1Vs14E7gB00b/JmYLdrmVClpJ+f6AR7ECLCT
7up1/63xhv4O1fnxqimFQ8E+4P208UewwI1VBNaFpEy9nXzrith1yrv8iIDGZ3RSAHHAh
UAl2BQjxUjC8yykrmCouuEC/BYHPUCgYEA9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+Vd
MCz0HgmdRWVeOutRZT+ZxBxCBgLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuz
pnWRbqN/C/ohNWLx+2J6ASQ7zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSoDg
YUAAoGBAPZBBrTPWjRiV4tizl/WtFcsuxkZwI+Ga9Hfpn0YXu/wr1KI3BPnhX4To5ndLU
V+4QfCuexBiDQkMfUyVyJzRX4h1ST37ZOdVsUkRQ81GxOwTznbByBGEf0bxHoMfrdeeDk
n35tq37lOAWYi3VEB397LFNQqeCVcW+Cd4ZETk1nr",

    "fs_pkeKey":
"IQAAAAAAAAACLO3nOJ0PMFjU+pXvglQr2BPR80+CoflkwS9Ww4N1FzRBAAAAAAAAAAOH
Qjk4qXQEeb+vEOckCCIEAk6OeIR/EiL1kpVwF7Dva0Q+RAYTFbfi/hqr1EbNzZERKjdbP
DofUNFAiom2hxtOIQAAAAAAAADOf40T0aFg6Q6G0Y9IuZhh6/e4TV86stsLBz39UCGjj
dBAAAAAAAAAAKF9rA+SFngeKWtGpqYsBM8k4480lfk04xfzq1nZPV5PSexeZwpSLS8XfX
MKTSlJ+ANrqP4L3JQZ2/NscU7ToaHHwAAAEEAAAAAAAAAA6v65C42ms29bhj26KWvhRk9
2AS/d/6qhgrHhy5422HJidgbAbWj6g5rvtGSu8V6aHULcWswukvrCWgNz5ag3lIhAAAAA
AAAAAJ5JbnpUMTfohcc3AhowF8d1box1XVbGz8401uSTRXvEkEAAAAAAAAAA6TMGGBa38
xgwnPSsCrccK1TbALY+h3T1kFrL1D4mpEpE6cqOV56B4YEUf0PB4g4f2yQG/o7FoPRU0J
YkKPdnZMfAAAAAAAAACEAAAAAAAAAA7TGymV2BlvNNzzpoI7bzojW4XRE9PZ3KvWcNOPO
T+HTIQAAAAAAAADa+XNdJ+z/xcuOe6T9qWWfuaxOv1T90zI6QfaSCCWd9ohAAAAAAAAA
AKnfdEmMmis81W0YgvHIDCCaDrh+yZzhhARqNNqdRUeKiEAAAAAAAAAAp4vfvgcwrgUP3
sk4Dqm5D/EhN3xyKtvHUFfj/xlja8tIQAAAAAAAADlIIBOrf49Xi0l8kGQwDC+NTlGff
CtbRtfInExzCZA9shAAAAAAAAANidWYEMlswm/gTwaFLyMAoB1qcEDqVbyjkuj7L2dE6
tCEAAAAAAAAAAnXELGx1PJXQMYvsp+PpoNxQ2MiDNmWtHIOIbO2Ce6gJIQAAAAAAAACb
soXvXwAn4vvweB78BXdqfW1bvLW6BWOp1ZsvODPc2ghAAAAAAAAAI2vevD8IPS8YRO/W
/rxYa8g8kt6yXPU0Z2LdrGorZtpCEAAAAAAAAA6PWzqnDRWvZkQRqmO+r7A4MQ7QSdFu
Je3ENusrNgUO/IQAAAAAAAAClLqXhl9cNQb7DfpzMUoABUj84i0N+WUWfzSN+pO/oysh
AAAAAAAAAMiLXliSprtKcZCCWQBWNGDbqEOATOAo30WSihNFy68+SEAAAAAAAAAl+
                                [.....]
