ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

ΑΞΙΟΛΟΓΗΣΗ ΦΟΡΤΟΥ ΚΡΥΠΤΟΓΡΑΦΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ ΣΕ ΕΞΥΠΗΡΕΤΗΤΕΣ ΔΙΑΔΙΚΤΥΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΤΑΥΡΟΥΛΑ ΚΑΡΑΓΙΑΝΝΗ
Επιβλέπων: Χ. Ξενάκης, Επίκουρος Καθηγητής
1/10/2014

ΜΠΔ: ΑΣΦΑΛΕΙΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πίνακας περιεχομένων

1. Περίληψη

Μια συνάρτηση παραγωγής κλειδιών βασισμένη στους κωδικούς ασφαλείας – ‘password – based KDF’ –συνάρτηση η οποία παράγει κλειδιά βάσει των κωδικών πρόσβασης – είναι συχνά υλοποιημένη σε εφαρμογές που απαιτούν εγγραφή και αποθήκευση χρηστών και κωδικών. Όπως και στις υπόλοιπες εφαρμογές και συναρτήσεις που η αυθεντικοποίηση χρήστη βασίζεται στην εισαγωγή συνθηματικού, οι KDFs υπόκεινται σε βασικές επιθέσεις αναζήτησης κωδικών πρόσβασης, όπως τιε επιθέσεις λεξικού – ‘dictionary attacks’ .

Οι συγκεκριμένες συναρτήσεις χρησιμοποιούν τις μεθόδους ενίσχυσης συνθηματικού με τυχαίο πρόθεμα – ‘salt / salting’ καθώς και την μέτρησης των κύκλων επανάληψης. Οι συγκεκριμένες μέθοδοι έχουν το αποτέλεσμα της αύξησης του φόρτου εργασίας και πόρων για μια επιτυχή επίθεση dictionary attack.
Οι προαναφερθείσες τεχνικές έχουν υιοθετηθεί ευρέως και από τα διάφορα πρότυπα του κλάδου, όπως PKCS και IETF .

Στην παρούσα μελέτη, μελετάται την πιο συχνή χρησιμοποιούμενη κατασκευή – υλοποίηση σε απλή σελίδα εγγραφής και σύνδεσης χρήστη και αποθήκευση κωδικού πρόσβαση με τη συνάρτηση BCRYPT. Επιπλέον, διερευνάται η καθυστέρηση που εισάγεται κατά την χρήση διάφορων τιμών της παραμέτρου cost –προσθήκη επαναλήψεων κατά την εκτέλεση της συνάρτησης Bcrypt- σε σύγκριση με διάφορα μήκη κωδικών πρόσβασης καθώς και σε αναλογία με τον αντίστοιχο χρόνο που απαιτείται κατά την εκτέλεση αντίστοιχης εργασίας με την εφαρμογή του αλγορίθμου SHA1.

2. Εισαγωγή

Όταν τα μέρη μοιράζονται ένα μυστικό συμμετρικό κλειδί, είναι συχνή η περίπτωση να χρειαστούν πρόσθετα κλειδιά. Μπορεί να χρειάζονται ξεχωριστά κλειδιά για διαφορετικούς σκοπούς κρυπτογράφησης - για παράδειγμα, μπορεί να απαιτείται ένα κλειδί για έναν αλγόριθμο κρυπτογράφησης, ενώ ένα άλλο κλειδί προορίζεται για χρήση από έναν αλγόριθμο προστασίας της ακεραιότητας, όπως ο κωδικός επαλήθευσης ταυτότητας μηνύματος. Σε άλλες περιπτώσεις, τα ξεχωριστά κλειδιά που απαιτούνται από πολλαπλές οντότητες μπορούν να παραχθούν από ένα έμπιστο μέρος από ένα μόνο κύριο κλειδί. Οι συναρτήσεις παραγωγής κλειδιών / παραγωγικές συναρτήσεις κλειδιών – 'Key Derivation Functions' χρησιμοποιούνται για την παραγωγή αυτών των πλήκτρων.

3. Εύρος και Σκοπός

Το παρών έγγραφο αποτελεί μία μελέτη πάνω σε μια από τις πιο διαδεδομένες συνάρτησης παραγωγής κλειδιών από κωδικούς ασφαλείας –'KDF', Bcrypt. Επιπλέον, στα πλαίσια της παρούσας μελέτης παρουσιάζονται και σενάρια εκμετάλλευσης των δυνατοτήτων της συγκεκριμένης συνάρτησης καθώς σύγκριση αυτών με αντίστοιχους αλγορίθμους (SHA1).

4. Ορισμοί

Κλειδί Κρυπτογράφησης	Μια δυαδική συμβολοσειρά που χρησιμοποιείται ως μυστική παράμετρος από έναν κρυπτογραφικό αλγόριθμο. Ένα κρυπτογραφικό κλειδί πρέπει να είναι είτε μια πραγματικά τυχαία δυαδική συμβολοσειρά μήκους που καθορίζεται από τον αλγόριθμο κρυπτογράφησης ή ψευδοτυχαία δυαδική συμβολοσειρά καθορισμένου μήκους που είναι εφικτό να διακριθεί υπολογιστικά από μία επιλεγμένη ομοιόμορφα στην τύχη από το σύνολο όλων των δυαδικών συμβολοσειρών με το συγκεκριμένο μήκος [1].
Συνάρτηση Κατακερματισμού	Μια συνάρτηση που απεικονίζει μια σειρά αυθαίρετου μήκους σε μια σειρά δυαδικών ψηφίων σταθερού μήκους. Οι εγκεκριμένες hash λειτουργίες έχουν σχεδιαστεί για να πληρούν τις ακόλουθες ιδιότητες [2]: 1. (One-way) Είναι υπολογιστικά ανέφικτο να βρεθεί οποιαδήποτε είσοδο που αντιστοιχεί σε κάποια προκαθορισμένο έξοδο, και 2. (Collision Resistant) Είναι υπολογιστικά ανέφικτο να βρεθούν δύο ξεχωριστές είσοδοι που αντιστοιχούν με την ίδια έξοδο.
Παραγωγή Κλειδιών – Key Derivation	Η διαδικασία κατά την οποία χρησιμοποιείται κλειδί για την παραγωγή κρυπτογραφικού υλικού [3]
Συναρτήσεις Παραγωγής Κλειδιών – 'Key Derivation Functions, KDF'	Μία συνάρτηση η οποία παράγει ένα η περισσότερα κλειδιά από μία μυστική τιμή όπως ένα κύριο κλειδί ("master key") ή άλλες γνωστές πληροφορίες όπως έναν κωδικό πρόσβασης συστήματος ή συνθηματική φράση χρησιμοποιώντας μία ψευδό- τυχαία συνάρτηση [3]
Bcrypt	Συνάρτηση παραγωγής κλειδιών που βασίζεται στον αλγόριθμο κρυπτογράφησης Blowfish. Εκτός από την ενσωμάτωση αλατιού - salting για την προστασία από τις επιθέσεις RAINBOW TABLE,το bcrypt είναι έχει μια προσαρμοστική λειτουργία: με την πάροδο του χρόνου, ο αριθμός των επαναλήψεων μπορεί να αυξηθεί και να γίνει πιο αργή, έτσι ώστε να παραμένει ανθεκτική σε brute-force επιθέσεις αναζήτησης, ακόμη και με την αύξηση της δύναμης υπολογιστικών πόρων. [4]
PBKDF	Συνάρτηση παραγωγής κλειδιών που βασίζεται στον κωδικό που εισάγεται από τον χρήστη. Είναι μία τεχνική επέκτασης κλειδιού. Μπορεί να χρησιμοποιηθεί για τον κατακερματισμό κωδικών πρόσβασης με υπολογιστικά εντατικό τρόπο, έτσι ώστε να μειωθεί η αποτελεσματικότητα επιθέσεων τύπου dictionary και brute-force. [5]
Blowfish	Συμμετρικός μπλοκ αλγόριθμος κρυπτογράφησης με μεταβλητό μήκος κλειδιού και 64 Bit μπλοκ. [6]
md5	Ο αλγόριθμος md5 είναι μία ευρέως διαδεδομένη κρυπτογραφική συνάρτηση κατακερματισμού η οποία παράγει μία τιμή κατακερματισμού μήκους 128 Bit (16

	byte) η οποία συνήθως εκφράζεται ως δεκαεξαδικών αριθμό μήκους 32 bit. Έχει χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών κρυπτογράφησης και συνήθως χρησιμοποιείται για την επαλήθευση της ακεραιότητας των δεδομένων [7].
SHA1	Η SHA-1 είναι μια κρυπτογραφική συνάρτηση κατακερματισμού που σχεδιάστηκε από τον Οργανισμό Ηνωμένων Πολιτειών Εθνικής Ασφάλειας και είναι μια αμερικανική Federal Πρότυπο Επεξεργασίας Πληροφοριών που δημοσιεύθηκε από τις Ηνωμένες Πολιτείες NIST. SHA-1 παράγει μία 160-bit (20-byte) τιμή κατακερματισμού. Μία SHA-1 hash τιμή συνήθως αποδίδεται ως ένας δεκαεξαδικός αριθμό, μήκους 40 ψηφία. SHA σημαίνει "secure hash algorithm». Οι τέσσερις εκδόσεις SHA αλγορίθμων έχουν διαφορετική δομή και ονομάζονται αντίστοιχα SHA-0, SHA-1, SHA-2, και SHA-3. SHA-0 είναι η αρχική έκδοση του hash λειτουργία 160-bit που δημοσιεύθηκε το 1993 με την επωνυμία "SHA": δεν υιοθετήθηκε από πολλές εφαρμογές. Δημοσιεύθηκε στο 1995, SHA-1 είναι πολύ παρόμοια με SHA-0, αλλά μεταβάλλει την αρχική προδιαγραφή hash SHA για να διορθώσει κάποιες αδυναμίες. SHA-2, που δημοσιεύθηκε το 2001, είναι σημαντικά διαφορετική από τη συνάρτηση κατακερματισμού SHA-1. [8]
Key stretching	‘Επέκταση κλειδιού’: Είναι η διαδικασία μέσα από την οποία ένα κλειδί, π.χ. κωδικός πρόσβασης, ενισχύεται μέσω ενός αλγορίθμου, π.χ. συνάρτησης κατακερματισμού καθώς και με την τεχνική προσθήκης salt. [9]
Salt	‘Αλας’: είναι τυχαία δεδομένα (τυχαίες συμβολοσειρές) που χρησιμοποιούνται ως πρόσθετη είσοδος σε μία μονόδρομη συνάρτηση που κατακερματίζει – hash έναν κωδικό ή συνθηματικό εισόδου. Η κύρια λειτουργία των αλάτων είναι να ενισχύσει ενάντια επιθέσεις τύπου ‘dictionary attacks’ ενάντια σε ένα κατάλογο με ήδη υπολογισμένα Hashes τον πιο διαδεδομένων κωδικών πρόσβασης ή / και κατά προ-υπολογισμένων επιθέσεων τύπου rainbow. Προτείνεται το salt να είναι ξεχωριστό για κάθε νέο κωδικό και να υπολογίζεται εκ νέου κάθε φορά που στο κωδικό εφαρμόζεται μια συνάρτηση ενίσχυσης του, π.χ. συνάρτηση κατακερματισμού. [10]
Dictionary Attacks	‘Επιθέσεις τύπου λεξικού’: Μέθοδος παράκαμψης των συστημάτων ασφαλείας τα οποία χρησιμοποιούν την αυθεντικοποίηση συνθηματικών πρόσβασης, κατά την οποία ο επιτιθέμενος ελέγχει συστηματικά όλους τους πιθανούς κωδικούς – κλειδιά πρόσβασης. Συνήθως αρχίζει με τις φράσεις – κλειδιά τα οποία έχουν μεγαλύτερη πιθανότητα να χρησιμοποιηθούν ως συνθηματικά

	εισόδου, όπως ονόματα και μέρη. Η λέξη 'λεξικό – dictionary' αναφέρεται στον εισβολέα να χρησιμοποιήσει όλα τα σχετικά λήμματα ενός λεξικού, σε μία προσπάθεια να ανακαλύψει των κωδικό πρόσβασης. Οι επιθέσεις λεξικού απαιτούν αυξημένους υπολογιστικούς πόρους. [11]
Rainbow table	Ο πίνακας 'Rainbow Table' είναι μια λίστα όλων των δυνατών συνδυασμών των 'plaintext' κρυπτογραφημένους κωδικούς πρόσβασης ειδικά για ένα συγκεκριμένο αλγόριθμο κατακερματισμού. Rainbow πίνακες χρησιμοποιούνται συχνά από λογισμικά ανάκτησης κωδικών πρόσβασης – 'password cracking' και κυρίως για επιθέσεις στην ασφάλεια δικτύων πληροφοριακών συστημάτων. Όλα τα συστήματα υπολογιστών που απαιτούν κωδικό πρόσβασης αποθηκεύουν σε μία βάση δεδομένων τους κωδικούς πρόσβασης που σχετίζονται με λογαριασμούς χρηστών, συνήθως κρυπτογραφημένους αντί ως απλό κείμενο ως ένα επιπλέον μέτρο ασφαλείας. Μόλις ο εισβολέας αποκτά πρόσβαση στη βάση δεδομένων, το λογισμικό ανάκτησης συγκρίνει τα δεδομένα της βάσης με την προ – υπολογισμένη λίστα του πίνακα rainbow, των δυνητικών hash των κωδικών που βρίσκονται στη βάση δεδομένων. Ο πίνακας rainbow συνδέει την πιθανότητα ύπαρξης ενός κωδικού με κάθε ένα από τα hash, το οποίο ο εισβολέας μπορεί στη συνέχεια να εκμεταλλευτεί για να αποκτήσει πρόσβαση στο δίκτυο, ως εξουσιοδοτημένος χρήστης. Οι Rainbow πίνακες κάνουν την διαδικασία ανάκτησης κωδικών πρόσβασης πιο γρήγορη από ότι οι προηγούμενες μέθοδοι, όπως η brute-force cracking ή οι επιθέσεις με τύπου dictionary. Ανάλογα με την υπολογιστική ισχύ και το λογισμικό που χρησιμοποιείται, οι πίνακες rainbow μπορούν να χρησιμοποιηθούν για να σπάσει κωδικούς πρόσβασης μήκους 14 αλφαριθμητικών χαρακτήρων σε περίπου 160 δευτερόλεπτα. Ωστόσο, η συγκεκριμένη προσέγγιση απαιτεί μεγάλου μεγέθους RAM μνήμη λόγω της μεγάλης ποσότητας δεδομένων σε ένα τέτοιο πίνακα. [12]
WPA2 (Wi –Fi Protected Access II)	Πρωτόκολλο ασφάλειας και πρόγραμμα πιστοποίησης ασφαλείας που αναπτύχθηκε από την Wi-Fi Alliance για να εξασφαλίσει τα ασύρματα δίκτυα υπολογιστών. Η Alliance δημιούργησε το συγκεκριμένο πρωτόκολλο σε απάντηση στις σοβαρές αδυναμίες που είχαν βρει στο προηγούμενο σύστημα, WEP (Wired Equivalent Privacy). [13]
Εικονικοποίηση Πλατφόρμας	Ένα λογισμικό ελέγχου («επόπτης» ή hypervisor) εκτελούμενο σε πραγματικό υλικό προσομοιώνει ένα υπολογιστικό περιβάλλον, μία «εικονική μηχανή», επάνω από το οποίο μπορεί να τρέξει κάποιο φιλοξενούμενο

	λογισμικό (συνήθως ένας πλήρης πυρήνας), απομονωμένο από το υπόλοιπο σύστημα. Η θεμελιώδης λογική πίσω από την εικονικοποίηση πλατφόρμας είναι η αρχή πως οποιαδήποτε λειτουργία μπορεί να εκτελεστεί είτε από λογισμικό είτε από εξειδικευμένο υλικό· οι μόνες διαφορές αφορούν την ευελιξία και την απόδοση. Είναι δυνατόν να προσομοιώνονται ταυτόχρονα πολλαπλές εικονικές μηχανές, εντελώς απομονωμένες μεταξύ τους, από το ίδιο λογισμικό ελέγχου. Η εικονικοποίηση πλατφόρμας εμφανίστηκε αρχικά τη δεκαετία του 1960, πριν από την επέλαση των μικροϋπολογιστών, σε μεγάλα, συγκεντρωτικά συστήματα (mainframes), αλλά μετά το 2000 και την αλματώδη αύξηση των επιδόσεων του υλικού των PC έχει γίνει πλέον κοινή πρακτική.[14]
Εικονικά Μηχανήματα	Στην επιστήμη των υπολογιστών, μια εικονική μηχανή (VM) είναι μια προσομοίωση ενός συγκεκριμένου συστήματος του υπολογιστή. Οι εικονικές μηχανές λειτουργούν με βάση την αρχιτεκτονική και τις λειτουργίες ενός πραγματικού ή υποθετικού υπολογιστή υπολογιστών, και τις εφαρμογές τους μπορεί να περιλαμβάνει εξειδικευμένο hardware, software, ή ένα συνδυασμό και των δύο. Η Κατάταξη των εικονικών μηχανών μπορεί να βασίζεται στο βαθμό στον οποίο εφαρμόζουν τη λειτουργικότητα των στοχευμένων πραγματικών μηχανών. Με αυτόν τον τρόπο, το σύστημα εικονικών μηχανών (επίσης γνωστή ως πλήρη εικονικοποίηση ΚΟ) παρέχει ένα πλήρες υποκατάστατο για την στοχευμένη πραγματική μηχανή και ένα επίπεδο λειτουργικότητας που απαιτείται για την εκτέλεση ενός πλήρους λειτουργικού συστήματος. Από την άλλη πλευρά, είναι η διαδικασία εικονικές μηχανές έχουν σχεδιαστεί για να εκτελέσει ένα πρόγραμμα υπολογιστή, παρέχοντας μια αφηρημένη και ανεξάρτητα από την πλατφόρμα περιβάλλον εκτέλεσης του προγράμματος. [15]
Λογισμικό Πακέτο Εικονικοποίησης	Λογισμικού εικονικοποίησης επιτρέπει σε ένα ενιαίο κεντρικό υπολογιστή για να δημιουργήσετε και να εκτελέσετε ένα ή περισσότερα εικονικά περιβάλλοντα. Το Λογισμικού εικονικοποίησης χρησιμοποιείται συχνά για να μιμηθεί ένα πλήρες πληροφοριακό σύστημα, ώστε να καταστεί δυνατό ένα φιλοξενούμενο λειτουργικό σύστημα να τρέξει. Ενδεικτικά αναφέρεται ότι για παράδειγμα, επιτρέπει σε ένα σύστημα Linux να λειτουργεί ως επισκέπτης πάνω από ένα PC που τρέχει εγγενώς ένα λειτουργικό σύστημα Microsoft Windows (ή το αντιστρόφως, τρέχει τα Windows ως guest στο Linux). [16]
Apache Server HTTP Server	Ο Apache HTTP είναι ένας εξυπηρετητής του παγκόσμιου ιστού (web). Όποτε ένας χρήστης επισκέπτεται ένα ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί με έναν διακομιστή (server) μέσω του πρωτοκόλλου HTTP,

	ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης. Ο Apache είναι ένας από τους δημοφιλέστερους εξυπηρετητές ιστού, εν μέρει γιατί λειτουργεί σε διάφορες πλατφόρμες όπως τα Windows, το Linux, το Unix και το Mac OS X. Κυκλοφόρησε υπό την άδεια λογισμικού Apache και είναι λογισμικό ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοικτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation). Ο Apache χρησιμοποιείται και σε τοπικά δίκτυα σαν διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων π.χ. Oracle, MySQL. [17] , [18]
PHP	Η PHP είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML. [19], [20]
PHPMyAdmin	Το PHPMyAdmin είναι ένα δωρεάν εργαλείο λογισμικού γραμμένο σε PHP, που προορίζεται για τη διαχείριση της MySQL στο Web. Το συγκεκριμένο εργαλείο υποστηρίζει ένα ευρύ φάσμα ενεργειών για MySQL και Drizzle βάσεις δεδομένων. Οι βασικές λειτουργίες με συχνή περίοδο χρήσης όπως η διαχείριση των βάσεων δεδομένων, των πινάκων, στηλών, σχέσεων, ευρετηρίων, χρηστών, δικαιωμάτων, κ.λπ., μπορεί να πραγματοποιηθεί μέσω της διεπαφής χρήστη, ενώ εξακολουθεί ο διαχειριστής να έχει τη δυνατότητα να εκτελέσει άμεσα κάποια δήλωση SQL. [21], [22]
MYSQL	Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που μετρά περισσότερες από 11 εκατομμύρια εγκαταστάσεις. Έλαβε το όνομά της από την κόρη του Μόντυ Βιντένιους, τη Μάι (αγγλ. My). Το πρόγραμμα τρέχει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων.[23], [24]
Message Authentication Code – Κωδικός Αυθεντικοποίηση Μηνυμάτων	Στη κρυπτογραφία, ένας κωδικός αυθεντικοποίησης μηνυμάτων (συχνά MAC) είναι ένα μικρό κομμάτι των πληροφοριών που χρησιμοποιούνται για την αυθεντικοποίηση ενός μηνύματος και να παρέχει ακεραιότητα και διαβεβαίωση αυθεντικότητας για το μήνυμα. Η διαβεβαίωση ακεραιότητας εντοπίζει τυχαίες και εκ προθέσεως αλλαγές στο μήνυμα, ενώ η διαβεβαίωση αυθεντικότητας επιβεβαιώνει την προέλευση του μηνύματος. Ως αλγόριθμος MAC, μερικές φορές ονομάζεται μια διαμορφωμένη (κρυπτογράφησης) συνάρτηση κατακερματισμού (ωστόσο, κρυπτογραφική συνάρτηση κατακερματισμού είναι μόνο ένας από τους πιθανούς

