Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Μαθηματική και αλγοριθμική ανάλυση κρυπτογραφικών τεχνικών

Γεώργιος Όγλου (ΑΜ: 3664)
E-mail: georgeoglou@yahoo.gr

Ηράκλειο – 2015

Επόπτης Καθηγητής: Δρ. Παπαδάκης Νίκος

Γιώργος Όγλου

Υπεύθυνη Δήλωση*:Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος ΕφαρμοσμένηςΠληροφορικής και Πολυμέσων του Τ.Ε.Ι. Κρήτης.*

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πρώτα απο όλα την μητέρα μου που τόσα χρόνια με στηρίζει σε κάθε μου βήμα δίνοντας μου εφόδια για να μπορέσω να σταθώ στη ζωή μου.

Έπειτα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητήμουΝίκο Παπαδάκη που χωρίς τις πολύτιμες συμβουλές, και την βοήθεια του δεν θα μπορούσα να τελειώσω την πτυχιακή.

Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές μέσης εκπαίδευσηςΝικήτας Χατζηκωνσταντίνου και Λουκάς Κώστα του φροντιστηρίου «Η ΔΟΜΗ» που βρίσκεται στο νησί της Ρόδου, για την πολύτιμη στήριξη που μου δώσαν όταν ήμουν ακόμη στο Λύκειο.Χαρη σε αυτούς οφείλεται η στροφή μου προς τις θετικές-τεχνολογικές επιστήμες καθως η αγάπη μου προς τα μαθηματικά και την φυσική.

Τέλος θα ήθελα να ευχαριστήσω την συμφοιτήτρια και κάτωχο του ECDLexpert Λαγουδάκη Στυλιανή για τις συμβουλές τις σε κάποια δύσκολα θέματα που είχα με την μορφοποιήση της πτυχιακής

Γιώργος Όγλου

Περίληψη

Η κρυπτογραφία είναι ένα απαραίτητο εργαλείο για την προστασία των πληροφοριών στα συστήματα υπολογιστών.Η αναφορά αυτή εξηγεί τις εσωτερικές λειτουργίες των αλγορίθμων και μηχανισμών χαμηλού επιπέδου που χρησιμοποιούνται συχνά στην κατασκευή κρυπτογραφικών πρωτοκόλλων για να δημιουργήσουμε ασφαλή υπολογιστικά συστήματα.Θα δούμε τον λόγο που είναι απαραίτητη η ασφάλεια στα κρυπτογραφικά συστήματα και θα χρησιμοποιήσουμε αυτήν την γνώση σε πραγματικές εφαρμογές.

Η αναφορά ξεκινά αρχικά με την ιστορία της κρυπτογραφίας , κατόπιν εξηγούμε πώς λειτουργούν οι αλγόριθμοι κατασκευής κρυπτογραφικών πρωτοκόλλων ενώ στη συνέχεια θα δείξουμε το πώς δύο άτομα μπορούν να επικοινωνήσουν με ασφάλεια όταν υπάρχει ένα κοινό κλειδί και ένας επιτιθέμενος προσπαθεί να κρυφακούσει και να παραποιήσει την πληροφορία μεταξύ των δύο ατόμων αυτών.

Θα εξετάσουμε επίσης αρκετά πρωτόκολλα που έχουν δημιουργηθεί και θα αναλύσουμε τα λάθη που έχουν και σε ποιές περιπτώσεις πρέπει να τα χρησιμοποιούμε και σε ποιές όχι.Επίσης θα ασχοληθούμε και με την κρυπτογραφία δημόσιου κλειδιού αλλά και τις τεχνικές όπου δύο ή περισσότερα άτομα θα μπορέσουν να επικοινωνήσουν με ασφάλεια.

Τέλος θα καλύψουμε αρκετά σημεία της θεωρίας αριθμών που θα μας βοηθήσει να περιγράψουμε σημαντικά πρωτόκολλα δημόσιας κρυπτογραφίας και ανταλλαγής κλειδιών.

Abstract

Cryptography is an indispensable tool for protecting computer information systems.This report explains the inner workings of algorithms and low-level mechanisms are often used in the construction of cryptographic protocols to create secure computing systimata.Tha see why that's necessary security to cryptographic systems and will use this knowledge in real-world applications.

The report initially starts with the history of cryptography, and then explains how the construction of cryptographic protocols and algorithms will then show how two people can communicate securely when there is a public key and an attacker tries to eavesdrop and skew information between two such persons.

We will also examine several protocols have been created and will analyze the mistakes are, and in which cases we have to use and them.Also will deal with public-key cryptography and techniques where two or more people will be able to communicate with safety.Finally we will cover several points of the number theory that will help us to describe important protocols aboutpublic cryptography and basickey exchange.

ΠίνακαςΠεριεχομένων

Πίνακας Εικόνων

Πίνακας πινάκων

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 Σκοπός της Πτυχιακής Εργασίας

1.2 Συνοπτική Περιγραφή Αναφοράς

Στο **κεφάλαιο 2** αναφερόμαστε στις βασικές έννοιες και λειτουργίες της κρυπτογραφίας.Στη συνέχεια προσθέτουμε τη σχετική ορολογία και κάνουμε μια πολύ σύντομη ιστορική αναδρομή.Τέλος επιγραμματικά αναφέρουμε τις εφαρμογές τς κρυπτογραφίας.

Στο **κεφάλαιο 3** περιγράφουμε τις βασικές ιδιότητες των αλγορίθμων ροής.Κατόπιν αναφερόμαστε στον αλγόριθμο μιας χρήσης(OTP) αλλα και στις γεννήτριες ψευδοτυχίας αριθμών.Δείχνουμε ποιές γεννήτριες είναι ασφαλείς και μπορούν να χρησιμοποιηθούν στην κρυπτογραφία.Στη συνέχεια κάνουμε μια αναφορά στα στατιστικά τέστ δείχνοντας με αυτόν τον τρόπο , ποιά τέστ είναι ικανά και αποδοτικά για την κρυπτογραφία.Τέλος δείχνουμε τις πιό σημαντικές επιθέσεις στους αλγορίθμους ροής και περιγράφουμε την σημασιολογική ασφάλεια.

Στο **κεφάλαιο 4** περιγράφουμε τις βασικές ιδιότητες των αλγορίθμων τμήματος και ορίζουμε τις ψευδοτυχαίες συναρτήσεις-ψεδοτυχαίες αντιμεταθέσεις.Κατόπιν περιγράφουμε το πρότυπο DESκαι δείχνουμε γιατί δεν θα πρέπει να χρησιμοποιείται στις μέρες μας.Κάνουμε επίσης μια αναφορά στο πρωτόκολλο 3DESδείχνοντας γιατ είναι ασφαλέστερο απο το πρωτόκολλο DES.Στη συνέχεια αναφερόμαστε στο πρωτόκολλο AESκαι στο γιατί είναι ασφαλές.Τέλος περιγραφουμε επιθέσεις στο πρώτυπο DESαλλα και περιγράφουμε τους τρόπους λειτουργίας των αλγορίθμων τμήματος.

Ο σκοπός μας στο **κεφάλαιο 5** είναι να περιγράψουμε το πώς μπορούμε να έχουμε ακεραιότητα σε ένα μήνυμα.Για τον λόγο αυτό δημιουργούμε-περιγράφουμε τους τρόπους λειτουργίας των κωδικών πιστοποίησης ταυτότητας(MAC).Δείχνουμε το πόσο ασφαλές μπορεί να είναι ένα MAC.Κατόπιν περιγράφουμε διεξοδικά αρκετά είδη MACαλγορίθμων αναφέροντας τα πλεονεκτήματα και τα μειονεκτήματα που έχει ένα είδος MACαπο ένα άλλο.Τέλος περιγράφουμε τι ειναι οι συναρτήσεις κατακερματισμού και αναλύουμε πρωτόκολλα που μπορούν να δώσουν αντοχές στις συγκρούσεις στις συναρτήσεις αυτές, μη ξεχνόντας να δείξουμε μερικές επιθέσεις πάνω στους αλγορίθμους MAC.

Στο **κεφάλαιο 6** αυτό θα συνδυάσουμε την εμπιστευτικότητα και τη ακεραιότητα για να πάρουμε ασφαλή συστήματα τα οποία θέλουμε να μας παρέχουν ασφάλεια πάνω σε επιθέσεις injectingκαι tampering.Αρχικά περιγράφουμε το πώς λειτουργέι ένα σύστημα αυθεντικής κρυπτογράφησης και δείχνουμε σχήματα αυτής όπως είναι το SSL,IPseckαι SSH.Κατόπιν αναφερόμαστε στο πρωτόκολλο TLSrecordκαι σε διάφορες επιθέσεις που μπορούν να συμβούν στα συστήματα αυθεντικής κρυπτογράφησης.Στη συνέχεια κάνουμε μια αναφορά στο πώς μπορούμε να παράγουμε απο ένα και μόνο κλειδί, πολλά κλειδία διαφορετικά μεταξύ τους.Τέλος περιγράφουμε την λειτουργία της ντετερμινιστικής κρυπτογραφίας και την tweakableκρυπτογράφηση

Στο **κεφάλαιο 7** περιγράφουμε διεξοδικά το πώς γίνεται η ανταλλαγή κλειδιών απο δύο ή περισσότερα άκρα επικοινωνίας.Αναφερόμαστε στο πρωτόκολλο Merkl-puzzleskαι δείχνουμε τον λόγο γιατι δεν πρεπει να χρησιμοποιείται στην πράξη.Στη συνέχεια αναφερόμαστε στο πολύ γνωστό και ευρέως χρησιμοποιούμενο πρωτόκολλο αναταλλαγής κλειδιών Diffie-Hellman.Τέλος αναλύουμε τον τρόπο

λειτουργίας της δημόσιας κρυπτογραφίας αλλα και παρουσιάζουμε βασικά στοιχεία της θεωρίας αριθμών που θα μας χρειαστούν για το επόμενο κεφάλαιο.

Στο **8 κεφάλαιο**θα χρησιμοποιήσουμε την θεωρία αριθμών για να κατασκευάσουμε σχήματα κρυπτογραφίας δημόσιου κλειδιού.Στη συνέχεια θα αναφερθούμε στις trapdoorfunctiosκαι στη συνάρτηση RSA.Τέλος θα περιγράψουμε το στάνταρντ PKCS1 αλλα και το σύστημα δημόσιας κρυπτογραφίας ElGamal.

1.3Σχεδιάγραμμα Αναφοράς

ΚΕΦΑΛΑΙΟ 2
Η ΕΠΙΣΤΗΜΗ ΤΗΣ ΚΡΥΠΤΟΓΡΑΦΙΑΣ

2.1 Βασικές έννοιες και ορισμοί

Η λέξη κρυπτογραφία προέρχεται από τα συνθετικά «κρυπτός»+ «γράφω» και είναι ένα διεπιστημονικό πεδίο που ασχολείται με την μελέτη, την ανάπτυξη και την χρήση ειδικών τεχνικών κρυπτογράφησης και αποκρυπτογράφησης, με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων.

Η σημασία της κρυπτολογίας είναι τεράστια στους τομείς της ασφάλειας υπολογιστικών συστημάτων και των τηλεπικοινωνιών. Ο κύριος στόχος της είναι να παρέχει μηχανισμούς ώστε 2 ή περισσότερα άκρα επικοινωνίας (π.χ. άνθρωποι, προγράμματα υπολογιστών κλπ.) να ανταλλάξουν μηνύματα, χωρίς κανένας τρίτος να είναι ικανός να διαβάζει την περιεχόμενη πληροφορία εκτός από τα δύο κύρια άκρα.

Ιστορικά, η κρυπτογραφία χρησιμοποιήθηκε για τη μετατροπή της πληροφορίας μηνυμάτων από μια κανονική, κατανοητή μορφή σε έναν «γρίφο», που χωρίς τη γνώση του κρυφού μετασχηματισμού θα παρέμενε ακατανόητος. Κύριο χαρακτηριστικό των παλαιότερων μορφών κρυπτογράφησης ήταν ότι η επεξεργασία γινόταν πάνω στη γλωσσική δομή του μηνύματος. Στις νεότερες μορφές, η κρυπτογραφία κάνει χρήση του αριθμητικού ισοδύναμου, ενώ η έμφαση έχει μεταφερθεί σε διάφορα πεδία των μαθηματικών, όπωςδιακριτά μαθηματικά, θεωρία αριθμών, θεωρία πληροφορίας, στατιστική και συνδυαστική ανάλυση.

Η κρυπτογραφία παρέχει τέσσερις βασικές λειτουργίες :

- **Εμπιστευτικότητα**: Η πληροφορία προς μετάδοση είναι προσβάσιμη μόνο στα εξουσιοδοτημένα μέλη. Η πληροφορία είναι ακατανόητη σε κάποιον τρίτο.
- **Ακεραιότητα**: Η πληροφορία μπορεί να αλλοιωθεί μόνο από τα εξουσιοδοτημένα μέλη και δεν μπορεί να αλλοιώνεται χωρίς την ανίχνευση της αλλοίωσης.
- **Μη αποποίηση**:Ο αποστολέας ή ο παραλήπτης της πληροφορίας δέν μπορεί να αρνηθεί την αυθεντικότητα της μετάδοσης ή της δημιουργίας της
- **Πιστοποίηση**:Οι αποστολέας και ο παραλήπτης μπορούν να εξακριβώνουν τις ταυτότητες τους καθώς και την πηγή και τον προορισμό της πληροφορίας με τη διαβεβαίωση ότι οι ταυτότητες τους δεν είναι πλαστές.

Για να κατανοήσουμε την επιστήμη της κρυπτογραφίας πρέπει να γνωρίζουμε την σχετική ορολογία που αναφέρεται σε αυτήν:

- **Κρυπτογράφηση**(encryption):ονομάζεται η διαδικασία μετασχηματισμού ενόςμηνύματος σε μια ακατανόητη μορφή με χρήση κάποιου κρυπτογραφικού αλγορίθμου με σκοπό να μην μπορεί το μήνυμα να διαβαστεί απο τρίτους.
- **Κρυπτογραφικός αλγόριθμος**(cipher):είναι η μέθοδος μετασχηματσιμού δεδομένων σε μια μορφή που να μην επιτρέπει την αποκάλυψη των περιεχομένων τους απο μη εξουσιοδοτημένα μέρη.
- **Αρχικό κείμενο**(plaintext):είνα το μήνυμα που αποτελεί την είσοδο σε μια διεργασία κρυπτογράφησης.
- **Κλειδί**(key):είναι ένας αριθμός αρκετών bitπου χρησιμοποιείται ως είσοδος στη συνάρτηση κρυπτογράφησης.

- **Κρυπτογραφημένο μήνυμα**(ciphertext):είναι το αποτέλεσμα της εφαρμογής ενος κρυπτογραφικού αλγορίθμου.
- **Κρυπτανάλυση**(cryptanalysis):είναι μια επιστήμη που ασχολείται με το «σπάσιμο» κάποιας κρυπτογραφικής τεχνικής.

Η κρυπτογράφηση και αποκρυπτογράφηση ενός μηνύματος γίνεται με τη βοήθεια ενός αλγόριθμου κρυπτογράφησης (cipher) και ενός κλειδιού κρυπτογράφησης (key).Συνήθως ο αλγόριθμος κρυπτογράφησης είναι γνωστός οπότε η εμπιστευτικότητα του κρυπτογραφημένου μηνύματος που μεταδίδεται βασίζεται ως επί το πλείστον στη μυστικότητα του κλειδιού κρυπτογράφησης.

Το μέγεθος του κλειδιού κρυπτογράφησης μετριέται σε αριθμό bits. Γενικά ισχύει ο εξής κανόνας: όσο μεγαλύτερο είναι το κλειδί κρυπτογράφησης, τόσο δυσκολότερα μπορεί να αποκρυπτογραφηθεί το κρυπτογραφημένο μήνυμα από επίδοξους εισβολείς. Διαφορετικοί αλγόριθμοι κρυπτογράφησης απαιτούν διαφορετικά μήκη κλειδιών για να πετύχουν το ίδιο επίπεδο ανθεκτικότητας κρυπτογράφησης.

Ένα κρυπτοσύστημα (σύνολο διαδικασιών κρυπτογράφησης - αποκρυπτογράφησης) αποτελείται από μία πεντάδα (P,C,k,E,D):

- Το **P** είναι ο χώρος όλων των δυνατών μηνυμάτων ή αλλιώς ανοικτών κειμένων
- Το **C** είναι ο χώρος όλων των δυνατών κρυπτογραφημένων μηνυμάτων ή αλλιώς κρυπτοκειμένων
- Το **k** είναι ο χώρος όλων των δυνατών κλειδιών ή αλλιώς κλειδοχώρος
- Η **E** είναι ο κρυπτογραφικός μετασχηματισμός ή κρυπτογραφική συνάρτηση.Η συνάρτηση κρυπτογράφησης E δέχεται δύο παραμέτρους, μέσα από τον χώρο P και τον χώρο k και παράγει μία ακολουθία που ανήκει στον χώρο C.
- Η **D** είναι η αντίστροφη συνάρτηση ή μετασχηματισμός αποκρυπτογράφησης.Η συνάρτηση αποκρυπτογράφησης D δέχεται 2 παραμέτρους, τον χώρο C και τον χώρο k και παράγει μια ακολουθία που ανήκει στον χώρο P

2.2 Ιστορική Αναδρομή

Η κρυπτογραφία είναι ένας απο τους δύο κλάδους της κρυπτολογίας(ο αλλος ειναι η κρυπτανάλυση) η οποία ασχολείται με την μελέτη της ασφαλούς επικοινωνίας.Ο κύριος στόχος της κρυπτολογίας ειναι να παρέχει μηχανισμούς ώστε δύο ή περισσότερα άκρα επικοινωνίας να μπορούν να ανταλλάξουν μηνύματα, χωρις κάποιος τρίτος να μπορεί να διαβάζει την περιεχόμενη πληροφορία, εκτος απο τα δύο κύρια άκρα.Η πρόοδος της κρυπτογραφίας χωρίζεται σε τρεις περιόδους.Η πρώτη περίοδος της κρυπτογραφίας διήρκησε απο το 1900πΧ εως το 1900μΧ,η δεύτερη περίοδος ήταν απο το 1900μΧ εως το 1950μΧ, ενώ η τρίτη περίοδος είναι απο το 1950μΧ εώς και σήμερα.

2.1.1 Πρώτη περίοδος της κρυπτογραφίας(1900πΧ-1900μΧ)

Το κυριότερο εργαλείο στην κρυπτανάλυση ηταν η χρησιμοποίηση των συχνοτήτων των γραμμάτων κειμένου, σε συνδυασμό με τις συχνότητες εμφάνισης στα κείμενα.Όμως, κατά τη διάρκεια της περιόδου(1900πΧ-1900μΧ) αναπτύχθηκε μεγάλο πλήθος μεθόδων και αλγορίθμων κρυπτογράφησης, που βασίζονταν κυρίως σε απλές αντικαταστάσεις γραμμάτων.Όλα αυτά τα συστήματα έχουν στις μέρες μας κρυπταναλυθεί.

Στην αρχαιότητα χρησιμοποιήθηκαν κυρίως συστήματα, τα οποία βασίζονταν στη στεγανογραφία και όχι τόσο στην κρυπτογραφία γραμμάτων. Ένα παράδειγμα είναι ο αλγόριθμος του Καίσαρα.Σύμφωνα με τον **αλγόριθμο του Καίσαρα**[1], το κάθε γράμμα του μηνύματος που θέλουμε να στείλουμε αντικαθίσταται απο το τρίτο επόμενο γράμμα του λατινικού αλφαβήτου.Ετσι παραδείγματος χάρη το γράμμα Α γινόταν Γ, το γράμμα Β γίνοταν Ε κτλ.Η διαδικασία έφτανε ώς το τελευταίο γράμμα του αλφαβήτου.