AAAAAEEAAAAAAAAAAoX2sD5IWeB4pa0ampiwEzyTjjzSV+TTjF/OrWdk9Xk9J7F5nClI
tLxd9cwpNKUn4A2uo/gvclBnb82xxTtOhodBAAAAAAAAANyBXC6z0kLGI4Xq3fxYDck8
hTr0LqLe7UvarwhmwsW9UtpapVWoZIVmtI+6AwMRYJksMOQ/sYMSOZc0/bbiMPP"
  },
  "signature":
"MCwCFBOiJA94n58Kk1ZHdG1oGx8epyEOAhR7yej1pRbWLrkqFMYti3VVhID0Cg=="
}
```


Εικόνα 18 Το μήνυμα περιέχει τα δημόσια κλειδιά του χρήστη και αποστέλλεται σε όλους τους χρήστες

5 Τεχνικό εγχειρίδιο

Η εφαρμογή αναπτύχθηκε σε περιβάλλον Ubuntu 18.0.4. Για την χρήση της εφαρμογής απαιτείται η εγκατάσταση των παρακάτω πακέτων.

- build-essential. Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "*sudo apt-get install build-essential*"
- git. Θα χρησιμοποιηθεί για την εγκατάσταση των επιπλέον βιβλιοθηκών. Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "sudo apt-get install git"
- libc-ares-dev. Απαιτείται για την μεταγλώττιση του mosquito broker. Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "sudo apt-get install libc-ares-dev"
- uuid-dev. Απαιτείται για την μεταγλώττιση του mosquito broker. Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "*sudo* apt-*get install uuid-dev*"
- Cmake. Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "sudo apt-get install cmake"
- Gmp Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "sudo apt-get install libgmp3-dev"
- JDK Το συγκεκριμένο πακέτο μπορεί να εγκατασταθεί με την εντολή "sudo apt-get install default-jdk"

5.1 Δημιουργία αρχείου κωδικών

Για την δημιουργία του αρχείου κωδικών πρέπει να εκτελεστούν οι παρακάτω εντολές.

- *touch password.srpv*
- openssl srp -srpvfile password.srpv -add user_name

Για παράδειγμα για την εισαγωγή ενός χρήστη με το όνομα alice εκτελούμε
openssl srp -srpvfile password.srpv -add alice

5.2 Εγκατάσταση και εκτέλεση mosquito broker

Για την εύκολη εγκατάσταση του Mosquitto και την προσθήκη των αλλαγών για την υποστήριξη του TLS-SRP έχει δημιουργηθεί το script build-srp-mosquitto.sh. Το συγκεκριμένο script κάνει clone την openssl 1.1.0d και το mosquito 1.4.11 και τέλος τα κάνει compile. Οι αλλαγές που έχουν γίνει για το mosquitto εφαρμόζονται ως ένα git patch(mosquitto_srp.patch).

Το παραπάνω script αφού δημιουργήσει το εκτελέσιμο για τον mosquitto broker δεν το εγκαθιστά στο σύστημα. Το script run-srp-mosquitto.sh εκτελεί τον mosquitto broker χρησιμοποιώντας το configuration που βρίσκεται στον φάκελο conf.d/mosquitto.conf.

Πριν την εκτέλεση είναι αναγκαίο να προσδιοριστούν οι παρακάτω τιμές στο αρχείο mosquitto.conf.

- srp_verifier_file: Προσδιορίζει το αρχείο κωδικών το οποίο θα χρησιμοποιηθεί κατά την αυθεντικοποίηση με το πρωτόκολλο SRP
- srp_seed_key: το κλειδί που θα χρησιμοποιηθεί για την παραγωγή του seed.