	τρόπους για να παραχθεί ένα MAC) και δέχεται ως είσοδο ένα μυστικό κλειδί και ένα μήνυμα αυθαίρετου μήκους για να πιστοποιηθεί, και εξάγει ένα MAC (μερικές φορές γνωστή ως tag). Η τιμή MAC προστατεύει τόσο την ακεραιότητα των δεδομένων ενός μηνύματος, καθώς και την αυθεντικότητα του, επιτρέποντας τους επαληθευτές - verifiers (οι οποίοι επίσης κατέχουν το μυστικό κλειδί) να εντοπίσουν τυχόν αλλαγές στο περιεχόμενο του μηνύματος. [25]
HMAC / Hash based Message Authentication Code	Στην κρυπτογραφία, ένας κωδικός αυθεντικοποίησης Μηνύματος βασισμένος σε συνάρτηση κατακερματισμού (HMAC) είναι μια ειδική κατασκευή για τον υπολογισμό του κωδικού επαλήθευσης ταυτότητας μηνύματος (MAC) που περιλαμβάνει μια κρυπτογραφική συνάρτηση κατακερματισμού σε συνδυασμό με ένα μυστικό κρυπτογραφικό κλειδί. Όπως και με οποιοδήποτε MAC, μπορεί να χρησιμοποιηθεί για την επαλήθευση ταυτόχρονα τόσο την ακεραιότητας των δεδομένων όσο και για την εξακρίβωση της γνησιότητας ενός μηνύματος. Κρυπτογραφικές συναρτήσεις hash, όπως MD5 ή SHA-1, μπορούν να χρησιμοποιούνται για τον υπολογισμό ενός HMAC. Το κρυπτογράφημα που προκύπτει ονομάζεται HMAC-MD5 ή HMAC-SHA1 αναλόγως. Η κρυπτογραφική δύναμη του HMAC εξαρτάται από την κρυπτογραφική δύναμη της υποκείμενης συνάρτηση κατακερματισμού, το μέγεθος της εξόδου του hash, και με το μέγεθος και την ποιότητα του κλειδιού. [26]
Replay Attack	Η παραβίαση της ασφάλειας κατά την οποία οι πληροφορίες αποθηκεύονται χωρίς άδεια και μετά αναμεταδίδονται για να ξεγελάσουν το δέκτη σε μη εξουσιοδοτημένες ενέργειες, όπως ψευδή ταυτοποίηση ή αυθεντικοποίηση ή αντίγραφο της συναλλαγής. Για παράδειγμα, τα μηνύματα από έναν εξουσιοδοτημένο χρήστη που συνδέεται σε ένα δίκτυο μπορεί να ληφθούν από έναν εισβολέα και να αποσταλούν εκ νέου την επόμενη μέρα. Ακόμα κι αν τα μηνύματα μπορεί να είναι κρυπτογραφημένα, και ο επιτιθέμενος δεν μπορεί να γνωρίζει ποια είναι τα πραγματικά κλειδιά και τους κωδικούς πρόσβασης, η αναμετάδοση των έγκυρων μηνυμάτων σύνδεσης είναι επαρκής για να αποκτήσει πρόσβαση στο δίκτυο. Επίσης γνωστή και ως "man-in-the-middle επίθεση," μια επίθεση replay μπορεί να προληφθεί με τη χρήση ισχυρών ψηφιακών υπογραφών που περιλαμβάνουν σημάνσεις χρόνου και με ένταξη μοναδικών πληροφοριών από την προηγούμενη συναλλαγή, όπως η αξία του αύξοντα αριθμού ακολουθίας. [27], [28]

botnet	Ένα botnet είναι μια συλλογή από συνδεδεμένα στο Internet προγράμματα που επικοινωνούν με άλλα παρόμοια προγράμματα, προκειμένου να εκτελέσουν κάποιες εργασίες. Αυτές οι εργασίες μπορεί να περιλαμβάνουν κάτι πολύ κοινό, όπως τη διατήρηση του ελέγχου καναλιού επικοινωνίας, ή η συμμετοχή σε επιθέσεις τύπου distributed denial of service DDOS. Η ονομασία botnet προέρχεται από τον συνδυασμό των λέξεων ρομπότ -robot και του δικτύου-network. [29]

5. Θεωρία

5.1 Κρυπτογραφική Συνάρτηση Κατακερματισμού

Τα κλειδιά κρυπτογράφησης είναι απαραίτητα σε όλες σχεδόν τις εφαρμογές ασφαλείας. Στην πράξη ωστόσο, η είσοδος σε εφαρμογές αποτελείται από βασικές συμβολοσειρές οι οποίες, με τη συγκεκριμένη αλληλουχία ψηφίων και μορφή, δεν μπορούν να χρησιμοποιηθούν ως κλειδιά ασφαλούς κρυπτογράφησης.

Επιπλέον, η ασφάλεια κατά την αποθήκευση κωδικών πρόσβασης, συνθηματικών εισόδου και αυθεντικοποίησης χρηστών αποτελεί μία από τις μεγαλύτερες προκλήσεις με την οποία έρχονται αντιμέτωποι οι διαχειριστές και αρχιτέκτονες πληροφοριακών συστημάτων και δικτύων. Τα συνθηματικά εισόδου και αυθεντικοποίησης χρηστών θεωρούνται από τα πιο εμπορεύσιμα είδη πληροφορίας. Πληθώρα επιθέσεων εξελίσσεται σε πραγματικό χρόνο με σκοπό την ανάκτηση του συγκεκριμένου είδους πληροφορίας (κωδικοί πιστωτικών καρτών, λογαριασμών τραπέζης κ.τ.λ.). Η ασφαλής αποθήκευση αυτού του είδους πληροφορίας πραγματοποιείται με διάφορες μεθόδους, μία εκ των οποίων είναι η χρήση των κρυπτογραφικών συναρτήσεων κατακερματισμού.

Μια συνάρτηση κατακερματισμού συνήθως σημαίνει μια λειτουργία που συμπιέζει, που σημαίνει ότι η παραγωγή είναι μικρότερη από ό, τι η είσοδος [30]. Συχνά, μια τέτοια συνάρτηση παίρνει μια είσοδο των αυθαίρετου ή σχεδόν οποιουδήποτε μήκους και δίνει έξοδο της οποίας το μήκος είναι ένας σταθερός αριθμός, όπως 160 bits. Υπάρχουν πολλοί διαφορετικοί τύποι συναρτήσεων κατακερματισμού, με διαφορετικές ιδιότητες ασφαλείας.

Η ακεραιότητα των δεδομένων είναι ένα κρίσιμο μέρος οποιοδήποτε ασφαλούς συστήματος. Χρησιμοποιώντας την έξοδο που παράγεται από μια κρυπτογραφική συνάρτηση κατακερματισμού ο διαχειριστής του συστήματος μπορεί να ανιχνεύσει μη εξουσιοδοτημένες αλλαγές στο αρχείο. Αυτό είναι ιδιαίτερα σημαντικό για την διασφάλιση των κρίσιμων εκτελέσιμων του συστήματος και των ευαίσθητων δεδομένων. Τα εργαλεία που αναφέρονται όλα βασίζονται σε κρυπτογραφικές συναρτήσεις κατακερματισμού όπως τα Tripwire, md5sum και sha1sum. Όταν χρησιμοποιείται από ένα γνώστες διαχειριστή του συστήματος, αυτά τα εργαλεία είναι πολύτιμα στο να εξακριβωθεί αν ένας κακόβουλος χρήστης παρέμβει στα σημαντικά αρχεία του συστήματος [31].

Οι συναρτήσεις κατακερματισμού μπορούν επίσης να συνδυαστούν με άλλες πρότυπες μεθόδους κρυπτογράφησης για την επαλήθευση της προέλευσης των δεδομένων. Όταν οι αλγόριθμοι κατακερματισμού σε συνδυασμό με κρυπτογράφησης, που παράγουν ειδικής επεξεργασίας μήνυμα που προσδιορίζουν την πηγή από την δεδομένων. Τα ειδικής επεξεργασίας μηνύματα ονομάζονται κωδικός αυθεντικοποίησης μηνυμάτων – ‘Message Authentication Codes / MAC’. Ο τυποποιημένος αλγόριθμος που χρησιμοποιείται ονομάζεται HMAC. Ο αλγόριθμος HMAC παρέχει τον έλεγχο της πηγής των δεδομένων, και αποτρέπει επίσης επιθέσεις όπως η επίθεση επανάληψης – replay attack [32].

5.1.1 Ακεραιότητα Δεδομένων

Από δύο διακριτά μηνύματα είναι εξαιρετικά απίθανο να παραχθούν πανομοιότυπα μηνύματα digest, μπορεί κανείς να χρησιμοποιήσει αυτή την ιδιότητα των κρυπτογραφικών συναρτήσεων κατακερματισμού για να ανιχνεύει εάν ένα μήνυμα έχει αλλοιωθεί. Αν κάποιος παίρνει ένα δυαδικό αρχείο και υπολογίζει ένα προϊόν επεξεργασία μέσω κατακερματισμού του αρχείου – digest message, μπορεί κανείς να καταγράψει τη διαδικασία [32] που ακολούθησε. Στο μέλλον, η σύνοψη μπορεί να υπολογισθεί εκ νέου στο αρχείο. Εάν το νέο μήνυμα Digest διαφέρει από το βασικό, τότε το αρχείο έχει τροποποιηθεί με κάποιο τρόπο (Sptizner).

Ο μόνος τρόπος που θα μπορούσε κανείς να υπολογίσει το προϊόν επεξεργασίας μέσω κατακερματισμού ενός αλλαγμένου αρχείου και να ταυτοποιείται με το αρχικό προϊόν επεξεργασίας μέσω κατακερματισμού θα ήταν είχε βρεθεί μία σύγκρουση. Οι συγκρούσεις είναι εξαιρετικά απίθανο να συμβούν εάν το νέο μήνυμα digest ταιριάζει με το αρχικό digest, είναι εξαιρετικά πιθανό ότι το αρχείο δεν έχει αλλοιωθεί. Ως εκ τούτου, οι ιδιότητες των κρυπτογραφικών συναρτήσεων κατακερματισμού μπορούν να χρησιμοποιηθούν για να βεβαιώσουν ότι τα αρχεία δεν έχουν αλλάξει και βοηθούν στο να μπορεί κανείς να καθορίσει γρήγορα την ακεραιότητα του αρχείου.

Η χρήση μηνύματος digest για την επαλήθευση της ακεραιότητας των δεδομένων δεν είναι δυνατή αν ο εισβολέας είναι σε θέση να τροποποιήσει τη θέση στην οποία αποθηκεύονται τα προϊόντα της επεξεργασίας μέσω κατακερματισμού. Ένας εισβολέας θα μπορούσε απλά να εκτελέσει μια μη εξουσιοδοτημένη αλλαγή, να υπολογίσει το νέο digest message για το αρχείο, και να τροποποιήσει τη βάση δεδομένων ώστε να συμπεριλάβει το νέο digest message. Ένα σύστημα διαχείρισης δεν θα γνώριζε τη διαφορά (εκτός αν το μήνυμα digest της βάσης δεδομένων ήταν αποθηκευμένο σε μια ανεξάρτητη θέση μη διαθέσιμη στον εισβολέα).

5.1.2 Αυθεντικοποίηση Δεδομένων

Μια άλλη εφαρμογή των κρυπτογραφικών συναρτήσεων κατακερματισμού είναι η αυθεντικοποίηση δεδομένων. Η αυθεντικοποίηση δεδομένων είναι η διαδικασία εξακρίβωσης της πηγής των δεδομένων. Με την αυθεντικοποίηση δεδομένων, είναι εφικτό να διακριθούν τα μηνύματα που προέρχονται από τον προβλεπόμενο αποστολέα και έναν εισβολέα. Οι συναρτήσεις κατακερματισμού από μόνες τους, δυστυχώς, δεν μπορούν να παράσχουν αυθεντικοποίηση δεδομένων. Δεδομένου ότι οι συναρτήσεις κατακερματισμού είναι ελεύθερα διαθέσιμες, έχει πρόσβαση ο καθένας, συμπεριλαμβανομένου ενός εισβολέα, ώστε να δημιουργήσει ένα μήνυμα digest για ένα μήνυμα. Για παράδειγμα, εάν ένα μήνυμα ηλεκτρονικού ταχυδρομείου αποστέλλεται με ένα μήνυμα digest συνημμένο, ο παραλήπτης μπορεί να χρησιμοποιήσει το μήνυμα digest για να επαληθεύσει την ακεραιότητα του μηνύματος. Ωστόσο, είναι πιθανό ότι ένας εισβολέας τροποποίησε τόσο το μήνυμα όσο και το μήνυμα digest. Αυτού του είδους η αλλαγή είναι μη εντοπίσιμη από τον παραλήπτη.

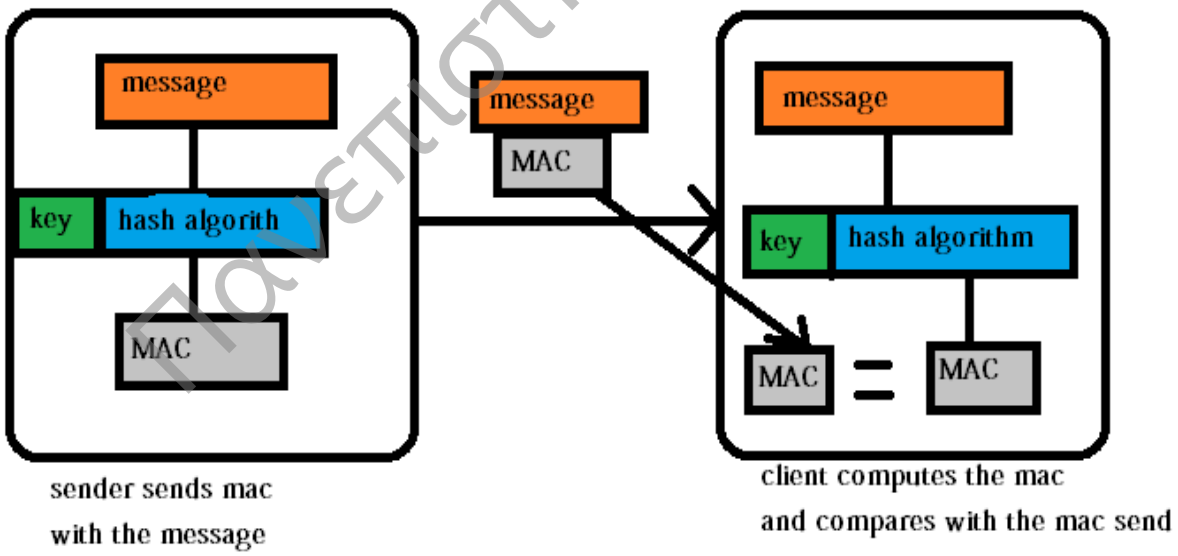
5.1.2.i Παράδειγμα τροποποίησης

Έστω ότι Πελάτης Α στέλνει ένα μήνυμα προς την τράπεζά του, ζητώντας να μεταφέρουν 5 δολάρια από τον έλεγχο τους σε λογαριασμό ταμιευτηρίου τους. Ο επιτιθέμενος Α τότε εμποδίζει

την μετάδοση του μηνύματος του πελάτη Α, και δημιουργεί μία δική του δηλώνοντας προς την τράπεζα να μεταφέρει 500 δολάρια από τον έλεγχο του λογαριασμού του πελάτη Α στο λογαριασμό του επιτιθέμενου Α. Ο Επιτιθέμενος Α υπολογίζει στη συνέχεια το κατάλληλο checksum md5 (κάτι παρόμοιο με b7ab99c9fc23453f77fb6bfef131bc07) για το πλαστό μήνυμα και στέλνει στην τράπεζα. Η τράπεζα θα μπορούσε στη συνέχεια, να βεβαιώσει ότι τα δεδομένα δεν τροποποιήθηκαν κατά τη μεταφορά, καθώς το μήνυμα digest ταιριάζει με το μήνυμα που έχει σταλθεί. Ωστόσο, το μήνυμα δεν προέρχεται από τον πελάτη Α, ο μόνος που έχει το δικαίωμα να κάνει συναλλαγές από το λογαριασμό του. Αυτή είναι μια πολύ κοινή επίθεση που ονομάζεται πλαστογραφία. Εάν η τράπεζα απλά επαληθεύει ότι το μήνυμα digest ταιριάζει με το αρχικό μήνυμα, δεν μπορεί ποτέ να είναι σίγουρη ότι ο αποστολέας ήταν στην πραγματικότητα ο Πελάτης Α. Αναζητείται μια μέθοδος με την οποία η αυθεντικότητα της πηγής δεδομένων μπορεί να επαληθευτεί. Ευτυχώς, με τη χρήση των κρυπτογραφικών συναρτήσεων κατακερματισμού και με τη χρήση μυστικού κλειδιού κρυπτογράφησης, αυτό μπορεί να επιτευχθεί. [32]

5.1.3 Message Authentication Codes – Κωδικός Αυθεντικοποίησης Μηνυμάτων

Κάθε φορά που κάποιος στέλνει ένα μήνυμα που έχει μεταμφιεστεί σε κάποιον άλλο χρήστη αυτό είναι πλαστογραφία, και όπως αναφέρεται στο παραπάνω παράδειγμα, αυτό είναι ένα πολύ μεγάλο πρόβλημα. Προκειμένου να αποφευχθεί αυτό το είδος επίθεσης, αναπτύχθηκαν οι κωδικοί αυθεντικοποίησης μηνυμάτων. Οι κωδικοί αυθεντικοποίησης μηνυμάτων είναι παρόμοιοι σε χρήση με ένα μήνυμα digest. Λαμβάνοντας το μήνυμα και εκτελώντας κάποιους υπολογισμούς, μπορεί κανείς να ελέγξει το ακεραιότητα των δεδομένων. Επιπλέον, οι κωδικοί αυθεντικοποίησης δεδομένων είναι επίσης σε θέση να επαληθεύουν την πηγή των δεδομένων. Οι κωδικοί αυθεντικοποίησης δεδομένων είναι ειδικά δημιουργημένα μηνύματα digest που μπορούν να δημιουργηθούν μόνο από τον αρχικό αποστολέα. [25], [32]



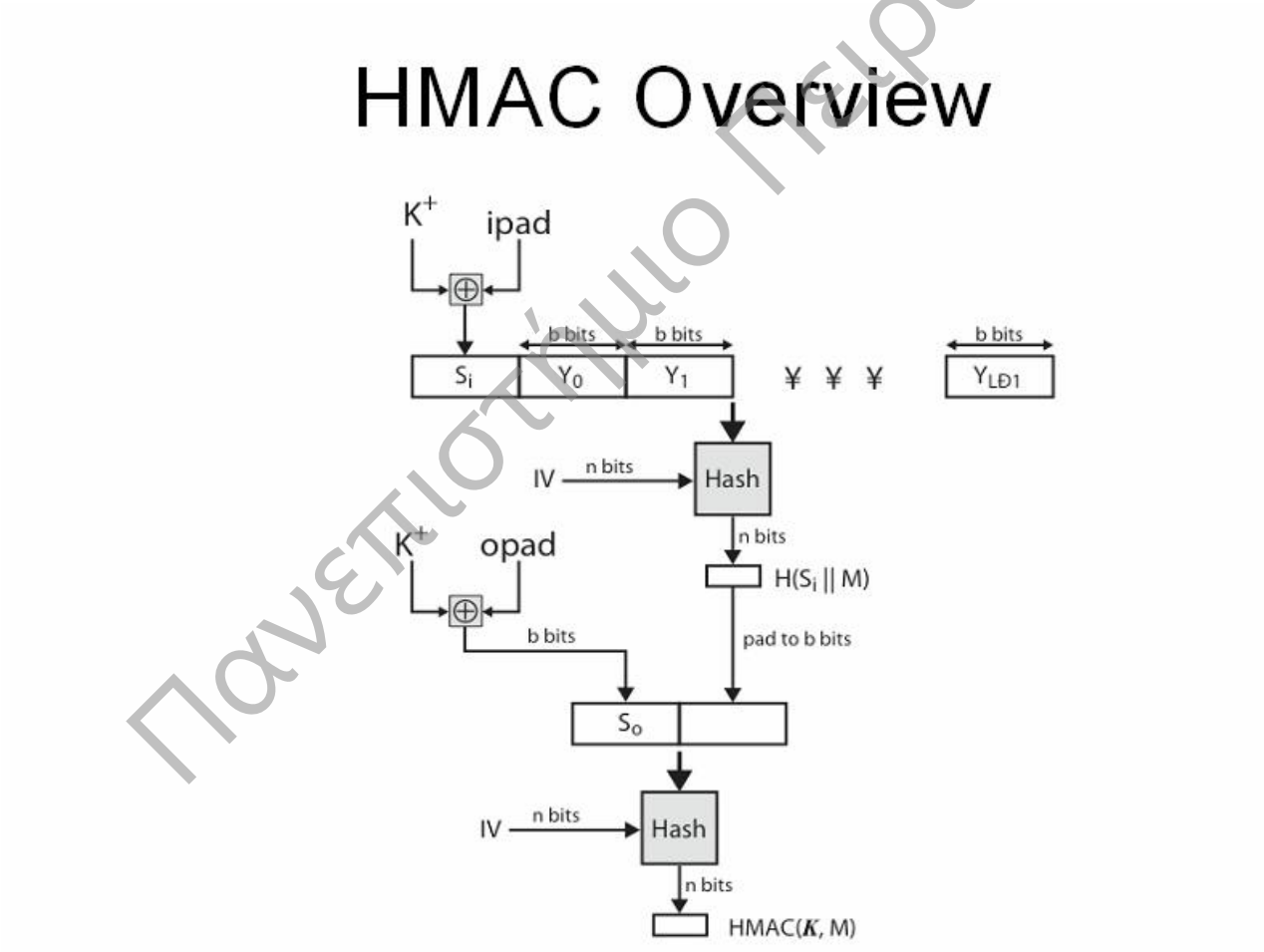
Εικόνα 1: Χρήση Message Authentication Code

Σε πολλές περιπτώσεις, όταν τα δύο μέρη επικοινωνούν δημιουργούν ένα κοινό μυστικό κλειδί που είναι γνωστό μόνο για τον εαυτό τους. Αυτό το κλειδί χρησιμοποιείται για την κρυπτογράφηση των δεδομένων κατά τη διάρκεια της συνόδου. Υπάρχουν διάφορες τεχνικές που χρησιμοποιούνται για να δημιουργήσουν αυτό το κοινόχρηστο κλειδί, χωρίς να εκθέτοντάς στον εισβολέα, όπως το πρωτόκολλο ανταλλαγής κλειδιού Diffie-Hellman. Έστω ότι τα δύο μέρη

μπορούν να δημιουργήσουν με ασφάλεια ένα μυστικό κλειδί, το κλειδί αυτό μπορεί να χρησιμοποιηθεί για να παράγει το κωδικός αυθεντικοποίησης δεδομένων.