Μια άλλη αναφορά που υποδηλώνει την ανάγκη της χρήσης της κρυπτογραφίας είναι ο **αλγόριθμος Vigenere**[2].Η κρυπτογράφηση Vigenère**(**πρωτοεμφανίστηκε απο τον LeonBatistaAlbertio το 1467)είναι μία μέθοδος κρυπτογράφησης σε αλφαβητικό κείμενο στο οποίο εφαρμόζονται διαφορετικοί αλγόριθμοι κρυπτογράφησης Καίσαρα με βάση τη θέση των γραμμάτων μιας λέξης ή φράσης κλειδί. Είναι μια απλή μορφή της πολυαλφαβητικής αντικατάστασης.Για την κρυπτογράφηση, ένας πίνακας του αλφάβητου μπορεί να χρησιμοποιηθεί ως πίνακας (**Εικόνα 1**)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Εικόνα 1:Ο πίνακας Vigenere

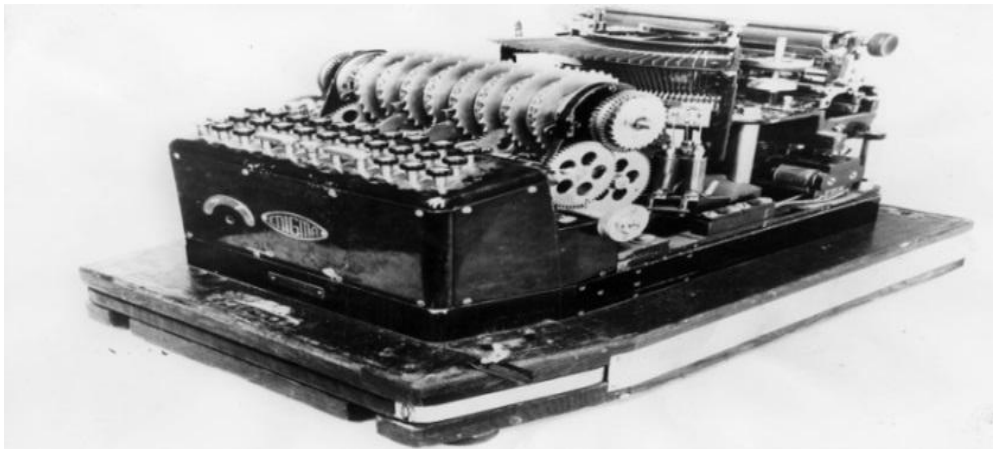
Αποτελείται από το αλφάβητο, που αναγράφεται σε διαφορετικές γραμμές (ή στήλες) τόσες φορές όσες και τα γράμματα του αλφαβήτου και κάθε αλφάβητο μετατοπίζεται κυκλικά σε σχέση με το προηγούμενο αλφάβητο, ώστε να υπάρχουν όλοι οι πιθανοί αλγόριθμοι κρυπτογράφησης του Καίσαρα.Κατά τη διαδικασία κρυπτογράφησης, χρησιμοποιείται διαφορετικό αλφάβητο σε κάθε ένα από τα γράμματα. Το αλφάβητο που χρησιμοποιείται σε κάθε γράμμα εξαρτάται από μια επαναλαμβανόμενη λέξη-κλειδί.Για παράδειγμα εάν έχουμε το plaintext«ATTACKATDAWN» και το κλειδί μας είναι «LEMON» τότε:

```
Plaintext: ATTACKATDAWN
Key:       LEMONLEMONLE
Ciphertext:LXFOPVEFRNHR
```

2.1.2 Δέυτερη περίοδος της κρυπτογραφίας(1900μΧ-1950μΧ)

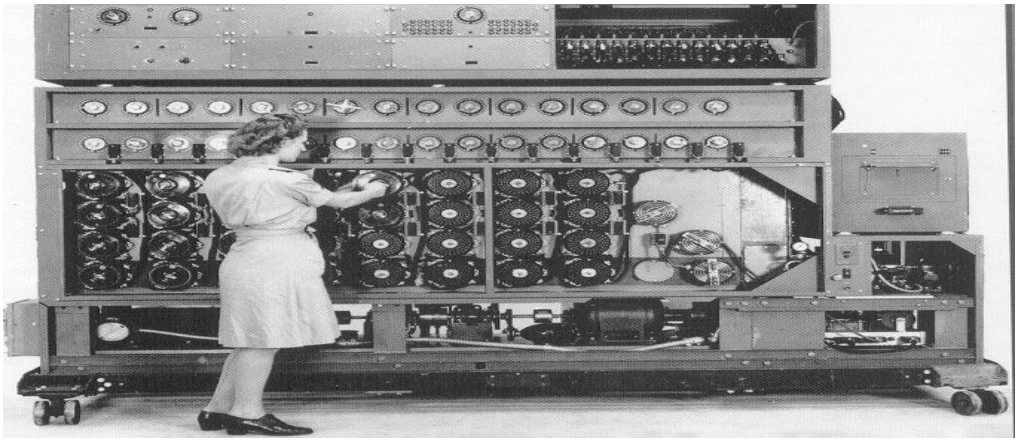
Επειδή η δέυτερη περίοδος φθάνει μέχρι το 1950 καλύπτει τους δύο παγκόσμιους πολέμους.Οι Γερμανοί χρήση ενός συστήματος που λεγοταν Enigma(**Εικόνα 2**)

[1] Αλγόριθμος του Καίσαρα, http://en.wikipedia.org/wiki/Caesar_cipher
[2]Αλγόριθμος του Vigenere, http://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher



Εικόνα 2:Η κρυπτομηχανή Enigma

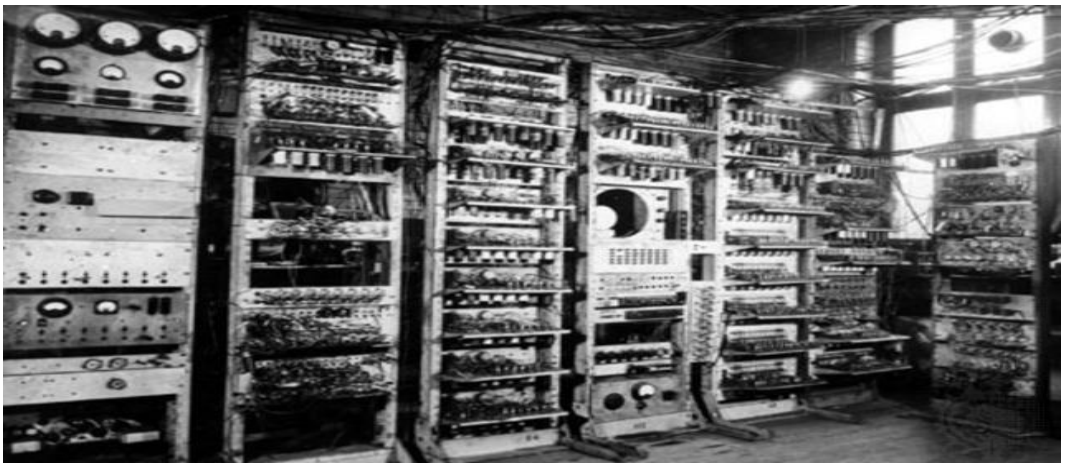
Μια συσκευή «Enigma[3]»είναι μια οποιαδήποτε συσκευή από μια οικογένεια συσχετιζόμενων ηλεκτρο-μηχανικών rotor συσκευών που χρησιμοποιήθηκαν για την κρυπτογράφηση και αποκρυπτογράφηση μυστικών μηνυμάτων.Η πρώτη συσκευή Enigma εφευρέθηκε από τον Γερμανό μηχανικό Άρθουρ Σέρμπιους[4].Αυτό το μοντέλο και οι παραλλαγές του χρησιμοποιήθηκαν εμπορικά από της αρχές της δεκαετίας του 1920και υιοθετήθηκαν εμπορικά απο στρατιωτικές και μυστικές υπηρεσίες και απο την Ναζιστική Γερμανία πρίν και κατά τη διάρκεια του Δεύτερου Παγκοσμίου Πολέμουενώ για την επίλυση του χρησιμοποιήθηκε απο τους Άγγλους κρυπταναλυτές το 1938 το Bombe[5](**Εικόνα 3**)



Εικόνα 3: Η μηχανή Bombe

Αργότερα όταν προστέθηκαν παραλλαγές στον κώδικα απο Πολωνούς, Βρετανούς, και Γάλλους μαθηματικούς και κρυπτογράφους και με την πολύτιμη βοήθεια του AlanTuring[6]δημιουργήθηκε το πολύτιμο Colossus(**Εικόνα 4**)

[3]Η κρυπτομηχανή Enigma, http://en.wikipedia.org/wiki/Enigma_machine
[4] Ο άρθουρ Σέρμπιους, https://en.wikipedia.org/wiki/Arthur_Scherbius
[5]Ημηχανήαποκρυπτογράφησης Bombe, https://en.wikipedia.org/wiki/Bombe
[6] Ο AlanTuringενας σπουδαίος επιστήμονας , https://en.wikipedia.org/wiki/Alan_Turing



Εικόνα 4:Η μηχανή Colossus

Από την άλλη πλευρά, το Ιαπωνικό Υπουργείο Εξωτερικών χρησιμοποίησετους κώδικες JN-25 και το μηχάνημα «Red» αλλα επειδή ήταν ανασφαλές τελικά ανέπτυξαν ένα τοπικά αναπτυγμένο κρυπτογραφικό σύστημα, που λέγεται «Purple[7]»(**Εικόνα 5**) στο οποίο η διάταξη άλλαζε αυτόματα κάθε 9 ημέρες.

Εικόνα 5:Η κρυπτομηχανή Purple

Οι κρυπτογράφοι του αμερικανικού ναυτικού σε συνεργασία με Βρετανούς και Ολλανδούς κρυπτογράφους μετά το 1940 έσπασαν αρκετά κρυπτοσυστήματα του Ιαπωνικού ναυτικού.

Το σπάσιμο του κώδικα JN-25 [8]οδήγησε στην αμερικανική νίκη στη Ναυμαχία της Μιντγουέι καθώς και στην εξόντωση του αρχηγού του Ιαπωνικού στόλου Ιζορόκου Γιαμαμότο.

Οι συμμαχικές κρυπτομηχανές που χρησιμοποιήθηκαν περιλάμβαναν το βρετανικό «TypeX»(**Εικόνα 7**) όπως και το πιό πολύπλοκο αμερικανικό «SIGABA»(**Εικόνα 6**)

[7]Η κρυπτομηχανή Purple, http://en.wikipedia.org/wiki/Purple_%28cipher_machine%29
[8]Οι κώδικες JN-25, https://en.wikipedia.org/wiki/Japanese_naval_codes

Εικόνα 6:Η κρυπτομηχανή SIGABA

Επίσης τα στρατεύματα στο πεδίο της μάχης χρησιμοποιήσαν το M-209[9]και τη λιγότερο ασφαλή οικογένεια κρυπτομηχανών M-94[10].

Εικόνα 7:Η μηχανή TypeX

2.1.3 Τρίτη περίοδος της Κρυπτογραφίας(1950μΧ-Σήμερα)

Η εποχή της σύγχρονης κρυπτογραφίας αρχίζει ουσιαστικά με τον Claude Shannon[11],που είναι αναμφισβήτητα ο πατέρας των μαθηματικών συστημάτων της κρυπτογραφίας ο οποίος έθεσε τη θεωρητική βάση για την κρυπτογραφία και την κρυπτανάλυση.Εκείνη την εποχή η κρυπτογραφία εξαφανίζεται και φυλλάσεται απο τις μυστικές υπηρεσίες κυβερνητικών επικοινωνιών όπως η NSA.

Στα μέσα της δεκαετίας του '70 έγιναν δύο σημαντικές δημόσιες πρόοδοι. Πρώτα ήταν η δημοσίευση του σχεδίου προτύπου κρυπτογράφησης DES (Data Encryption Standard) στον ομοσπονδιακό κατάλογο της Αμερικής στις 17 Μαρτίου 1975. Το προτεινόμενο DES υποβλήθηκε από την IBM, στην πρόσκληση του Εθνικού Γραφείου των Προτύπων (τώρα γνωστό ως NIST), σε μια προσπάθεια να αναπτυχθούν ασφαλείς ηλεκτρονικές εγκαταστάσεις επικοινωνίας για επιχειρήσεις όπως τράπεζες και άλλες μεγάλες οικονομικές οργανώσεις.

Μετά από τις συμβουλές και την τροποποίηση από την NSA, αυτό το πρότυπο υιοθετήθηκε και δημοσιεύθηκε ως ένα ομοσπονδιακή τυποποιημένο πρότυπο

[9]Η κρυπτομηχανή M-209, https://en.wikipedia.org/wiki/M-209
[10]Ο κρυπτογραφικός εξοπλισμός M-94, https://en.wikipedia.org/wiki/M-94
[11]Ο Claudde Shannon, http://en.wikipedia.org/wiki/Claude_Shannon

επεξεργασίας πληροφοριών το 1977 (αυτήν την περίοδο αναφέρεται σαν FIPS 46-3). Ο DES ήταν ο πρώτος δημόσια προσιτός αλγόριθμος κρυπτογράφησης που εγκρίνεται από μια εθνική αντιπροσωπεία όπως η NSA. Η απελευθέρωση της προδιαγραφής της από την NBS υποκίνησε μια έκρηξη δημόσιου και ακαδημαϊκού ενδιαφέροντος για τα συστήματα κρυπτογραφίας.

Ο DES αντικαταστάθηκε επίσημα από τον AES το 2001 όταν ανήγγειλε ο NIST το FIPS 197. Μετά από έναν ανοικτό διαγωνισμό, ο NIST επέλεξε τον αλγόριθμο Rijndael, που υποβλήθηκε από δύο Φλαμανδούς κρυπτογράφους, για να είναι οAES. Ο DES και οι ασφαλέστερες παραλλαγές του όπως ο 3DES ή TDES χρησιμοποιούνται ακόμα σήμερα, ενσωματωμένος σε πολλά εθνικά και οργανωτικά πρότυπα. Εντούτοις, το βασικό μέγεθος των 56-bit έχει αποδειχθεί ότι είναι ανεπαρκές να αντισταθεί στις επιθέσεις ωμής βίας (μια τέτοια επίθεση πέτυχε να σπάσει τον DES σε 56 ώρες ενώ το άρθρο που αναφέρεται ως το σπάσιμο του DES δημοσιεύτηκε από τον O'Reilly and Associates).

Κατά συνέπεια, η χρήση απλής κρυπτογράφησης με τον DES είναι τώρα χωρίς την αμφιβολία επισφαλής για χρήση στα νέα σχέδια των κρυπτογραφικών συστημάτων και μηνύματα που προστατεύονται από τα παλαιότερα κρυπτογραφικά συστήματα που χρησιμοποιούν DES, και όλα τα μηνύματα που έχουν αποσταλεί από το 1976 με τη χρήση DES, διατρέχουν επίσης σοβαρό κίνδυνο αποκρυπτογράφησης. Ανεξάρτητα από την έμφυτη ποιότητά του, το βασικό μέγεθος του DES (56-bit) ήταν πιθανά πάρα πολύ μικρό ακόμη και το 1976, πράγμα που είχε επισημάνει ο Whitfield Diffie. Υπήρξε επίσης η υποψία ότι κυβερνητικές οργανώσεις είχαν ακόμα και τότε ικανοποιητική υπολογιστική δύναμη ώστε να σπάσουν μηνύματα που είχαν κρυπτογραφηθεί με τον DES.

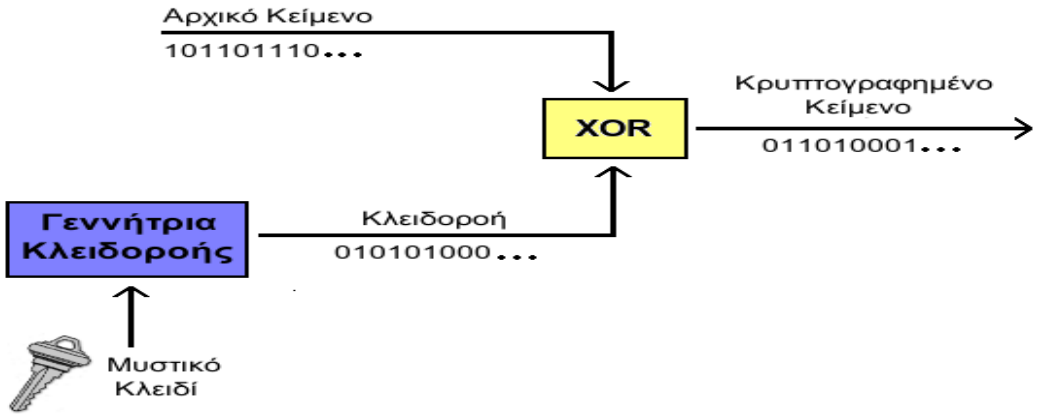
Σήμερα χρησιμοποιείται σε πολλές εφαρμογές όπως:

- Ασφάλεια συναλλαγών σε τράπεζες δίκτυα-ATM
- Κινητή τηλεφωνία(GSM)
- Σταθερή τηλεφωνία(cryptophones)
- Διασφάλιση εταιρικών πληροφοριών
- Στρατιωτικά δίκτυα
- Διπλωματικά δίκτυα
- Ηλεκτρονικές επιχειρήσεις(πιστωτικές κάρτες, πληρωμές)
- Ηλεκτρονική ψηφοφορία
- Ηλεκτρονική δημοπρασία
- Ηλεκτρονικό γραμματοκιβώτιο(email)
- Συστήματα συναγερμών
- Συστήματα βιομετρικής αναγνώρισης
- Έξυπνες κάρτες
- Ιδιωτικά δίκτυα
- World Wide Web
- Δορυφορικές εφαρμογές και ασύρματα δίκτυα
- Συστήματα ιατρικών δεδομένων και άλλων βάσεων δεδομένων
- Τηλεδιάσκεψη(VOIP)

ΚΕΦΑΛΑΙΟ3
ΑΛΓΟΡΙΘΜΟΙ ΡΟΗΣ

3.1 Αλγόριθμοι ροής και ιδιότητες

Οι **κρυπτογραφικοί αλγόριθμοι ροής (stream ciphers)** χρησιμοποιούνται για την κρυπτογράφηση μίας συνεχούς ροής δεδομένων (data stream).Για την κρυπτογράφηση επιλέγεται αρχικά μία γεννήτρια κλειδοροής (keystream generator), η οποία δέχεται ως είσοδο το μυστικό κλειδί και παράγει στην έξοδό της μία ψευδοτυχαία ακολουθία bits, η οποία ονομάζεται κλειδοροή (keystream). Στην συνέχεια εφαρμόζεται η συνάρτηση XOR ανάμεσα στο αρχικό κείμενο και στην κλειδοροή και το αποτέλεσμα της συνάρτησης είναι η τελική κρυπτογραφημένη ροή δεδομένων.(**Εικόνα 8**)



Εικόνα 8:Κρυπτογράφηση με αλγόριθμο ροης

Η αποκρυπτογράφηση γίνεται με την ακριβώς αντίστροφη διαδικασία. Εάν χρησιμοποιηθεί το ίδιο κλειδί ως είσοδο στην γεννήτρια κλειδοροής, τότε η δεύτερη θα παράγει ακριβώς την ίδια ακολουθία bits (κλειδοροή) όπως και προηγουμένως κατά την διαδικασία της κρυπτογράφησης. Εφαρμόζοντας την συνάρτηση XOR ανάμεσα στην κρυπτογραφημένη ακολουθία δεδομένων και την κλειδοροή παράγεται τελικά το αρχικό κείμενο.

Για να είναι ασφαλής ο κρυπτογραφικός αλγόριθμος ροής, θα πρέπει να πληρούνται ορισμένες προϋποθέσεις όσον αφορά την γεννήτρια κλειδοροής και την ψευδοτυχαία ακολουθία bits που αυτή παράγει. Συγκεκριμένα η ασφάλεια του αλγορίθμου εξαρτάται από τις εξής παραμέτρους:

- Η ψευδοτυχαία ακολουθία bits που παράγεται από την γεννήτρια κλειδοροή θα πρέπει να έχει αρκετά μεγάλη περίοδο επανάληψης.
- Η ακολουθία bits της κλειδοροής θα πρέπει να μοιάζει πολύ με τυχαία
- Η κλειδοροή θα πρέπει να έχει μεγάλη γραμμική ισοδυναμία

3.2 Ο αλγόριθμος μιάς χρήσης

Στηνκρυπτογραφία αλγόριθμο μιας χρήσης**one time pad**(OTP) ή αλλιώς κρυπτογραφική τεχνική OTP[12] ονομάζουμε ένα σύστημα κατα το οποίο ένα μυστικό κλειδί γεννίεται μέσω μιας γεννήτριας κλειδιών και χρησιμοποιείται μόνο μια φορά για να κρυπτογραφήσει ένα μήνυμα το οποίο συνήθως αποκρυπτογραφείται απο τον

[12]Η κρυπτογραφική τεχνική OTP, https://en.wikipedia.org/wiki/One-time_pad

παραλήπτη του μηνύματος χρησιμοποιώντας κάθε φορά το κατάλληλο μπλόκ και το κλειδί.

Τα μηνύματα τα οποία κρυπτογραφούνται βασιζόμενα σε τυχαία κλειδιά θεωρούνται αδύνατον να σπάσουν.Κάθε κρυπτογράφηση είναι μοναδική και δέν έχει καμία απολύτως σχέση με τις επόμενες κρυπτογραφήσεις.Με τον OTPέχουμε το **πρόβλημα** της ασφαλούς μετάδοσης του κλειδιού, μιας και ο αποστολέας και ο παραλήπτης χρησιμοποιούν **το ίδιο κλειδί** για την κρυπτογράφηση απο τη μεριά του αποστολέα και την αποκρυπτογράφηση απο την πλευρά το παραλήπτη.

Εφευρέθηκε το 1882 απο τον FrankMiller[13]ο οποίος χρησιμοποιήσε το OTPσύστημα για την ασφαλή μετάδοση της πληροφορίας μέσω τηλεγραφίας,ενώ επαναχρησιμοποιήθηκε και πατενταρίστηκε απο τον GilbertVernam[14](VernamCipher).Η τελική ανακάλυψη και ουσιαστική βελτιώση του συστήματος οφείλεται στον Shannon, γύρω στο 1940.

Σύμφωνα με τις βελτίωσειςη κρυπτογράφηση του μηνύματος γίνεται χρησιμοποίωντας πράξη XOR μεταξύ του μηνύματος και του κλειδιού ενώ η αποκρυπτογράφηση γίνεταιπολύ απλά κάνοντας πράξη XORμεταξύ του ciphertextκαι του κλειδιου.Είναι γρήγορος και ασφαλής αλγόριθμος μόνο έαν το μήκος του κλειδιού είναι μεγαλύτερο ή ίσο με το μήκος του μηνύματος προς αποστολή.Αυτό δεν μας συμφέρει άρα είναι πολύ δύσκολο να χρησιμοποιηθεί στην πράξη.Για ένα OTP σύστημα θα λέμε ότι έχει τέλεια ασφάλεια έαν ισχυέι οτι :

$$\forall m,c: \text{αν } E(k,m)=c \Longrightarrow k \oplus m=c \Longrightarrow k=m \oplus c \Longrightarrow \#\{k \in K : E(k,m)=c=1\}$$

Η ιδέα ειναι να χρησιμοποιήσουμε το κλειδί μας ως σπορά(seed) και μια γενήτρια αριθμών για να επεκτείνουμε το κλειδί μας σε μια πολυ μεγαλύτερη και πιό τυχαία ακολουθία αριθμων.Θα θεωρήσουμε μια γεννήτρια που παράγει αυτές τις τυχαίες ακολουθίες και θα την ονομάσουμε G(k).Θα κάνουμε πράξη XOR μεταξύ της G(k) και του μηνύματος,όπως ακριβώς κάναμε και στον αλγόριθμο OTPκαι θα μας δώσει ως αποτέλεσμα το ciphertext του μηνύματος μας ενώ με την αντίστροφη διαδικασία γίνεται η αποκρυπτογράφηση.Δηλαδή:

$$c=E(m,k)=m \oplus G(k)$$
$$D(k,c)=c \oplus G(k).$$

3.3 Γεννήτριες Ψευδοτυχίας

3.3.1 Αδύναμες Γεννήτριες ψευδοτυχίας

Μια **γεννήτρια τυχαίων** αριθμών είναι μια υπολογιστική ή μηχανική συσκευή που έχει σχεδιαστεί για να παράγει μια ακολουθία αριθμών ή συμβόλων που δεν ακολουθούν κάποιο μοτίβο, δηλαδή εμφανίζονται τυχαία.Υπάρχουν αρκετές υπολογιστικές μέθοδοι για την παραγωγή τυχαίων αριθμών, πολλές όμως δεν μπορούν να παράγουν πραγματικά τυχαίους αριθμούς.Ωστόσο, υπάρχουν προσεκτικά σχεδιασμένα υπολογιστικά συστήματα που παράγουν τυχαίους αριθμούς με ασφάλεια, όπως αυτά που βασίζονται στον αλγόριθμο Yarrow, τον Fortuna [15](PRNG) και άλλους.Οι γεννήτριες αριθμών, χρησιμοποιούνται επίσης στην **κρυπτογραφία**όταν η **σπορά**(seed) είναι μυστική.