- use_identity_as_username: Αν είναι true θα χρησιμοποιηθεί ως username η ταυτότητα που δόθηκε από τον πελάτη κατά την εκτέλεση του SRP για τον έλεγχο πρόσβασης. Σε άλλη περίπτωση θα πραγματοποιηθεί έλεγχος με το όνομα χρήστη και τον κωδικό που προσδιορίζεται στο πακέτο CONNECT.
- acl_file: Το αρχείο με τους κανόνες ελέγχου πρόσβασης.

5.3 Εγκατάσταση και εκτέλεση client

Αρχικά πρέπει να εκτελεστεί το script build-libforwardsec.sh το οποίο θα κάνει compile την βιβλιοθήκη libforwardsec και τις εξαρτήσεις αυτής.

Έπειτα πρέπει να εκτελεστεί το script build-mqttsrp-client.sh. Το script υποθέτει πως είναι εγκατεστημένη η έκδοση openjdk-11. Αν έχετε εγκαταστήσει κάποιο άλλο τροποποιήστε την μεταβλητή JAVA_HEADERS_DIR στην αρχή του script.

Στον φάκελο mqttsrpclient/output/ έχει δημιουργηθεί το αρχείο mqttsrpclient-1.0.jar το οποίο είναι και το εκτελέσιμο της εφαρμογής. Στον φάκελο έχουν αντιγραφεί και οι απαραίτητες βιβλιοθήκες.

Αφού μεταβούμε στον φάκελο mqttsrpclient/output/, πρέπει να ρυθμιστεί η μεταβλητή περιβάλλοντος LD_LIBRARY_PATH ώστε ο loader να μπορεί να βρει τις απαραίτητες βιβλιοθήκες. Αυτό μπορεί να γίνει εκτελώντας την εντολή *export LD_LIBRARY_PATH=$(pwd)*

Για να εκτελέσουμε την εφαρμογή ως ο χρήστης alice με παραλήπτη τον χρήστη bob εκτελούμε «`java -Djava.library.path=$(pwd) -jar mqttsrpclient-1.0.jar -h ssl://127.0.0.1:8883 -pqos 2 -ptopic otr/alice/bob -sqos 2 -stopic otr/bob/alice -u alice -id alice`»

Για να εκτελέσουμε την εφαρμογή ως ο χρήστης bob με παραλήπτη των μηνυμάτων την alice «`java -Djava.library.path=$(pwd) -jar mqttsrpclient-1.0.jar -h ssl://127.0.0.1:8883 -pqos 2 -ptopic otr/bob/alice -sqos 2 -stopic otr/alice/bob -u alice -id bob`»

6 Συμπεράσματα και μελλοντικές κατευθύνσεις

Η παρούσα μελέτη ως στόχο είχε να εξεταστεί ένα σχήμα αμοιβαίας αυθεντικοποίησης πελάτη και broker το οποίο να μην βασίζεται στην χρήση της υποδομής δημοσίου κλειδιού. Στις περισσότερες περιπτώσεις οι χρήστες αυθεντικοποιούνται με την χρήση κωδικών. Δόθηκε ιδιαίτερη έμφαση σε τεχνικές στις οποίες οι χρήστες χρειάζεται να θυμούνται μόνο τον κωδικό τους. Αναλύθηκαν οι συχνά χρησιμοποιούμενες τεχνικές καθώς και τα σχήματα PAKE τα οποία επιτρέπουν αμοιβαία αυθεντικοποίηση με την χρήση κωδικών. Ο χρήστης συμμετέχει στο πρωτόκολλο με τον κωδικό του και ο broker διατηρεί έναν verifier ο οποίος αποτελεί έναν μετασχηματισμό του κωδικού. Το πρωτόκολλο SRP μπορεί να χρησιμοποιηθεί ως τρόπος αυθεντικοποίησης στο TLS επιτρέποντας την δημιουργία ενός ασφαλούς καναλιού μεταξύ πελάτη και broker.