5.1.4 HMAC

Μια δημοφιλής εφαρμογή του κωδικού αυθεντικοποίησης δεδομένων είναι το HMAC (Hash Message Authentication Code) σύστημα (Krawczyk). Το πρότυπο πρωτόκολλο για τη δημιουργία και επαλήθευση των κωδικών αυθεντικοποίησης μηνυμάτων που δημιουργούνται μέσω συναρτήσεων κατακερματισμού έχει πολλές μεθόδους για την αντιμετώπιση αυτών των επιθέσεων. Το πρωτόκολλο που χρησιμοποιείται είναι γνωστό ως αλγόριθμος HMAC. Ο αλγόριθμος HMAC (που σημαίνει Κατακερματισμός Κωδικού Αυθεντικοποίησης μηνυμάτων) ορίζεται στο RFC 2085 και αναπτύχθηκε από ερευνητές του NIST το έτος 1997 (Oehler). Η χρήση του είναι του HMAC είναι κοινή σε οποιοδήποτε σύστημα όπου τα μηνύματα απαιτούν αυθεντικοποίηση της πηγής. Πολλά ασφαλή πρωτόκολλα Internet χρησιμοποιούν τον αλγόριθμο HMAC για να παρέχουν αυθεντικοποίηση των δεδομένων, συμπεριλαμβανομένων μερικών παραλλαγών του IPSec (Frankel). [26]

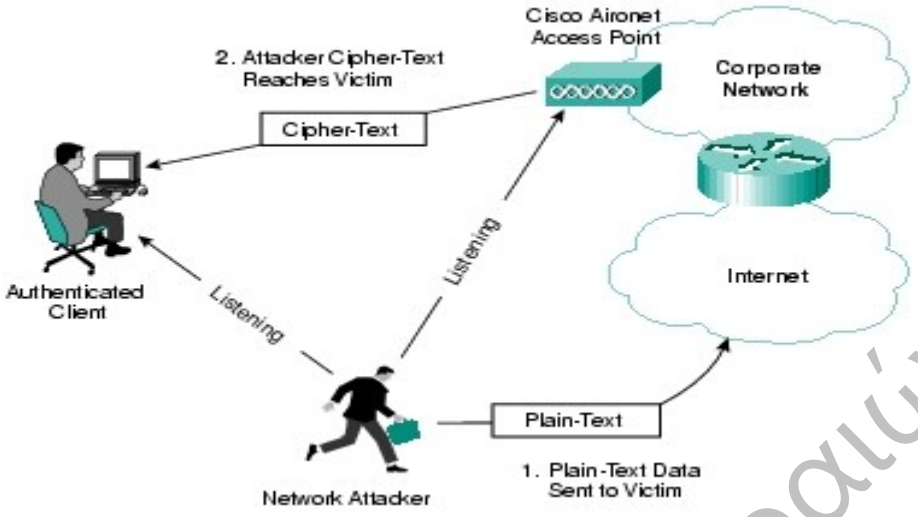


Εικόνα 2: Παράδειγμα χρήσης HMAC

5.1.5 Εξουδετέρωση Επιθέσεων τύπου Replay από τον HMAC

Οι κωδικοί αυθεντικοποίησης μηνυμάτων όπως HMAC αποτρέπουν την πλαστογράφηση των δεδομένων, ανιχνεύοντας πότε τα μηνύματα που αποστέλλονται από οποιονδήποτε άλλο από το

αρχικό αποστολέα. Υπάρχει ένα άλλο είδος της επίθεσης που είναι ιδιαίτερα ανησυχητική, η επίθεση replay (Oehler).

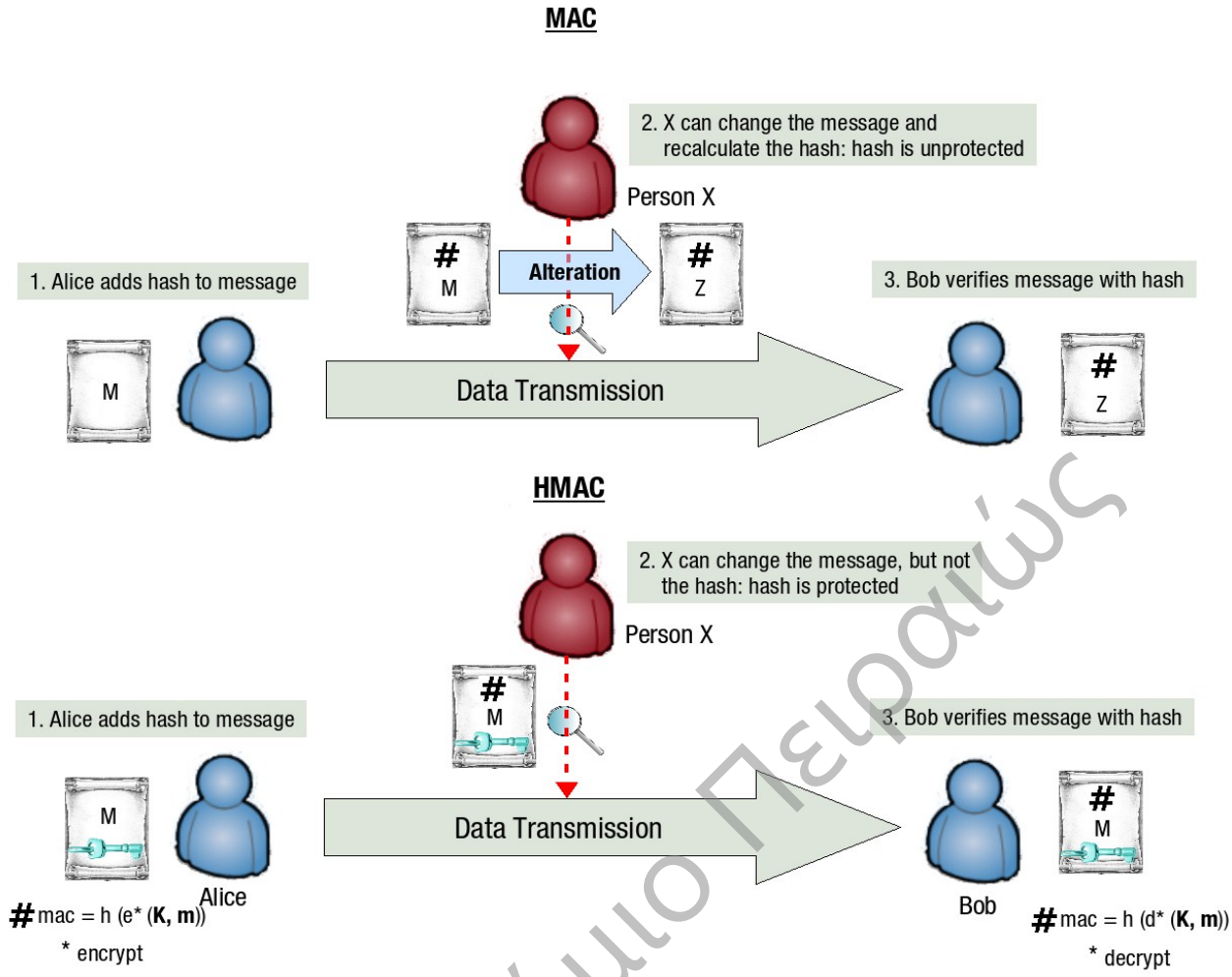


Εικόνα 3: Παράδειγμα Επίθεσης τύπου Replay

Ένας εισβολέας μπορεί να μην είναι σε θέση να δημιουργήσει με επιτυχία ένα κωδικό αυθεντικοποίησης μηνύματος για ένα νέο μήνυμα. Ωστόσο, ένας εισβολέας έχει την πιθανότατα να δει προηγούμενους έγκυρων κωδικούς αυθεντικοποίησης μηνύματος κατά τη μεταφορά των δεδομένων. [32]

5.1.5.i Παράδειγμα Εξουδετέρωσης τύπου Επιθέσεων Replay

Επιτιθέμενος Α είναι ένας έμπορος Διαδικτύου που εμπορεύεται βιβλία στην κρυπτογραφία. Κάθε φορά που γίνεται μία αγορά, βλέπει τα μηνύματα που αποστέλλονται στην τράπεζα για να εξουσιοδοτήσει την τράπεζα και να μεταφέρει τα χρήματα από το λογαριασμό του πελάτη στο δικό του. Ο επιτιθέμενος έχει δει τώρα ένα έγκυρο μήνυμα (μεταφορά χρημάτων από τον λογαριασμό του στο δικό του λογαριασμός) και τον αντίστοιχος κωδικός αυθεντικοποίησης μηνύματος. Ο εισβολέας μπορεί τότε να στείλει αυτό το μήνυμα, μαζί με τον έγκυρο κωδικό αυθεντικοποίησης μηνύματος επανειλημμένα, με αποτέλεσμα τη μεταφορά ολόκληρου του λογαριασμού του πελάτη στο δικό του. Ο αλγόριθμος HMAC αποτρέπει αυτού του είδος επίθεση προσαρτώντας μια μορφή timestamp σε κάθε μήνυμα (Oehler). Ο παραλήπτης μπορεί τότε να επαληθεύσει ότι το μήνυμα δεν έχει ληφθεί ξανά σε προηγούμενο χρόνο. Σε τέτοια περίπτωση όπου πολλαπλά μηνύματα του ίδιου τύπου αποστέλλονται, το νέο timestamp θα διαφοροποιήσει τα μηνύματα. Οι μαθηματικές εργασίες πίσω από τον αλγόριθμο HMAC είναι εξαιρετικά πολύπλοκες.[32]



Εικόνα 4: Παράδειγμα εξουδετέρωσης επίθεσης τύπου Replay

5.2 Συνάρτηση Παραγωγής Κλειδιών – Key derivation Functions

Οι συναρτήσεις παραγωγής κλειδιών χρησιμοποιούνται συνήθως σε συνδυασμό με άλλες γνωστές παραμέτρους ασφαλείας με σκοπό την παραγωγή ενός ή περισσοτέρων κλειδιών κρυπτογράφησης από μία κοινή μυστική τιμή. Κάποιες από τις πιο γνωστές συναρτήσεις τέτοιου είδους είναι [33]:

- PBKDF
- SCRYPT
- BCRYPT

Οι συναρτήσεις παραγωγής κλειδιών μπορούν επίσης να χρησιμοποιηθούν και για την παραγωγή κλειδιών κρυπτογράφησης με συγκεκριμένες επιθυμητές ιδιότητες για αποφυγή αδύναμων κλειδιών σε ορισμένα συστήματα ασφαλείας. Επιπλέον, η πιο διαδεδομένη εφαρμογή των συγκεκριμένων συναρτήσεων είναι για ‘password hashing’ σε εφαρμογές όπου απαιτείται επαλήθευση κωδικού πρόσβασης χρηστών. Σε παραγωγικά συστήματα, όπου διακινούνται μεγάλοι όγκοι πληροφορίας, συνηθίζεται να χρησιμοποιείται ως λειτουργικό σύστημα η οικογένεια UNIX. Σε αυτό τον τύπο λειτουργικών συστημάτων, οι πληροφορίες των χρηστών αποθηκεύονται σε ένα από τα βασικότερα αρχεία συστήματος, το αρχείο ‘passwd’. Το συγκεκριμένο αρχείο περιλαμβάνει τις κύριες πληροφορίες για τους χρήστες του συστήματος όπως:

- Όνομα χρήστη
- Κατάσταση χρήστη
- 'Hash' κωδικού πρόσβασης
- Τελευταία ημερομηνία εισόδου στο σύστημα
- Ρυθμίσεις κωδικού πρόσβασης
- Τελευταία ημερομηνία αλλαγής κωδικού πρόσβασης\

5.2.1 Password Hashing

Οι Key Derivation Function χρησιμοποιούνται κυρίως για 'password hashing' καθώς διαθέτουν τα απαραίτητα χαρακτηριστικά αν και δεν ήταν αυτός ο αρχικό σκοπός δημιουργίας τους. Επιπλέον αποτελούν ένα από τα συστατικά των πρωτόκολλων συμφωνίας κλειδιών. Παρέχουν τη δυνατότητα εισαγωγής επιθυμητής χρονοκαθυστέρησης με σκοπό την αποτροπή επιθέσεων 'Brute – force' ή/και ' dictionary'[33].

Επιπλέον ιδιότητα των key derivation Function είναι η δημιουργία κλειδιού κατάλληλου για χρήση ως είσοδο σε αλγόριθμο κρυπτογράφησης. Αυτό σημαίνει τη λήψη ενός κωδικού πρόσβασης και επεξεργασίας του μέσω ενός αλγορίθμου όπως PBKDF2, HMAC ή HKDF. Η διαδικασία αυτή είναι συνήθως γνωστή ως 'key stretching'[33].

5.2.2 Key stretching

Key stretching είναι οι τεχνικές που χρησιμοποιούνται για την ενίσχυση ενός κλειδιού, το οποίο χρησιμοποιείται συνήθως ως συνθηματικό εισόδου, και χαρακτηρίζονται από την χρήση τους για την ενίσχυση της ασφάλειας μηχανισμών αυθεντικοποίησης. Το βασικό χαρακτηριστικό της τεχνικής 'key stretching' είναι η αύξηση του μεγέθους ενός κλειδιού, και η μη δυνατότητα προσδιορισμού της αρχικής φράσης μη γνωρίζοντας τις παραμέτρους που χρησιμοποιήθηκαν[9]. Η πιο διαδεδομένη διαδικασία που ακολουθείται είναι η εξής: Η φράση κλειδί υπολογίζεται μέσω μιας συνάρτησης κατακερματισμού παραπάνω από μία φορές, εμπλουτισμένη με 'salt'.

Μερικά παραδείγματα από τεχνικές που χρησιμοποιούνται κατά την διαδικασία επέκτασης κλειδιού είναι οι ακόλουθοι:

- Κρυπτογραφική συνάρτηση κατακερματισμού
- Μπλοκ Κρυπτογράφησης
- Προσθήκη άλατος –salting

5.2.3 PBKDF1/2

Η PBKDF1 δημιουργήθηκε αρχικά από την ίδια ομάδα που δημιούργησε τον αλγόριθμο RSA - RSA Security LLC. Δημιουργήθηκε στο πλαίσιο των Public Key Cryptography Standards (PKCS). Αργότερα, αντικαταστάθηκε από την συνάρτηση PBKDF2, ως νέα και βελτιωμένη έκδοση.

Η συγκεκριμένη συνάρτηση εφαρμόζει μια ψευδοτυχαία λειτουργία, όπως ένα κρυπτογραφικό hash, κρυπτογράφημα ή HMAC με τον κωδικό πρόσβασης εισόδου ή τη φράση πρόσβασης μαζί με 'salt' και επαναλαμβάνει τη διαδικασία πολλές φορές για να παραχθεί ένα προερχόμενο κλειδί, το οποίο μπορεί στη συνέχεια να χρησιμοποιηθεί ως ένα κρυπτογραφικό κλειδί. Η

προστιθέμενη υπολογιστική ισχύ καθιστά την ανάκτηση του κωδικού με μεθόδους Password cracking πολύ πιο δύσκολη, και είναι γνωστή και ως το επέκταση κλειδιού. [5]

Το αρχικό πρότυπο της συγκεκριμένης συνάρτησης παρουσιάσθηκε το έτος 2000, στο οποίο η συνιστώμενος ελάχιστος αριθμός των επαναλήψεων ήταν 1000, αλλά η παράμετρος έχει ως στόχο να αυξηθεί με την πάροδο του χρόνου, ανάλογα με την αύξηση της απόδοσης και ταχύτητας των CPU. Προσθέτοντας αλάτι στον κωδικό μειώνεται η ικανότητα να χρησιμοποιηθούν προϋπολογισμένα hashes (πίνακες rainbow) για επιθέσεις. Επιπλέον κάθε κωδικός πρέπει να ελεγχθεί μεμονωμένα, το οποίο απαιτεί αυξημένη υπολογιστική ισχύ. Το συγκεκριμένο πρότυπο συνιστά ένα μήκος άλατος τουλάχιστον 64 bits.

5.2.3.i Παράμετροι εισόδου

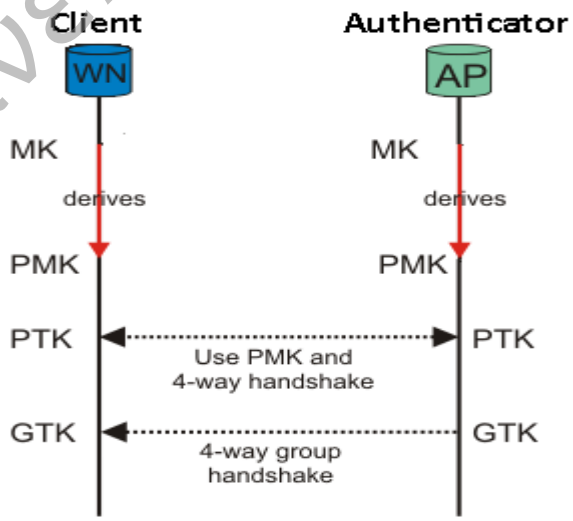
Η PBKDF έχει 5 παραμέτρους εισόδου [5]:
KD = PBKDF 1/2 (PRF, Password, Salt, c, dkLen)

1. RPF: μια ψευδοτυχαία συνάρτηση δύο παραμέτρων, με μήκος εξόδου hLen
2. Password: Ο κύριος κωδικός από τον οποίο παράγεται το κλειδί
3. Salt: Το άλας το οποίο χρησιμοποιείται για την επέκταση του κλειδιού
4. c: Ο επιθυμητός αριθμός επαναλήψεων
5. dkLen: Το επιθυμητό μήκος του παραγόμενου κλειδιού
6. DK: Το παραγόμενο κλειδί

Παράδειγμα εφαρμογής της παραπάνω συνάρτησης είναι το πρωτόκολλο WPA2, όπου οι παράμετροι έχουν ορισθεί:

DK = PBKDF2(HMAC−SHA1, passphrase, ssid, 4096, 256)

Το WPA βασίζεται σε ένα σύστημα Pre-Shared Key (PSK), στο οποίο μπορεί να εισαχθεί ένα κύριο κλειδί, αλλά το Pairwise Παροδικές Key (PTK) δεν είναι γνωστό κατά τη διάρκεια της «four-way χειραψίας." [34]



Εικόνα 5: WPA2

Η Ταυτότητα στηρίζεται στην εξαγωγή του PTK από ένα ζεύγος Master Key, το οποίο με τη σειρά του προέρχεται από ένα κύριο κλειδί. Χρειάζονται περίπου πέντε ή έξι περισσότεροι μετασχηματισμοί για να πάει από το PMK στο PTK, αλλά η ταχύτητα με την οποία μπορεί να επιτευχθεί password cracking στο WPA παρουσιάζεται συχνά σε μονάδες PMK, το οποίο αποτελεί το πιο εντατικής υπολογιστικής τμήμα μιας επίθεσης brute-force.

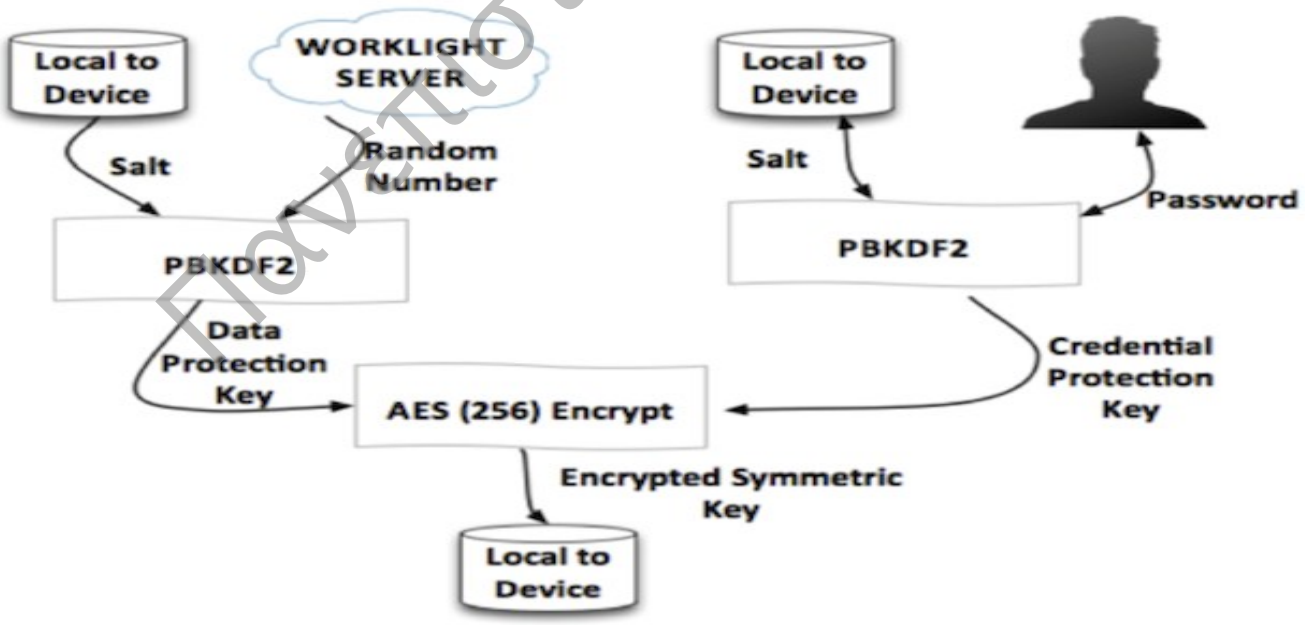
5.2.3.ii Διαδικασία εφαρμογής PBKDF

Η διαδικασία αποτελείται από δύο βήματα:

1. Δημιουργία data1 και data2 από τον κωδικό πρόσβασης και το salt.
2. Υπολογισμός του κλειδιού χρησιμοποιώντας επικλήσεις μετασχηματισμού χρησιμοποιώντας ένα βρόχο - loop, το οποίο μπορεί να υλοποιηθεί ακόλουθα:

```
for (int i = 0? i <iteration_count? i + +)
        {
data1 = SHA1_Transform (data1, data2)?
data2 = SHA1_Transform (data2, data1)?
        }
```

Η είσοδος αποτελείται από τον κωδικό πρόσβασης και το salt (τυχαία bits) προκειμένου να δημιουργηθεί η πρώτη παράμετρος δεδομένων. Αυτό αντιπροσωπεύει το κλειδί το οποίο είναι σε μη τελική μορφή. Από εκεί, το κλειδί κατακερματίζεται συνεχώς σε έναν κύκλο, όπου ο επόμενος υπολογισμός βασίζεται στο προηγούμενο, προκειμένου να συνεχιστεί. Για κάθε διάστημα, η τιμή του κλειδιού αλλάζει. Η διαδικασία επαναλαμβάνεται όσες φορές έχει ορισθεί από τον προγραμματιστή μέχρι να παραχθεί το τελικό κλειδί [34].