---

[13]OFrank Miller, https://en.wikipedia.org/wiki/Frank_Miller_(cryptography)
[14]OGilbert Vernam, http://en.wikipedia.org/wiki/Gilbert_Vernam
[15]Η γεννήτρια ψευδοτυχίας αριθμών Fortuna, https://en.wikipedia.org/wiki/Fortuna_(PRNG)

Οσο αφορά την κρυπτογραφία η δημιουργία γεννητριών τυχαίων αριθμών γίνεται με την χρήση υπολογιστικών αλγορίθμων που μπορούν να παράγουν μεγάλες σειρές φαινομενικά τυχαίων αποτελεσμάτων, τα οποία είναι στην πραγματικότητα πλήρως προκαθορισμένα από μια μικρότερη αρχική τιμή(σπορά).Αυτοί οι αλγόριθμοι ονομάζονται **γεννήτριες ψευδοτυχαίων αριθμών.**Οι γεννήτριες ψευδοτυχαίων αριθμών (PRNGs) είναι αλγόριθμοι που μπορούν να δημιουργήσουν αυτόματα μεγάλες σειρές αριθμών με καλές ιδιότητες τυχαιότητας, όμως η σειρά σε κάποιο σημείο αναγκάζεται να επαναληφθεί.

3.3.2 Στατιστικά τεστ[16]

Θα προσπαθήσουμε να περιγράψουμε **τι είναι ένα στατιστικό τεστ**.Ενα στατιστικό τέστ$\in \{0,1\}^n$είναι ένας αλγόριθμος (εστω A) ο οποίος παίρνει ώς είσοδο μια συμβολοσειρά μήκους Nbits(έστω x) και απλά μας δίνει μια έξοδο μεταξύ 0 και 1.Θα θεωρήσουμε ότι αν το τεστ βγάλει ώς έξοδο 0 τότε η είσοδος που δώσαμε στο τέστ δέν είναι τυχαία.Ενω αν μας βγάλει ώς έξοδο το 1 τότε η είσοδος που του δώσαμε στο τέστ θα θεωρείται τυχαία.Επίσης το στατιστικό τέστ ανήκει στο.Μερικα παραδείγματα παρακάτω:

Παράδειγμα 1

$$\mathbf{A(x)=1\ εαν\ |\ \#0(x)-\#1(x)| \leq 10\cdot \sqrt{n}}$$

Το στατιστικό μας τέστ θα μας βγάλει έξοδο 1 εαν ο αριθμός των μηδένικών δοσμένης συμβολοσειράς χμειον τον αριθμό των άσσων δοσμένης της ίδιας συμβολοσειράς είναι μικρότεροςή ίσος απο 10 φορές την τετραγωνική ριζα του n τότε η συμβολοσειρά είναι τυχαία.

Παράδειγμα 2

$$\boldsymbol{A(x)=1\ εάν\ |\ \#00(x)-\frac{n}{4}| \leq 10\cdot\sqrt{n}}$$

Ο αριθμός των ζευγαρίων {0,0} που θα βρούμε στη συμβολοσειρά μας μείον το n/4 να είναι μικρότερο ή ίσο απο 10 φορές την τετραγωνική ρίζα του n.

Παράδειγμα 3

$$\boldsymbol{A(x) = 1 \quad max\text{-}run\text{-}of\text{-}0(x) \leq 10\cdot log_2 N.}$$

Δηλαδητομπλόκ που έχει την μεγαλύτερη ακολουθία απο μηδενικά πρέπει το πλήθος τους να είναι μικρότερο ίσο απο τον αριθμο $\log_2(n)$.Για παράδειγμα έαν όμως δώσουμε παραδείγματος χάρη την συμβολοσειρά 1111 το τέστ θα μας βγάλει ώς έξοδο 1.Δηλαδή θα θεωρήσει οτι η συμβολοσειρά μας είναι τυχαία ενώ πραγματικά δέν είναι!Επειδή υπάρχουν εκατοντάδες στατιστικά τέστ θα πρέπει να βρούμε ένα τρόπο για να **κρίνουμε** εάν ένα τέστ είναι καλό ή οχι.

Για να γίνει αυτό θα πρέπει να εισάγουμε την έννοια του **πλεονεκτήματος**(advantage).Εστω μια γεννήτρια ψευδοτυχίας $G:\{0,1\}^n$ και ένα στατιστικό τέστ στο $\{0,1\}^n$.Ορίζουμε ως **πλεονέκτημα** της γεννήτριας, το πλεονέκτημα του στατιστικού τέστ A σε σχέση με τηνG, δηλαδη την διαφορά της πιθανότητας του στατιστικού τέστ να βγάλει έξοδο 1 μείον την πιθανότητα του

[16] Τα στατιστικά τέστ, https://en.wikipedia.org/wiki/Statistical_hypothesis_testing

στατιστικού τέστ να βγάλει 1 ότανδίνεται μια πραγματικά τυχαία συμβολοσειρά(**Εικονα 9**).

Αν το αποτέλεσμα είναι **κοντά** στο 1 τότε το στατιστικό τέστ μπορεί να διακρίνει την έξοδο της Gόσο τυχαία και αν είναι αυτη.Αρα το τέστ αυτό σαν να λέμε οτι «σπάει» την γεννήτρια G.Το ακριβώς αντίθετο συμβαίνει έαν η έξοδος μας είναι 0.Δεν μπορεί το τέστ να διακρίνει αν η έξοδος της γεννήτριας είναι τυχαία ή ψευδοτυχαία τελικά.

$$Adv_{PRG}(A,G) = \left| P[(A,G(k)]=1 - P[A,G(r)]=1 \right| \quad \text{ανήκει } [0,1]$$

$k \xleftarrow[R]{} K$

Εικόνα 9:Το πλενέκτημα ενός αντίπαλου

Παράδειγμα

Έστω ότι έχουμε μια γεννήτρια G:K→$\{0,1\}^n$η οποία ικανοποιεί μια ιδιότητα.Τυχαίνει στα 2/3 των κλειδιών της γεννήτριας, το πρώτο bitης εξόδου να είναι 1, δηλαδή msb(G(k))=1.Εάν επιλέξω ένα τυχαίο κλειδί έχω πιθανότητα 2/3 να μου βγάλει το πρώτο bittου κλειδιού να είναι 1.Ορίζουμε και ένα στατιστικό τέστ το οποίο μας λέει ότι θα βγάζει ώς εξοδο 1 εάντο πιο σημαντικό bitης συμβολοσειράς που θα του δώσουμε είναι 1 και έαν το πιο σημαντικό bitης συμβολοσειράς που θα το δώσουμε δέν είναι 1, τότε θα βγάζει ώς έξοδο 0:

if[msb(x)=1] output 1, elseoutput 0

Η ερώτηση μας είναι ποιό είναι το πλεονέκτημα του στατιστικού τέστ πάνω στην γεννήτρια G;Δηλαδή:

$$\mathbf{Adv_{PR}[A,G]=| Pr[A(G(k))=1]-Pr[A(r)=1] |}$$

Απάντηση

Η Pr[A(G(k)))=1]έχει τιμή ακριβώς 2/3.Ας δούμε για την Pr[A(r)]=1.Εάν μας δωθεί μια τυχαία ακολουθία , πόσο πιθανό είναι ότι το πιο σημαντικό bitης θα είναι 1;Για μια τυχαία ακολουθία συμβαίνει ακριβώς στον μισό χρόνο δηλαδή 1/2 .Αρα το αποτέλεσμα μας είναι 2/3-1/2=1/6 το οποίο δέν ειναι αμελητέο.

3.3.3 Ασφαλείς γεννήτριες ψευδοτυχίας

Τωρα που γνωρίζουμε τι είναι ένα στατιστικό τέστ και τι πλεονέκτημα αυτου μπορούμε πλέον να δουμε τι προϋποθέσεις πρέπει να πληροί για να θεωρείται μια γεννήτρια ασφαλής.Έστω μια G:→ $\{0,1\}^n$.Η γεννήτρια θα είναι ασφαλής **εάν**για κάθε στατιστικό τέστ A(αποδοτικό τέστ), το πλεονέκτημα ώς προς τηGθα είναι αμελητέο ή κοντά στο μηδέν.Με άλλα λόγια το στατιστικό τέστ δεν θα πρέπει να είναι ικανό να προβλέψει την έξοδο τηςγεννήτριας.Μια γεννήτρια πρέπει να είναι μη προβλέψιμη.Θα αποδείξουμε οτι αν η γεννήτρια μας είναι προβλεπόμενη τότε δέν είναι ασφαλής.

Αποδειξη

Εστω οτι μας δίνεται ένας αποδοτικός αλγόριθμος στον οποίο αν δώσω την έξοδο της γεννήτριας A(G(k))για τα πρώτα ibitstης εξόδου,αυτός θα μπορέσει να προβλέψει το επόμενο bitης εξόδου με πιθανότητα 1/2 +ϵ,για ε μη αμελητεο(ε=1/1000).Θα χρησιμοποιήσουμε αυτό τον αλγόριθμο για να «σπάσουμε» τη γεννήτρια μας.Να δείξουμε δηλαδή οτι η γεννήτρια μας είναι διακρίσιμη άρα είναι και μή

ασφαλής.Αςορίσουμε ένα στατιστικό τέστ Β.Το τέστ παίρνει τιμή 1 όταν μπορεί να προβλέψει την έξοδο, ενώ παίρνει 0 όταν δέν μπορεί να προβλέψει την έξοδο.

Έστω οτι δίνεται μια τυχαία συμβολοσειρά.Ρωτάμε ποιά είναι η πιθανότητα το τέστ να μας βγάλει 1.Η πιθανότητα αυτή είναι ακριβώς 1/2 .Εστω οτι δίνουμε μια ψευδοτυχαία ακολουθία και ρωτάμε το ίδιο.Εξ ορισμού, οταν δίνουμε τα πρώταibitsτης γεννήτριας μπορεί να προβλέψει το επόμενο bitμε πιθανότητα $1/2+\epsilon$.Για εμάς μεγαλύτερο απο $1/2+\epsilon$.Αν κοιτάξουμε το πλεονέκτημα του στατιστικού μας τέστ θα δούμε ότι είναι μεγαλύτερο απο ε.

Αυτο σημαίνει οτι αν ο αλγόριθμος Α είναι ικανός να προβλέψει το επόμενο bitμε πλεονέκτημα ϵ, τοτε ο αλγόριθμος Β(στατιστικο τεστ) είναι ικανός να διακρίνει την έξοδο της γεννήτριας μια πλεονέκτημα ϵ.Αν ο Α είναι καλός αλγόριθμος πρόβλεψης τοτε το Β είναι ικανό να «σπάσει» τη γεννήτρια.Αρα αν η G είναι ασφαλής γεννήτρια τοτε δεν θα πρέπει να υπάρχουν αλγόριθμοι πρόβλεψης■

3.4Επιθέσεις στους Αλγορίθμους Ροής

Το προτζεκτ της Verona[17]

Το πρότζεκτ της Veronaήταν ένα πρόγραμμα αντικατασκοπείας το οποίο συντονίστηκε απο το στρατό των ηνωμένων πολιτειών της αμερικής και είχε διάρκεια απο το 1943 έως το 1980.Το πρόγραμμα αυτό είχε σκοπό να αποκρυπτογραφήσει τα μηνύματα της κατασκοπείας της Σ.Ε.Κατα την διάρκεια αυτού του προγράμματος περίπου τρείς χιλίαδες μηνύματα αποκρυπογραφήθηκαν και μεταφράστηκαν.Αυτο έγινε διότι οι σοβιετικοί χρησιμοποιήσαν το onetimepadσύστημα αλλα χρησιμοποιώντας το ίδιο κλειδί.

MS-PPTP(WindowsNT)

Εκεινη την εποχή η επικοινωνία με αυτόν τον τρόπο ήταν ευάλωτη διότι ένας υπολογιστής επικοινωνούσε με εναν σέρβερ αλλα το πρόβλημα ειναι ότι τα μηνύματα του πελάτη κρυπτογραφούνταν με τοίδιο seedτης ψευδοτυχίας αριθμων με αυτήν του σέρβερ με αποτέλεσμα,ουσιαστικά να χρησιμοποιείται το ίδιο κλειδί αλγορίθμου ροής.Το σωστό είναι να έχουμε ένα κλειδί για τον κάθε υπολογιστή ξεχωριστά.

802.11bWEP

Στο WEPυπάρχει ενας πελάτης και ένα σημείο πρόσβασης(accespoint).Και τα δύο άκρα μοιράζονται το ίδιο κλειδί.Έστω ότι θέλει να στείλει ένα μήνυμα ο πελάτης.Για να συμβεί αυτό,ο πελάτης,πρέπει να στείλει ενα frame το οποίο εμπεριέχει το plaintext Μ.Επίσης πρίν το στείλει προσθέτει και ένα checksumστο μήνυμα.Το μήνυμα και το checksumκρυπτογραφούνται χρησιμοποιώντας εναν αλγόριθμο ροήςοπου το κλειδί του αλγορίθμου ροής ειναι η αλληλουχία της IVκαι του μόνιμου κλειδιού k.Το IVείναι μια συμβολοσειρά εικοσιτεσσάρων bitπου ξεκινάει απο το μηδέν(μπορεί πιθανόν να είναι ένας counterπου να μετράει τον αριθμό των πακέτων μηνυμάτων που στάλθηκαν).

Οισχεδιαστές του WEPθεώρησαν οτι πρέπει να κρυπτογραφούν μόνο το ένα μήνυμα άρα κάθε frameθα κρυπτογραφείται με διαφορετικό κλειδί,χρησιμοποιώντας πάλι το

[17]Το πρότζεκ της Verona, http://en.wikipedia.org/wiki/Venona_project

IV πάνω στο νέο κλειδί.Οπότε ο παραλήπτης γνωρίζει το κλειδί,το IV,και τώρα μπορεί να επαναπροσαρμόσει την γεννήτρια ψευδοτυχαίων αριθμών του κατάλληλα,και να πάρει το μήνυμα.Το **πρόβλημα** είναι όμως,οτι το IVείναι μονο εικοσιτέσσερα bitμακρύ,με αποτέλεσμα να υπάρχουν μόνο 2^{24}IV'S,δηλαδή δεκαέξι εκατομμύρια frames.Ομως κάθε εξήντα εκατομμύρια frames το IVκάνει έναν κύκλο,αρα το ίδιο IVθα χρησιμοποιηθεί για να κρυπτογραφήσει το ίδιο μήνυμα!Αρα ο επιτιθέμενος μπορεί να πάρει αυτά τα δύο μηνύματα(γνωρίζει φυσικά και το μόνιμο κλειδί που δέν αλλάζει ποτέ.).

Μια αλλη επίθεσημας δείχνει οτι ο OTPκαι οι αλγόριθμοι ροής γενικότερα δεν μας παρέχουν ακεραιότηταλλα μας παρέχουν εχεμύθεια.Αυτη η επίθεση ονομάζεται **malleability**.Υποθέστε οτι έχουμε ένα μήνυμα το οποίο κρυπτογραφείται μέσω ενός αλγορίθμου ρόης και παράγει ενα **ciphertext=m⊕k**.Ο επιτιθέμενος δεν μπορεί να βρέι το μήνυμα καθώς γνωρίζει μόνο το ciphertext.Αν αποφασίσει όμως να γίνει πιο δραστικός στις επιθέσεις του,μπορεί να αλλάξει αυτό το ciphertextκάνοντας XORτο ciphertextπου έλαβε μαζι με μια τιμή p(subpermutationkey).

Το αποτέλεσμα αυτών θα ειναι **(m⊕k) ⊕p**.Οταν αυτό το μήνυμα το στέιλουμε πίσω στον παραλήπτη για αποκρυπτογράφηση, το αποτέλεσμα που θα λαβει ο παραλήπτης θάναι **m⊕p**.Δηλαδή ο επιτιθέμενος κατάφερε να τροποποιήσει το περιεχόμενο του μηνύματος!Εκτός απο τον αλγόριθμο OTPυπάρχουν και άλλοι αλγόριθμοι ροής οι οποίοι **δεν θεωρούνται ασφαλείς**.Δύο απο αυτούς είναι ο**RC4**και ο **CSS**.Στην συνέχεια θα προσπαθήσουμε να τους αναλύσουμε και να δούμε γιατί δεν πρέπει να τους χρησιμοποιούμε στις μέρες μας.

Αλγόριθμος RC4(1987)

Ο αλγόριθμος RC4[18] ειναι πιο συχνά χρησιμοποιούμενος αλγόριθμος ροής καθως έχει χρησιμοποιηθεί στα πιο δημοφιλή πρωτόκολλα ίντερνετ οπως το TLS[19]και το HTTPS[20]και WEP.Δημιουργήθηκε απο τονRonRivest[21]της RSASecurityτο 1987.Ο τροπος λειτουργίας του είναι ο εξής:

Keyscheduling

Έστω ένας πίνακας S.Χρησιμοποιούμε αυτή τη διαδικασία για να αρχικοποιήσουμε τον πίνακα S.Το μήκος του κλειδιού ορίζεται απο τον αριθμό των bytesπου θα έχει το κλειδί και μπορεί να έχει μέγεθος απο 1 εώς 256 bytes.Συνήθως έχουμε 5-16 bytes που μας δίνει μήκος κλειδιού 40-128 bits.Ο αλγόριθμος φαίνεται παρακάτω :

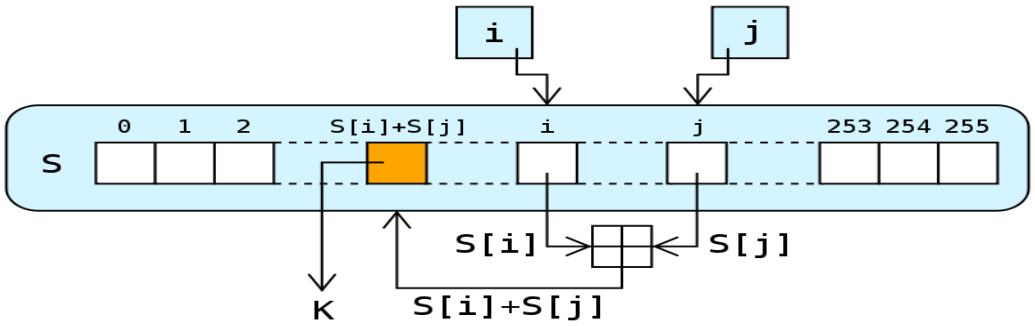
```
for i from 0 to 255
S[i]:= i
Endfor
j := 0
for i from 0 to 255
j := (j + S[i] + key[i mod keylength]) mod 256
swap values of S[i] and S[j]
endfor
```

Pseudo-random generation algorithm (PRGA)

[18]Ο αλγόριθμος RC4, http://en.wikipedia.org/wiki/RC4
[19] Το πρωτόκολλο TLS, https://en.wikipedia.org/wiki/Transport_Layer_Security
[20] Το πρωτόκολλο HTTPS, https://en.wikipedia.org/wiki/HTTPS
[21] Ο κρυπτογράφος RonRivest, https://en.wikipedia.org/wiki/Ron_Rivest

Για όσες επαναλήψεις χρειαστεί η PRGAμε ανάλογη τροποποίηση εξάγει ένα bytetης κλειδοροής.Σε κάθε επανάληψη η PRGAαυξάνει το ικατα 1 και ψάχνει το ιστοιχείο του πίνακα Sδηλαδή το S[i] και το προσθέτει stoj.Στη συνέχεια εναλλάσει τις τιμές των S[i], S[j] και χρησιμοποιεί το άθροισμα S[i]+S[j] mod 256, σαν δείκτη για να φέρει το τρίτο στοιχείο του πίνακα Στο οποίο θα γίνει πράξη XORμε το επόμενο bytetου μηνύματος για να παράγει το επόμενο bytetου ciphertextή plaintext.Κάθε στοιχείο του S, εναλλάσεται με το επόμενο στοιχείο το λιγότερο 256 φορές.Η παραπάνω διαδικασία φαίνεται απο τον παρακάτω αλγόριθμο αλλα και στο σχήμα (**Εικόνα 10**)

```
i := 0
j := 0
while GeneratingOutput:
       i := (i + 1) mod 256
       j := (j + S[i]) mod 256
       swap values of S[i] and S[j]
       K := S[(S[i] + S[j]) mod 256]
       output K
endwhile
```



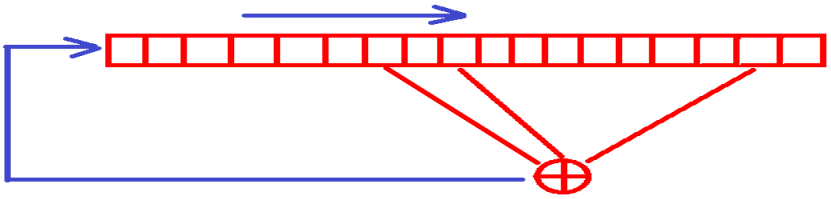
Εικόνα 10:Αλγόριθμος RC4

Παρολα αυτά ομως το ο αλγόριθμος RC4 **παρουσίαζει αδυναμίες** και δεν θα πρέπει να χρησιμοποιείται πλέον:

- **Προδιάθεση πιθανότητας του δευτερου bit**.Το δεύτερο byte της εξόδου του RC4 έχει μια προδιάθεση πιθανότητας κοντά στο μηδέν με πιθανότητα το δεύτερο byte να έχει πιθανότητα μηδέν είναι 1/128.Αν ο RC4 ήταν απολύτως τυχαίος, η πιθανότητα θα ήτανε 2/256 μίας και υπάρχουν 256 πιθανά bytes.Αυτο σημαίνει οτι άν χρησιμοποιούμε τον RC4 το δέυτερο byte δεν θα κρυπτογραφηθεί.Αυτή η προδιάθεση συμβαίνει διότι στην κανονική κατάσταση του τρίτου στοιχείου είναι μηδέν και αφού το δεύτερο byte δέν είναι ίσο με δύο, τοτε η έξοδος του δεύτερου byte θα είναι πάντα μηδέν.Αν θέλετε να χρησιμοποιήσετε τον RC4 θα πρέπει να αγνοήσετε τα πρώτα 256 bytes ξεκινώντας απο το 257 byte.
- **Πιθανότερη ακολουθία η 00**.Αν πάρουμε μια μεγάλη σε μήκος έξοδο του RC4 θα δείτε οτι η πιό πιθανή ακολουθία που θα συναντήσετε είναι η 00.Αρα η πιθανότητα να πάρουμε την ακολουθία 00 είναι ακριβώς 1/2562.Επειδή υπάρχει και μια προδιάθεση πιθανότητας προσθέτουμε στη συνολική πιθανότητα 1/2563.Αυτή η αδυναμία μπορεί να κάνει την γεννήτρια μας αρκετά προβλεπόμενη.