Επιπλέον, διερευνήθηκαν σχήματα τα οποία εγγυώνται την εμπιστευτικότητα και την ακεραιότητα των δεδομένων που μεταφέρονται από το MQTT πρωτόκολλο. Μπορούμε να διακρίνουμε δυο περιπτώσεις επικοινωνίας. Στην πρώτη περίπτωση και οι δύο χρήστες είναι συνδεδεμένοι στον broker επομένως μπορούν να ανταλλάξουν μηνύματα ταυτόχρονα. Στην άλλη περίπτωση ένα από τα δύο μέρη δεν είναι συνδεδεμένο. Το κύριο πρόβλημα αυτής της περίπτωσης είναι ότι δεν μπορούμε να παρέχουμε εύκολα forward secrecy καθώς δεν μπορούν να χρησιμοποιηθούν ratchet αλγόριθμοι.

Το σχήμα που επιλέχθηκε λύνει το παραπάνω πρόβλημα εφαρμόζοντας διαφορετικό κρυπτογραφικό αλγόριθμο ανάλογα την κατάσταση του παραλήπτη. Αν ο παραλήπτης είναι συνδεδεμένος εφαρμόζεται το πρωτόκολλο OTR. Αν είναι αποσυνδεδεμένος χρησιμοποιείται το forward secure σχήμα δημοσίου κλειδιού. Στο συγκεκριμένο σχήμα κάθε χρήστης διαθέτει ένα ζεύγος κλειδιών(ιδιωτικό/δημόσιο) το οποίο χρησιμοποιεί για κρυπτογράφηση και αποκρυπτογράφηση. Ο κάθε χρήστης ανά συγκεκριμένα χρονικά διαστήματα ανανεώνει το ιδιωτικό κλειδί του ώστε να μην είναι δυνατή η αποκρυπτογράφηση των προηγούμενων μηνυμάτων. Επιπλέον μπορεί να ανακαλέσει την ικανότητα αποκρυπτογράφησης για συγκεκριμένα μηνύματα χωρίς να χρειάζεται να χαθεί η ικανότητα αποκρυπτογράφησης των υπόλοιπων μηνυμάτων που λάβαμε σε αυτό το διάστημα. Το κύριο

πλεονέκτημα του σχήματος είναι ότι παρέχει forward secrecy στην περίπτωση offline μηνυμάτων χωρίς να απαιτείται η ύπαρξη ενός εξυπηρετητή για την διανομή pre-keys.

Τα κλειδιά για την λειτουργία του σχήματος διανέμονται μέσω του MQTT. Ωστόσο η εξακρίβωση της αυθεντικότητας των κλειδιών εναπόκειται στον χρήστη. Ο χρήστης πρέπει να ελέγξει τα fingerprints των κλειδιών ή να εκτελέσει το πρωτόκολλο SMP για να εξακριβώσει την αυθεντικότητα των κλειδιών.

Ο συνδυασμός των παραπάνω μεθόδων επιτρέπει την εύκολη αυθεντικοποίηση από την μεριά του πελάτη καθώς χρειάζεται μόνο η γνώση του κωδικού. Παρόμοια ο broker χρειάζεται μόνο τον verifier. Δεν απαιτείται η υλοποίηση υποδομής δημοσίου κλειδιού. Το forward secure σχήμα δημοσίου κλειδιού επιτρέπει να κρυπτογραφούνται μηνύματα όταν οι χρήστες είναι αποσυνδεδεμένοι χωρίς να είναι αναγκαία η ύπαρξη ενός εξυπηρετητή για την διανομή pre-keys όπως στον Signal. Αξίζει να σημειωθεί ότι ο εξυπηρετητής πρέπει να είναι προσβάσιμος κάθε φορά που κάποιος χρήστης επιθυμεί να στείλει ένα μήνυμα. Επιπλέον το αποθηκευτικό κόστος μπορεί να είναι αρκετά μεγάλο.