Εικόνα 6: Παράδειγμα χρήσης συνάρτησης PBKDF2

5.2.4 SCRYPT

Η συνάρτηση παραγωγής κλειδιών Scrypt δημιουργήθηκε και υλοποιήθηκε από το 'Tarnashp online back up system'. Χρησιμοποιήθηκε σε απάντηση των ήδη υλοποιημένων συναρτήσεων Bcrypt και PBKDF. Η ομάδα που δημιούργησε τη συγκεκριμένη συνάρτηση υποστηρίζει ότι σε σύγκριση με τις δύο 'αντίπαλες' συναρτήσεις, στην περίπτωση που χρειάζονται 5 δευτερόλεπτα για την παραγωγή ενός κλειδιού με σύγχρονο υλικό – Hardware, το κόστος μιας επίθεσης brute force εναντίον μιας υλοποίησης η οποία στηρίζεται στην scrypt συνάρτηση είναι περίπου 4000 φορές μεγαλύτερο από το κόστος μιας παρόμοιας επίθεσης εναντίον bcrypt (για να βρεθεί το ίδιο password), και 20.000 φορές μεγαλύτερη από ό, τι μια παρόμοια επίθεση εναντίον PBKDF2 [35].

5.2.4.i Βοηθητικό πρόγραμμα Κρυπτογράφησης βασισμένο στη συνάρτηση Scrypt

Στην παρουσίαση της συγκεκριμένης συνάρτησης παρουσιάζεται επίσης και ένα απλό βοηθητικό πρόγραμμα κρυπτογράφησης. Το συγκεκριμένο πρόγραμμα με κωδικό πρόσβασης αποτελεί μια επίδειξη της scrypt συνάρτησης. Με βάση το σύγχρονο υλικό - hardware και με τις προεπιλεγμένες παραμέτρους, το κόστος της ανάκτησης ενός κωδικού πρόσβασης σε ένα κρυπτογραφημένο αρχείο με scrypt είναι περίπου 100 δισεκατομμύρια φορές περισσότερο από το κόστος της ανάκτησης του ίδιου κωδικού πρόσβασης σε ένα κρυπτογραφημένο αρχείο με openssl. Αυτό σημαίνει ότι ένας κωδικός πέντε χαρακτήρων με τη χρήση scrypt είναι ισχυρότερος από ένα κωδικό δέκα χαρακτήρων χρησιμοποιώντας openssl [36].

Η παραπάνω υλοποίηση μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση δεδομένων καθώς και για την αντίστροφη διαδικασία.

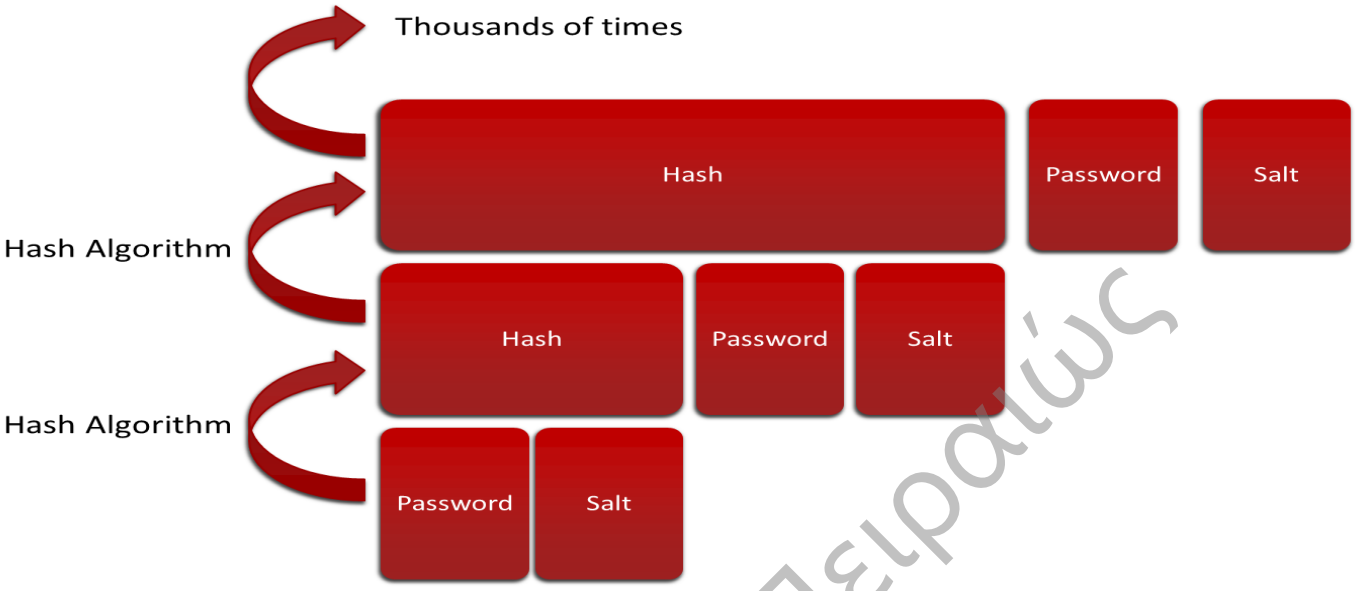
5.2.4.ii Παράμετροι Εισόδου

Η συνάρτηση scrypt υποστηρίζει επίσης τρεις επιλογές γραμμής εντολών [36]:

- -t maxtime , η οποία αναθέτει στη scrypt να περάσει το μέγιστο 'maxtime' δευτερόλεπτα για τον υπολογισμό του κλειδιού κρυπτογράφησης που προέρχεται από τον κωδικό πρόσβασης. Για την κρυπτογράφηση, η τιμή αυτή θα καθορίσει το πόσο ασφαλές είναι το κρυπτογραφημένα δεδομένα, ενώ για την αποκρυπτογράφηση αυτή η τιμή χρησιμοποιείται ως ανώτατο όριο (αν η scrypt ανιχνεύει ότι θα πάρει πάρα πολύ χρόνο για να αποκρυπτογραφήσει τα δεδομένα, θα δημιουργηθεί ένα μήνυμα σφάλματος).
- -m maxmemfrac είναι η εντολή scrypt που χρησιμοποιείται για να καθορίσει το μέγιστο καθορισμένο κλάσμα της διαθέσιμης μνήμης RAM για τον υπολογισμό του παραγόμενου κλειδιού κρυπτογράφησης. Για την κρυπτογράφηση, αυξάνοντας την συγκεκριμένη τιμή μπορεί να αυξήσει την ασφάλεια των κρυπτογραφημένων δεδομένων, ανάλογα με την τιμή 'maxtime'. Για την αποκρυπτογράφηση, αυτή η τιμή χρησιμοποιείται ως ανώτατο όριο και μπορεί να προκαλέσει τη scrypt να δημιουργήσει μήνυμα σφάλματος.
- -M Maxmem η εντολή scrypt η οποία χρησιμοποιείται για τον καθορισμό του μέγιστου αριθμό των bytes της μνήμης RAM κατά τον υπολογισμό του παραγόμενου κλειδιού κρυπτογράφησης

Εάν η κρυπτογραφημένα δεδομένα είναι 'corrupted', h scrypt θα εκτελεστεί με μία μη-μηδενική κατάσταση. Ωστόσο, η scrypt μπορεί να παράγει έξοδο πριν κρίνει ότι τα κρυπτογραφημένα

δεδομένα ήταν corrupted. Για αυτό το λόγο για εφαρμογές που απαιτούν πιστοποίηση δεδομένων, θα πρέπει η έξοδος της scrypt να αποθηκευτεί σε μια προσωρινή θέση και να ελεχθεί ο κωδικός εξόδους scrypt πριν από τη χρήση των κρυπτογραφημένων δεδομένων.



Εικόνα 7: Παράδειγμα για την κατανόηση της λειτουργίας των συναρτήσεων KDF

5.2.5 Bcrypt

Η συνάρτησης παραγωγής κλειδιών Bcrypt δημιουργήθηκε και παρουσιάσθηκε το έτος 1999 από τους Niels Provos και David Mazieres . Η παρουσίαση έγινε στα πλαίσια του USENIX Association – Advanced Computing Systems Association. Η συνάρτηση βασίζεται στον αλγόριθμο Blowfish. Κύριο χαρακτηριστικό της είναι ότι εισάγει ένα κόστος παραγωγής κλειδιού το οποίο μειώνει δραστικά τις πιθανότητες ανάκτησης ενός κωδικού πρόσβασης [4].

Η συνάρτηση Bcrypt αποτελεί τον προεπιλεγμένο αλγόριθμο κατακερματισμού κωδικού πρόσβασης για το BSD, ανάμεσα σε πολλά άλλα συστήματα. Υπάρχουν εφαρμογές της bcrypt για Ruby, Python, C, C #, Perl, PHP, Java και άλλες γλώσσες προγραμματισμού.

5.2.5.i Χαρακτηριστικά Bcrypt

Ειδικότερα, εκτός από την ενσωμάτωση άλατος – 'salt' για την προστασία ενάντια στις επιθέσεις τύπου 'raibow tables', η συγκεκριμένη συνάρτηση έχει μια προσαρμοστική λειτουργία: με την πάροδο του χρόνου, ο αριθμός των επαναλήψεων μπορεί να αυξηθεί και η εκτέλεση της συνάρτησης να γίνει περισσότερο χρονοβόρα. Η συγκεκριμένη λειτουργία καθιστά τις υλοποιήσεις της συνάρτησης Bcrypt ανθεκτικές σε επιθέσεις τύπου 'brute force'.

Το πρόθεμα "$ 2a $" ή "2y" σε μια συμβολοσειρά hash σε ένα αρχείο κωδικού πρόσβασης υποδεικνύει ότι η συνάρτηση κατακερματισμού που χρησιμοποιείται είναι η Bcrypt. [4]

5.2.5.ii Παράμετροι Εισόδου

Το υπόλοιπο της συμβολοσειράς κατακερματισμού περιλαμβάνει [4]:

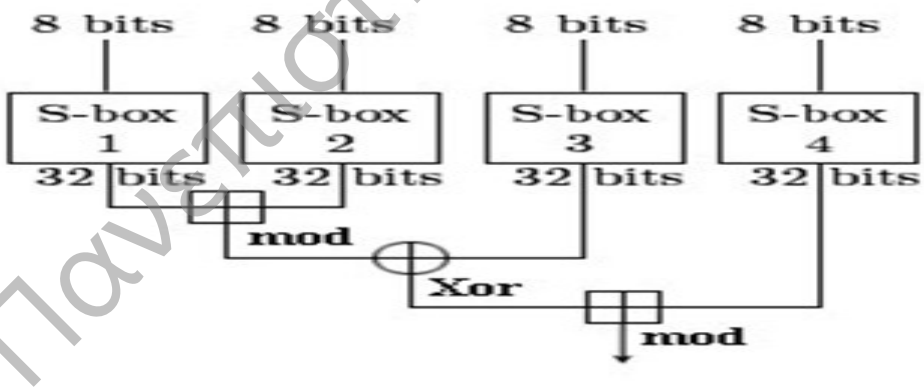
- την παράμετρο του κόστους,
- ένα αλάτι ‘ salt’ μήκους 128- bit (οι base-64 κωδικοποιούνται ως 22 χαρακτήρες),
- και την τιμή κατακερματισμού 192-bit (οι base-64 κωδικοποιούνται ως 31 χαρακτήρες)

5.2.5.iii Διαδικασία Εφαρμογής Bcrypt

Πριν την αναφορά στη διαδικασία εφαρμογής της συνάρτησης Bcrypt, πρέπει να αναφερθεί η λειτουργία του αλγορίθμου Blowfish, πάνω στον οποίο βασίστηκαν οι Pravos και Mazieres, για τη συνάρτηση Bcrypt.

5.2.5. iv Λειτουργία Blowfish

Ο αλγόριθμος Blowfish είναι αξιοσημείωτος μεταξύ των κρυπταλγόριθμων για το αυξημένο κόστος υλοποίησης και εγκατάστασης του. Ειδικότερα, η διαδικασία έναρξης λειτουργίας του ξεκινά με τα δευτερεύοντα κλειδιά σε μια τυπική κατάσταση. Έπειτα χρησιμοποιεί αυτή την κατάσταση για να εκτελέσει μια κρυπτογράφηση μπλοκ χρησιμοποιώντας μέρος του κλειδιού, και χρησιμοποιεί το αποτέλεσμα της κρυπτογράφησης (η οποία προσφέρει περισσότερη ακρίβεια σε σύγκριση με συνάρτηση κατακερματισμού) για να αντικαταστήσει ορισμένα από τα δευτερεύοντα κλειδιά. Στη συνέχεια, χρησιμοποιεί αυτό την τροποποιημένη κατάσταση για να κρυπτογραφήσει ένα άλλο μέρος του κλειδιού, και χρησιμοποιεί το αποτέλεσμα για να αντικαταστήσει περισσότερα από τα δευτερεύοντα κλειδιά. Προχωρά με αυτό τον τρόπο, χρησιμοποιώντας μια προοδευτική τροποποιημένη κατάσταση για τον κατακερματισμό του κλειδιού και να αντικαταστήσει τα κομμάτια για το σύνολο των δευτερευόντων κλειδιών. [6]



Εικόνα 8: Παράδειγμα λειτουργίας αλγορίθμου Blowfish

5.2.5.v Λειτουργία Bcrypt

Οι δημιουργοί του bcrypt εκμεταλλεύτηκαν, και πήρε περαιτέρω. Ανέπτυξαν ένα νέο βασικό αλγόριθμο ρύθμισης για Blowfish, πραγματοποιώντας ‘debugging’ στο αποτέλεσμα της κρυπτογράφησης "Eksblowfish" ("ακριβό βασικό πρόγραμμα Blowfish"). Η βασική υλοποίηση ξεκινάει με μια τροποποιημένη μορφή του προτύπου Blowfish κλειδιού εγκατάστασης, στην οποία τόσο το salt και όσο ο κωδικός πρόσβασης χρησιμοποιούνται για να ορίσουν όλα τα δευτερεύοντα κλειδιά. [4], [6]

Στη συνέχεια πραγματοποιούνται μια σειρά από γύρους κατά τους οποίους εφαρμόζεται ο πρότυπος αλγόριθμος Blowfish, χρησιμοποιώντας εναλλάξ το salt και τον κωδικό πρόσβασης ως το κλειδί. Κάθε γύρος ξεκινά με είσοδο το δευτερεύον κλειδί από τον προηγούμενο γύρο. Από πλευρά Κρυπτογραφίας, η συγκεκριμένη εφαρμογή δεν αποτελεί ισχυρότερη λύση. Ωστόσο, ο αριθμός των γύρων μπορεί να ορισθεί ανεξάρτητα και να αυξήσει κατά αυτό τον τρόπο τη διάρκεια και το χρόνο που θα χρειαζόταν μία επίθεση τύπου Brote force για να ανακτήσει έναν κωδικό πρόσβασης ο οποίος έχει αποθηκευτεί ως αποτέλεσμα της συνάρτησης Bcrypt. [4], [6]

Ο αριθμός των επαναλήψεων είναι μια δύναμη του δύο, η οποία είναι μία από τις παραμέτρους που ορίζονται για τον αλγόριθμο. Ο αριθμός των επαναλήψεων κωδικοποιείται στο αποτέλεσμα.

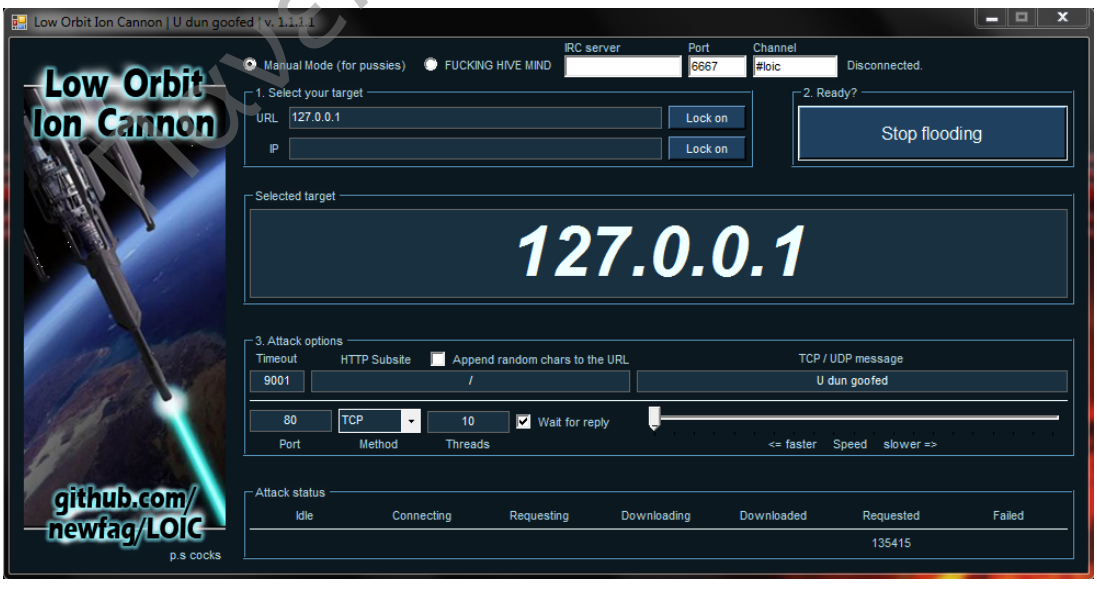
5.3 LOIC – Εργαλείο Ανοιχτού Κώδικα για DDOS επιθέσεις

5.3.1 Λειτουργία LOIC

Το LOIC – Low Orbit Ion Cannon είναι ένα εργαλείο ανοιχτού κώδικα το οποίο δημιουργήθηκε αρχικά για προσομοιώσεις ακραίων καταστάσεων στα δίκτυα υπολογιστών – 'network stress testing' και αργότερα για εκτέλεση επιθέσεων τύπου denial-of-service. Αναπτύχθηκε αρχικά από την Praetox Technologies, αλλά αργότερα μετατράπηκε σε εργαλείο ανοιχτού κώδικα και δόθηκε ελεύθερα στο κοινό. [37]

Το LOIC χρησιμοποιείται για να εκτελέσει denial-of-service (DoS) επιθέσεις σε ένα στόχο κατά τις οποίες κατακλύζεται ο διακομιστής με TCP ή UDP πακέτα με την πρόθεση να διαταραχθεί η υπηρεσία που προσφέρεται από ένα συγκεκριμένο κεντρικό υπολογιστή. [37]

Συχνή χρήση του LOIC πραγματοποιείται κατά την εκτέλεση επιθέσεων DDOS, όταν εγκαθίσταται από πληθώρα χρηστών σε διαφορετικού ηλεκτρονικού υπολογιστές οι οποίοι λειτουργούν ως botnets και λαμβάνουν μέρος στην επίθεση.



Εικόνα 9: Γραφικό περιβάλλον εργαλείου LOIC

5.3.2 Εκτέλεση Επίθεσης χρησιμοποιώντας το εργαλείο LOIC

Δημοφιλές και γνωστό στο ευρύ κοινό, έγινε κατόπιν της γνωστής κινητοποίησης Operation Global Blackout. Η συγκεκριμένη εφαρμογή έχει ενσωματωμένη μία λειτουργία που ονομάζεται *Hivemind.* Η λειτουργία αυτή προσφέρει στους χρήστες τη δυνατότητα σύνδεσης αντίγραφου της εφαρμογής που έχει κατεβάσει με κάποιον IRC server.

Ο IRC Server αποτελεί τον ενδιάμεσο σταθμό μέσω του οποίου κάποιος άλλος χρήστης αποκτά πρόσβαση στην υπολογιστική δύναμη των τερματικών στα οποία έχει εγκατασταθεί η εφαρμογή LOIC. Με την έναρξη της πρόσβασης, η ισχύς των πόρων περνά στα χέρια του κακόβουλου χρήστη, οποίος έχει τη δυνατότητα να στρέψει το δίκτυο πλέον των υπολογιστών προς ένα συγκεκριμένο στόχο.

Κατά κύριο λόγο απαιτούνται αρκετά μεγάλος αριθμός τερματικών για να πραγματοποιηθεί μία επίθεση DDOS που έχει ως στόχο ένα μέτριου μεγέθους server. Καθώς όμως δεν υπάρχουν γεωγραφικοί περιορισμοί στην επιλογή των πόρων οι οποίοι θα χρησιμοποιηθούν ως botnets, οι επιλογές είναι απεριόριστες. Μοναδική προϋπόθεση είναι η εγκατάσταση και ενεργοποίηση της του συγκεκριμένου εργαλείου.

Ο λόγος που καθιστά επιτυχείς τις επιθέσεις μέσω LOIC είναι η ταυτόχρονη αποστολή αιτημάτων TCP/UDP στον server στόχο από όλα τα συνδεδεμένα botnets. Η συνεχής αποστολή τέτοιου είδους αιτημάτων μπορεί να πραγματοποιηθεί και με ένα μόνο τερματικό. Ωστόσο, η επίθεση δεν θα στεφθεί με επιτυχία καθώς τα αιτήματα που αποστέλλονται κατά αυτόν τον τρόπο αγνοούνται εύκολα από τον στόχο.

Η κεντρική διαχείριση αυτού του δικτύου υπολογιστών γίνεται από ένα και μόνο άτομο, ο οποίος αποφασίζει τη χρονική στιγμή της επίθεσης. Αυτό φυσικά συνεπάγεται πως δίνεται σε κάποιον τρίτο, ο έλεγχος του υπολογιστή. [38]

6. Υλοποίηση

6.1 Περίληψη Υλοποίησης

Στα πλαίσια της διπλωματικής εργασίας εξετάσθηκε η καθυστέρηση που προσδίδει η εφαρμογή της συνάρτησης Bcrypt για διάφορες τιμές της παραμέτρου 'cost' και σε αντιπαραβολή με την αντίστοιχη που προσδίδει η εφαρμογή του αλγορίθμου SHA1 και με μεταβλητό μήκος κωδικού πρόσβασης.