Αλγόριθμος CSS

OCSS[22]είναι ένα σύστημα κρυπτογράφησης και χρησιμοποιήθηκε με σκοπό να κρυπτογραφήσειDVDδισκους.Το σύστημα αυτό τοποθετούνταν πάνω σε hardware.Βασίζονταν σε ένα μηχανισμο που λέγεται LFSR[23].Ενα LFSRείναι ένας καταχωρητής που περιέχει κελιά όπου το κάθε κελί αντιπροσωπεύει και ένα bit.Η θέση κάθε κελιού ονομάζεται tap(**Εικόνα 11**).



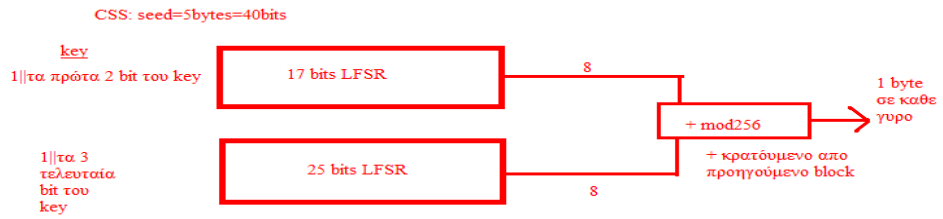
Εικόνα 11:XOR πράξη σε LFSR

Αυτά τα tapsδίνονται σε μια XOR και σε κάθε κύκλου του ρολογιού ο καταχωρητής ολισθαίνει αριστερά.Το τελευταίο bitβγαίνει εκτός και το πρώτο bittου αποτελέσματος της πράξης XORγίνεται το πρώτο bittου LFSR.Ο μηχανισμός αυτός είναι πολυ εύκολος να ενσωματωθεί σε ένα hardwareμε πολύ λίγα τρανζίστορς.Η σπορά για αυτό το LFSRείναι η αρχική κατάσταση του LFSR.Τα LFSR'Sείναι η βάση αρκετών αλγορίθμων ροής πχ ο CSSαπαιτεί 2 LFSR, το GSM[24]χρειάζεται 3 LFSRενω το wifi[25]5 LFSR.

Το seedτου αλγορίθμου είναι 5 bytesή αλλιώς 40 bits.Ο CSSχρησιμοποιεί 2 LFSRS.Το ενα LFSRεχει χωρητικότητα 17bitsενω το άλλο 25 bits.Η αρχική κατάσταση του 17bitsLFSR, είναι οαριθμός 1(σε bits)και τα 2 πρώτα bytestου κλειδιού κρυπτογράφησης ενώ η αρχική κατάσταση του 25bitsLFSR,είναι ο αριθμός 1(σε bit) και τα 3 τελευταία bytestου κλειδιού.

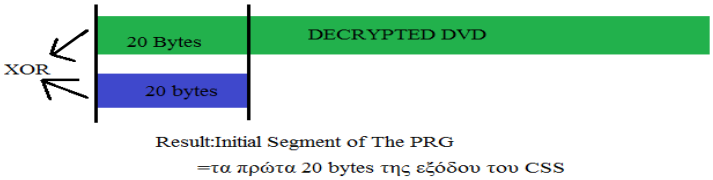
Αυτα τα δύο LFSRτρέχουν για 8 κύκλους επεξεργασίας και δημιουργούν 8bits ώς έξοδο ο κάθε ένας.Αυτα τα 16 bitsαθρόιζονται και μετα περνάνε απο μια πράξη mod256 σύν ένα κρατούμενο απο το προηγούμενο μπλόκ.Πάντος σε κάθε πράξη άθροισης και έπειτα moduloαυτή της πράξης,το κρατούμενο αυτό μπορεί να είναι 0 ή 1 και δίνεται στον επόμενο γύρο.Η έξοδος όλης αυτής της διαδικασίας είναι 1 bytεσε κάθε κύκλο.Αυτό το byteγίνεται XOR με το υποτιθέμενο bytetης ταινίας που βρίσκεται στο DVD.Μετά το πέρας όλης της διαδικασίας όλη η ταινία θα είναι κρυπτογραφημένη.Η διαδικασία φαίνεται στην παρακάτω εικόνα(**Εικονα 12)**

[22]Ο αλγόριθμος CSS, http://en.wikipedia.org/wiki/Content_Scramble_System
[23] Γραμμικός καταχωρητής ολίσθησης, https://en.wikipedia.org/wiki/Linear_feedback_shift_register
[24] Το στάνταρ GSM, https://en.wikipedia.org/wiki/GSM
[25] Η τεχνολογία Wifi, https://en.wikipedia.org/wiki/Wi-Fi



Εικόνα 12:Αλγόριθμος CSS

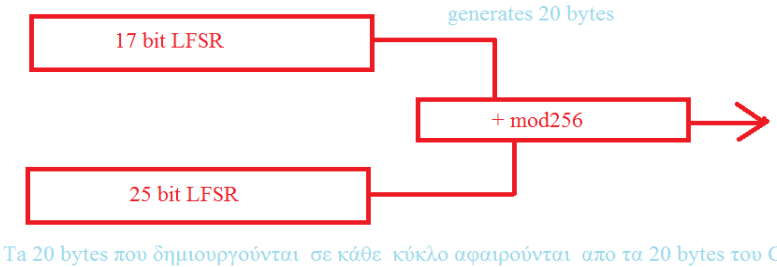
Παρόλααυτά όμως αυτά ο CSSείναι πολύ έυκολος να **σπάσει** σε χρόνο περίπου 2^{17}. Έστω ότι έχουμε την κρυπτογραφημένη ταινία και θέλουμε να την αποκρυπτογραφήσουμε.Επειδή τα DVD χρησιμοποιούν MPEG αρχεία , συμβαίνει ότι έαν γνωρίζουμε ενα πρόθεμα(prefix) του plaintext(έστω ότι έχει μέγεθος 20 bytes) και κάνουμε XORαυτά τα 20 bytesμε τα 20 πρώτα bytestου ciphertext, τότε αμέσως θα γνωρίζουμε την **αρχικο segment**της γεννήτριας ψευδοτυχίας.(**Εικόνα 13**)



Εικόνα 13:Ευρεση αρχικής κατάστασης της γεννήτριας

Αυτό που κάνουμε είναι να δοκιμάζουμε όλες τις 2^{17} πιθανές τιμές του πρώτου LFSR για 20 bytes.Αρα θα έχουμε τελικά 20 bytesώς έξοδο απο τον πρώτο LFSR.Κατόπιν παίρνουμε τα 20 bytetης εξόδου του CSSσυστήματος (**Εικονα 14**τα οποία γνωρίζουμε και τα αφαιρούμε απο τα 20 bytestης εξόδου του πρώτου LFSR.Με αυτόν τον τρόπο θα πάρουμε τα πρώτα 20 bytestου δέυτερου LFSR.Τώρα είναι εύκολο να αναγνωρίσουμε αν αυτή η ακολουθία των 20 bytesαπο ποιόνLFSR ήρθε.

Αν δέν ήρθαν απο τον 25bitLSFRτότε ξαναδοκιμάζουμε απο την αρχή για τον πρώτο 17bitLFSR.Η διαδικασία συνεχίζεται μέχρι να βρούμε την αρχική κατάσταση του πρώτου LFSR.Ετσι θα μάθουμε και την αρχική κατάσταση του δέυτερου LFSRκαι έτσι μπορούμε να προβλέψουμε το υπόλοιπο κομμάτι του CSSτο οποίο θα χρησιμοποιηθεί για την αποκρυπτογράφηση του υπόλοιπου plaintext.



Εικόνα 14: Αφαιρεση 20bytestου πρωτου LFSRαπο τα 20CSSbytes

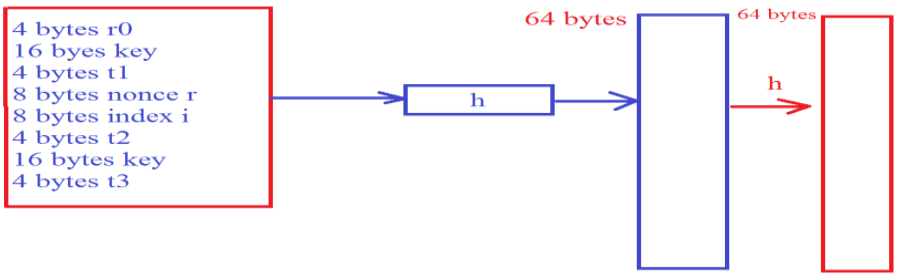
Υπάρχουν όμως μοντέρνοι αλγόριθμοιροής που **θεωρουνται ασφαλείς στις μέρες** μας.Μια κατηγορία αλγορίθμων ειναι οι **eStream**[26].Οι σύγχρονοι αυτοί αλγόριθμοι δεν χρησιμοποιούν απλά μια γεννήτρια ψευδοτυχιας αλλάεμπεριέχεταισε αυτήν και μιά τιμή Rπου λέγεται nonceκαι δέν αλλάζει ποτέ για οποιοδήποτε δωσμένο κλειδί.Δηλαδη ισχύει $\{0,1\}^s \cdot R \rightarrow \{0,1\}^n n \gg s$.Τώρα πλέον η διαδικασία κρυπτογράφησης έχει ώς εξης $E(k,m;r)=m \oplus PRG(k;r)$.Οπότε το ζεύγος (k,r) δεν χρησιμοποιείται για πάνω απο μία φορά.

Ένα παράδειγμα αλγοριθμου τυπο streamειναι ο **Salsa20**.OSalsa20 παίρνει ως είσοδο κλειδί 128ή 256 bitκαι ένα nonceμήκους 64 bits.Για πιό απλή εξήγηση του αλγορίθμου ας πάρουμε κλειδί των 128bit.Η συνάρτηση λειτουργίας του αλγοριθμου είναι :

$$H(k,r(0))||H(k,r(1))||\ldots$$

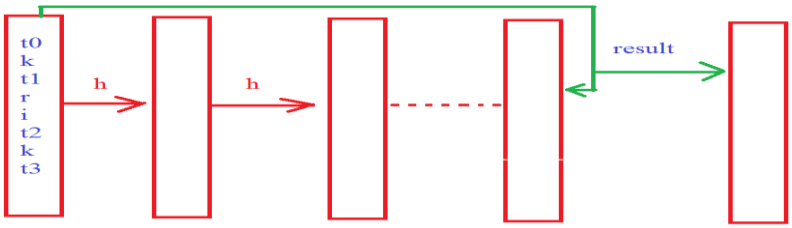
Η συνάρτηση αυτή δέχεται 3 παραμέτρους.Το kπου υποδηλώνει το κλειδί της κρυπτογράφησης,το rπου δηλώνει το nonce, μια τρίτη παράμετρος που είναι ουσιαστικά ένας μετρητής(counter) που αυξάνεται κατά 1 κάθε φορα.Η λειτουργία της συνάρτησης είναι αρκέτα απλη.Αρχικά θα επεκτείνουμε τις καταστάσεις k,r,a σε ένα μεγάλο μπλόκ των 64 bits.Στο μπλόκ αυτό δέν έχουμε μόνο αυτές τις 3 παραμέτρους αλλα και 3 σταθερές(t_1,t_2,t_3), και έναν δείκτη i.

Μέσω μιας αντιστρέψιμης συνάρτησης hόλες αυτές οι τιμές πηγαίνουν σε ένα επόμενο μπλοκ των 64bytesκαι τα δωσμένα στοιχεία του κάθε μπλόκ αντιστρέφονται, και παίρνονται ώς είσοδο στο επόμενο 64 bytesμπλόκ μέσω της hκαι αυτή η διαδικασία συνεχίζεται.(**Εικονα 15**).



Εικόνα 15:Η λειτουργια της συναρτησης h

Έτσι όπως φαίνεται η συνάρτηση δε παράγει τελείως τυχαίες ακολουθίες αριθμών καθώς, αν δωθεί το τελευταίο μπλόκ μπορούμε πολύ έυκολα να μαντέψουμε το αρχικό μπλόκ.Επίσης υπάρχει ένα τελευταίο βήμα που κάνουμε πρωτού τελειώσει η διαδικασία.Κάνουμε άθροιση byteμε bytetου αρχικού μπλόκ με το τελευταίο μπλόκ της διαδικάσιας και παίρνουμε την τελική μας έξοδο(**Εικονα 16**)



Εικόνα 16:Η τελικη έξοδος της συναρτησης

[26]Το πρότζεκτ eStream, http://en.wikipedia.org/wiki/ESTREAM

3.5Σημασιολογική Ασφάλεια

Όπως έχουμε πεί σύμφωνα με τον Shannon, ένας αλγόριθμος **έχει τέλεια ασφάλεια** αν ισχύει οτι:

$$\forall m_0,m_1 \in M , (|m_0|=|m_1|),\{E(k,m_0)\}=\{E(k,m_1)\} \text{ όταν } k \leftarrow K.$$

Αυτή η δήλωση όμως απαιτεί οι αλγόριθμοι κρυπτογράφησης συνεπώς και οι αλγόριθμοι ροής να έχουν αρκέτα μεγάλα κλειδιά που αυτό δέν είναι πάντα εφικτό.Ετσι θα προσπαθήσουμε να τροποποιήσουμε λίγο τον ορισμό.Αντί να απαιτήσουμε οι 2 κατανομές να είναι ταυτόσημεςμπορούμε να υποθέσουμε οτι **είναι υπολογιστικά μη διακρίσιμες**.Με άλλα λόγια ένας επιτιθέμενος δεν μπορεί να διακρίνει τις 2 κατανομές ακόμη και άν αυτές μοιάζουν πολύ.

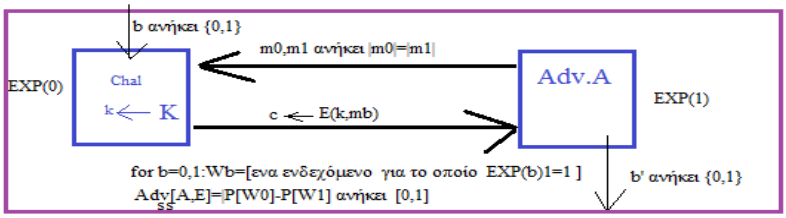
Ο ορισμός για να είναι ταιριαστός πρέπει να υποστεί ακόμη μία τροποποιήση.Αντί να υποθέσουμε οτι ο ορισμός μας ισχύει για κάθε m_0,m_1 θα υποθέσουμε οτι ισχύει για τα m_0,m_1τα οποία ο επιτιθέμενος χρησιμοποιεί κάθε φορά για την επίθεση του.Ο τροποποιήμενος ορισμός πλέον μπορεί να μας οδηγήσει σε στην **σημασιολογική ασφάλεια.**Για να ορίσουμε την **σημασιολογική ασφάλεια**(semanticsecurity)θα πρέπει να χρησιμοποιήσουμε ένα πείραμα.Εστω το πείραμα EXP(0), το πείραμα EXP(1)και ο A να είναι ένας επιτιθέμενος.Εστω οτι υπάρχουν 2 διεκδικητες(challengers) αλλα αυτοί είναι τόσο όμοιοι που θα τον θεωρήσουμε σάν ένανδιεκδικητή

Αυτό που κάνει ο καθε διεκδικητης είναι να παίρνει ένα τυχαίο κλειδί και ο επιτιθέμενος να προσπαθεί να εξάγει τα 2 μηνυματα m_0,m_1τα οποία θεωρούμε ότι έχουν ίδιο μέγεθος.Ο διεκδικητής είτε δίνει την κρυπτογράφηση του m_0είτε του m_1.Επομένως στο πείραμα 0, ο διεκδικητής θα έχει έξοδο την κρυπτογράφηση του μηνύματος m_0ενώ ο διεκδικητής στο πείραμα 1 θα δίνει ώς έξοδο την κρυπτογράφηση του μηνύματος m_1.Ο επιτιθέμενος θα προσπαθήσει να μαντέψει αν του δώθηκε η κρυπτογράφηση του m_0ή κρυπτογράφηση του m_1

Ας ορίσουμε το ενδεχόμενα W_bνα είναι τα ενδεχόμενα του πειράματος B όπου ο επιτιθέμενος έχει ώς έξοδο 1.Ορίζουμε το ενδεχόμενο W_0οπού ο επιτθέμενος όπως έχει έξοδο 0 και το ενδεχόμενο W_1όταν ο επιτιθέμενος έχει έξοδο 1.Ποιό όμως είναι το πλεονέκτημα του επιτιθέμενου;

Αυτό το πλεονέκτημα λέγεται **σημασιολογικό πλεονέκτημα ασφάλειας**(semanticsecurityadvantage) του επιτιθέμενου A ενάντια στο σχήμα E, το οποίο έχει τιμή στο κλειστό διάστημα [0,1] και είναι η διαφορά των πιθανοτήτων τών δύο αυτών ενδεχομένων W_0 και W_1.Αυτό που προσπαθούμε να δούμε είναι, αν ο επιτιθέμενος αντιδρά διαφορετικά αν του δίνεται η κρυπτογράφηση του m_0 ή κρυπτογράφηση του m_1.

Με άλλα λόγια, ενδιαφερόμαστε για το αν ο επιτιθέμενος έχει έξοδο 1 ή όχι.Αν και στα δύο πειράματα ο επιτιθέμενος έχει ώς έξοδο 1 με την ίδια πιθανότητα τότε δέν θα μπορεί να διακρίνει το ένα πείραμα απο το άλλο.Αν ο επιτιθέμενος μας βγάλει 1 στο ένα πείραμα και μια εντελώς διαφορετική πιθανότητα στο άλλο πείραμα τότε ο επιτιθέμενος είναι ικανός να διακρίνει το ένα πείραμα απο το άλλο.Και αυτό σημαίνει οτι μπορεί να διακρίνει την κρυπτογράφηση του μηνύματος m_0απο το m_1.Η διαδικασία φαίνεται και στο (**Εικονα 17**).



Εικόνα 17:Η διαδικασία του πειράματος

Ορισμός σημασιολογικής ασφάλειας

Κανένας αποδοτικός επιτιθέμενος δεν μπορεί να διακρίνει το κρυπτογραφημένο μήνυμα m_0απο το κρυπτογραφημένο μήνυμα m_1.Με διαφορετική προσέγγιση ένα σχήμα Ε είναι σημασιολογικά ασφαλές, αν για όλους τους αποδοτικά επιτιθέμενους το πλεονέκτημα τους είναι αμελητέο.

3.6 Ερωτήσειςκεφαλαίου

Ερώτηση 1

Data compression is often used in data storage and transmission. Suppose you want to use data compression in conjunction with encryption. Does it make more sense to:

a) The order does not matter -- either one is fine.
b) Encrypt then compress.
c) Compress then encrypt
d) The order does not matter -- neither one will compress the data.

Απάντηση

The correct answer is **c**. Ciphertexts tend to look like random strings and therefore the only opportunity for compression is prior to encryption.

Ερώτηση 2

Let$G:\{0,1\}^s \rightarrow \{0,1\}^n$be a secure PRG. This of the following is a secure PRG (there is more than one correct answer):

a)$G'(k)=G(k) \oplus 1^n$
b)G´(k)=G(k)||0 (here || denotes concatenation)
c)G´(k)=G(0)
d)G´(k)=reverse (G(k))where reverse(x) reverses the string x so that the first bit of x is the last bit of reverse(x), the second bit of x is the second to last bit of reverse(x),and so on.
e)G´(k)=G(k)||G(k)(here || denotes concatenation)
f)$G'(k)=G(k \oplus 1^s)$

Απάντηση

The correct answers are **a**, **d** ,**f**.For example :
$G'(k)=G(k \oplus 1^s)$) is secure because XOR with 1 just flips the bits and doesnot change the information content of the string.With the same way, **G'(k) = reverse(G(k))** is secure because a secure PRG does not yeild additional information when reversed.

Ερώτηση 3

LetG(k)→$\{0,1\}^n$be a secure PRG. DefineG(k)=G (k_1) ∧G(k_2) where ∧ is the bit-wise AND function. Consider the following statistical test A on $\{0,1\}^n$. A(x) outputs LSB(x), the least significant bit of x.What is $Adv_{PRG}[A,G']$? You may assume that LSB(G(k)) is 0 for exactly half the seeds k in K

Απάντηση

If A outputs LSB(x), then

Adv[A, G'] = |Pr[A(G')=1] - Pr[A(r)=1]|= | Pr[LSB(G_1 and G_2) = 1] - 1/2 |= 1/4

Ερώτηση 4

Let (E,D) be a (one-time) semantically secure cipher with key space $K=(0,1)^{\ell}$. A bank wishes to split a decryption keyk∈ $\{0, 1\}^{\ell}$ into two pieces p_1 and p_2 so that both are needed for decryption. The piece p_1 can be given to one executive and p_2 to another so that both must contribute their pieces for decryption to proceed. The bank generates random k_1 in$\{0,1\}^{\ell}$and sets $k'_1 \leftarrow k \oplus k_1$.

Note that $k_1 \oplus k'_1 = k$.The bank can give k_1 to one executive and k'_1 to another. Both must be present for decryption to proceed since, by itself, each piece contains no information about the secret key k (note that each piece is a one-time pad encryption of k).Now, suppose the bank wants to split k into three pieces p_1,p_2, p_3 so that any two of the pieces enable decryption using k. This ensures that even if one executive is out sick, decryption can still succeed.