Παρόλα αυτά στο σχήμα μας η αποκρυπτογράφηση είναι σχετικά δαπανηρή κάτι που μπορεί να αποτελέσει πρόβλημα για τις κινητές συσκευές ή άλλες συσκευές με περιορισμένους πόρους. Επιπλέον δεν μπορούμε να κρυπτογραφήσουμε ένα offline μήνυμα για κάποιο topic χωρίς να ξέρουμε ποιοι είναι εγγεγραμμένοι σε αυτό το topic καθώς θα πρέπει να χρησιμοποιήσουμε τα κλειδιά τους.

Το σχήμα μπορεί να βελτιωθεί περαιτέρω. Το κόστος αποκρυπτογράφησης μπορεί να μειωθεί αναθέτοντας τις δαπανηρές λειτουργίες της αποκρυπτογράφησης σε έναν εξυπηρετητή στο cloud. Υπάρχουν τεχνικές [56] [57] που επιτρέπουν την αποκρυπτογράφηση ABE αλγορίθμων χωρίς να χρειάζεται ο χρήστης να γνωστοποιήσει το ιδιωτικό κλειδί του.

Ακόμα μια χρήσιμη προσθήκη θα ήταν η χρήση επιπλέον ετικετών εκτός του Guid όπως το θέμα μιας συζήτησης ή το όνομα του παραλήπτη. Αυτό θα επέτρεπε να διαγράψουμε μηνύματα που έχουν ένα συγκεκριμένο περιεχόμενο ή παραλήπτη.

Επιπλέον θα ήταν ενδιαφέρον να εξεταστεί η εφαρμογή ABE αλγορίθμων [11] [12]που βασίζονται σε πολλαπλές έμπιστες αρχές(threshold multi authority ABE) για την κρυπτογράφηση μηνυμάτων που αποστέλλονται σε πολλούς χρήστες. Η χρήση ενός ABE σχήματος θα επέτρεπε την αποστολή μηνυμάτων χωρίς την ανάγκη ανταλλαγής κλειδιών μεταξύ των συμμετεχόντων. Επιπλέον η χρήση πολλών αρχών θα καθιστούσε το σύστημα πιο ανθεκτικό στην περίπτωση επίθεσης σε αντίθεση αν χρησιμοποιούταν μια κεντρική αρχή.

7 Αναφορές

[1] L. Lundgren, «Light Weight Protocol Critical Implications,» σε *DEF CON 24*, 2016.

[2] R. G. Andrew Banks, «MQTT Version 3.1.1,» Οκτώβριος 2014. [Ηλεκτρονικό]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. [Πρόσβαση Αύγουστος 2019].

[3] X. Wang, J. Zhang, E. Schooler και M. Ion, «Performance evaluation of Attribute-Based Encryption: Toward data privacy in the IoT,» 2014.

[4] P. Pal, G. Lauer, J. Khoury, N. Hoff και J. Loyall, «P3S: A Privacy Preserving Publish-Subscribe Middleware,» 2012.

[5] M. Ion, G. Russello και B. Crispo, «Supporting Publication and Subscription Confidentiality in Pub/Sub Networks,» 2010.

[6] M. A. Tariq, Non-functional requirements in publish/subscribe systems, 2013.

[7] M. Singh, M. A. Rajan, V. L. Shivraj και P. Balamuralidhar, «Secure MQTT for Internet of Things (IoT),» σε *Fifth International Conference on Communication Systems and Network*

Technologies, 2015.

[8] A. Sahai και B. Waters, «Fuzzy Identity-Based Encryption,» σε *Advances in Cryptology -- EUROCRYPT 2005*, Berlin, 2005.

[9] H. Lee, J. Lim και T. Kwon, «MQTLS: Toward Secure MQTT Communication with an Untrusted Broker,» 2019.

[10] S. Shin, K. Kobara, C.-C. Chuang και W. Huang, «A security framework for MQTT,» σε *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016.