Αναλυτικότερα εκτελέσθηκαν τα ακόλουθα σενάρια:

- 3 σενάρια στα οποία πραγματοποιήθηκε αλλαγή του μήκους κωδικού πρόσβασης με σταθερή τιμή της παραμέτρου κόστους της συνάρτησης Bcrypt, με τον κωδικό πρόσβασης να παίρνει τις ακόλουθες τιμές:
  - 8
  - 10
  - 15
- 3 σενάρια στα οποία πραγματοποιήθηκε αλλαγή της παραμέτρου κόστους της συνάρτησης Bcrypt, με σταθερό μήκος ψηφιών κωδικού πρόσβασης ανά 3 σενάρια, με την τιμή της παραμέτρου να παίρνει τις ακόλουθες τιμές:
  - 7
  - 10
  - 15
- 3 σενάρια στα οποία πραγματοποιήθηκε αλλαγή του μήκους κωδικού πρόσβασης με χρήση του αλγορίθμου SHA1, με τον κωδικό πρόσβασης να παίρνει τις ακόλουθες τιμές:
  - 8
  - 10
  - 15

6.2 Χαρακτηριστικά Υλοποίησης

6.2.1 Υλικό - Hardware

Η υλοποίηση της διπλωματικής εργασίας πραγματοποιήθηκε σε φορητό ηλεκτρονικό υπολογιστή με χαρακτηριστικά:

- Επεξεργαστής: Intel core i5
- Μνήμη RAM: 4GB
- Λειτουργικό Σύστημα: Windows 7 Home Premium

Στα πλαίσια της υλοποίησης, δημιουργήθηκαν στη σουίτα Back Player 2 εικονικά μηχανήματα με τα ακόλουθα χαρακτηριστικά:

1. Μνήμη RAM: 1 GB
2. Επεξεργαστής: Intel Core i5, 2,53 GHz
3. Χωρητικότητα Σκληρού Δίσκου: 10GB

6.2.2 Λογισμικό -Software

Στα δύο εικονικά συστήματα εγκαταστάθηκε Λειτουργικό Σύστημα Ubuntu 12.4.

Σε κάθε μηχάνημα εγκαταστάθηκε ο Apache web server και υλοποιήθηκε ένα σύστημα Online εγγραφής και αυθεντικοποίησης χρηστών.

Αναλυτικότερα, εγκαταστάθηκαν τα ακόλουθα χαρακτηριστικά, απαραίτητα για την υλοποίηση ενός συστήματος online εγγραφής και αυθεντικοποίησης:

1. Apache Server HTTP Server 2.2, για τη χρήση του ως διακομιστής του συστήματος.
2. PHP v.5.5, για την υλοποίηση της φόρμας εγγραφής, εισόδου και ενδιάμεσων καταστάσεων (συνδεδεμένος χρήστης, αποσύνδεση κ.τ.λ.)
3. PHP MyAdmin, για την διαχείριση των δεδομένων της βάσης δεδομένων στην οποία αποθηκεύονται τα στοιχεία των χρηστών του συστήματος)
4. MYSQL database, για την αποθήκευση των δεδομένων του συστήματος (στοιχεία χρηστών κτλ.)

6.2.3 Λειτουργίες Υλοποίησης

Το συγκεκριμένο σύστημα υλοποιήθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού php 5.5 σε συνδυασμό με τη HTTP.

Επιγραμματικά οι βασικές λειτουργίες που προσφέρει η παρούσα υλοποίηση είναι:

1. Online Εγγραφή νέου χρήστη
2. Αποθήκευση στη βάση δεδομένων MYSQL των ακόλουθων στοιχείων ανά χρήστη:
    - Όνομα Χρήστη - ΄Name’
    - Επώνυμο Χρήστη – ‘Surname’
    - Διεύθυνση Ηλεκτρονικής Αλληλογραφίας – ‘Email’
    - Συνθηματικό Χρήστη – ‘Username’
    - Κωδικός Πρόσβασης Χρήστη – ‘Password’
    - Ταυτότητα Χρήστη – ‘ User_ID’
3. Online Αυθεντικοποίησης ήδη εγγεγραμμένου χρήστη

6.2.4 Βασικά Στοιχεία Υλοποίησης

Η υλοποίηση αποτελείται από τα ακόλουθα βασικά στοιχεία:

1. Μία σελίδα Εγγραφής Χρήστη
2. Μια σελίδα Εισόδου Χρήστη
3. Μία βάση δεδομένων που περιέχει τα στοιχεία των χρηστών

Για τη δημιουργία των συγκεκριμένων εφαρμογών, χρειάστηκε να δημιουργηθούν κάποιες βασικές λειτουργίες, οι οποίες περιέχονται στα αντίστοιχα κύρια αρχεία:

1. ‘login.php’: όπου περιέχει τον κώδικα για τον έλεγχο των στοιχείων του χρήστη.
   Η διαδικασία που ακολουθείται είναι η εξής:
   Ο χρήστης εισάγει τον λογαριασμό ηλεκτρονικού ταχυδρομείου καθώς και το όνομα χρήστη με το οποίο έχει εγγραφεί στο σύστημα. Έπειτα τα στοιχεία ελέγχονται με τα αντίστοιχα στοιχεία που βρίσκονται αποθηκευμένα στη βάση δεδομένων. Σε περίπτωση μη αντιστοίχησης των εισαγόμενων στοιχείων με τα αντίστοιχα αποθηκευμένα, εμφανίζεται μήνυμα σφάλματος ενημερώνοντας τον χρήστη για την μη αντιστοίχηση. Στην περίπτωση που δεν έχουν εισαχθεί από τον χρήστη το όνομα χρήστη ή η διεύθυνση ηλεκτρονικής αλληλογραφίας, εμφανίζονται μηνύματα σφάλματος ενημερώνοντας τον χρήστη για την ελλιπή εισαγωγή των στοιχείων του.

2. ‘logout.php’: όπου δίνεται η δυνατότητα εξόδου από το σύστημα σε ένα ήδη επιθετικοποιημένο χρήστη και τερματισμός της συνεδρίας στο σύστημα.
3. ‘register.php’: όπου περιέχεται η σελίδα εγγραφής νέου χρήστη. Αναλυτικά η διαδικασία που ακολουθείται:
   Ο χρήστης εισέρχεται στη σελίδα και εισάγει τα ακόλουθα στοιχεία:
   - Όνομα χρήστη – ‘first_name’
   - Επώνυμο χρήστη –‘last_name’
   - Διεύθυνση ηλεκτρονικού ταχυδρομείου – ‘email’
   - Κωδικός πρόσβασης – ‘password’
   - Επιβεβαίωση κωδικού πρόσβασης – ‘Confirm password’

   Εν συνεχεία, ελέγχεται εάν έχουν συμπληρωθεί όλα τα απαιτούμενα πεδία στη φόρμα εγγραφής. Σε περίπτωση που δεν έχουν συμπληρωθεί όλα τα στοιχεία, εμφανίζεται μήνυμα λάθους το οποίο ενημερώνει το χρήστη για την έλλειψη στοιχείων ανά πεδίο. Εν συνεχεία η σελίδα συνδέεται με τη βάση δεδομένων και αποθηκεύει τα στοιχεία του χρήστη στην αντίστοιχη στήλη του πίνακα ‘users’ το καθένα.
4. ‘mysqli_connect’: το συγκεκριμένο αρχείο χρησιμοποιείται για την σύνδεση της φόρμας με τη βάση δεδομένων.
5. ‘login_functions.inc.php’: το συγκεκριμένο αρχείο περιέχει συναρτήσεις – δομές για την εκτέλεση του κώδικα αυθεντικοποίησης χρήστη – login.php
6. ‘login_page.inc.php’: περιέχει τη δομή της φόρμας αυθεντικοποίησης – εισόδου χρήστη.

Για τον αναλυτικό κώδικα αναφορικά με τη συνάρτηση Bcrypt και τα αρχεία που χρησιμοποιήθηκαν παρακαλώ αναφερθείτε στο συμπληρωματικό παράρτημα Α.

6.2.5 Βασική Διαφορά των δύο εικονικών μηχανών

Η βασική διαφορά των δύο εικονικών μηχανών προσδιορίζεται στο πεδίο αποθήκευσης του κωδικού πρόσβασης. Στην μία εικονική μηχανή, ο εισαγόμενος κωδικός πρόσβασης αποθηκεύεται στην αντίστοιχη στήλη του πίνακα στη βάση δεδομένων με τη μορφή όπως έχει αυτή επεξεργαστεί βάσει του αλγορίθμου SHA1.

Στην δεύτερη εικονική μηχανή, ο εισαγόμενος κωδικός πρόσβασης αποθηκεύεται στην αντίστοιχη στήλη του πίνακα στη βάση δεδομένων με τη μορφή όπως έχει αυτή επεξεργαστεί βάσει της συνάρτησης Bcrypt.

Οι προαναφερθέντες αλγόριθμοι χρησιμοποιούνται κατά την εγγραφή νέου χρήστη καθώς και κατά την αυθεντικοποίηση των στοιχείων του κατά την είσοδο του στο σύστημα, στα αντίστοιχες εικονικές μηχανές ο καθένας.

Ειδικότερα η διαδικασία που ακολουθείται είναι η ακόλουθη:

- Ο χρήστη εγγράφεται στη σελίδα Register
- Τα στοιχεία του αποθηκεύονται στη βάση δεδομένων. Ο κωδικός ασφαλείας που ορίζει ο χρήστης αποτελεί την είσοδο στην εκάστοτε συνάρτηση κατακερματισμού SHA & Bcrypt αντίστοιχα.
- Έπειτα ο χρήστης εισέρχεται στη σελίδα εισόδου ‘login page’
- Πραγματοποιείται έλεγχος των στοιχείων του χρήστη με τα ήδη αποθηκευμένα.

- Κατά τον έλεγχο που πραγματοποιείται, επιλέγονται τα στοιχεία από τις αντίστοιχες στήλες της βάσης δεδομένων βάσει της διεύθυνσης ηλεκτρονικού ταχυδρομείου που εισήγαγε ο χρήστης. Ειδικότερα, λαμβάνεται η διεύθυνση ηλεκτρονικού ταχυδρομείου και αναζητεί στη βάση τα στοιχεία του χρήστη με το αντίστοιχο email.
- Στην περίπτωση που δεν βρεθεί η διεύθυνση ηλεκτρονικού ταχυδρομείου στα δεδομένα που βρίσκονται στη βάση δεδομένων, ο χρήστης ενημερώνεται ότι δεν βρέθηκαν στοιχεία.
- Στην περίπτωση που δοθεί λανθασμένο όνομα χρήστη ή διεύθυνση ηλεκτρονικού ταχυδρομείου, ο χρήστης ενημερώνεται με αντίστοιχα μηνύματα σφάλματος.
- Στην περίπτωση που βρεθούν τα αντίστοιχα στοιχεία στην βάση, επιλέγεται ο το πεδίο το οποίο περιέχει τον κωδικό πρόσβασης.
- Ο αποθηκευμένος κωδικός πρόσβασης συγκρίνεται με τον αντίστοιχο που έχει εισαχθεί στην συγκεκριμένη συνεδρία από τον χρήστη. Στην περίπτωση όπου δεν ταυτίζονται οι δύο πληροφορίες εμφανίζεται μήνυμα λάθους, ενημερώνοντας τον χρήστη για την εσφαλμένη εισαγωγή του κωδικού πρόσβασης.
- Στην περίπτωση που τα δύο πεδία ταυτίζονται, πραγματοποιείται είσοδος του χρήστης στο σύστημα και εμφανίζεται μήνυμα επιτυχούς εισόδου.
- Επιπλέον δίνεται στο χρήστη η επιλογή αποχώρησης από το σύστημα –‘ logout’

Σημαντική πληροφορία

Η πρόκληση έγκειται στο γεγονός ότι απαιτείται διαφορετικός τρόπος υλοποίησης για την συνάρτηση κατακερματισμού SHA και την BCRYPT αντίστοιχα.

Στην υλοποίηση της συνάρτησης SHA ο κατακερματισμός της πληροφορίας πραγματοποιείται στο επίπεδο της βάσης δεδομένων. Αναλυτικότερα, ο κωδικός εισάγεται από το χρήστη στη φόρμα εγγραφής χρήστη σε μορφή ‘plaintext’. Έπειτα πραγματοποιείται σύνδεση με τη βάση δεδομένων και κατά την αποθήκευση του κωδικού στη βάση δεδομένων, εφαρμόζεται η συνάρτηση SHA. Στη βάση MYSQL δίνεται η δυνατότητα αποθήκευσης πληροφορίας με τη χρήστη της SHA συνάρτησης. Η δυνατότητα αυτή βρίσκεται ενσωματωμένη στις λειτουργίες που προσφέρει ο συγκεκριμένος τύπος βάσης δεδομένων

6.2.6 Συναρτήσεις PHP

Στην υλοποίηση της συνάρτησης Bcrypt, ο κατακερματισμός της πληροφορίας πραγματοποιείται σε επίπεδο ιστοσελίδας. Αναλυτικότερα, ο κωδικός εισάγεται από το χρήστη στη φόρμα εγγραφής χρήστη σε μορφή ‘plaintext’. Έπειτα χρησιμοποιείται η συνάρτηση Password_hash, με παράμετρο Password_Default. Οι συγκεκριμένες δύο συναρτήσεις προσφέρονται στη σουίτα της PHP, PHP API και χρησιμοποιούνται για τον κατακερματισμό πληροφορίας με νεότερες συναρτήσεις κατακερματισμού. Η συνάρτηση που υποστηρίζεται έως τώρα είναι η Bcrypt:

- Η συνάρτηση password_hash ορίζει την διαδικασία κατακερματισμού Παίρνει ως είσοδο την τιμή προς κατακερματισμό, και την συνάρτηση κατακερματισμού(Password_Default, Password_Bcrypt) [41]
- Η συνάρτηση Password_Default δημιουργήθηκε ως τη συνάρτηση την οποία μπορεί να ορίσει ο χρήστης ως την εξορισμού χρησιμοποιούμενη συνάρτηση κατακερματισμού. Προς το παρόν, η συνάρτηση που έχει υλοποιηθεί από την ομάδα υποστήριξης της γλώσσας

προγραμματισμού PHP είναι η Bcrypt. Η Ομάδα υποστήριξης έχει ενημερώσει στη σελίδα της θα ακολουθήσουν και άλλες συναρτήσεις κατακερματισμού [41]

- Η συνάρτηση Password_Bcrypt είναι η υλοποίηση σε PHP της συνάρτησης κατακερματισμού BCRYPT.[41]
- Επιπλέον δίνεται η δυνατότητα επιλογής με την δομή OPTIONs του κόστους και του άλας τα οποία θα εφαρμοστούν κατά την εφαρμογή της συνάρτησης BCRYPT. Με τον ορισμό του κόστους ορίζονται οι επαναλήψεις οι οποίες εφαρμόζονται κατά την εκτέλεση της Bcrypt.
- Έχει δημιουργηθεί μία συνάρτηση για την αυθεντικοποίηση του εισαγόμενου κωδικού με τον αποθηκευμένο-Bcrypt χωρίς την εγγραφή επιπλέον κώδικα, η password_verify [42]

Έπειτα πραγματοποιείται σύνδεση με τη βάση δεδομένων και κατά την αποθήκευση του κωδικού στη βάση δεδομένων, εφαρμόζεται η συνάρτηση SHA. Στη βάση MYSQL δίνεται η δυνατότητα αποθήκευσης πληροφορίας με τη χρήστη της SHA συνάρτησης. Η δυνατότητα αυτή βρίσκεται ενσωματωμένη στις λειτουργίες που προσφέρει ο συγκεκριμένος τύπος βάσης δεδομένων.

Ο κώδικας που χρησιμοποιήθηκε για την μέτρηση της καθυστέρησης που εισάγεται από τη χρήση της συνάρτησης BCRYPT στην αποθήκευση των κωδικών πρόσβασης στη βάση είναι η ακόλουθη συνάρτηση [39]:

$time_start= micortime(true);

$time_end = microtime(true);

$delay= $time_end - $time_start

Η συνάρτηση microtime είναι ενσωματωμένη στις βιβλιοθήκες της γλώσσας προγραμματισμού php. Η συγκεκριμένη συνάρτηση επιστρέφει την τρέχουσα χρονική στιγμή –Timestamp σε συστήματα UNIX. Η λογική που ακολουθήθηκε ήταν εκτέλεση της συνάρτησης microtime στην αρχή εκτέλεσης του κώδικα ελέγχου στοιχείων του χρήστη και μετά το πέρας της εκτέλεσης του συγκεκριμένου κώδικα. Ο υπολογισμός της χρονικής διάρκειας της εκτέλεσης ορίζεται ως την αφαίρεση της τιμής της αρχικής τιμής από την τελική τιμή.

Η καθυστέρηση που εισάγεται ορίζεται ως την αφαίρεση του χρόνου συνολικής εκτέλεσης του κώδικα για την υλοποίηση SHA1 από τον χρόνο συνολικής εκτέλεσης του κώδικα για την υλοποίηση Bcrypt.

Ο χρόνος συνολικής εκτέλεσης του ελέγχου στοιχείων του χρήστη κατά την υλοποίηση Bcrypt αλλάζει με τον ορισμό της παραμέτρου κόστους –cost με διαφορετικές τιμές.

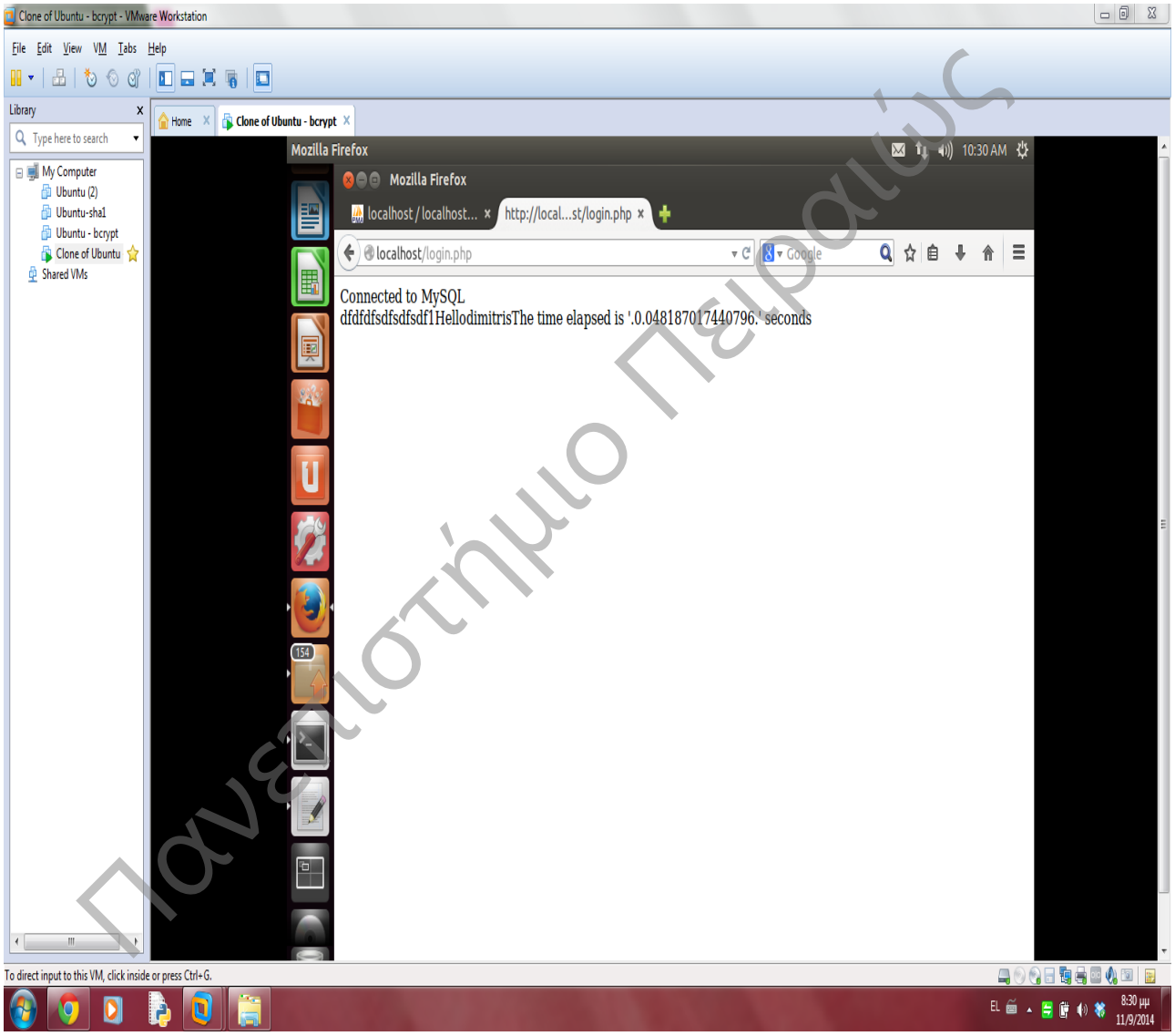
7. Πειράματα και Αποτελέσματα

Παρακάτω παρατίθενται τα σενάρια που εκτελέστηκαν καθώς και τα αντίστοιχα αποτελέσματα.