To do so the bank generates two random pairs (k_1,k'_1) and (k_2,k'_2) as in the previous paragraph so that $k_1 \oplus k_1 = k_2 \oplus k'_2 = k$.How should the bank assign pieces so that any two pieces enable decryption using k, but no single piece can decrypt?

a)$p_1=(k_1,k_2)$,$p_2=(k_1',k_2)$,$p_3=(k_2')$
b)$p_1=(k_1,k_2)$,$p_2=(k_2,k_2')$,$p_3=(k_2')$
c)$p_1=(k_1,k_2)$, $p_2=(k_1')$,$p_3=(k_2')$
d)$p_1=(k_1,k_2)$,$p_2=(k_1,k_2)$,$p_3=(k_2')$
e)$p1=(k_1,k_2)$,$p_2=(k_1',k_2')$,$p3=(k_2')$

Απάντηση

Executives 1 and 2 can decrypt using k_1,k_1', executives 1 and 3 can decrypt using k_2,k_2', and executives 2 and 3 can decrypt using k_2,k_2'. Moreover, a single executive has no information about k.

Second solution :

The bank has generated the following keys:

k, k_1, and k_2 are random
$k_1' = k \oplus k1$
$k_2' = k \oplus k_2$

All choices involve $p_1 = (k_1, k_2)$ and $p_3 = (k_2')$, so we consider p_2.p_2 cannot be (k_1, k_2) because p_2 U p_1 would give p_2 no additional information.p_2 cannot be (k_1', k_2') because then p_2 U p_3 would give p_2 no more info. p_2 must be (k_1', k_2). p_1 U p_2 can derive k and p_2 U p_3 can derive k.p_2 cannot be (k_1') becaue p_2 U p_3 would be insufficient to derive k.p_2 cannot be (k_2,k_2') because $k_2' \oplus k_2 = k$ so p could derive k So the correct answer is **a**

Ερώτηση 5

LetM=C=…K={0,1…255} and consider the following cipher defined over (M,K,C).

E(k,m)=m+k(mod256) καιD(k,c)=c-k(mod256).

Does this cipher have perfect secrecy?

Απάντηση

As with the one-time pad,there is exactly one key mapping a given message m to a given ciphertext c.

Second solution:

We aim to show that for all c in C, m_1 in M, m_2 in M:
P[E(k,m_1) = c] = P[E(k,m_2) = c], so we simply compute, for any m in M:
P[E(k,m) = c] = P[m + k mod 256 = c] = P[c - m mod 256 == k] = 1/256.So the cipher has perfect secrecy.

Ερώτηση 6

Let (E,D) be a (one-time) semantically secure cipher where the message and ciphertext space is$\{0,1\}^n$. Which of the following encryption schemes are (one-time) semantically secure?

a) E′(k,m)=reverse(E(k,m))
b) E′(k,m)=0‖‖E(k,m) (i.e prepend 0 to the ciphertext)
c) E′(k,m)=compute c←E(k,m) and output c‖‖c
d) E′(k,m)=E(k,m)‖ k
e) E′(k,m)=E(0^n,m)
f) E′(k,m)=E(k,m)‖ LSB(m)

Απάντηση

The correct answers are **a**, **b**, **c** because an attack on E′ gives an attack on E.The answer **d** is false because, to break semantic security, an attacker would read the secret key from the challenge ciphertext and use it to decrypt the challenge ciphertext. Basically, any ciphertext reveals the secret key.The answer **e** is false because, To break semantic security, an attacker would ask for the encryption of 0^n and 1^n and can easily distinguish EXP(0) from EXP(1) because it knows the secret key, namely 0^n.The answer f is false because, To break semantic security, an attacker would ask for the encryption of 0^n and $0^{n-1}1$ and can distinguish EXP(0) from EXP(1).

Second solution :

E'(k,m) = E(k,m) || E(k,m) appears to be semantical secury(ss).E'((k,k'), m) = E(k,m) || E(k', m) appears to be ss.E'(k,m) = E(k,m) || LSB(m) is not ss because the adversary can send two messages with different LSBs to determine which ciphertext is which.E'(k,m) = E(0^n,m) is not ss since the adversary can send any two distinct messages and determine which is which.E'(k,m) = 0 || E(k, m) appears to be ss.E'(k,m) = E(k,m) || k is obviously not ss because the attacker can send $m_0=0_s$, m1=1s and just look for the message that is k||k.

Ερώτηση 7

Suppose you are told that the one time pad encryption of the message "attack at dawn" is 09e1c5f70a65ac519458e7e53f36 (the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in hex).What would be the one time pad encryption of the message "attack at dusk" under the same OTP key?

Απάντηση

m_1=attack at dawn έπεταιότι$\mathbf{c_1}=\mathbf{m_1}\oplus\mathbf{k_1}$=09e1c5f70a65ac519458e7e53f36 αρα$k_1=m_1\oplus c_1 m_2$=attackatdusk.Έπεταιότι$c_2=m_2\oplus$κάρα$k_2=m_2\oplus c_2$ΕπειδήέχουμεOTP καικρυπτογράφησημετοίδιοκλειδίτοτε**k1=k2=k**.Οπότεισχύειότι:

$$\mathbf{m_1\oplus c_1=m_2\oplus c2\Rightarrow c_2=m_1\oplus c_1\oplus m2}\ (1)$$

Μετατρέπουμε το m_2σε δεκαεξαδικό: 61747461636b206174206475736b.Κάνοντας την πράξη που μας δείχνει η σχέση (1) εχουμε 6c73d5240a948c86981bc2808548

Ερώτηση 8

The movie industry wants to protect digital content distributed on DVD's. We develop a variant of a method used to protect Blu-ray disks called AACS. Suppose there are at most a total of n DVD players in the world. We view these n players as the leaves of a binary tree of height $\log_2 n$. Each node in this binary tree contains an AES key k_i.

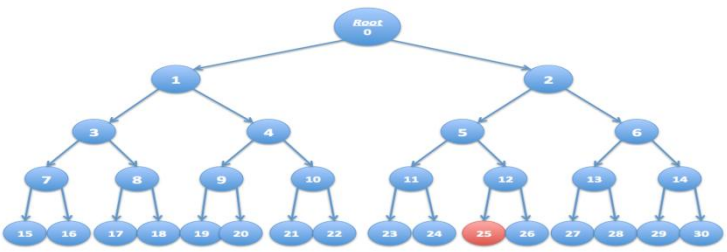
These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player is assigned a serial number $i \in [0, n-1]$. Consider the set of nodes S_i along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the keys associated with the nodes in the set S_i. A DVD movie m is encrypted as:

$$\mathbf{E(k_{root},k)\|\|E(k,m)}$$

where k is a random AES key called a content-key and kroot is the key associated with the root of the tree. Since all DVD players have the key k_{root}all players can decrypt the movie m. We refer to $E(k_{root},k)$ as the header and E(k,m) as the body. In what follows the DVD header may contain multiple ciphertexts where each ciphertext is the encryption of the content-key k under some key $k_i$in the binary tree.

Suppose the keys embedded in DVD player number r are exposed by hackers and published on the Internet. In this problem we show that when the movie industry distributes a new DVD movie, they can encrypt the contents of the DVD using a slightly larger header (containing about $\log_2 n$ keys) so that all DVD players, except for player number r, can decrypt the movie. In effect, the movie industry disables player number r without affecting other players(**Εικονα 18**).

s shown below, consider a tree with n=16 leaves. Suppose the leaf node labeled 25 corresponds to an exposed DVD player key. Check the set of keys below under which to encrypt the key k so that every player other than player 25 can decrypt the DVD. Only four keys are needed.



Εικόνα 18:Το δυαδικό δένδρο με 25 φύλλα

a)11

b)5
c)6
d)3
e)26
f)15
g)1
h)20

Απάντηση
The answer r**a** is correct because: You cannot encrypt k under key 5, but 11's children must be able to decrypt k.
The answer **c** is correct because because: You cannot encrypt k under 2, but 6's children must be able to decrypt k.
The answer **e** is correct because: You cannot encrypt k under any key on the path from the root to node 25. Therefore 26 can only decrypt if you encrypt k under key k_{26}.
The answer **g** is correct because: You cannot encrypt k under the root, but 1's children must be able to decrypt k.

Ερώτηση 9
Continuing with the previous question, if there are n DVD players, what is the number of keys under which the content key k must be encrypted if exactly one DVD player's key needs to be revoked?

a)$\sqrt{n}$
b)n-1
c)2
d)$\log_2 n$
e)n/2

Απάντηση
The correct answer is **d**.The key will need to be encrypted under one key for each node on the path from the root to the revoked leaf. There are log2n nodes on the path.

Ερώτηση 10
Continuing with question 8, suppose the leaf nodes labeled 16, 18 and 25 correspond to exposed DVD player keys. Check the smallest set of keys under which to encrypt the key k so that every player other than players 16,18,25 can decrypt the DVD.Only six keys are needed.

a)19
b)3
c)23
d)26
e)4
f)11
g)17
h)6
i)29
k)15

Απάντηση

The answer**d**is a correct answer beacause this will let player 26 decrypt.
The answer **e**is a correct answer beacause this will let players 19-22 decrypt.
The answer **f** is a correct answer beacause this will let players 23,24 decrypt.
The answer **g** is a correct answer beacause this will let player 17 decrypt.
The answer **h** is a correct answer beacause this will let players 27-30 decrypt.
The answer **k** is a correct answer beacause this will let player 15 decrypt.

ΠρογραμματιστικήΆσκηση 1

Let us see what goes wrong when a stream cipher key is used more than once. Below are eleven hex-encoded ciphertexts that are the result of encrypting eleven plaintexts with a stream cipher, all with the same stream cipher key. Your goal is to decrypt the last ciphertext, and submit the secret message within it as solution.**Hint**: XOR the ciphertexts together, and consider what happens when a space is XORed with a character in [a-zA-Z].

ciphertext #1:

```
315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cd
f2d3aff021dfff5b403b510d0d0455468aeb98622b137dae857553ccd8883a7bc37520e06e515
d22c954eba5025b8cc57ee59418ce7dc6bc41556bdb36bbca3e8774301fbcaa3b83b220809560
987815f65286764703de0f3d524400a19b159610b11ef3e
```

ciphertext #2:

```
234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d
9606ef4081ffde5ad46a5069d9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919
cf3a9f1cb74151f6d551f4480c82b2cb24cc5b028aa76eb7b4ab24171ab3cdadb8356f
```

ciphertext #3:

```
32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8
a2368e51d04e0e7b207b70b9b8261112bacb6c866a232dfe257527dc29398f5f3251a0d47e503
c66e935de81230b59b7afb5f41afa8d661cb
```

ciphertext #4:

```
32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad
3306ef5021eafe1ac01a81197847a5c68a1b78769a37bc8f4575432c198ccb4ef63590256e305
cd3a9544ee4160ead45aef520489e7da7d835402bca670bda8eb775200b8dabbba246b130f04
0d8ec6447e2c767f3d30ed81ea2e4c1404e1315a1010e7229be6636aaa
```

ciphertext #5:

```
3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d
9607cee021dafe1e001b21ade877a5e68bea88d61b93ac5ee0d562e8e9582f5ef375f0a4ae20e
d86e935de81230b59b73fb4302cd95d770c65b40aaa065f2a5e33a5a0bb5dcaba43722130f042
f8ec85b7c2070
```

ciphertext #6:

```
32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8d
e287be84d07e7e9a30ee714979c7e1123a8bd9822a33ecaf512472e8e8f8db3f9635c1949e640
c621854eba0d79eccf52ff111284b4cc61d11902aebc66f2b2e436434eacc0aba938220b08480
0c2ca4e693522643573b2c4ce35050b0cf774201f0fe52ac9f26d71b6cf61a711cc229f77ace7
aa88a2f19983122b11be87a59c355d25f8e4
```

ciphertext #7:

```
32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba47b01c909ba7696c
f606ef40c04afe1ac0aa8148dd066592ded9f8774b529c7ea125d298e8883f5e9305f4b44f915
cb2bd05af51373fd9b4af511039fa2d96f83414aaaf261bda2e97b170fb5cce2a53e675c154c0
d9681596934777e2275b381ce2e40582afe67650b13e72287ff2270abcf73bb028932836fbdec
fecee0a3b894473c1bbeb6b4913a536ce4f9b13f1efff71ea313c8661dd9a4ce
```

ciphertext #8:

```
315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54ce56801d943ba708b8
a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f89ee04132ecb9298f5fd2d5e4b45e40e
cc3b9d59e9417df7c95bba410e9aa2ca24c5474da2f276baa3ac325918b2daada43d671215044
1c2e04f6565517f317da9d3
```

ciphertext #9:

```
271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4aad6e04d513e96d99d
e2569bc5e50eeeca709b50a8a987f4264edb6896fb537d0a716132ddc938fb0f836480e06ed0f
cd6e9759f40462f9cf57f4564186a2c1778f1543efa270bda5e933421cbe88a4a52222190f471
e9bd15f652b653b7071aec59a2705081ffe72651d08f822c9ed6d76e48b63ab15d0208573a7ee
f027
```

ciphertext #10:

```
466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b3399cc
fafb61d0315fca0a314be138a9f32503bedac8067f03adbf3575c3b8edc9ba7f537530541ab0f
9f3cd04ff50d66f1d559ba520e89a2cb2a83
```

target ciphertext (decrypt this one):

```
32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd
8257bf14d13e6f0a803b54fde9e77472dbff89d71b57bddef121336cb85ccb8f3315f4b52e301
d16e9f52f904
```

Απάντηση

Ο κώδικας της προγραμματιστικής άσκησης 1 βρίσκεται εδώ.Αρχικά δημιουγούμε έναν πίνακα που τον ονομάζουμε CTS.Ο πίνακας αυτός περιέχει τα 10 ciphetexttα οποία όμως έχουμε αποκωδικοποιήσει μέσω της decode() απο δεκαεξαδικό σε συμβολοσειρές..Μετά φτιάξαμε μια μεταβλητή που την ονομάσαμε TCTη οποία περιέχει το αποκωδικοποιημένο targetciphertextσε δυαδικό επίσης.Κατόπιν δημιουργούμε μια συνάρτηση που την ονομάζουμε strxorη οποία παίρνει ως ορίσματα 2 συμβολοσειρές η οποία η δουλειά της είναι να εφαρμόζει πράξη XORμεταξύ αυτών των δύο συμβολοσειρών.Για να γίνει αυτό χρησιμοποιήσαμε 4 builtinσυναρτήσεις της Python.

Τη συνάρτηση len() που μετράει το μήκος μια συμβολοσειράς, τη συνάρτηση zip() η οποία σταματά την επανάληψη μιας διαδικασίας όταν βρεί το τέλος της μικρότερης συμβολοσειράς απο τις δυο.Η διαδικασία φυσικα ειναι η πράξη XORαυτών των συμβολοσειρών.Χρησιμοποιούμε την ord() για να μετατρέψουμε απο το δυαδικό σε ASCIIvalues , δηλαδή σε ακεραίους μιας και η ΄Λ΄ απαιτεί αριθμούς.Τέλος μόλις γίνει η πράξη XORτις μετατρέπουμε σε συμβολοσειρες.Στη συνέχεια δημιουργούμε και μια άλλη συνάρτηση που την ονομάζουμε mainπου δεν έχει ορίσματα.

Τότε για κάθε CT1, CT2 που ανήκουν στην itertools.permutationsαναθέτουμε στη μεταβλητή xorτο αποτέλεσμα της συνάρτησης strxor(CT1 ,CT2).Η itertools.permutationsπαίρνει ώς όρισμα μια συμβολοσειρά και έναν αριθμο που μας δείχνει το σύνολο των χαρακτήρων που θα έχει το κάθε ανατιθέμενο στοιχείο που πηγάζει απο το πρώτο όρισμα της.Μετά για κάθε i, cθα κάνουμε κάποιους ελέγχους αφού πρώτα μέσω της enumerateαριθμήσουμε την κάθε τιμή xorπου βρήκαμε απο το προηγούμενο βήμα.

Πρώτα ελέγχουμε για το έαν το iείναι μεγαλύτερο ή ισο απο το μέγεθος του κλειδιούKEY.Εαν είναι ίσο τότε η τιμή KEY[i]=΄_΄ ενώ αν i>len(KEY) τότε δε κάνουμε τίποτα.Εαν όμως i<len(KEY) προχωράμε στο να ελέγξουμε ξεχωριστά για κεφαλαία και μικρά γράμματα για το εαν το c(που είναι σε ASCII) που ανήκει στο CTSείναι τιμή του κλειδιού.Μόλις γίνουν οι απαραίτητοι έλεγχοι θα βρούμε το κλειδί και τέλος θα κάνουμε XORμεταξύ του κλειδιού και του targetciphertextμας για να βρούμε το plaintextπου ζητάμε.Έξοδος προγράμματος :

['f', '9', 'n', '\x89', '\xc9', '\xdb', '\xd8', '\xcb', '\x98', 't', '5', '*', '\xcd', 'c', '\x95', '\x10', '.', '\xaf', '_', 'x', '\xaa', '\x7f', '\xed', '(', '\xa0', 'n', '~', '\xc9', '\x8d', ')', '\xc5', 'C', 'i', '\xe1', 'f', '\xdb', 'W', '\xf8', '\xaa', '@', '\x1a', '\x9c', 'm', 'p', '\x8f', '\x80', '\xc0', 'f', '\xc7', 'c', '\xf0', '\xf0', '\x12', '1', 'H', '\x8e', '_', '\xe8', '\x02', '\xd0', '[', '\xa9', '\x87', '%', '3', ']', '\xed', '\xfc', '\xec', '\xd5', '\x9c', 'C', ':', 'k', '&', '\x8b', '`', '\xbf', 'N', '\xb5', '<', '\x9a', 'p', 'q', '\x98', '\xbb', '_', '\xce', '1', 'a', '\xed', '\xc7', '_', 'K', '\xe4', '{', '"', '\xcf', '\xd2', '_', '\xd2', '_',

'\x8b', 'W', '7', 'n', '\xdb', '\xa8', '\xc2', '_', '_', 'O', '|', '_', '$', 'a', '\xe2', '\xa1', '_', '_', 'E', '\x02', '\x1b', '\x1d', '\x1d', '\xc0', '\xd3', '\xba', '_', '`', 'x', '_', '\x91', '\x11', '\x00', '_', '_', '_', '_', '\xbb', '_', '\x02', '_', '\xc4', '\xab', '_', '_', '_', '\xa9', '_', '_', '_', '_', '_', '_', '\x8a', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_', '_']
decryptedtext: ['T', 'h', 'e', ' ', 's', 'e', 'c', 'u', 'e', 't', ' ', 'm', 'e', 's', 's', 'a', 'g', 'e', '\xb1', 'i', 's', ':', ' ', 'W', 'h', 't', '{', ' ', 'u', 's', 'i', '&', 'g', 'q', '4', 'a', '=', 't', 'r', 'e', 'a', 'm', ' ', 'c', 'i', 'p', 'h', 'e', 'r', ',', '.', 'n', 'e', 'v', 'e', '1', '\xa7', 'u', 's', 'e', ' ', 't', 'h', '7', ' ', 'k', '&', 'y', ' ', 'm', 'o', 'r', 'e', ' ', 't', 'h', 'a', 'n', ' ', '*', 'n', 'c', 't']

Απο ότι βλέπουμε το ζητούμενο μήνυμα είναι :
The secret message is: When using a stream cipher, never use the key more than once

3.7 Αναλυτική βαθμολογία

You submitted this homework on **Sat 2 May 2015 2:31 AM EEST**. You got a score of **5.71** out of **8.45**.

Question 1

Data compression is often used in data storage and transmission. Suppose you want to use data compression in conjunction with encryption. Does it make more sense to:

Your Answer		Score	Explanation
The order does not matter -- either one is fine.			
Encrypt then compress.			
Compress then encrypt.	✔	1.00	Ciphertexts tend to look like random strings and therefore the only opportunity for compression is prior to encryption.
The order does not matter -- neither one will compress the data.			
Total		1.00 / 1.00	

Εικόνα 19:Ερωτηση 1-Week 1

Question 2

Let $G : \{0,1\}^s \rightarrow \{0,1\}^n$ be a secure PRG. Which of the following is a secure PRG (there is more than one correct answer):

Your Answer		Score	Explanation
☑ $G'(k) = G(k) \oplus 1^n$	✔	0.17	a distinguisher for G' gives a distinguisher for G.
☑ $G'(k) = G(k) \| 0$ (here $\|$ denotes concatenation)	✖	0.00	A distinguisher will output *not random* whenever the last bit of its input is 0.
☐ $G'(k) = G(0)$	✔	0.17	A distinguisher will output *not random* whenever its input is equal to $G(0)$.
☑ $G'(k) = \text{reverse}(G(k))$ where reverse(x) reverses the string x so that the first bit of x is the last bit of reverse(x), the second bit of x is the second to last bit of reverse(x), and so on.	✔	0.17	a distinguisher for G' gives a distinguisher for G.
☑ $G'(k) = G(k) \| G(k)$ (here $\|$ denotes concatenation)	✖	0.00	A distinguisher will output *not random* whenever the first n bits are equal to the last n bits.
☑ $G'(k) = G(k \oplus 1^s)$	✔	0.17	a distinguisher for G' gives a distinguisher for G.
Total		0.67 / 1.00	

Εικόνα 20:Ερώτηση 2-Week 1

Question 3

Let $G : K \rightarrow \{0,1\}^n$ be a secure PRG. Define $G'(k_1, k_2) = G(k_1) \bigwedge G(k_2)$ where $\bigwedge$ is the bit-wise AND function. Consider the following statistical test A on $\{0,1\}^n$:

$A(x)$ outputs $\text{LSB}(x)$, the least significant bit of x.

What is $Adv_{\text{PRG}}[A, G']$? You may assume that $\text{LSB}(G(k))$ is 0 for exactly half the seeds k in K.

Note: Please enter the advantage as a decimal between 0 and 1 with a leading 0. If the advantage is 3/4, you should enter it as 0.75

You entered:

0.16

Your Answer		Score	Explanation
0.16	✖	0.00	
Total		0.00 / 1.00	

Εικόνα 21:Ερώτηση 3-Week 1

Question 4

Let (E, D) be a (one-time) semantically secure cipher with key space $K = \{0,1\}^\ell$. A bank wishes to split a decryption key $k \in \{0,1\}^\ell$ into two pieces p_1 and p_2 so that both are needed for decryption. The piece p_1 can be given to one executive and p_2 to another so that both must contribute their pieces for decryption to proceed.