[11] H. Lin, Z. Cao, X. Liang και J. Shao, «Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority,» σε *Progress in Cryptology - INDOCRYPT 2008*, Berlin, 2008.

[12] M. Chase, «Multi-authority Attribute Based Encryption,» σε *Theory of Cryptography*, Berlin, 2007.

[13] H. T. P. Eronen, «Pre-Shared Key Ciphersuites for Transport Layer Security (TLS), RFC 4279,» 2005.

[14] D. E. D. a. G. M. Sacco, «Timestamps in Key Distribution Protocols,» 1981.

[15] J. Schmidt, *Requirements for Password-Authenticated Key Agreement (PAKE) Schemes, RFC 8125,* 2017.

[16] S. M. B. a. M. Merritt, «Encrypted Key Exchange: Password-Based Protocols SecureAgainst Dictionary Attacks,» σε *Proceedings of the I.E.E.E. Symposium on Research in Securityand Privacy*, 1992.

[17] D. P. Jablon, «Strong Password-Only Authenticated Key Exchange,» 1996.

[18] D. P. Jablon, «Extended Password Key Exchange Protocols Immune to Dictionary Attack,» σε *Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1997.

[19] F. &. R. P. Hao, «J-PAKE: Authenticated Key Exchange Without PKI,» *Transactions on Computational Science,* pp. 192-206, 2010.

[20] T. Kwon, «Authentication and Key Agreement Via Memorable Passwords,» 2001.

[21] G. T. a. M. W. M. Steiner, «Refinement and Extension of Encrypted KeyExchange,» σε *Operating Systems Review*, 1995, pp. 22-30.

[22] S. Pohlig και M. Hellman, «An improved algorithm for computing logarithms overGF(p)and its cryptographic significance (Corresp.),» *IEEE Transactions on Information Theory,* τόμ. 24, pp. 106-110, 1978.

[23] S. F. S. Feng Hao, «The SPEKE Protocol Revisited,» Newcastle University, UK.

[24] T. Wu, «The Secure Remote Password Protocol,» Stanford University, 1997.

[25] T. Wu, «SRP-6: Improvements and Refiments to the Secure Remote Password Protocol,» Arcot Systems, 2002.

[26] H. K. J. X. Stanislaw Jarecki, «OPAQUE: An Asymmetric PAKE ProtocolSecure Against Pre-Computation Attacks,» σε *Eurocrypt*, 2018.

[27] D. J. a. A. M. S. Blake-Wilson, «Key agreement protocols and their security analysis,» σε *Proceedings of the 6th IMA InternationalConference on Cryptography and Coding*, London, Springer, 1997, pp. 30-45.

[28] C. Hlauschek, M. Gruber, F. Fankhauser και C. Schanes, «Prying Open Pandora's Box: KCI Attacks Against TLS,» σε *Proceedings of the 9th USENIX Conference on Offensive Technologies*, Berkeley, 2015.

[29] H. Krawczyk, «HMQV: A High-performance Secure Diffie-hellman Protocol,» σε *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, Berlin, 2005.

[30] M. Abdalla και D. Pointcheval, «Simple Password-Based Encrypted Key Exchange Protocols,» σε *Topics in Cryptology -- CT-RSA 2005*, Berlin, 2005.

[31] O. Goldreich και Y. Lindell, «Session-Key Generation Using Human Passwords Only,» *Journal of Cryptology,* τόμ. 19, pp. 241-340, 01 7 2006.

[32] J. Becerra, D. Ostrev και M. Škrobot, «Forward Secrecy of SPAKE2,» σε *Provable Security*, Cham, 2018.

[33] V. Moscaritolo, G. Belvin και P. Zimmermann, «Silent Circle Instant Messaging Protocol Protocol Specification,» 2012.

[34] S. R. Verschoor and T. Lange, (In-)Secure messaging with the Silent Circle instant messaging protocol, IACR, 2016.