7.1 Σενάρια Εκτέλεσης Πειραμάτων – Bcrypt

7.1.1 Σενάριο 1

- Μήκος κωδικού πρόσβασης: 8 χαρακτήρες
- Κόστος επαναλήψεων: 7
- Χρόνος εκτέλεσης: 0.0481 δευτερόλεπτα



Εικόνα 10: Αποτέλεσμα Σεναρίου 1

7.1.2 Σενάριο 2

- Μήκος κωδικού πρόσβασης: 8 χαρακτήρες
- Κόστος επαναλήψεων: 10
- Χρόνος εκτέλεσης: 0.2350 δευτερόλεπτα

7.1.3 Σενάριο 3

- Μήκος κωδικού πρόσβασης: 8 χαρακτήρες
- Κόστος επαναλήψεων: 15
- Χρόνος εκτέλεσης: 5.0159 δευτερόλεπτα

7.1.4 Σενάριο 4

- Μήκος κωδικού πρόσβασης: 10 χαρακτήρες
- Κόστος επαναλήψεων: 07
- Χρόνος εκτέλεσης: 0.0525 δευτερόλεπτα

7.1.5 Σενάριο 5

- Μήκος κωδικού πρόσβασης: 10 χαρακτήρες
- Κόστος επαναλήψεων: 10
- Χρόνος εκτέλεσης: 0.2536 δευτερόλεπτα

7.1.6 Σενάριο 6

- Μήκος κωδικού πρόσβασης: 10 χαρακτήρες
- Κόστος επαναλήψεων: 15
- Χρόνος εκτέλεσης: 5,7068 δευτερόλεπτα

7.1.7 Σενάριο 7

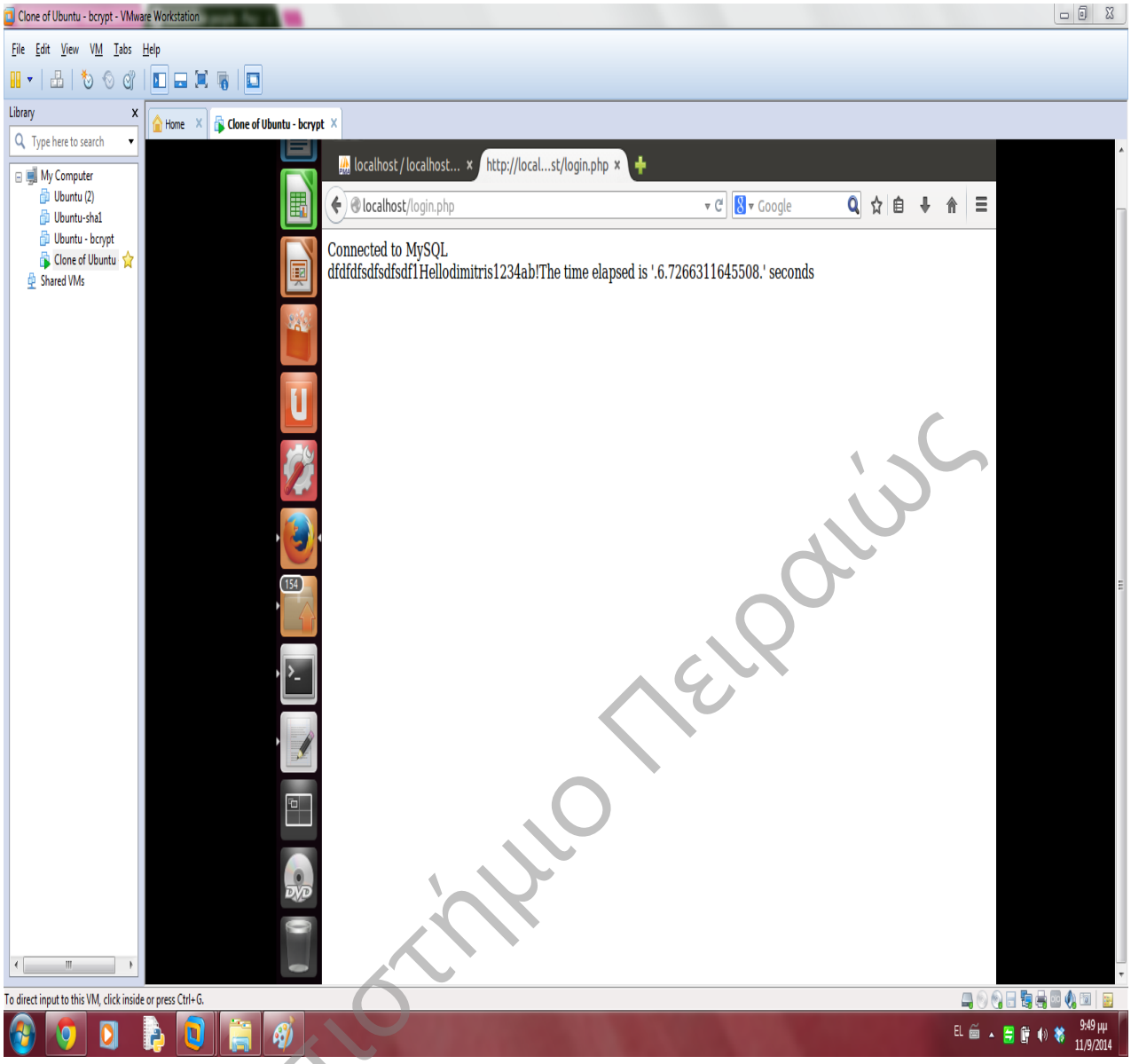
- Μήκος κωδικού πρόσβασης: 15 χαρακτήρες
- Κόστος επαναλήψεων: 7
- Χρόνος εκτέλεσης: 0.0625 δευτερόλεπτα

7.1.8 Σενάριο 8

- Μήκος κωδικού πρόσβασης: 15 χαρακτήρες
- Κόστος επαναλήψεων: 10
- Χρόνος εκτέλεσης: 0.2571 δευτερόλεπτα

7.1.9 Σενάριο 9

- Μήκος κωδικού πρόσβασης: 15 χαρακτήρες
- Κόστος επαναλήψεων: 15
- Χρόνος εκτέλεσης: 6.7266 δευτερόλεπτα

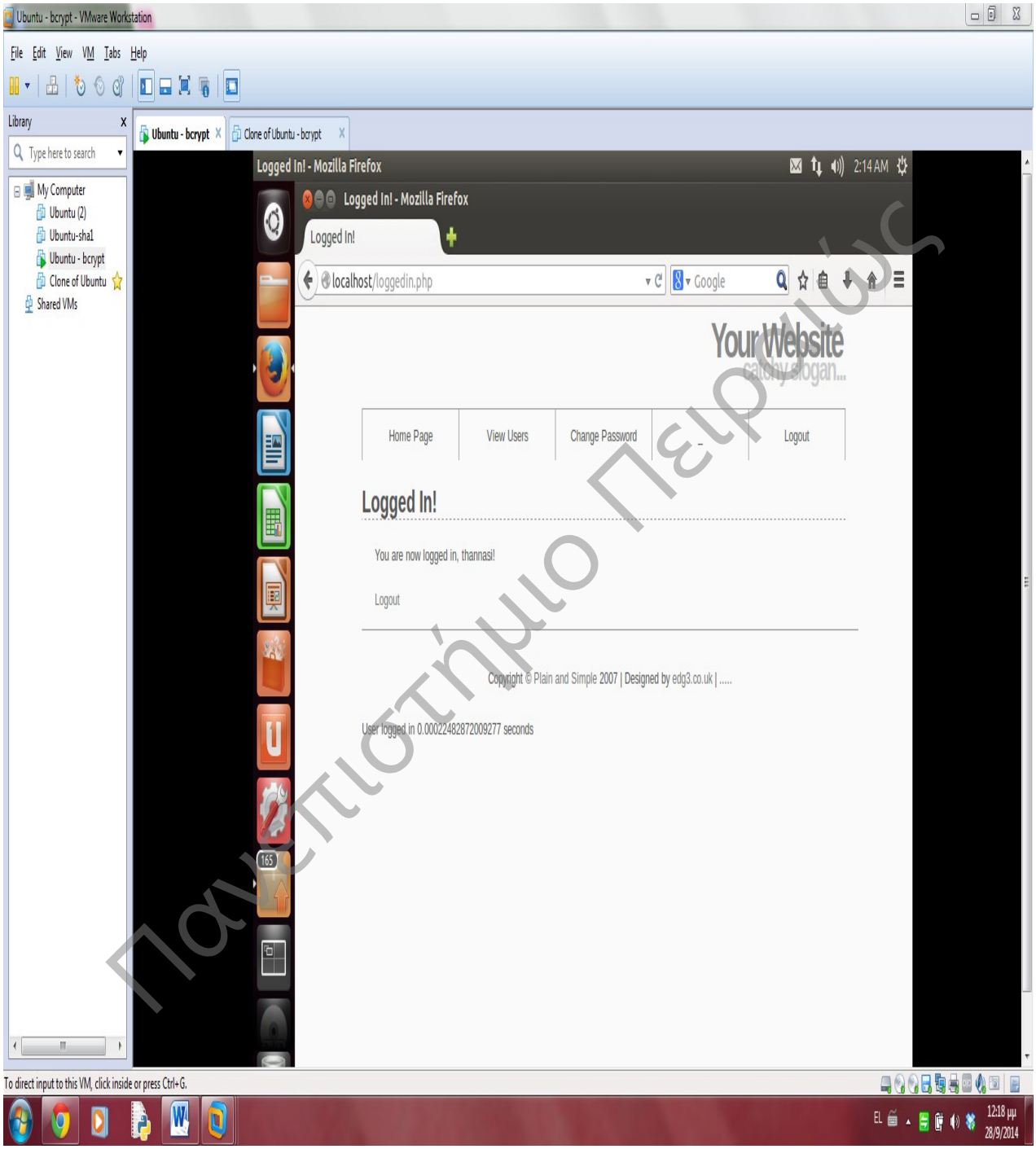


Εικόνα 11: Αποτέλεσμα Σεναρίου 9

7.2 Σενάρια Εκτέλεσης Πειραμάτων – SHA1

7.2.1 Σενάριο 10

- Μήκος κωδικού πρόσβασης: 8
- Χρόνος Εκτέλεσης: 0.00022 δευτερόλεπτα



Εικόνα 12: Αποτέλεσμα Σεναρίου 10

7.2.2 Σενάριο 11

- Μήκος κωδικού πρόσβασης: 10
- Χρόνος Εκτέλεσης: 0.00023 δευτερόλεπτα

7.2.3 Σενάριο 12

- Μήκος κωδικού πρόσβασης: 15
- Χρόνος Εκτέλεσης: 0.00024 δευτερόλεπτα

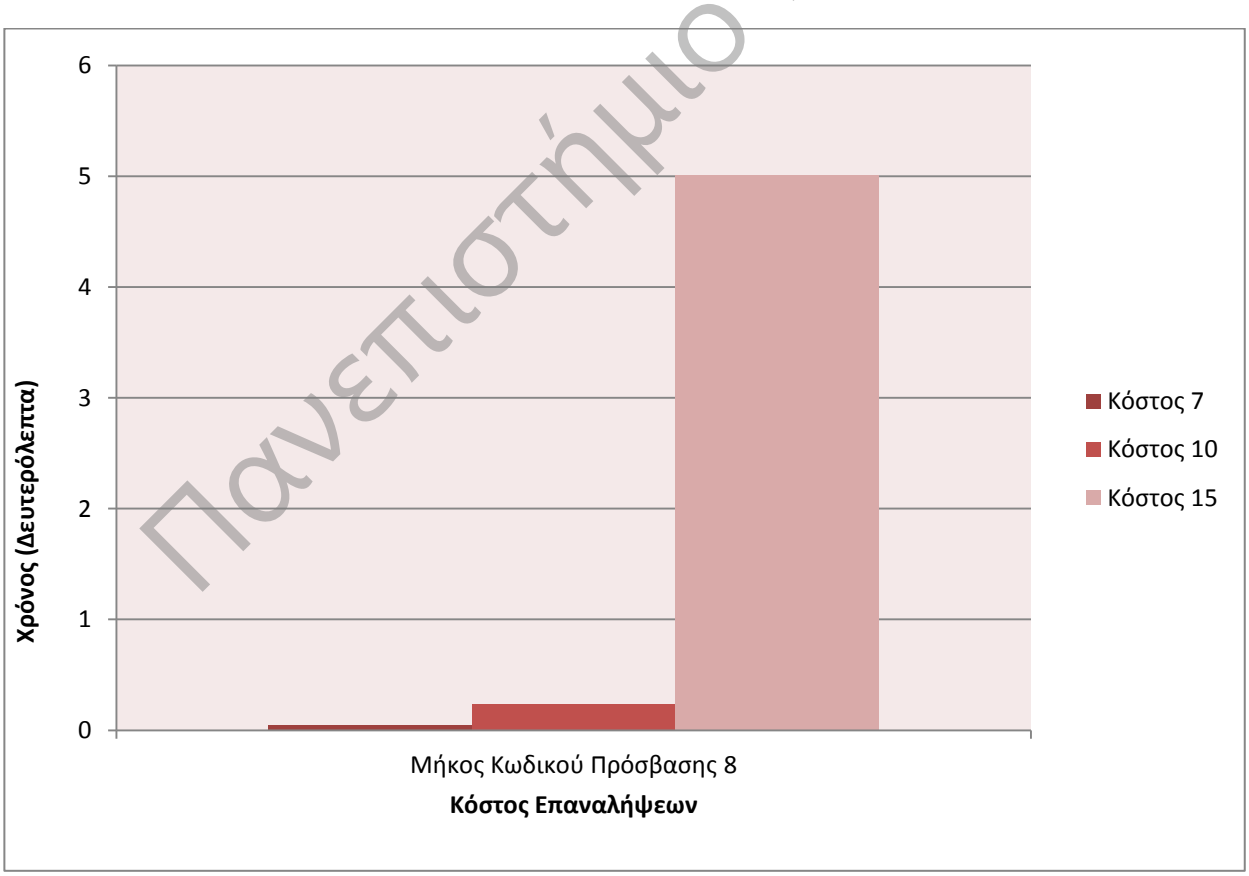
7.3 Ανάλυση και Αποτελέσματα

7.3.1 Σύγκριση Εισαγόμενης Καθυστέρησης Κόστους ανά Μήκος Κωδικού Πρόσβασης - Bcrypt

7.3.1.i Μήκος κωδικού Πρόσβασης: 8 ψηφία – Κόστη:7, 10 & 15

Πίνακας 1: Αναλυτικά Αποτελέσματα Σεναρίων 1, 2 & 3

α/α	Κόστος	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	7	0.0481	0,1869
Β	10	0.2350	
Γ	15	5.0159	4,7809

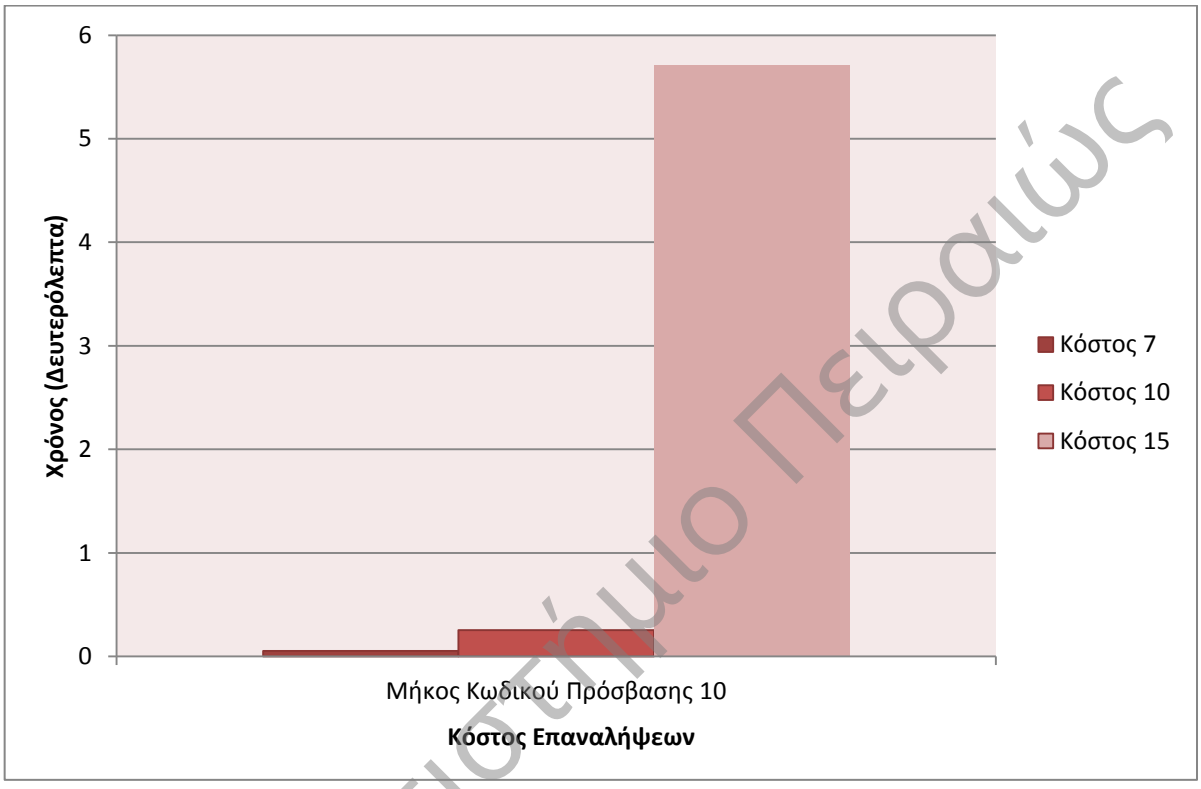


Γράφημα 1: Σχηματική Απεικόνιση Σεναρίων 1,2,3

7.3.1.ii Μήκος κωδικού Πρόσβασης: 10 ψηφία – Κόστη:7, 10 & 15

Πίνακας 2: Αναλυτικά Αποτελέσματα Σεναρίων 4, 5 &6

α/α	Κόστος	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	7	0.0525	0,2011
Β	10	0.2536	
Γ	15	5,7068	5,4532

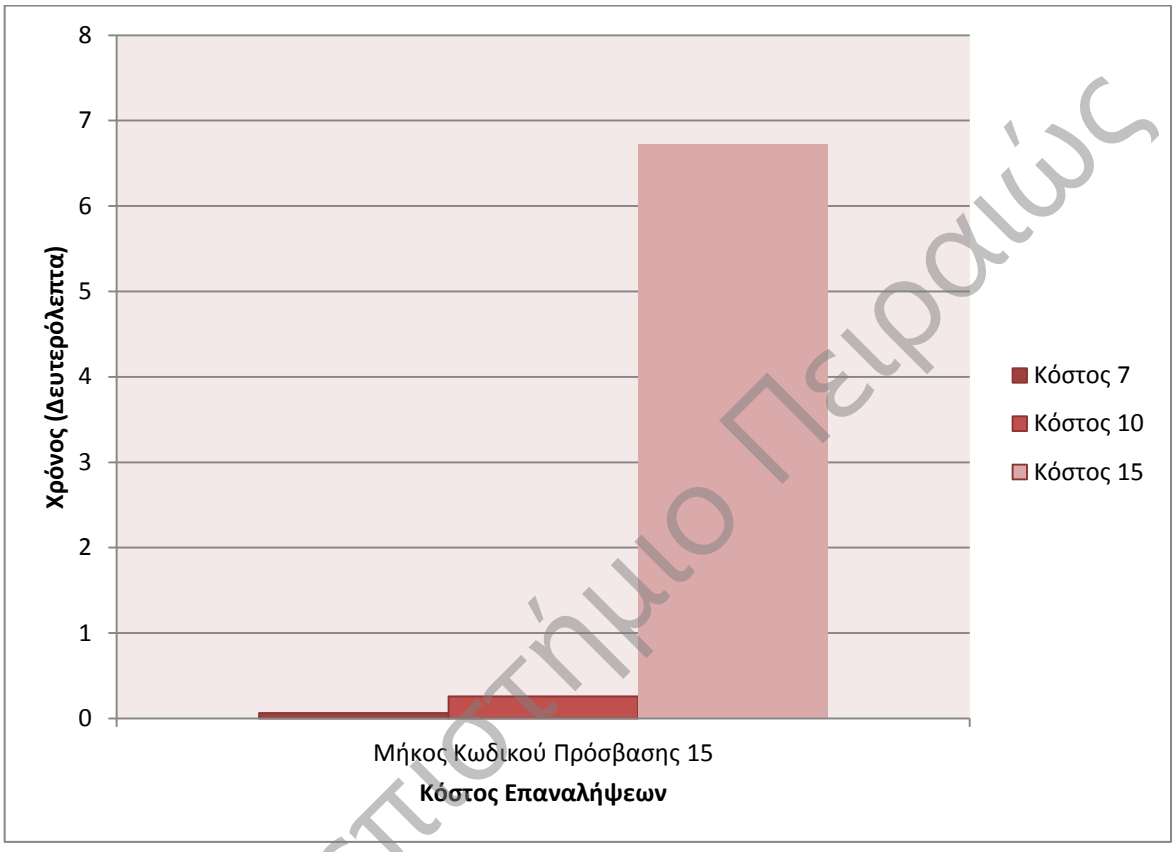


Γράφημα 2: Σχηματική Απεικόνιση Σεναρίων 4,5,6

7.3.1.ii Μήκος κωδικού Πρόσβασης: 15 ψηφία – Κόστη:7, 10 & 15

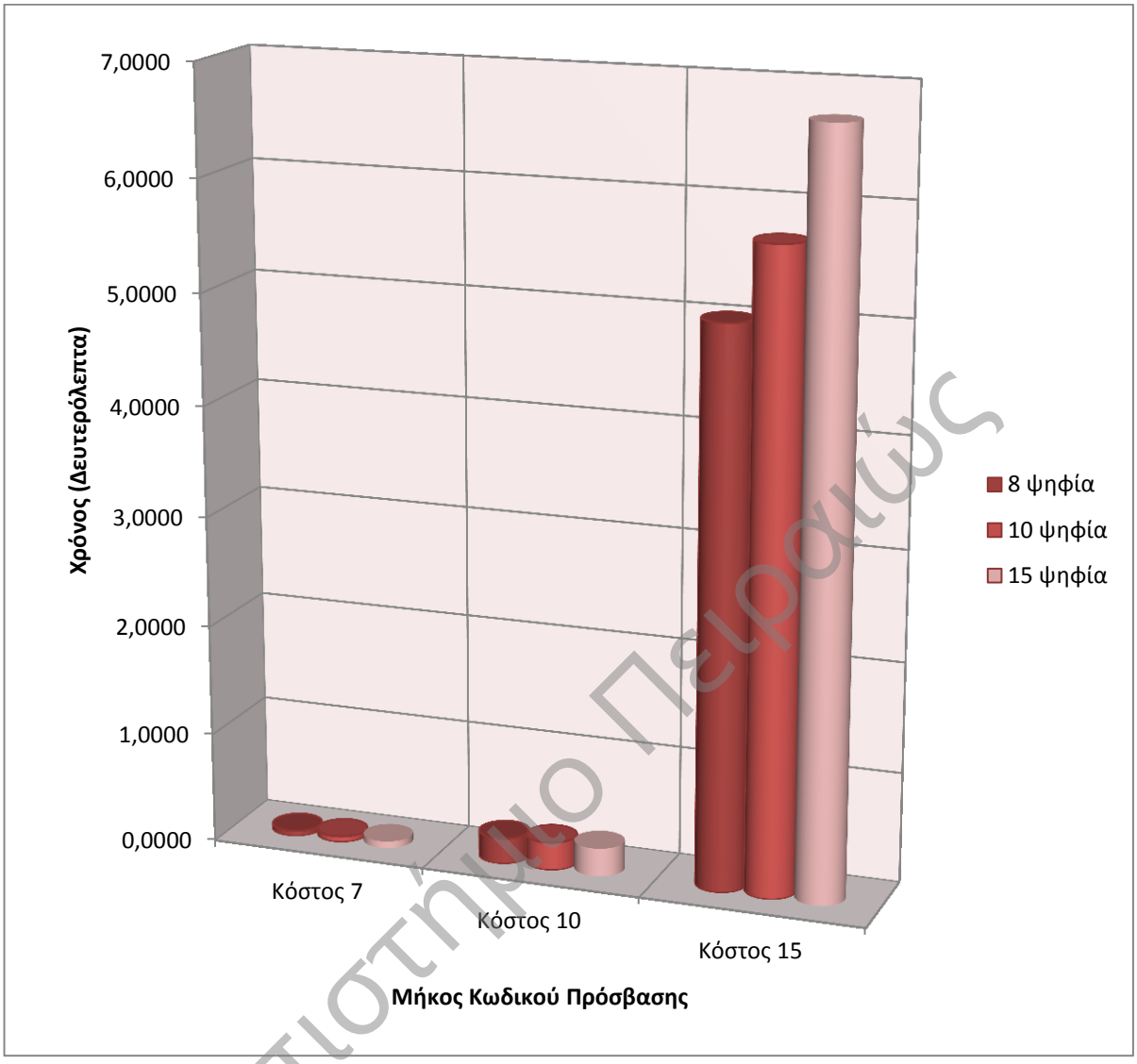
Πίνακας 3: Αναλυτικά Αποτελέσματα Σεναρίων 7, 8 & 9