The bank generates random k_1 in $\{0,1\}^\ell$ and sets $k_1' \leftarrow k \oplus k_1$. Note that $k_1 \oplus k_1' = k$. The bank can give k_1 to one executive and k_1' to another. Both must be present for decryption to proceed since, by itself, each piece contains no information about the secret key k (note that each piece is a one-time pad encryption of k).

Now, suppose the bank wants to split k into three pieces p_1, p_2, p_3 so that any two of the pieces enable decryption using k. This ensures that even if one executive is out sick, decryption can still succeed. To do so the bank generates two random pairs (k_1, k_1') and (k_2, k_2') as in the previous paragraph so that $k_1 \oplus k_1' = k_2 \oplus k_2' = k$. How should the bank assign pieces so that any two pieces enable decryption using k, but no single piece can decrypt?

Your Answer		Score	Explanation
◉ $p_1 = (k_1, k_2)$, $p_2 = (k_1', k_2)$, $p_3 = (k_2')$	✔	1.00	executives 1 and 2 can decrypt using k_1, k_1', executives 1 and 3 can decrypt using k_2, k_2', and executives 2 and 3 can decrypt using k_2, k_2'. Moreover, a single executive has no information about k.
○ $p_1 = (k_1, k_2)$, $p_2 = (k_2, k_2')$, $p_3 = (k_2')$			
○ $p_1 = (k_1, k_2)$, $p_2 = (k_1')$, $p_3 = (k_2')$			
○ $p_1 = (k_1, k_2)$, $p_2 = (k_1, k_2)$, $p_3 = (k_2')$			
○ $p_1 = (k_1, k_2)$, $p_2 = (k_1', k_2')$, $p_3 = (k_2')$			
Total		1.00 / 1.00	

Εικόνα 22:Ερώτηση 4-Week 1

Question 5

Let $M = C = K = \{0, 1, 2, \ldots, 255\}$ and consider the following cipher defined over (K, M, C):

$$E(k, m) = m + k \pmod{256} \quad ; \quad D(k, c) = c - k \pmod{256}.$$

Does this cipher have perfect secrecy?

Your Answer		Score	Explanation
○ No, there is a simple attack on this cipher.			
◉ Yes.	✔	1.00	as with the one-time pad, there is exactly one key mapping a given message m to a given ciphertext c.
○ No, only the One Time Pad has perfect secrecy.			
Total		1.00 / 1.00	

Εικόνα 23:Ερώτηση 5-Week 1

Question 6

Let (E, D) be a (one-time) semantically secure cipher where the message and ciphertext space is $\{0, 1\}^n$. Which of the following encryption schemes are (one-time) semantically secure?

Your Answer		Score	Explanation
☑ $E'(k, m) = \text{reverse}(E(k, m))$	✔	0.17	an attack on E' gives an attack on E.
☐ $E'(k, m) = 0 \,\|\| E(k, m)$ (i.e. prepend 0 to the ciphertext)	✖	0.00	an attack on E' gives an attack on E.
☐ $E'(k, m) =$ compute $c \leftarrow E(k, m)$ and output $c \,\|\| c$ (i.e., output c twice)	✖	0.00	an attack on E' gives an attack on E.
☐ $E'(k, m) = E(k, m) \,\|\| k$	✔	0.17	To break semantic security, an attacker would read the secret key from the challenge ciphertext and use it to decrypt the challenge ciphertext. Basically, any ciphertext reveals the secret key.
☐ $E'(k, m) = E(0^n, m)$	✔	0.17	To break semantic security, an attacker would ask for the encryption of 0^n and 1^n and can easily distinguish EXP(0) from EXP(1) because it knows the secret key, namely 0^n.
☐ $E'(k, m) = E(k, m) \,\|\| \text{LSB}(m)$	✔	0.17	To break semantic security, an attacker would ask for the encryption of 0^n and $0^{n-1}1$ and can distinguish EXP(0) from EXP(1).
Total		0.67 / 1.00	

Εικόνα 24:Ερώτηση 6-Week 1

Question 7

Suppose you are told that the one time pad encryption of the message "attack at dawn" is *09e1c5f70a65ac519458e7e53f36* (the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in hex). What would be the one time pad encryption of the message "attack at dusk" under the same OTP key?

You entered:

6895b196690e8c30e07883904c5d

Your Answer		Score	Explanation
6895b196690e8c30e07883904c5d	✖	0.00	
Total		0.00 / 1.00	

Εικόνα 25:Ερώτηση 7-Week 1

Your Answer		Score	Explanation
11	✔	0.03	You cannot encrypt k under key 5, but 11's children must be able to decrypt k.
5	✔	0.03	No, this will let node 25 decrypt the DVD.
6	✔	0.03	You cannot encrypt k under 2, but 6's children must be able to decrypt k.
3	✔	0.03	There is a better solution that does not require encrypting on the key of this node.
26	✔	0.03	You cannot encrypt k under any key on the path from the root to node 25. Therefore 26 can only decrypt if you encrypt k under key k_{26}.
15	✖	0.00	There is a better solution that does not require encrypting on the key of this node.
1	✔	0.03	You cannot encrypt k under the root, but 1's children must be able to decrypt k.
20	✔	0.03	There is a better solution that does not require encrypting on the key of this node.
Total		0.22 / 0.25	

Εικόνα 26:Ερώτηση 8-Week 1

Question 9

Continuing with the previous question, if there are n DVD players, what is the number of keys under which the content key k must be encrypted if exactly one DVD player's key needs to be revoked?

Your Answer		Score	Explanation
$\sqrt{n}$			
$n-1$			
2			
$\log_2 n$	✔	1.00	That's right. The key will need to be encrypted under one key for each node on the path from the root to the revoked leaf. There are $\log_2 n$ nodes on the path.
$n/2$			
Total		1.00 / 1.00	

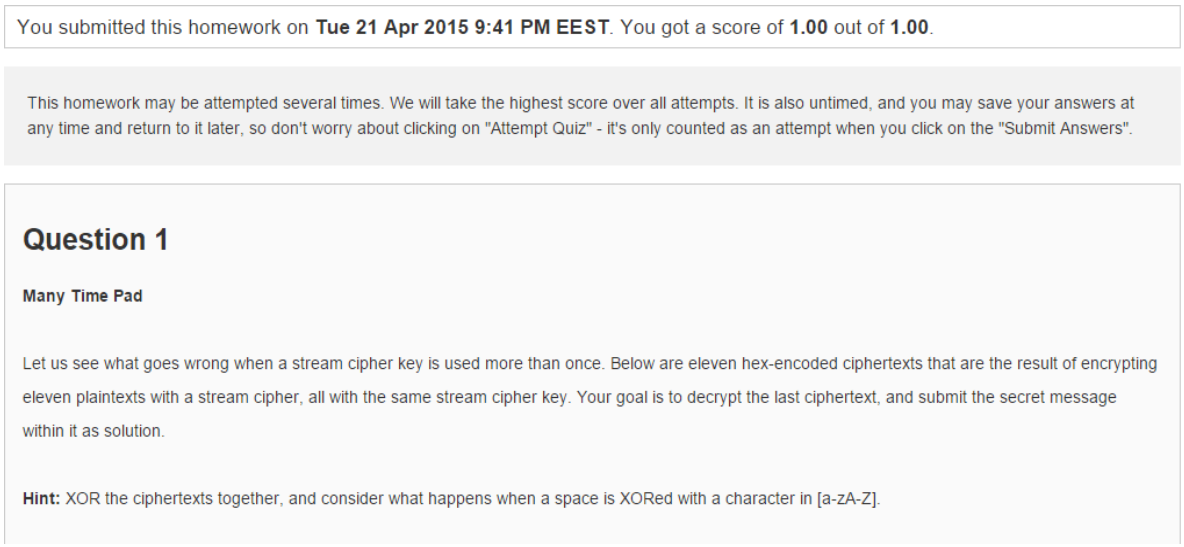
Εικόνα 27:Ερώτηση 9-Week 1

Question 10

Continuing with question 8, suppose the leaf nodes labeled 16, 18, and 25 correspond to exposed DVD player keys. Check the smallest set of keys under which to encrypt the key k so that every player other than players 16,18,25 can decrypt the DVD. Only six keys are needed.

Your Answer		Score	Explanation
19	✔	0.02	
3	✔	0.02	
23	✔	0.02	
26	✖	0.00	Yes, this will let player 26 decrypt.
4	✔	0.02	Yes, this will let players 19-22 decrypt.
11	✔	0.02	Yes, this will let players 23,24 decrypt.
17	✔	0.02	Yes, this will let player 17 decrypt.
6	✔	0.02	Yes, this will let players 27-30 decrypt.
29	✖	0.00	
15	✔	0.02	Yes, this will let player 15 decrypt.
Total		0.16 / 0.20	

Εικόνα 28:Ερώτηση 10-Week 1



Εικόνα 29:Προγραμματιστική ασκήση 1a-Week 1

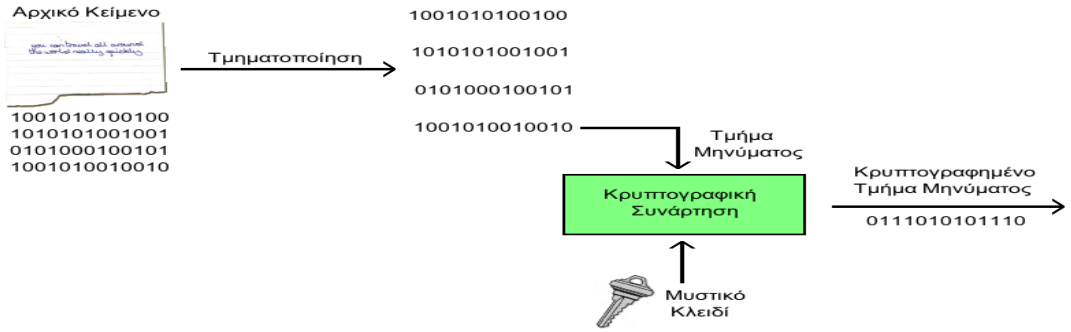
Πίνακας 1:Βαθμολογίες-Week 1

	$1^{η}$ προσπάθεια	$2^{η}$ προσπάθεια	$3^{η}$ προσπάθεια	$4^{η}$ προσπάθεια
Ερωτήσεις	4.36/8.45	4.75/8.45	5.41/8.45	5.71/8.45
Άσκηση	0.00/1.00	0.00/1.00	0.00/1.00	1.00/1.00

ΚΕΦΑΛΑΙΟ 4
ΑΛΓΟΡΙΘΜΟΙ ΤΜΗΜΑΤΟΣ

4.1 Αλγόριθμοι τμήματος και ιδιότητες

Οι κρυπτογραφικοί αλγόριθμοι τμήματος ή τμήματος (block ciphers) τεμαχίζουν σε τμήματα (blocks) το αρχικό κείμενο που πρόκειται να κρυπτογραφηθεί και κρυπτογραφούν κάθε τμήμα ξεχωριστά. Συνηθισμένα μεγέθη ενός τμήματος δεδομένων είναι τα 64 ή 128 bits. Η κρυπτογράφηση κάθε ενός τμήματος γίνεται χρησιμοποιώντας μία μαθηματική συνάρτηση κρυπτογράφησης και το μυστικό κλειδί. Το αποτέλεσμα της διαδικασίας κρυπτογράφησης είναι η παραγωγή ενός κρυπτογραφημένου τμήματος το οποίο στην πλειοψηφία των περιπτώσεων έχει το ίδιο μήκος με το αντίστοιχο τμήμα του αρχικού κειμένου. Η διαδικασία κρυπτογράφησης φαίνεται στο σχήμα(**Εικόνα 30**)



Εικόνα 30:Διάγραμμα λειτουργίας αλγορίθμων τμήματος

Ένας αλγόριθμος τμήματος τμήματος (blockcipher) χρησιμοποιεί δύο αλγορίθμους για να μπορέσει να λειτουργήσει.Έναν αλγόριθμοΕπου χρησιμοποιείται για την κρυπτογράφησηκαι έναν αλγόριθμοD που χρησιμοποιείται για την αποκρυπτογράφησητου μηνύματος.Οι δύο αυτοί αλγόριθμοι δέχονται δύο εισόδους. Το μήκος του τμήματος(block) το οποίο έχει μέγθος nbitsένα κλειδί το οποίο έχει μήκος kbits.Ο αλγόριθμος **αποκρυπτογράφησης** ορίζεται να είναι μια αντιστρέψιμη συνάρτηση κρυπτογράφησης δηλαδή $D=E^{-1}$.

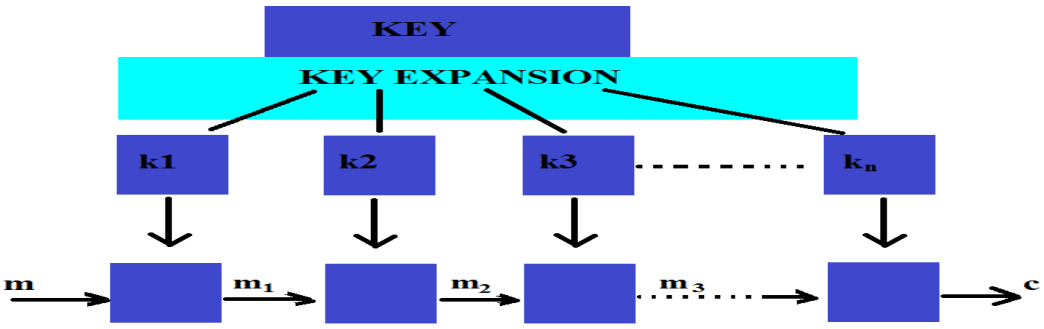
Με μαθηματικούς συμβολισμούς έαν θεωρήσουμε ότι Κείναι το κλειδί εισόδου, Στο ciphertextενώ Ρτο plaintextτότε:

$$E_K(P)=E(K, P):\{0, 1\}^k\times\{0, 1\}^n\rightarrow\{0, 1\}^n$$

$$E^{-1}_K(C)=D_K(C)=D(K, C):\{0, 1\}^k\times\{0, 1\}^n\rightarrow\{0, 1\}^n$$

Παραδείγματα τέτοιων αλγορίθμων είναι ο 3DES[27]και οAES.Οσο αφορά τον 3DES γνωρίζουμε ότι το μήκος του κάθε μπλόκ είναι 64bitsκαι το κλειδί του έχει μήκος 128bitsενώ όσο αφορά των AESτο μήκος του κάθε μπλόκ είναι 128bitsκαι κλειδί μπορεί να είναι 128, 192 ή 256 bits.Όμως οσο πιο μεγάλο είναι το κλειδί τοσο πιο αργός είναι ο αλγόριθμος αλλα και πιό ασφαλής.

Το κύριο χαρακτηριστικό των αλγορίθμων τμήματος είναι η επανάληψη διότι παίρνουν ως είσοδο ένα κλειδί Κκαι το πρώτο πράγμα που κάνουν είναι να επεκτείνουν το κλειδί σε μια ακολουθία κλειδίων K_1….K_nπου λέγονται **κυκλικά κλειδιά** (roundkeys).Στη συνέχεια ο τρόπος με τον οποίο ο αλγόριθμος χρησιμοποιεί αυτα τα κυκλικά κλειδιά είναι απλα κρυπτογραφόντας το μήνυμα που θέλουμε επαναληπτικά συνεχώς χρησιμοποιώντας μια **κυκλική συνάρτηση**(roundfunction).Η συνάρτηση αυτή παίρνει δύο ορίσματα.Το κυκλικό κλειδί και την κατάσταση στην οποία το μήνυμα βρίσκεται εκείνη την στιγμή που έχουμε ενα συγκεκριμένο κυκλικό κλειδί.
Ειδικά για τον AES αφού πρώτα γεννηθούν k_nστο σύνολο κυκλικά κλειδιά απο το αρχικό κλειδί, κρυπτογραφούμε με το κλειδί k_1το μήνυμα μας και βγάζουμε ώς έξοδο το κρυπτογραφημένο μήνυμα m_1.Στη συνέχεια κρυπτογραφούμε το m_1με το δεύτερο κλειδί της κυκλικής συνάρτησης και βγάζουμε ένα κρυπτογραφημένο μήνυμα m_2.Η διαδικασία αυτή συνεχίζεται μέχρι όλοι οι γύροι(rounds) να έχουν εμφανιστεί.Αυτή η διαδικασία φαίνεται και στη παρακάτω εικόνα(**Εικόνα 31**)



Εικόνα 31:Η λειτουργία ενός blockcipher

Διαφορετικοί αλγόριθμοι έχουν διαφορετικές κυκλικές συναρτήσεις.Για παράδειγμα στον 3DESο αριθμός των γύρων είναι 48 ενώ για τον AESείναι 10.Πριν προχωρήσουμε παρακάτω πρέπει να ορίσουμε κάποιες απαραίτητες αφαιρετικές εννοίες(abstractconcepts) των αλγορίθμων τμήματος.Αρχικά θα πρέπει να ορίσουμε τι σήμαινει **ψευδοτυχαία συνάρτηση**(pseudorandomfunction) και τι **ψευδοτυχαία αντιμετάθεση (**pseudorandom pemutation).

Ορισμός 3.1.1
Μια **ψευδοτυχαία συνάρτηση**(PRF) ορίζεται πάνω στο (K,X,Y) οπου Κείναι ο κλειδόχωρος, ένα πεδίο εισόδου(X) και ένα πεδίο εξόδου(Y) και όλο αυτο που κάνει

[27] Ο αλγόριθμος τμήματος 3DES, https://en.wikipedia.org/wiki/Triple_DES

είναι να παίρνει ένα κλειδί, μία είσοδο και μας δίνει ένα στοιχείο απο το πεδίο εξόδου.Αυτό φαίνεται και σχηματικά:

$$F:K\times X\rightarrow Y$$

Η μόνη μας απαίτηση είναι να υπάρχει ένας αποδοτικός αλγόριθμοςπου να μπορεί να αξιολογεί(evaluate) την συνάρτηση μας F(k,x).Για τις συναρτήσεις που δέναπαιτούμε να είναι αντιστρέψιμες απαιτούμε να υπάρχει αυτός ο αποδοτικός αλγόριθμος αξιολόγησης της συνάρτησης.

Ορισμός 3.1.2

Η**ψευδοτυχαία αντιμετάθεση** (PRP) ορίζεται στο (K,X) όπου Κείναι ο κλειδόχωρος ενώ Χ είναι ενα σέτ τιμών.Και αυτό που κάνει είναι να παίρνει ένα στοιχείο απο τον κλειδόχωρο και ένα στοιχείο απο το Χ και να μας δίνει ώς έξοδο ένα στοιχείο που ανήκει πάλι στο Χ.Με μαθηματικό συμβολισμο έχουμε :

$$E:K\times X\rightarrow X$$

Πρέπει να τηρούνται κάποιες απαιτήσεις:

- Να υπάρχει **αποδοτικός αλγόριθμος** που να αξιολογεί την παραπάνω συνάρτηση E(k,x)
- Η συνάρτηση E(k,·) να είναι **ένα προς ένα**.Δηλαδή κάθε στοιχείο του Χ να αντιστοιχίζεται με ένα **ακριβώς**στοιχείο Χ.Επειδή όμως είναι ένα προς ένα τότε είναι και **αντιστρέψιμη συνάρτηση**.Δηλαδή για κάθε δοσμένη είσοδο θα πρέπει να υπάρχει ακριβώς μία έξοδο που να αντιστοιχίζεται στην συγκεκριμένη έισοδο.
- Να υπάρχει ένας **αντίστροφος αποδοτικός αλγόριθμος**D που όταν του δίνουμε συγκεκριμένη έξοδο, θα μας δώσει ώς έξοδο μια εικόνα που αντιστοιχίζεται **ακριβώς** σε αυτήν την έξοδο, δηλαδη D(k,y).

4.2 Πρότυπο κρυπτοθέτησης δεδομένων(DES)

Στις αρχές του 1970 ο HorstFiestel[28]σχεδιάζει τον Lucifer[29]στα ερευνητικά εργαστήρια της IBM[30].Ο Luciferχρησιμοποιούσε 128bitμπλόκ και 128bitsκλειδί.Παρόλα αυτά όμως ήταν ευάλωτος σε επιθέσεις διαφορικής κρυπτανάλυσης [31]για περίπου στα μισά πρώτα κλειδία του.ΗIBMτον υπέβαλλε ώς υποψήφιο για το πρότυπο κρυπτοθέτησης αλλα όμως, το 1973 η NBSυπέβαλλε ένα άλλο πρότυπο που ήταν ουσιαστικά μια διαφοροποιήση στον ήδη υπάρχοντα αλγόριθμοπου ονομάστηκε DES[32] και επιλέχτηκε απο το Ομοσπονδιακό Πρότυπο Επεξεργασίας Πληροφοριών(FIPS) για τις Ηνωμένες Πολιτείες το 1976.