[35] I. Goldberg, D. Goulet, J. Appelbaum και J. van Bergen, «Off-the-Record Messaging Protocol version 3,» 2012. [Ηλεκτρονικό]. Available: https://otr.cypherpunks.ca/Protocol-v3-4.1.1.html.

[36] C. Alexander και I. Goldberg, «Improved User Authentication in Off-the-Record Messaging,» σε *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, New York, NY, USA, 2007.

[37] T. P. a. M. Marlinspike, «The X3DH Key Agreement Protocol,» 2016.

[38] T. P. a. M. Marlinspike, «The Double Ratchet Algorithm,» 2016.

[39] A. Shamir, «Identity-Based Cryptosystems and Signature Schemes,» σε *Advances in Cryptology*, Berlin, 1985.

[40] D. Boneh και M. K. Franklin, «Identity-Based Encryption from the Weil Pairing,» σε *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, London, 2001.

[41] E. Fujisaki και T. Okamoto, «Secure Integration of Asymmetric and Symmetric Encryption Schemes,» σε *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, Berlin, 1999.

[42] R. Canetti, S. Halevi και J. Katz, «A Forward-secure Public-key Encryption Scheme,» σε *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, Berlin, 2003.

[43] D. Boneh, X. Boyen και E.-J. Goh, «Hierarchical Identity Based Encryption with Constant Size Ciphertext,» σε *Advances in Cryptology -- EUROCRYPT 2005*, Berlin, 2005.

[44] R. Ostrovsky, A. Sahai και B. Waters, «Attribute-based Encryption with Non-monotonic Access Structures,» σε *Proceedings of the 14th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2007.

[45] A. Beimel, «Secure Schemes for Secret Sharing and Key Distribution,» 1996.

[46] M. D. Green και I. Miers, «Forward Secure Asynchronous Messaging from Puncturable Encryption,» σε *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, Washington, 2015.

[47] L. Zhang, «Building Facebook Messenger,» Αύγουστος 2011. [Ηλεκτρονικό]. Available: https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920. [Πρόσβαση Αύγουστος 2019].

[48] Duncan Wood - Sans Institute, PKI, The What, The Why, and The How, 2002.

[49] C. Ellison και B. Schneier, «Ten risks of PKI: What you're not being told about Public Key Infrastructure,» *Computer Security Journal,* 2000.

[50] J. A. Helou και S. Tilley, «Multilingual web sites: Internationalized Domain Name homograph attacks,» σε *Timisoara, Romania 12th IEEE International Symposium on Web Systems Evolution (WSE)*, Timisoara, Romania, 2010.

[51] «IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques,» *IEEE Std 1363.2-2008,* pp. 1-140, 1 2009.

[52] Apple Inc, «iOS Security,» May 2019. [Ηλεκτρονικό]. Available: https://www.apple.com/tr/business/docs/site/iOS_Security_Guide.pdf. [Πρόσβαση June 2020].

[53] R. Fillion, «Developers: How we use SRP, and you can too,» 1Password, 14 February 2018. [Ηλεκτρονικό]. Available: https://blog.1password.com/developers-how-we-use-srp-and-you-can-too/. [Πρόσβαση 27 June 2020].

[54] D. Taylor, T. Perrin, T. Wu και N. Mavrogiannopoulos, *Using the Secure Remote Password (SRP) Protocol for TLS Authentication,* RFC 5054, 2007.

[55] N. Kobeissi, K. Bhargavan και B. Blanchet, «Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach,» σε *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017.

[56] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache και M. Scott, «Secure Delegation of Elliptic-Curve Pairing,» σε *Smart Card Research and Advanced Application*, Berlin, 2010.

[57] M. Green, S. Hohenberger και B. Waters, «Outsourcing the decryption of ABE ciphertexts,» 2011.