α/α	Κόστος	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	7	0.0625	0,1956
Β	10	0.2571	
Γ	15	6.7266	6,4695



Γράφημα 3: Σχηματική Απεικόνιση Σεναρίων 7, 8, 9

7.3.2 Συγκεντρωτικά Αποτελέσματα Σεναρίων 1-9 /1η Απεικόνιση



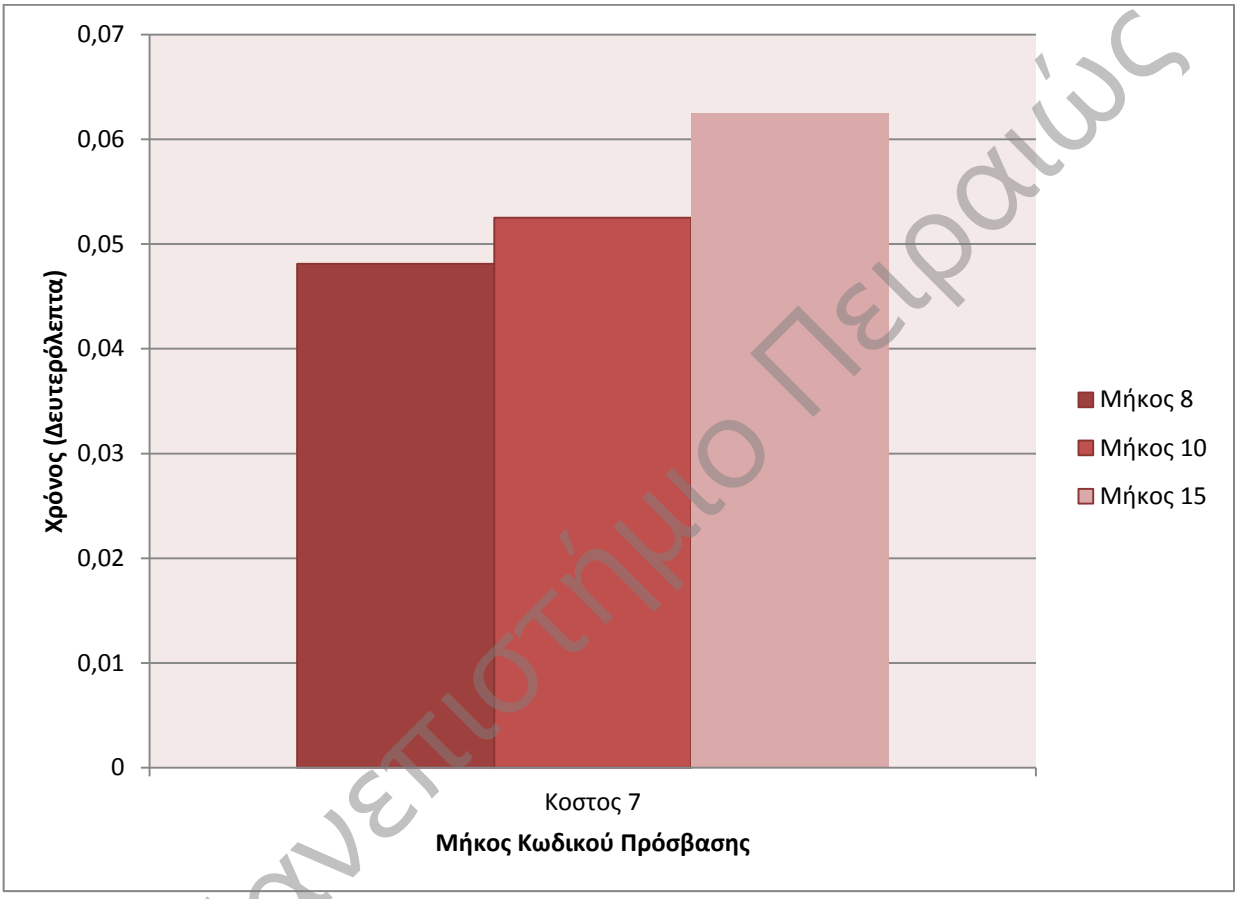
Γράφημα 4: Σχηματική Απεικόνιση Σεναρίων 1-9

7.3.3 Σύγκριση Εισαγόμενης Καθυστέρησης Μήκους Κωδικού Πρόσβασης ανά Κόστος

7.3.3.i Κόστος:7 – Μήκη Κωδικών Πρόσβασης: 8,10 & 15

Πίνακας 4: Αναλυτικά Αποτελέσματα Σεναρίων 1, 4 & 7

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.0481	0,0044
Β	10	0.0525	
Γ	15	0.0625	0,01

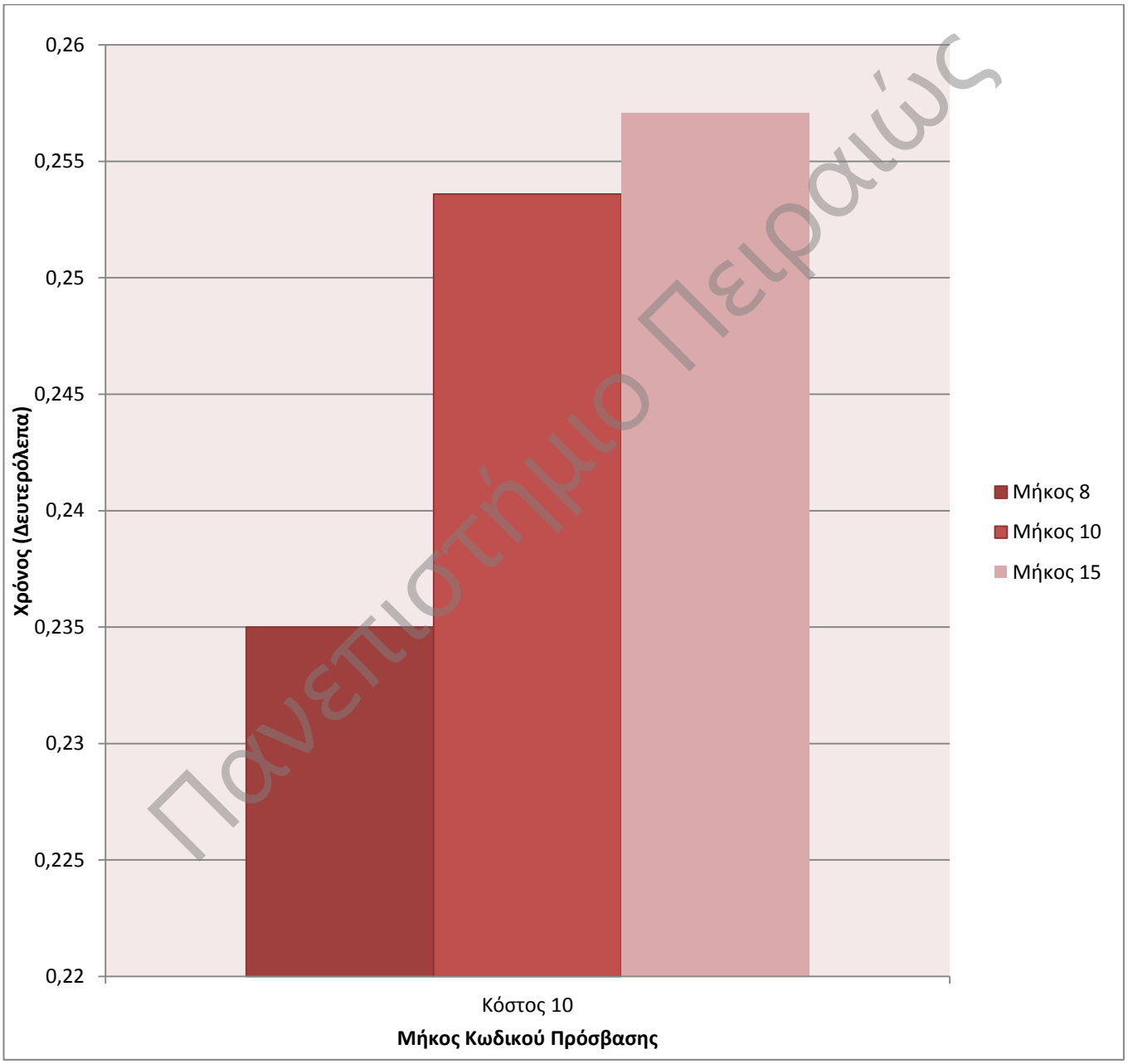


Γράφημα 5: Σχηματική Απεικόνιση Σεναρίων 1,4,7

7.3.3.ii Κόστος:10 – Μήκη Κωδικών Πρόσβασης: 8,10 & 15

Πίνακας 5: Αναλυτικά Αποτελέσματα Σεναρίων 2, 5 & 8

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.2350	0,0186
Β	10	0.2536	
Γ	15	0.2571	0,0035

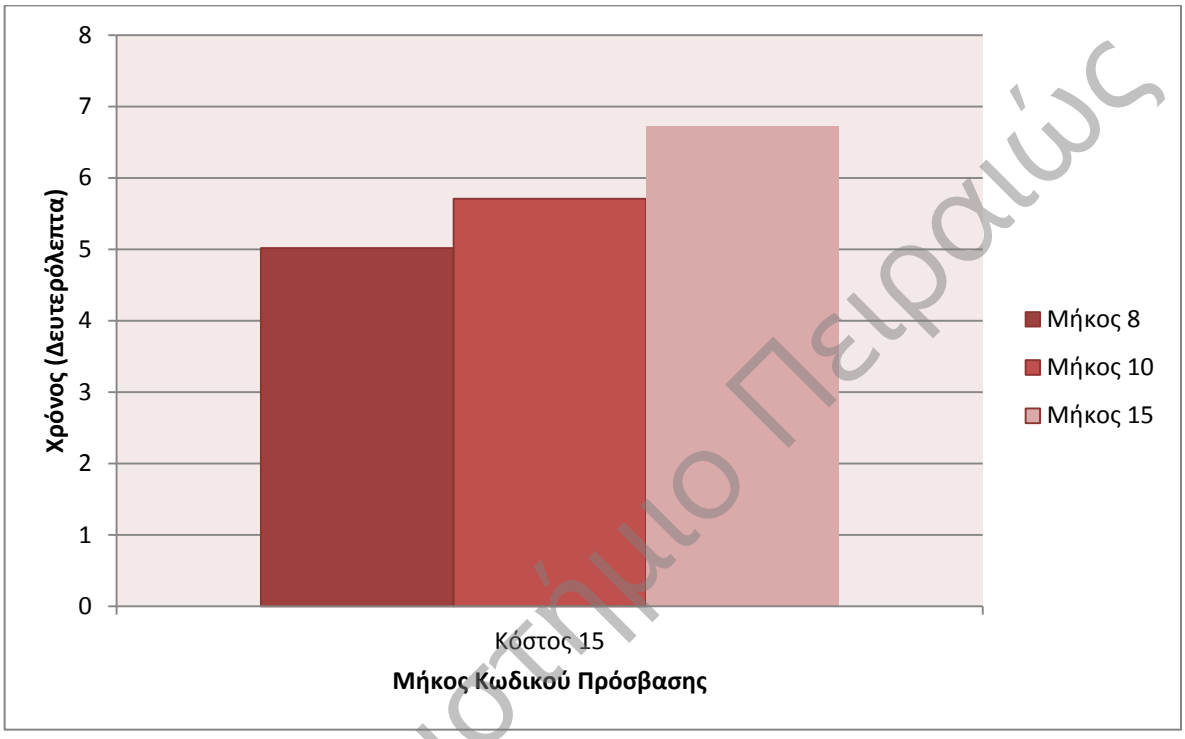


Γράφημα 6 Σχηματική Απεικόνιση Σεναρίων 2, 5, 8

7.3.3.iii Κόστος:15 – Μήκη Κωδικών Πρόσβασης: 8,10 & 15

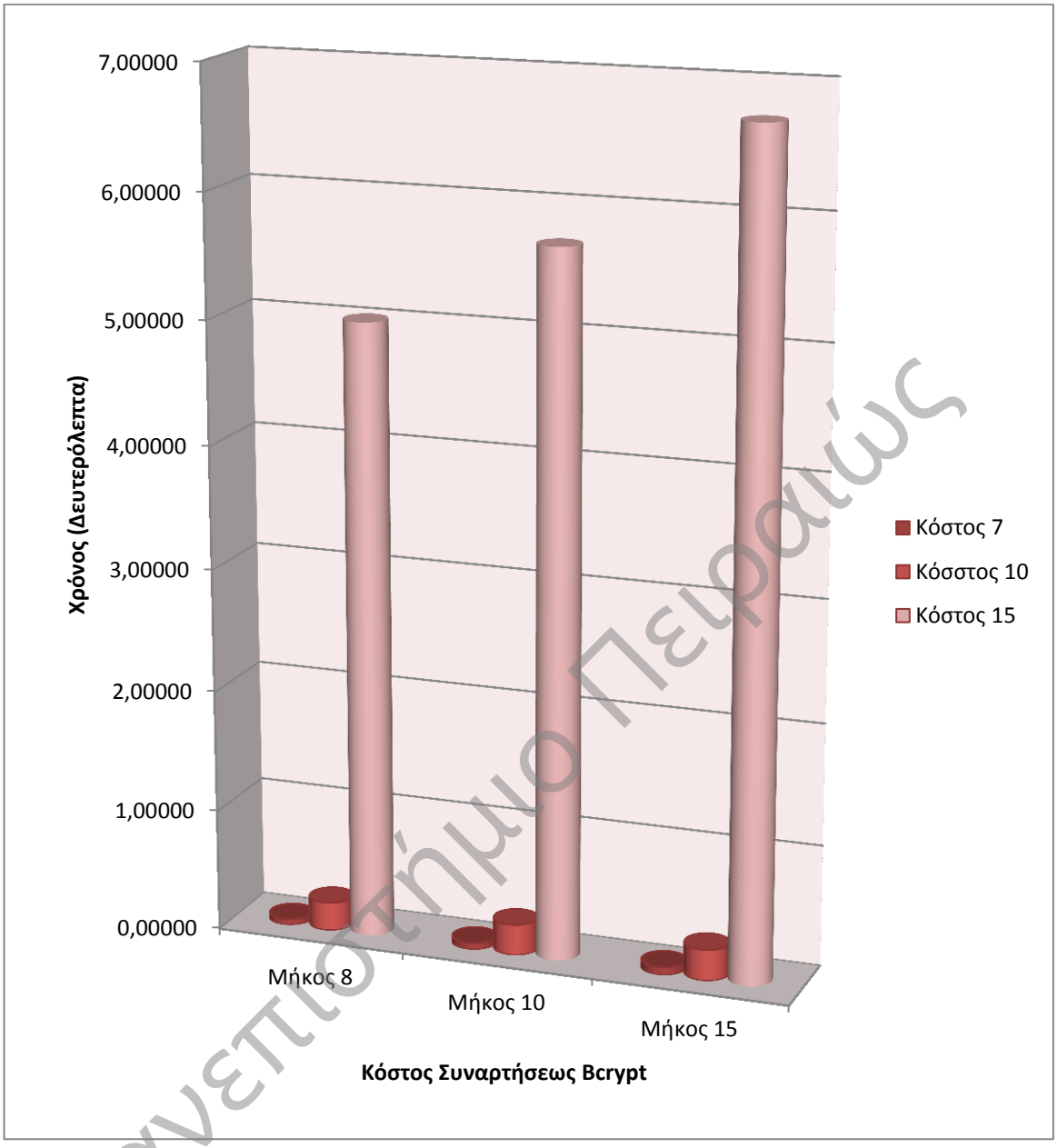
Πίνακας 6: Αναλυτικά Αποτελέσματα Σεναρίων 3, 6 & 9

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	5.0159	0,6909
Β	10	5,7068	
Γ	15	6.7266	1,0198



Γράφημα 7 Σχηματική Απεικόνιση Σεναρίων 3, 6, 9

7.3.4 Συγκεντρωτικά Αποτελέσματα Σεναρίων 1-9 /2η Απεικόνιση



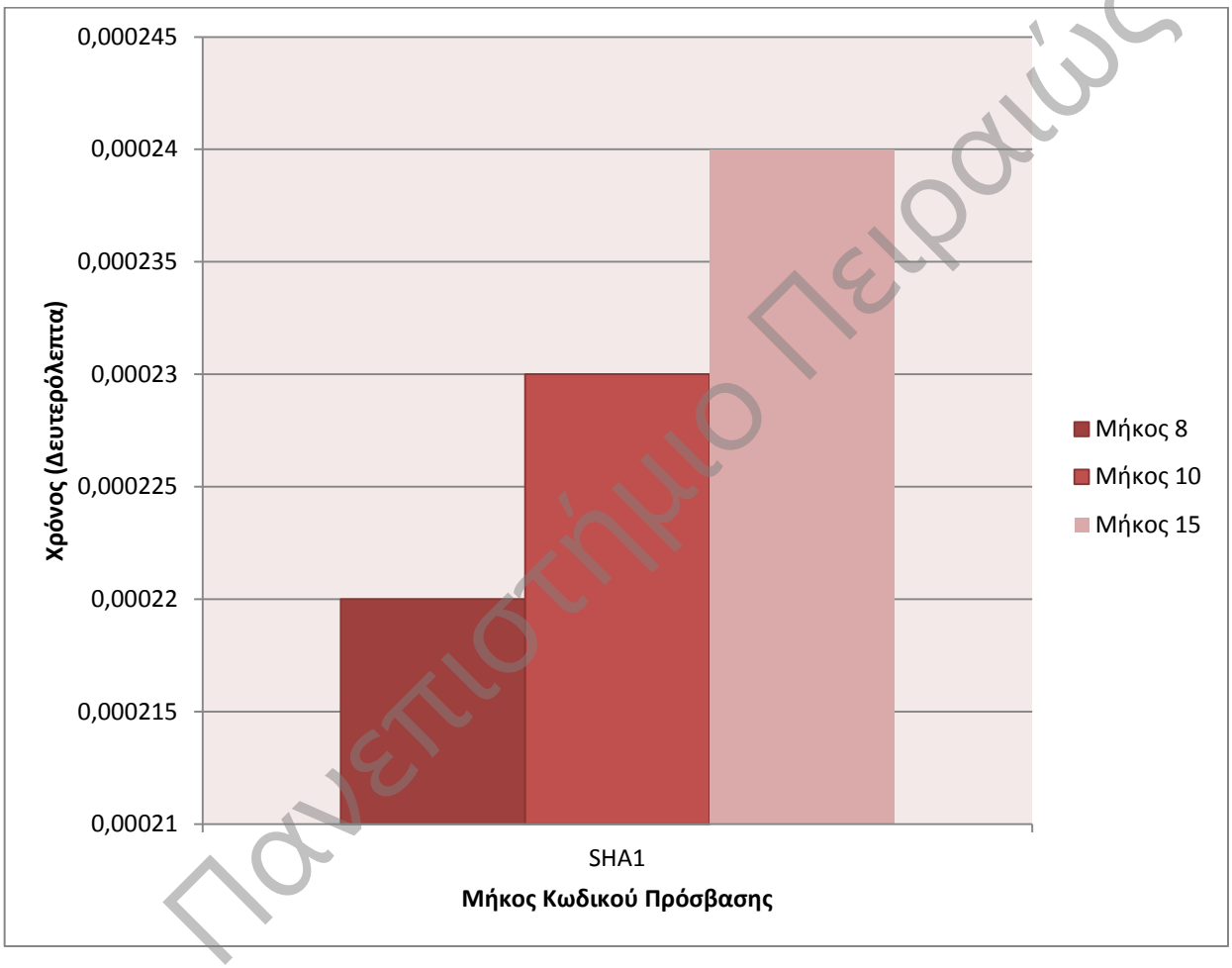
Γράφημα 8 Σχηματική Απεικόνιση Σεναρίων 1-9

7.3.5 Σύγκριση Εισαγόμενης Καθυστέρησης Μήκους Κωδικού Πρόσβασης SHA1

7.3.5.i Κωδικών Πρόσβασης: 8,10 & 15

Πίνακας 7: Αναλυτικά Αποτελέσματα Σεναρίων 10, 11 & 12

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.00022	0,00001
Β	10	0.00023	
Γ	15	0.00024	0,00001



Γράφημα 9: Σχηματική Απεικόνιση Σεναρίων 10, 11, 12

7.3.6 Σύγκριση εισαγόμενης καθυστέρησης ανά μήκος κωδικού πρόσβασης σε αντιπαραβολή με αποτελέσματα SHA1