Παρόλο που ο διάδοχος του Luciferδιέθετε άμυνα σε επιθέσεις διαφορικής κρυπτανάλυσης είχε πολύ μικρό κλειδί(56 bits).Έτσι στις μέρες μας θεωρείται **πλέον**ανασφαλής για πολλές εφαρμογές.Μάλιστα τον Ιανουάριο του 1999 οι εταιρείες «Distributed.net[33]» και «Electronic Frontier Foundation[34]» κατόπιν συνεργασίας, «έσπασαν» δημοσίως ένα κλειδί του DES μέσα σε 22 ώρες και 15 λεπτά. Υπάρχουν επίσης ορισμένα αναλυτικά αποτελέσματα που καταδεικνύουν θεωρητικές αδυναμίες

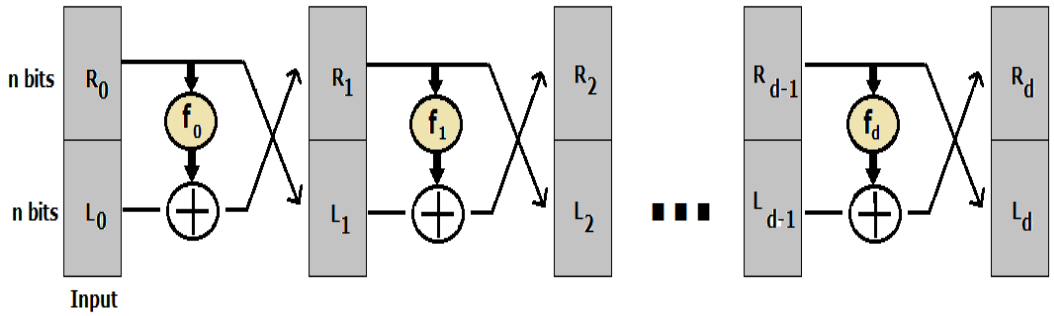
[28]Ο HorstFiestel ενας γερμανός κρυπτογράφος, http://en.wikipedia.org/wiki/Horst_Feistel
[29] Ο αλγόριθμος Lucifer , https://en.wikipedia.org/wiki/Lucifer_(cipher)
[30] Η εταιρία IBM, https://en.wikipedia.org/wiki/IBM
[31] Διαφορική κρυπτανάλυση, https://en.wikipedia.org/wiki/Differential_cryptanalysis
[32]Ο αλγόριθμος DES, http://el.wikipedia.org/wiki/Data_Encryption_Standard
[33]Ηεταιρείαdistributed.net , http://www.distributed.net/Main_Page
[34]Ηεταιρείαelectronic frontier foundation, https://www.eff.org/

στον κρυπταλγόριθμο αν και είναι ανέφικτο να υλοποιηθούν στην πράξη. Θεωρείται πως ο αλγόριθμος είναι **πρακτικά ασφαλής** υπό τη μορφή του τριπλού DES (triple DES) αν και υπάρχουν θεωρητικές αμφισβητήσεις

Η λειτουργία του DESβασίζεται στο **δίκτυο Fiestel**(Fiestelnetwork).Το δίκτυο Fiesteleίναι μια πολύ έξυπνη ιδέα για να δημιουργήσουμε έναν αλγόριθμο τμήματος από αυθαίρετες(arbitrary) συναρτήσεις.Ας φανταστούμε τις συναρτήσειςαυτές να είναι οι $f_1 \ldots f_d$ οι οποίες δεν χρειάζεται να είναι αντιστρέψιμες ή κάτι άλλο.Αυτό που θα κάνουμε είναι δημιουργήσουμε μία αντιστρέψιμη συνάρτηση από όλες.Ο τρόπος που θα γίνει αυτό είναι δημιουργώντας μια συνάρτηση Fη οποία χαρτογραφεί(mapping)2nbitsσε 2nbits

<u>Περιγραφή λειτουργίας</u>

Έστω το R_0(right) και L_0(left) είναι οι είσοδοι μας οι οποίες είναι 2 μπλόκ μήκους nbitsτο κάθε ένα.Ουσιαστικά η έισοδος είναι 2nbits.η ΤιμηR_0πηγαίνει στο L_1 χωρίς κάποια αλλαγή του R_0.Στη συνέχεια ηR_0περνάει μέσα από την συνάρτηση f_0 και το αποτέλεσμα γίνεται πράξη XORμεταξύ της εξόδου του L_0.Το αποτέλεσμα αυτής της διαδικασίας γίνεται το νέο R_1.Ολη αυτή η διαδικασία που μόλις περιγράψαμε ονομάζεται ένας γύρος του δικτύου Fiestelo οποίος έγινε με την χρήση της συνάρτησης f_1(**Εικονα 32)**



Εικόνα 32:Η λειτουργία του δικτύου Fiestel

Η διαδικασία αυτή συνεχίζεται μέχρι μέχρι να πάμε στον τελευταίο γύρο ο οποίος χρησιμοποιεί την συνάρτηση f_d και να πάρουμε ως έξοδο τα R_d, L_d.Με συμβολισμούς έχουμε :

$$\mathbf{L_i=R_i\ ,\ R_i=f_i(R_{i-1})\oplus L_i \gamma\iota\alpha i=1,\ldots,d}$$

Εάν θεωρήσουμε ότι οι δοσμένες είσοδοι είναι οι R_{i+1},L_{i+1}τότε μπορούμε να υπολογίσουμε τα R_i,L_iμε αντίστροφη κατεύθυνση:

$$\mathbf{R_i=L_{i+1}}\text{ενώ}\mathbf{L_i=f_{i+1}(L_{i+1})\oplus R_{i+1}.}$$

<u>Θεώρημα4.2.1</u>

Το θεώρημα αυτό έχει δημιουργθεί από τους Luby και Rackoff το 1985 και μας δέιχνει ότι έαν έχουμε μια συνάρτηση η οποία είναι ασφαλής και ψευδοτυχαία(PRF) τοτε είναι αρκετοί 3 γύροι Fiestel για να πάρουμε ως αποτέλεσμα μιαPRF.Δηλαδή το αποτέλεσμα θα είναι μια αντιστρέψιμη συνάρτηση η οποία δεν θα διακρίνεται από μια πραγματικά τυχαία αντιστρέψιμη συνάρτηση.

Ειναι χρήσιμο διότι όπως έχουμε μάθει για να είναι ένας αλγόριθμος τμήματος ασφαλής πρέπει να υποστηρίζει**ασφαλή ψευδοτυχαία αντιμετάθεση**(PRF).Πιο

απλοικά, αυτό που μας λέει το θέωρημα είναι ότι ένα ένα ξεκινήσουμε με μία ασφαλή ψευδοτυχαία συνάρτηση τότε θα μπορέσουμε να καταλήξουμε με έναν ασφαλή αλγόριθμο τμήματος.

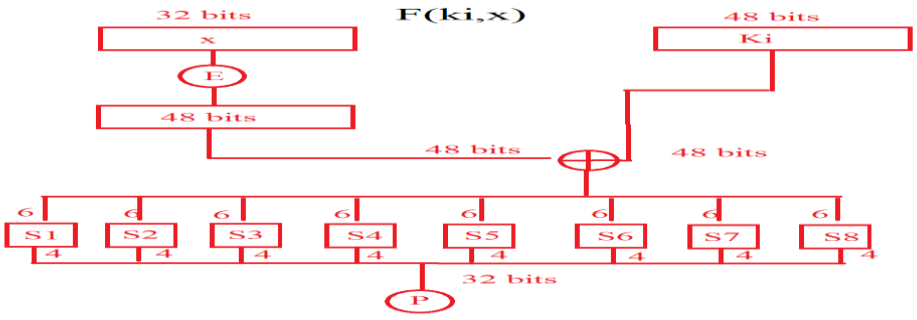
Η λειτουργία του DES

Τώρα που είδαμε το πώς λειτουργεί το δίκτυο Fiestelμπορούμε να τεκμηριώσουμε τη λειτουργία του αλγορίθμου DES.Ο αλγόριθμος DESαποτελείται από 16 γύρους δικτύου Fiestel.Έστω οι συναρτήσεις $F_1 \ldots F_{16}$ οι οποίες χαρτογραφούν 32bitsσε 32bits$\{0,1\}^{32} \rightarrow \{0,1\}^{32}$ με αποτέλεσμα το μπλόκ εισόδου στον DESνα είναι 64bits.Ολες αυτές οι 16 κυκλικές συναρτήσεις παράχθηκαν από μία και **μόνο συνάρτηση F**με την χρησιμοποιήση διαφορετικών κυκλικών κλειδιών.Σε επιπεδο highlevel αυτό που κάνει ο DESείναι να μεταθέτει αυτά τα 64bits.Το αποτέλεσμα των μεταθέσεων εισέρχεται στο δίκτυο Fiestel για 16 γύρους.Οι έξοδοι μετα τον τελευταίο γυρο του Fiestelπάλι περνάνε μέσα από μια αντιμετάθεση (finalpermutation)που είναι η αντιστροφή της αρχικής αντιμετάθεσης(initialpermutation)

Τα κλειδιά όπως είπαμε παράγονται **από ένα συγκεκριμένο κλειδί K**.Το κλειδί αυτό έχει μήκος 56bits το οποίο επεκτείνεται σε 16 κυκλικά κλειδιά που χρειαζόμαστε όπου το κάθε κυκλικό κλειδί έχει μήκος 48bits .Εαν θέλει κάποιος να αντιστρέψει τον αλγόριθμο τότε μπορεί να χρησιμοποιήσει αυτά τα 16 κλειδιά με αντίστροφη σειρά.Το μόνο που μένει είναι να τεκμηριώσουμε την συνάρτηση $\mathbf{F(k_i,x)}$

Για να το κάνουμε αυτό αρχικά θα δούμε πώς λειτουργεί.Η συνάρτηση Fπαίρνει ως είσοδο μια τιμή xη οποία έχει τιμή μήκος 32bits και το κυκλικό κλειδί που έχει μήκος 48bits.Το πρώτο που κάνει είναι να περνά μέσα από ένα **expansionbox(E)** το οποίο παίρνει τα 32 αυτά bits της τιμής xκαι τα χαρτογραφείμε κατάλληλο τρόπο ώστε αν φτάσουν τα 48 bits.

Κατόπιν υπολογίζουμε την πράξη XORμεταξύ του Εκαι του κυκλικού κλειδιού.Το αποτέλεσμα της πράξης σπάει σε ομάδες των 6 bits.Η κάθε 6αδα bitsπηγαίνει δηλαδή σε ένα **S-Box**.Έχουμε δηλαδή 8 S-Boxes στο σύνολο.Στη συνέχεια το κάθε S-Boxχαρτογραφεί αυτά τα 6 bitsσε 4 bits.Αρα συνολικά αυτά τα 8 S-Boxesμας δίνουν 32 bits.Τελικα τα 32 bitsμετατίθενται(P) και το αποτέλεσμα της μετάθεσης είναι η έξοδος της συνάρτησης F(**Εικονα 33**)



Εικόνα 33:Η λειτουργια της F

Οσο αφορά για τα τα **S-Boxes**, δέν είναι τίποτα άλλο απο συναρτήσεις που παίρνουν ως αυτά τα 6 bitskαι δίνουν ως έξοδο 4 bits $\{0,1\}^6 \rightarrow \{0,1\}^4$.Αυτό γίνεται μέσω ενός

ειδικού πίνακα(lookuptable[35]) ο οποίος διαθέτει 64 τιμές στο σύνολο μιας και 2^6=64 διαφορετικές τιμές(**Εικόνα 34**)

S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Εικόνα34:Ενα S-Box look up table

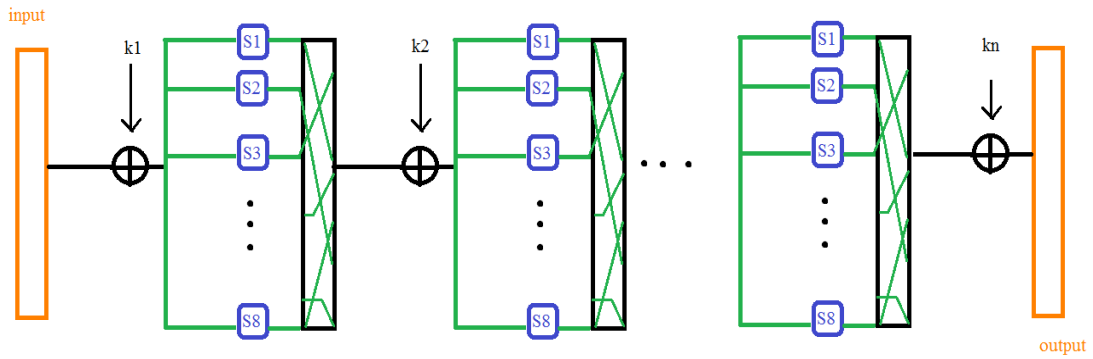
4.3 Το προηγμένο πρότυπο κρυπτογράφησης(AES[36])

Ο AESέχει μήκος κλειδιού 128 bitsαλλά μπορεί να φτάσει μέχρι τα 256 bits.Ο AESδημιουργήθηκε με βάση το**δικτύοαντικατάστασης και αντιμετάθεσης**(substitutionpermutationnetwork) το οποίο δεν είναι Fiesteldίκτυο.Να σημειώσουμε ότι στο Fiesteldίκτυο τα μισά bitsμένουν αναλλοίωτα σε κάθε γύρο ενώ στο δίκτυο αντικατάστασης και αντιμετάθεσης όλα τα bitsαλλάζουν σε κάθε γύρο.

Δίκτυο αντικατάστασης και αντιμετάθεσης

Εστω ότι έχουμε τον πρώτο γύρο του δικτύουαντιμετάθεσης.Το πρώτο πράγμα που κάνουμε είναι μια πράξη XORτης εισόδου με το πρώτο κυκλικό κλειδί και το αποτέλεσμα το περνάμε από ένα επίπεδο αντικατάστασης(substitutionlayer) όπου τα μπλόκ κατάστασης (S_1,S_2, S_3…,S_8) αντικαθίστανται απο άλλα μπλόκς σύμφωνα με έναν πίνακα αντικατάστασης.

Μετά περνάμε μέσα απο ένα permutationlayer με σκοπό τα bitsμετατίθενται και ανακατεύτονται αναμεταξύ τους.Αυτή τη διαδικασία την κάνουμε συνεχώς αλλα κάθε φορά χρησιμοποιούμε διαφορετικό κυκλικό κλειδί(**Εικόνα 35**).Ενα σημαντικό σημείο σε αυτόν τον σχεδιασμό είναι ότι λόγο της σχεδίασης του δικτυου πρέπει κάθε βήμα σε αυτό να είναι αντιστρέψιμο.



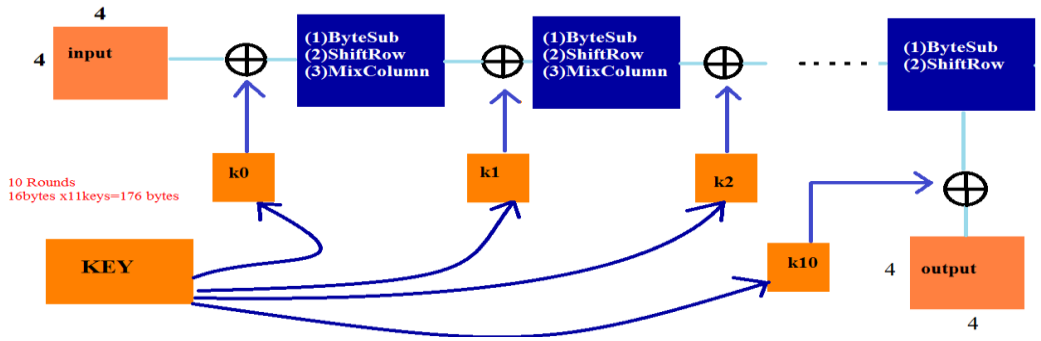
Εικόνα 35:Δικτυο αντικατάστασης και αντιμετάθεσης

Αρα όλη η διαδικασία είναι αντιστρέψιμη.Ο τρόπος που αποκρυπτογραφούμε είναι να πάρουμε την έξοδο και να ακολουθήσουμε κάθε βήμα με αντίστροφη σειρά.Ας

[35] Ένας πίνακας lookup, https://en.wikipedia.org/wiki/Lookup_table
[36]Προηγμένο πρότυπο κρυπτογράφησης,http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

δούμετην διαδικασία ειδικά για τον **AES128**.Ο AES128χρησιμοποιεί 128bitsμπλοκδηλαδή 16 bytes.Γράφουμε αυτά τα 16 bytesσαν έναν 4×4 πίνακα στον οποίο στο κάθε κελί του έχει 1 byte και ξεκινάμε τον πρώτο γύρο.Πρώτα κάνουμε πράξη XORμε το πρώτο κυκλικό κλειδί k_0.

Μετά εφαρμόζουμε κάποια συνάρτηση η οποία περιλαμβάνει κάποιες αντικαταστάσεις(SubBytes) και αντιμεταθέσεις(ShiftRow) και άλλες διαδικασίες στην παρούσα κατάσταση(MixColumn).Στη συνέχεια κάνουμε πράξηXOR μεταξύ του αποτέλεσματος που πήραμεκαι του κυκλικού κλειδιου k_1.Επαναλαμβάνουμε την ίδια διαδικασία με διαφορετικό κλειδί κάθε φορά.Την διαδικασια αυτή θα την πραγματοποιήσουμε για γύρους.Στον 10^o γύρο δέν θα χρησιμοποιήσουμε όμως τη συνάρτηση mixcolumn(**Εικόνα 36**)



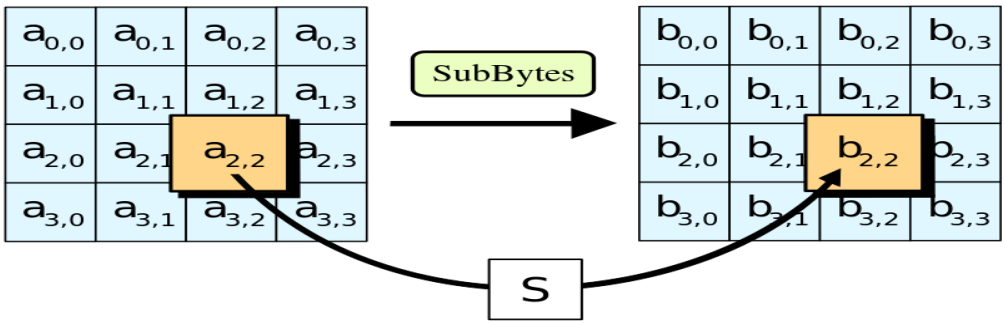
Εικόνα 36:Η λειτουργία του αλγορίθμου AES

Να τονίσουμε ότι τακυκλικά κλειδιά έρχονται απο το 16byteAESkey χρησιμοποιώντας επέκταση.Η επέκταση φτάνει σε 11× 16bytes=176bytes.Το κάθε κλειδί απο αυτά είναι επίσης ένας πίνακας 4× 4 τα οποία έχουν υποστεί πράξη XORσε κάθε κατάσταση

<u>Η συνάρτηση SubBytes</u>

Σε κάθε βήμα της συνάρτησης SubBytesκάθε $bytea_{i,j}$αντικαθίσταται με το $S(a_{i,\ j})$ χρησιμοποιώντας ένα 8bitboxαντικατάστασης(Rijndael S-box[37]).Αυτός ο μηχανισμός μας παρέχει μή γραμμικότητα στον αλγόριθμο.Το S-Boxπαράγεται απο ένα multiplicativeinverse[38]πάνω στο $GF(2^8)$το οποίο είναι γνωστό για τις μή γραμμικές ιδιότητες του.Για να αποφύγουμε επιθέσεις που βασίζονται σε απλές αλγεβρικές ιδιότητες το S-Boxκατασκευάζεται χρησιμοποιώντας μια αντίστροφη συνάρτηση και έναν μετασχηματισμό που ονομάζεται affinetransformation[39](**Εικόνα37**)

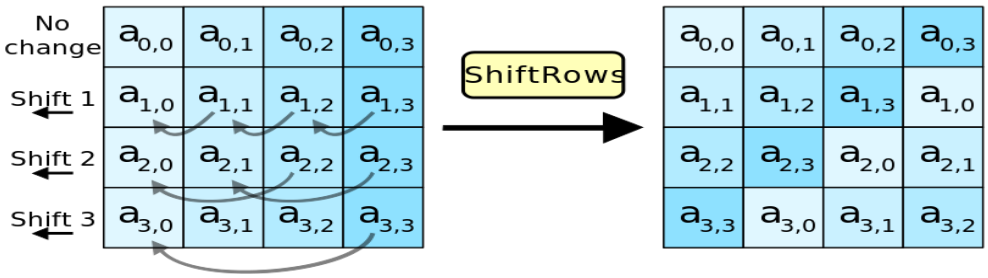
[37]Το S-Box τουRijndael, https://en.wikipedia.org/wiki/Rijndael_S-box
[38]Ημαθηματικήέννοιαmultiplicative inverse, https://en.wikipedia.org/wiki/Multiplicative_inverse
[39] Μια γεωμετρική έννοια, https://en.wikipedia.org/wiki/Affine_transformation



Εικόνα 37:Η λειτουργία της συνάρτησης SubBytes

Η συνάρτηση ShiftRows

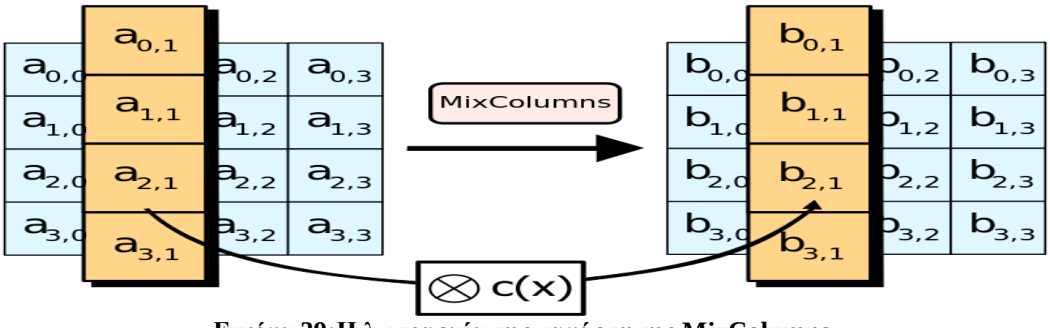
Ησυνάρτηση ShiftRowsεπεξεργάζεται τις γραμμές της εκάστοτε κατάστασης.Κυκλικά ολισθαίνει τα byte της κάθε γραμμής κατα 1 προς τα αριστερά (**Εικόνα 38**)



Εικόνα 38:Η λειτουργία της συνάρτησης ShiftRows

Η συνάρτηση MixColumns

Σε καθε βήμα τα 4 bytes της κάθε στήλης της κατάστασης συνδιάζονται με έναν γραμμικό μετασχηματισμό(lineartransformation).Η συνάρτηση MixColumnsπαίρνει 4 bytes σαν είσοδο και εξάγει 4 bytes ώς έξοδο όπου το κάθε byte επηρεάζει και τα 4 bytes της εξόδου.Σε κάθε βήμα η στήλη μετασχηματίζεται χρησιμοποιώντας έναν fixed πίνακα(**Εικονα 39**)



Εικόνα 39:Η λειτουργία της συνάρτησης MixColumns

4.4 Επιθέσεις στους αλγόριθμους τμήματος

4.4.1 Επιθέσεις εξαντλητικής αναζήτησης(DES)

Θα ξεκινήσουμε με ένα **λήμμα4.4.1**.Ας υποθέσουμε ότι ο DESείναι **ένας ιδανικός αλγόριθμος**.Θα υποθέσουμε δηλαδή ότι ο DESέχει δημιουργηθεί μέσα από τυχαίες αντιστρέψιμες συναρτήσεις.Πιο συγκεκριμένα για κάθε κλειδί ο DESεφαρμόζει και μία αντιστρέψιμη συνάρτηση.Μιας και έχουμε 2^{56} πιθανά κλειδιά στον DES τότε θα υποκριθούμε ότι ο DESείναι μια συλλογή από 2^{56} συναρτήσεων που είναι αντιστρέψιμες από το $\{0,1\}^{64}$ στο $\{0,1\}^{64}$.Τότε θα βγάλουμε ως συμπέρασμα ότι όταν

μας δίνουν ένα μήνυμα και ένα ciphertextυπάρχει ένα και μόνο ένα κλειδί το οποίο χαρτογραφεί αυτό το μήνυμα στο συγκεκριμένο ciphertext.Η πιθανότα να βρούμε το κλειδί ενώ έχουμε το μήνυμα και το ciphertextαγγίζει το 99,5%.