Πίνακας 8: Σύγκριση Αποτελεσμάτων Σεναρίων 1,4,7, 10, 11 & 12

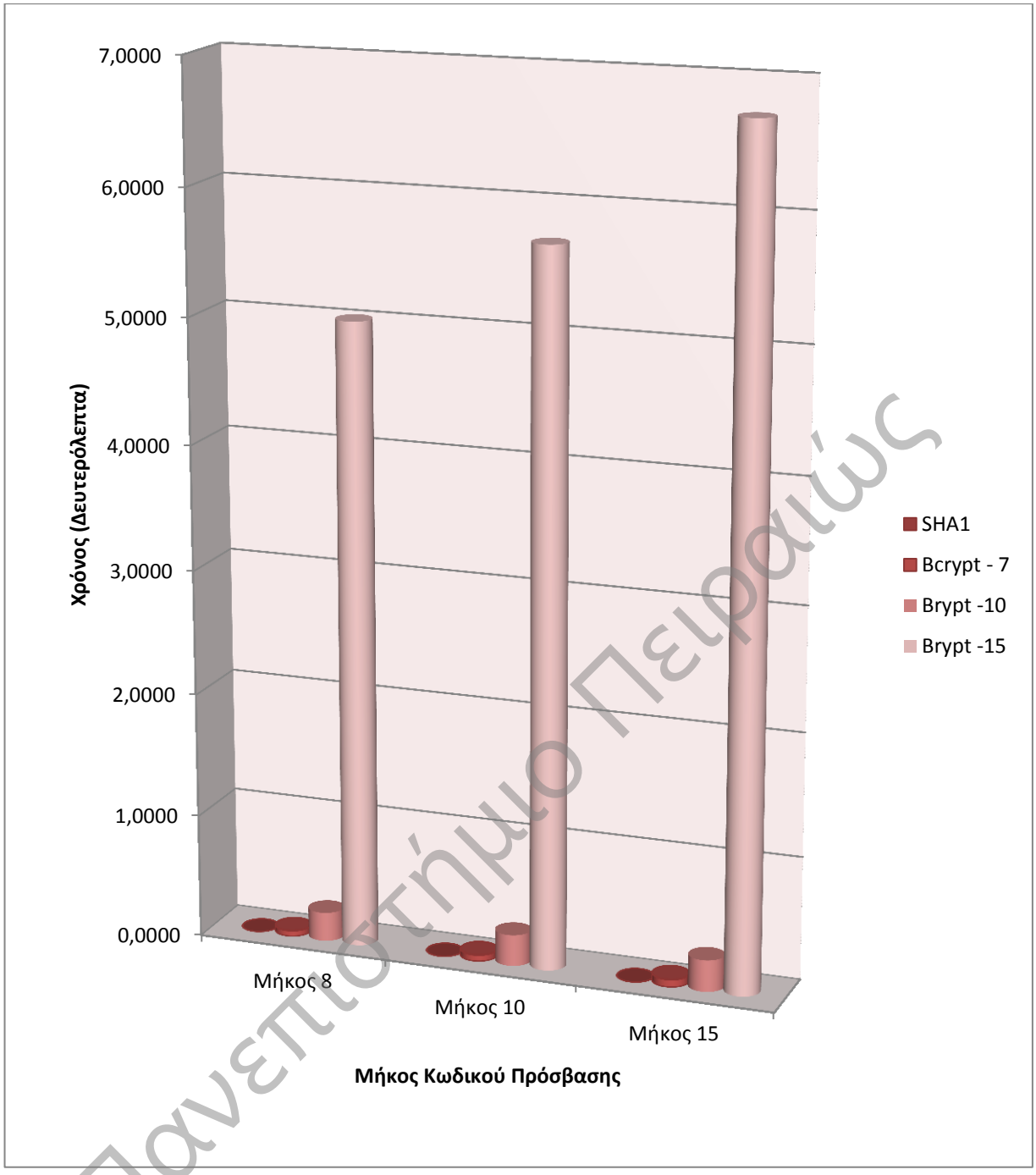
Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης - SHA1 (δευτερόλεπτα)	Χρονική Διάρκεια Εκτέλεσης - Bcrypt Κόστος 7 (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.00022	0.0481	0,0478
Β	10	0.00023	0.0525	0,0522
Γ	15	0.00024	0.0625	0,0622

Πίνακας 9: Σύγκριση Αποτελεσμάτων Σεναρίων 2,5,8, 10, 11 & 12

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης - SHA1 (δευτερόλεπτα)	Χρονική Διάρκεια Εκτέλεσης - Bcrypt Κόστος 10 (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.00022	0.2350	0,2347
Β	10	0.00023	0.2536	0,2533
Γ	15	0.00024	0.2571	0,2568

Πίνακας 9: Σύγκριση Αποτελεσμάτων Σεναρίων 3,6,9, 10, 11 & 12

Α/Α	Μήκος Κωδικού Πρόσβασης	Χρονική Διάρκεια Εκτέλεσης - SHA1 (δευτερόλεπτα)	Χρονική Διάρκεια Εκτέλεσης - Bcrypt Κόστος 15 (δευτερόλεπτα)	Διαφορά χρονικής διάρκειας (Β-Α & Γ-Β, δευτερόλεπτα)
Α	8	0.00022	5.0159	5,01568
Β	10	0.00023	5,7068	5,7065
Γ	15	0.00024	6.7266	6,7263



Γράφημα 10: Απεικόνιση Σεναρίων 1-12

8. Συμπεράσματα

8.1.1 Συμπεράσματα σύγκρισης μεγεθών χρονικής καθυστέρησης

Από την ανάλυση που παρατίθεται παραπάνω καταλήγουμε στα ακόλουθα συμπεράσματα

I. Η χρονική καθυστέρηση που εισάγεται με την αλλαγή του κόστους για σταθερό μήκος κωδικού πρόσβασης αυξάνεται με την αύξηση του κόστους
II. Η μεγαλύτερη χρονική καθυστέρηση εισάγεται για κόστος επαναλήψεων 15
III. Η χρονική καθυστέρηση που εισάγεται με την αλλαγή μήκους κωδικού πρόσβασης για σταθερό κόστος αυξάνεται με την αύξηση του μήκους κωδικού πρόσβασης
IV. Η μεγαλύτερη χρονική καθυστέρηση εισάγεται για μήκος κωδικού πρόσβασης 15
V. Το μέγεθος της χρονικής καθυστέρηση που εισάγεται κατά την αύξηση του κόστους επαναλήψεων σε σχέση με την αντίστοιχη που εισάγεται κατά την αύξηση του μήκους κωδικού πρόσβασης είναι κατά πολύ μεγαλύτερο. Τα αντίστοιχα μεγέθη κυμαίνονται σε τιμές 0,060-6,47 sec για την χρονική καθυστέρηση αύξησης κόστους και τιμές 0,004 - 1,001 sec για την χρονική καθυστέρηση αύξησης μήκους κωδικού πρόσβασης.

8.1.2 Συμπεράσματα σύγκρισης μεγεθών χρονικής καθυστέρησης Bcrypt με SHA1

Από την ανάλυση των δεδομένων που παρατίθεται παραπάνω καταλήγουμε στα ακόλουθα συμπεράσματα:

I. Απαιτείται ελάχιστος χρόνος για την εισαγωγή του χρήστη στην εφαρμογή
II. Η εισαγόμενη καθυστέρηση με την εφαρμογή του αλγορίθμου Bcrypt σε σύγκριση με την εκτέλεση της αντίστοιχης λειτουργίας με τη χρήση του αλγορίθμου SHA1 είναι σχετικά μεγάλη.

8.1.3 Βέλτιστος Συνδυασμός κόστους επαναλήψεων με μήκος κωδικού πρόσβασης

Βέλτιστη επιλογή αναλογίας μήκους κωδικού πρόσβασης, κόστους επαναλήψεων και εισαγόμενης χρονικής καθυστέρησης αποτελεί ο συνδυασμός κόστους 10 επαναλήψεων και μήκους κωδικού πρόσβασης 10 ψηφίων. Αναλυτικότερα:

- Ο συγκεκριμένος συνδυασμός δεν ξεπερνά σε χρονική διάρκεια εκτέλεσης τα 0,25 δευτερόλεπτα, χρονική διάρκεια η οποία περνά απαρατήρητη στο χρήστη της εφαρμογής.
- Επιπλέον με τον ορισμό του κόστους σε 10, δεν αυξάνεται ο υπολογιστικός φόρτος για κοινή εγκατάσταση υλικού. Μπορεί να εφαρμοσθεί σε υλικό ευρείας κυκλοφορίας με απλά γενικά χαρακτηριστικά (π.χ. 4GB RAM) χωρίς να εισάγεται καθυστέρηση αισθητή στον χρήστη.
- Η τιμή 10 στην παράμετρο cost της συνάρτησης Bcrypt, ορίζει ένα καλό επίπεδο δυσκολίας σε επιθέσεις εύρεσης κωδικών πρόσβασης. Επιπλέον ορίζει μια χρονική καθυστέρηση η οποία αυξάνει τους υπολογιστικούς πόρους καθώς και το χρόνο που χρειάζεται ένας επιτιθέμενος για να ανακτήσει έναν κωδικό πρόσβασης.
- Ο κωδικός πρόσβασης 10 ψηφίων αποτελεί μία κοινώς αποδεχτή επιλογή των βέλτιστων πρακτικών και οδηγιών ασφαλείας πληροφοριακών συστημάτων. Εισάγει ελάχιστη χρονική καθυστέρηση (μεγέθους 0,186 δευτερολέπτων) σε σχέση με την

αντίστοιχη χρονική διάρκεια για μήκος 8 ψηφιών ενώ μειώνονται οι πιθανότητες ανάκτησης του κωδικού πρόσβασης λόγω αυξημένων ψηφίων.

Εναλλακτική επιλογή αποτελεί το μήκος κωδικού πρόσβασης 8 ψηφιών και το κόστος εκτέλεσης 10. Συνοπτικά:

- Το μήκος κωδικού πρόσβασης 8 ψηφιών αποτελεί το ελάχιστο αποδεκτό μήκος από τις βέλτιστες πρακτικές και οδηγίες ασφαλείας.
- Το κόστος 10 προσθέτει την κατάλληλη κάλυψη σε επιθέσεις ανάκτησης κωδικών πρόσβασης, χωρίς ανάγκη για επιπλέον υπολογιστικούς πόρους.
- Η χρονική διάρκεια εκτέλεσης δεν ξεπερνά τα 0,2350 δευτερόλεπτα, ποσότητα μη αισθητή από το χρήστη.

9. Αναφορές

[1] Wikipedia, 'Key (Cryptography)' , http://en.wikipedia.org/wiki/Key_(cryptography)
[2] Wikipedia, 'Hash Function', http://en.wikipedia.org/wiki/Hash_function
[3] Wikipedia, 'Key Derivation Function', http://en.wikipedia.org/wiki/Hash_function
[4] Wikipedia, 'Bcrypt', http://en.wikipedia.org/wiki/Bcrypt
[5] Wikipedia, 'PBKDF2', http://en.wikipedia.org/wiki/PBKDF2
[6] Wikipedia, 'Blowfish (Cipher)', http://en.wikipedia.org/wiki/Blowfish_(cipher)
[7] Wikipedia, 'MD5', http://en.wikipedia.org/wiki/MD5
[8] Wikipedia, 'SHA1', http://en.wikipedia.org/wiki/SHA-1
[9] Wikipedia, 'Key stretching', http://en.wikipedia.org/wiki/Key_stretching
[10] Wikipedia, 'Salt (Cryptography)', http://en.wikipedia.org/wiki/Salt_(cryptography)
[11] Wikipedia, 'Dictionary Attack' , http://en.wikipedia.org/wiki/Dictionary_attack
[12] Wikipedia, 'Rainbow Table', http://en.wikipedia.org/wiki/Rainbow_table
[13] Wikipedia, 'Wi-Fi_Protected_Access' ', http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access
[14] Wikipedia, 'Software Virtualization', http://en.wikipedia.org/wiki/Category:Virtualization_software
[15]Wikipedia, 'Virtual Machines', http://en.wikipedia.org/wiki/Virtual_machine
[16] Wikipedia, 'Virtualization', http://en.wikipedia.org/wiki/Virtualization
[17] Wikipedia, 'Apache HTTP Server', http://en.wikipedia.org/wiki/Apache_HTTP_Server
[18] Apache org. , http://httpd.apache.org/
[19] Wikipedia, 'PHP', http://en.wikipedia.org/wiki/PHP
[20] PHP.net, http://php.net/
[21] Wikipedia, 'phpMyAdmin', http://en.wikipedia.org/wiki/PhpMyAdmin
[22] phpMyadmin.net, http://www.phpmyadmin.net/home_page/index.php
[23] Wikipedia, 'MySQL ', http://en.wikipedia.org/wiki/MySQL
[24] Mysql.com. http://www.mysql.com/
[25] Wikipedia, 'Message Authentication Code', http://en.wikipedia.org/wiki/Message_authentication_code
[26] Wikipedia, 'Hash Based Message Authentication Code', http://en.wikipedia.org/wiki/Hash-based_message_authentication_code
[27] Wikipedia, 'Replay Attack', http://en.wikipedia.org/wiki/Replay_attack
[28] PCmag., 'Definition of Replay attack', http://www.pcmag.com/encyclopedia/term/50439/replay-attack
[29] Wikipedia, 'Botnet', http://en.wikipedia.org/wiki/Botnet
[30] Wikipedia, 'Hash Function', http://en.wikipedia.org/wiki/Hash_function
[31] NIST Special Publication 800-108, 'Recommendation for Key Derivation using Pseudorandom Functions' , Computer Security Division, Information Technology Laboratory, October 2009
[32] SANS Institute, 'An Overview of Cryptographic Hash Functions and their Uses', InfoSec Reading Room, SANS Institute 2003
[33] Wikipedia, 'Key Derivation Function', http://en.wikipedia.org/wiki/Key_derivation_function

[34] Tomshardware.com, 'WiFi Security', http://www.tomshardware.com/reviews/wireless-security-hack,2981-5.html
[35] Wikipedia, 'scrypt', http://en.wikipedia.org/wiki/Scrypt
[36] Tarnsnap, 'The scrypt key derivation function', http://www.tarsnap.com/scrypt.html
[37] Wikipedia, 'Low Orbit Ion Cannon', http://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon
[38] Osarena, 'LOIC: Ανοιχτού Κώδικα Εργαλείο για DDOS επιθέσεις από απλούς χρήστες', http://osarena.net/logismiko/applications/loic-anichtou-kodika-ergalio-gia-ddos-epithesis-apo-aplous-christes.html
[39] PHP Manual, 'microtime', http://php.net/manual/en/function.microtime.php
[40] PHP Manual, 'Password_hash', http://php.net/manual/en/function.password-hash.php
[41] PHP Manual, 'Password_verify', http://php.net/manual/en/function.password-verify.php

Παράρτημα Α

Όνομα Αρχείου	Κώδικας

Mysqli_connect.php	<?php
	
	define('DB_USER', 'root');
	define('DB_PASSWORD', 'password');
	define('DB_HOST', 'localhost');
	define('DB_NAME', 'sphy111');
	
	$dbc = @mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)
	OR
	die('Could not connect to MySQL: ' . mysqli_connect_error());
	
	?>

Login_functions.inc.php	<?php
	//$start_time=microtime (true);
	
	function absolute_url($page = 'index.php')
	{
	$url = 'http://' . $_SERVER['HTTP_HOST'] .
	dirname($_SERVER['PHP_SELF']);
	$url = rtrim($url, '/\\');
	$url .= '/' . $page;
	return $url;
	}
	
	function check_login($dbc, $email='', $pass='')
	{
	$errors = array();
	if (empty($email)) {
	$errors[] = 'You forgot to enter your email address.';
	} else {
	$e = mysqli_real_escape_string($dbc, trim($email));
	}
	
	if (empty($pass)) {
	$errors[] = 'You forgot to enter your password.';
	} else {
	$p = mysqli_real_escape_string($dbc, trim($pass));
	}
	
	if (empty($errors)) {
	
	$q = "SELECT user_id, first_name FROM users
	WHERE email='$e' AND pass=SHA1('$p')";
	$r = @mysqli_query($dbc, $q);
	if (mysqli_num_rows($r) == 1) {
	$row = mysqli_fetch_array($r, MYSQLI_ASSOC);
	return array(true, $row);

	} else {
	$errors[] = 'The email address and password entered do not match those on file.';
	}
	}
	
	return array(false, $errors);
	}
	
	//$end_time=microtime(true);
	//$delay= $end_time - $start_time;
	// echo "The delay is '$delay' δευτερόλεπτα";
	?>

Login.php (bcrypt)	<?php
	$time_start= microtime (true);
	ob_start();
	session_start();
	
	$email = $_POST['username'];
	$userpass = $_POST['password'];
	
	$username = "root";
	$password = "password";
	$hostname = "localhost";
	
	//connection to the database
	$dbhandle = mysql_connect($hostname, $username, $password)
	or die("Unable to connect to MySQL");
	echo "Connected to MySQL
	";
	
	//select a database to work with
	$selected = mysql_select_db("users",$dbhandle)
	or die("Could not select examples");
	
	//execute the SQL query and return records
	$query="SELECT username,pass FROM important WHERE email='$email'";
	
	$result = mysql_query($query);
	
	//If the query returned results, loop through
	// each result
	if($result)
	{
	echo "dfdfdfsdfsdfsdf";
	while($row = mysql_fetch_array($result))
	{
	$pass1 = $row['pass'];
	$name1= $row['username'];
	// echo "Passhash: " . $name;

```
            echo "1";
 }

//$row = mysql_fetch_array($result)


//echo $row['pass'];

if (password_verify($userpass, $pass1))
{

echo "Hello" . $name1;
}
else
{
echo "NOTOK";
}
$time_end= microtime (true);

$delay= $time_end-$time_start;
echo "The time elapsed is '.$delay.' δευτερόλεπτα";

}



//close the connection
mysql_close($dbhandle);




//$options = [
//'cost' => 15,
//];


//$res=password_hash('dimitris1234ab!',PASSWORD_BCRYPT, $options);
//echo $res;

//if (password_verify('rasmuslerdorf',
'$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq'))
//{

//echo "OK";
//}
//else

//{
//echo "NOTOK";
```

	//} ?>
Login.html (BCRYPT)	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html xmlns="http://www.w3.org/1999/xhtml"> <head> <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <title>Login Form</title> </head> <body> <form id="form1" name="form1" method="post" action="login.php"> <table width="510" border="0" align="center"> <tr> <td colspan="2">Login Form</td> </tr> <tr> <td>Username:</td> <td><input type="text" name="username" id="username" /></td> </tr> <tr> <td>Password</td> <td><input type="password" name="password" id="password" /></td> </tr> <tr> <td> </td> <td><input type="submit" name="button" id="button" value="Submit" /></td> </tr> </table> </form> </body> </html>
important DATABASE (BCRYPT)	-- phpMyAdmin SQL Dump -- version 3.4.10.1deb1 -- http://www.phpmyadmin.net -- -- Host: localhost -- Generation Time: Sep 30, 2014 at 11:54 AM -- Server version: 5.5.37 -- PHP Version: 5.5.13-1+deb.sury.org~precise+1 SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO"; SET time_zone = "+00:00"; /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */; /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */; /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */; /*!40101 SET NAMES utf8 */;

```
--
-- Database: `users`
--

-- --------------------------------------------------------

--
-- Table structure for table `important`
--

CREATE TABLE IF NOT EXISTS `important` (
 `username` varchar(30) NOT NULL,
 `pass` char(255) NOT NULL,
 `email` varchar(255) NOT NULL,
 `user_id` int(10) unsigned NOT NULL,
 PRIMARY KEY (`user_id`),
 UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `important`
--

INSERT INTO `important` (`username`, `pass`, `email`, `user_id`) VALUES
('dimitris',
'$2y$13$t9IJurS8s7Zt0JbNRREfgOeYVI7cx8LBXSvKq6OKiDPk4IbC4lmES',
'dimitris@gmail.com', 1),
('rasmuslerdorf',
'$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq',
'ras@gmail.com', 2),
('dimitris',
'$2y$07$3.EPkfvcjaU1Ieqt75aJ8.ZxrTDqRm9f9a7M7Wp8j3aDdpHkQzFjO',
'dimitris@yahoo.com', 3),
('dimitris',
'$2y$10$V9XeAiD9LhzwngV4VfKn/uwfi.5jzenGKYldkb2oSJ2bahybQBU62',
'dimitris@bandu.com', 4),
('dimitris',
'$2y$15$6dM9KAstXr5Y.4rNS6s3Y.b5nIE4NOqHC3eyOX59/2L1W6zeAfIau',
'dimitris@zandu.com', 5),
('dimitris12',
'$2y$07$B6vWqc8x6mDnt7adwKoISOHJkhtmj2b5whayXFdRv58qp/Apn/re.',
'dimitris12@candu.com', 6),
('dimitris12',
'$2y$10$/9vZbxMKOT/78WgjtZkfveDyjGawiQPrOtQV5RUhRsUXfo7UcGkpa',
'dimitris12@dandu.com', 7),
('dimitris12',
'$2y$15$PHFtxmA31wdY7z.qRUCnmu/9SqjX3W0UltQkCzWAiI7m47ybOPXh6',
'dimitris12@fandu.com', 8),
('dimitris1234ab!',
'$2y$07$J2AL5DrvkA3MZRvcf6G/yeMWbsO8d4nG7tBxdO6W0D/yauerTqP4O',
'dimitris1234ab!@gandu.com', 9),
('dimitris1234ab!',
'$2y$10$k4mFrICZZZUrwstlXus9UOZAtkPqib2/Gw.3ca5sDFnGRvoo9rYWq',
'dimitris1234ab!@handu.com', 10),
('dimitris1234ab!',
'$2y$15$5/.4x2hqI925UyZpdysup.mFc9AojQ2MmS10EbGRumY.NIGfIU1mC',
'dimitris1234ab!@kandu.com', 11);

/*!40101 SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET
```

	COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
Sphy111(SHA1)	-- phpMyAdmin SQL Dump -- version 3.4.10.1deb1 -- http://www.phpmyadmin.net -- -- Host: localhost -- Generation Time: Sep 30, 2014 at 12:47 PM -- Server version: 5.5.37 -- PHP Version: 5.5.12-2+deb.sury.org~precise+1 SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO"; SET time_zone = "+00:00"; /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */; /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */; /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */; /*!40101 SET NAMES utf8 */; -- -- Database: `sphy111` -- -- -- -- -- Table structure for table `users` -- CREATE TABLE IF NOT EXISTS `users` (`user_id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT, `first_name` varchar(20) NOT NULL, `last_name` varchar(40) NOT NULL, `email` varchar(80) NOT NULL, `pass` char(40) NOT NULL, `registration_date` datetime NOT NULL, PRIMARY KEY (`user_id`)) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=25 ; -- -- Dumping data for table `users` -- INSERT INTO `users` (`user_id`, `first_name`, `last_name`, `email`, `pass`, `registration_date`) VALUES (22, 'thannasi', 'kouroubas', 'thannasi@zouvas.com', 'f93f91a3a33a3cf96c6c14403a68caf763fc1d5b', '2014-09-27 10:03:11'), (23, 'thannasi12', 'gouvas', 'thannasi12@gouvas.com', 'd3a676f9295fba67117c921d2a767f33966e0d7c', '2014-09-27 10:04:05'), (24, 'thannasi1223ab!', 'douvas', 'thannasi1234ab!@douvas.com', '05356f2080afd781515736074191096b101ca283', '2014-09-27 10:05:50'); /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */; /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */; /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;