Απόδειξη

Αυτό που ρωτάμε είναι ποιά είναι η πιθανότητα να υπάρχει κάποιο κλειδί k΄ το οποίο δεν είναι ίδιο με το k, τέτοιο ώστε ώστε το c=DES(k,m)=DES(k΄,m).Αν το κλειδί k΄υπάρχει τότε δεν μπορούμε να αποφασίσουμε ποίο από τα δυο αυτά κλειδιά είναι το σωστό αφού και τα δύο αυτά κλειδιά φαίνονται να λειτουργούν κανονικά.Θα υποθέσω ότι αυτό συμβαίνει με μικρή πιθανότητα.Δηλαδή έχουμε την παρακάτω πρόταση :

$$\mathbf{P[\exists\, k' \neq k : C=DES(k, m)=DES(k', m)]}$$

Ολή αυτή η πρόταση θα πρέπει να είναι μικρότερη ή ίση από το άθροισμα των πιθανοτήτων **P[DES(k,m)=DES(k΄,m)]** όπου το $k' \in \{0,1\}^{56}$.Ρωτάμε δηλαδή πώς γίνεται η τυχαία αντιμετάθεση των κλειδιών στο (k΄,m) να παράγει την ίδια τυχαία αντιμετάθεση των (k,m).Εαν αναλογιστούμε ότι για ένα κλειδί η πιθανότητα είναι $1/2^{64}$ αφού οι υπάρχουν 2^{64} πιθανές εξόδους αντιμετάθεση s.Επισης έχουμε συνολικά 2^{56} κλειδιά τοτε η πιθανότητα το κλειδί να μην είναι μοναδικό είναι:

$$\mathbf{1/2^{64} \cdot 1/2^{256}}$$

Η συνολική πιθανότητα είναι πάντα 1 επομένως $1-1/2^{256}$.Ενω εάν το μετατρέψουμε σε ποσοστό έχουμε τότε περίπου **99,5%.**Αρα έαν δωθεί ένα plaintextκαι ένα ciphertexttοτε το**κλειδί που τα συνδέει είναι μοναδικό**■

Μέθοδος εξαντλητικής αναζήτησης

Η μέθοδος αυτή στοχέυει στο να βρούμε το ζητούμενο κλειδί δοκιμάζοντας όλα τα πιθανα κλειδιά.Οπως καταλαβαίνουμε η μέθοδος αυτή ειναι χρονοβόρα και συνήθως δέν μας δίνει τα επιθυμητά αποτελέσματα διότι το σύνολικο πλήθος των κλειδιών είναι απλα ασσύληπτο.Ομως για τον απλό DESδέν ισχύει αυτό μιας και το κλειδί του έχει μήκος μόνο 56 bit.Η RSAπροσπάθησε να σπάσει τον DESορίζοντας μια πρόκληση για το ευρύ κοινό

Αυτό που έκανε ήταν να δημοσιέυσει έναν μεγάλο αριθμό από ciphertexts ενώ για 3 μόνοαπό αυτά τα ciphertextseίχαν δωθεί τα plaintexts τους.Το ζητούμενο ήταν με αυτά τα δεδομένα ένα μπορούσαν να αποκρυπτογραφήσουν τα μηνύματα όλων των ciphertexts.Μιά αλλη προσπάθεια έγινε το 1997 χρησιμοποιώντας το disturbed.netΜπόρεσαν σε 3 μόλις μήνες να βρούν το σωστό κλειδί μέσα από τον κλειδόχωρο που έχει σύνολο 2^{56}κλειδιά.Επίσης το 1998 η EFFεκλείσαν συμβόλαιο με τον PaulKocher[40]με σκοπό να φτιάξει ένα hardwareτο οποίο θα μπορούσε να σπάσει τον DES.Η μηχανή που δημιούργησε ονομάστηκε deepcrack[41], κόστιζε 250.000$ και έσπασε τον DESσε μόλις 3 ημέρες!

Μια άλλη προσπάθεια έγινε το 1999 όταν η RSAεισήγαγε ακόμη μία πρόκληση που ήταν ο συνδυασμός της μεθόδουinternetsearchκαι deepcrack.Το αποτελεσμα αυτής της πρόκλησης ήταν να σπάσει ο DES σε 22 ώρες.Τέλος το 2006 ένα άλλο hardwareπου ονομάζονταν COPACOBANAπου κόστιζε μόνο 10.000$ και χρησιμοποιούσε 120

[40]Ο κρυπτογράφος PaulKocher, http://en.wikipedia.org/wiki/Paul_Kocher
[41]Η μηχανη deepcrackκαι COPACOBANA, https://en.wikipedia.org/wiki/Custom_hardware_attack

FPGA'S μπόρεσε να σπάσει τον DESσε 7 ημέρες.Δηλαδή τα 56 bitkey του DESείναι **εντελώς ανασφαλή** και σπάνε πολύ έυκολα όπως έιδαμε.

Επειδή ο DES όμως είχε μεγάλη υποστήριξη από το κοινό και μεγάλο hardwaresupportto θέμα ήταν έαν μπορούμε να βελτιώσουμε τον DES έτσι ώστε να γίνει περισσότερο ασφαλής.Δηλαδή να πάρουμε τον ήδη υπάρχοντα DESνα προσπαθήσουμε να επεκτείνουμε το μήκος του κλειδιού ώστε να γίνει πιο ανεκτικός στις επιθέσεις εξαντλητικής αναζήτησης.Ετσι δημιουργήθηκε ο **3DES**.

Υποθέστε ότι μας δίνουν ένα αλγόριθμο τμήματος Επου έχει ένα κλειδόχωρο K,ένα χώρο μηνυμάτων Μκαι ένα χώρο εξόδου μηνυμάτων Μ.Ο αλγόριθμος 3DES χρησιμοποιεί 3 ανεξάρτητα μεταξύ τους κλειδιά τα οποία κρυπτογραφούν το ίδιο μήνυμα.Πρώτα κρυπτογραφούμε με το κλειδί k_3, μετά αποκρυπτογραφούμε χρησιμοποιώντας το κλειδί k_2και τέλος κρυπτογραφούμε πάλι χρησιμοποιώντας όμως το κλειδί k_1 δηλαδή :

$$\mathbf{E(k_1,D(k_2,E(k_3,m)))}$$

Γιατί όμως κάνουμε 2 κρυπτογραφήσεις και 1 αποκρυπτογράφηση και δεν κάνουμε 3 κρυπτογραφήσεις;Αυτό συμβαίνει διότι αν τα 3 κλειδιά είναι ίδια τότε θα έχουμε τον απλό DES.Αυτο μας προστατέυει από κάποιο hackσε περίπτωση που κάποιος προσθέσει σε ένα hardware ένα μηχανισμό που να μπορεί να θέτει τα 3 κλειδιά γίνουν ίσα μεταξύ τους.

Οσο αφορά την ταχύτητα ο 3DESείναι 3 φορές πιο αργός από τον απλό DES καθώς το κλειδί του έχει μήκος $3\cdot56=168$bits.Επομένως θα χρειαστεί πολύ περισσότερος χρόνος για να γίνει μια σωστή επίθεση στον 3DES.Ένα ακόμη ερώτημα είναι γιατί δεν χρησιμοποιήσαμε **2DES**;Θα περιγράψουμε μια επίθεση στον 2DES που τον κάνει ανασφαλή παρόλου που είναι πιο γρήγορος από τον 3DESκαι έχει και ένα ικανοποιητικού μεγέθους κλειδί(112bits).

Meetinthemiddleattack(DES)

Αυτή η επίθεση απαιτέι να γνωρίζουμε μερικά ζευγάρια ciphertext/plaintext.Ας υποθέσουμε ότι έχουμε ένα ζευγάρι από αυτα.Ας δούμε πρώτα πώς γίνεται η διπλή κρυπτογράφηση:

$$\mathbf{p\rightarrow E(k_1,\ p)\ \rightarrow E(k_2,\ E(k_1,\ p)\ )=C}$$

Κρυπτογραφούμε το ρχρησιμοποιώντας 2^{56} πιθανά κλειδιά και αποθηκέυουμε τα αποτελέματα(η αποθήκευση αυτών είναι ένα πρόβλημα).Τα αποθηκευμένα αποτελέματα επίσης θα περιλαμβανουν όλες τις πιθανές κρυπτογραφήσειςp→E(k_1, p).Θα αποκρυπτογραφήσουμε το Cχρησιμοποιώντας όλα τα 2^{56} κλειδιά.Επομένως η αποκρυπτογράφηση γίνεται ώς εξής :

$$\mathbf{D(k_2,\ C)=D(k_2,\ E(k_2,\ E(k_1,\ p)\ )\ )\ \rightarrow E(k_1,\ p)}$$

Μετά την αποκρυπτογράφηση με το κάθε κλειδί θα ελέγχουμε για το έαν ταιριάαζει με κάποια απο τις 2^{56} αποθηκευμένες τιμές.Έαν έχουμε τάυτιση τότε βρήκαμε ένα σωστό ζεύγος κλειδιών.Άρα μας παίρνει χρόνο 2^{57} για να σπάσουμε τον 2DESμε χρήση εξαντλητικής αναζήτησης.

4.4.2Επιθέσεις σχετικά με την εκτέλεση υλικό(DES)

Υποθέστε ότι έχετε μια έξυπνη κάρτα(smartcard)η οποία έχει κάποιον αλγόριθμο τμήματος.Για παράδειγμα μια έξυπνη κάρτα μπορεί να χρησιμοποιηθεί για

creditcardπληρωμές άρα πιθανόν να έχει ένα κρυφό κλειδί μέσα σε αυτήν για να επιβεβαιώσει τις αγορές σας μέσω αυτης οταν την τοποθετείτε μέσα σε κάποιο μηχάνημα πληρωμών.

Έστω ότι κάποιος επιτιθέμενος σας κλέβει την έξυπνη κάρτα.Αυτό που μπορεί να κάνει είναι να πάει την έξυπνη κάρτα σε ένα εργαστήριο και να μετρήσει με κάποιο τρόπο πόσο χρόνο χρειάζεται η κάρτα για να κάνει κρυπτογράφηση και αποκρυπτογράφηση.Αν ο χρόνος της εκτέλεσης της διαδικασίας είναι σχετικός με το κλειδί τότε μετρώντας τον χρόνο, ο επιτιθέμενος μπορεί να μάθει κάποια πράγματα για το μυστικό κλειδί.

Μια άλλη επίθεση είναι αντί να μετρήσουμε τον χρόνο, μπορούμε να μετρήσουμε την κατανάλωση ρεύματος της κάρτας ενώ είναι σε λειτουργία.Μπορούμε απλά να την συνδέσουμε σε μία συσκευή και να φτιάξουμε ένα γράφημα για το πώς συμπεριφέρεται ανα χρονική στιγμή.Επειδή αυτές οι κάρτες δεν είναι πού γρήγορες μπορούμε να μετρήσουμε ακριβώς την ποσότητα ρέυματος που χρησιμοποιείται σε κάθε κύκλο ρολογιού την ώρα που ήταν σε λειτουργία.

Απο ένα ειδικό γράφημα μπορούμε να δόυμε πότε κάνει αρχική αντιμετάθεση και πότε τελική αντιμετάθεση .Επίσης βλεπούμε καθαρά τους 16 γύρους επεξεργασίας του DESαφού σε κάθε ύψωμα του γραφήματος θεωρείται και ένας γύρος του DES.Μπορούμε να βρόυμε κάθε bittου κλειδιού με απλά παρατηρώντας το πώς συμπεριφέρεται η κάρτα σε κάθε λειτουργία.Το παραπάνω είδος επίθεσης ανήκει στο είδος των **sidechannelattacks**[42].Οι πρώτοι που τις εφήυραν έιναι οι Kocher,Jane,Junto 1998.Ένα άλλο είδος επιθέσεων είναι οι επιθέσεις **σφάλματος**(faultattacks).Θα μπορούσαμε να υπερθερμάνουμε έναν επεξεργαστή με αποτέλεσμα να τον φέρουμε σε κατάστασηoverclockingμε σκοπο να μας δώσει μια δώσει δεδομένα από errors.

4.4.3 Γραμμικες[43] και διαφορικές επιθέσεις [44](DES)

Οι επιθέσεις αυτές ανακαλύφθηκαν απο τον Bihamκαι τον Shamirτο 1989.Θα αναφερθούμε σε μια versionτης επίθεσης που ανακαλύφθηκε απο τον Matsuito 1993.Ο σκοπός μας είναι όταν μας δώσουν πολλά ζευγάρια plaintext-ciphertextνα μπορέσουμε να ανακαλύψουμε το κλειδί σε χρόνο μικρότερο απο 2^{56}.Άρα η επίθεση αυτή να είναι καλύτερη απο την εξαντλητική αναζήτηση.Το παράδειγμα που θα σας δείξουμε ονομάζεται γραμμική κρυπτανάλυση(linearcryptanalysis).Υποθέστε ότι cείναι η κρυπτογράφηση ενός μηνύματος mμε κλειδί κρυπτογράφησης να είναι το k.

Υποθέστε επίσης ότι έαν κοιτάξω σε ένα τυχαίο κλειδί και σε ένα τυχαίο μήνυμα θα βρώ μια εξάρτηση μεταξύ του ciphertext, του μηνύματος και του κλειδιού.Πιο συγκεκριμένα έαν κάνω πράξη XORμεταξύ ενός υποσυνόλου που περιέχει bitsτων μηνυμάτων και ενός υποσυνόλου που περιέχει τα bitsτων ciphertextκαι συγκρίνουμε το αποτέλεσμα με μια πράξη XORπου θα συμβεί στα bitsτου κλειδιού τότε έαν τα αποτελέσματα αυτά είναι τελειώς ανεξάρτητα(που αυτό θέλουμε ουσιαστικά ώστε να μη μπορεί κάποιος με το plaintextκαι το ciphertextνα μπορεί να προβλέψει τα bitsτου

[42]Επιθέσειςside channel, https://en.wikipedia.org/wiki/Side-channel_attack
[43]Γραμμικές επιθέσεις κρυπτανάλυσης, http://en.wikipedia.org/wiki/Linear_cryptanalysis
[44]Διαφορικές επιθέσεις κρυπτανάλυσης,http://en.wikipedia.org/wiki/Differential_cryptanalysis

κλειδιού) τότε αυτή ισότητα των δύο αποτελεσμάτων στέκει για πιθανότητα 1/2 θα θεώρησουμε και ένα biasΕέτσι ώστε να έχουμε 1/2 + Ε.Με συμβολισμούς έχουμε :

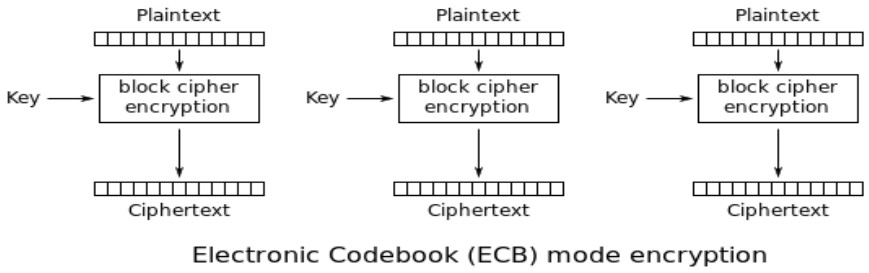
$$\mathbf{P[m[i_1]\oplus\ldots\oplus m[i_r]\ \oplus c[j_j]\oplus\ldots\oplus c[j_v] = k[l_1]\oplus\ldots\oplus k[l_u]\ ]=1/2+\varepsilon}$$

Όμως υπάρχει αυτή η σχέση και οφείλεται σε ένα bugστον σχεδιασμό του πέμπτου S-Boxτου DES διότι συμβαίνει να είναι πολύ κοντά στο να μοιάζει με γραμμική συνάρτηση με αποτέλεσμα να μας γεννάει αυτή την σχέση που προαναφερθήκαμε.Αυτό το πολύ μικρό λάθος στο σχεδιασμό όχι μόνο δημιουργεί αυτη τη σχέση αλλα μας θέτει και το Επολύ μικρό($\varepsilon=1/2^{21}$).

4.5Τρόποιλειτουργίας (modesofoperation)

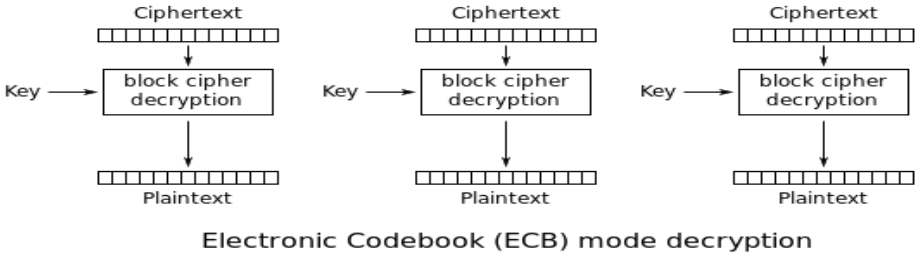
4.5.1ECBmode

Αυτός ο τρόπος λειτουργίας μας περιγράφει το πώς μπορούμε να κάνουμε τις απαραίτητες επαναλήψεις ενός μπλόκ με σκοπό να μπορέσουμε να μετασχηματίσουμε με ασφαλή τρόπο δεδομένα μεγαλύτερα απο ένα μπλόκ.Ο τρόπος λειτουργίαςECBείναι πολύ απλός.Το μήνυμα διαιρείται σε μπλόκς και κάθε μπλόκ κρυπτογραφείται ξεχωριστά απο τα υπόλοιπα μπλόκς(**Εικόνα 40**)



Εικόνα 40:Κρυπτογράφηση ECB

Το μόνο μειονέκτημα σε αυτόν τον τρόπο λειτουργίας είναι οτι όμοια plaintextsθα έχουν όμοια ciphertexts.Δέν παρέχει σημαντική εμπιστευτικότητα και δέν θα πρέπει να το χρησιμοποιούν τα κρυπτογραφικά πρτόκολλα.Επίσης το ECBμπορεί να κάνει τα πρωτόκολλα που δέν παρέχουν ακεραιότητα ακόμη πιο ανασφαλή μιας και θα είναι επιρρεπή σε **replayattacks** μιας και το κάθε μπλόκ κρυπτογραφείται με ακριβώς τον ίδιο τρόπο καθε φορα.Η αποκρυπτογράφηση φαίνεται στο παρακάτω σχήμα(**Εικόνα 41**)

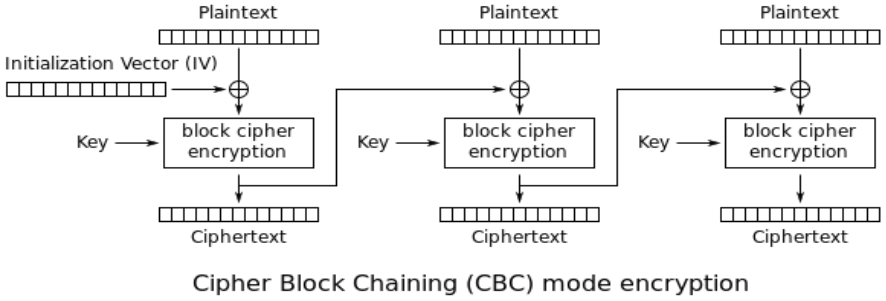


Εικόνα 41:Αποκρυπτογράφηση ECB

4.5.2 CBCmode

Το μήνυμα σπάει σε μπλόκς.Καθε plaintextμπλοκ υφίσταται πράξη XORμε το προηγούμενο ciphertextπρωτού κρυπτογραφηθεί(**Εικόνα 42**).Ειδικά για το πρώτο

μπλόκ η πράξη XORγίνεται μεταξύ μιας ακολουθίας που ονομάζεται IVκαι το αποτέλεσμα της πράξης δίνεται στο επόμενο μπλόκ για να συνεχιστεί η διαδικασία κρυπτογραφησης.



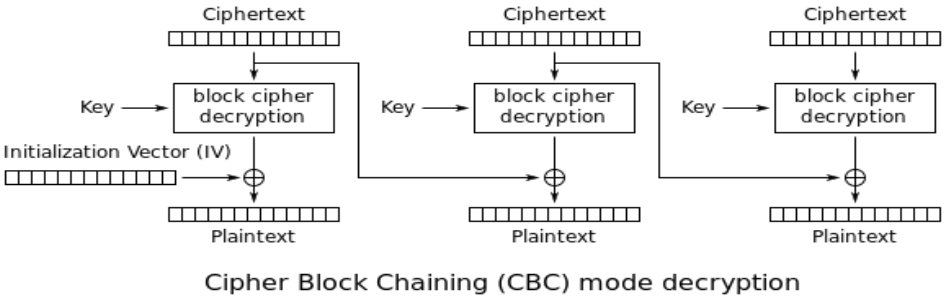
Εικόνα42:Κρυπτογράφηση CBC

Με μαθηματικούς συμβολισμούς έχουμε για την κρυπτογράφηση και την αποκρυπτογράφηση ότι:

$$\mathbf{C_i=E_K(P_i \oplus C_{i-1}),\ C_0=IV}$$

$$\mathbf{P_i=D_k(C_i) \oplus C_{i-1},\ C_0=IV}$$

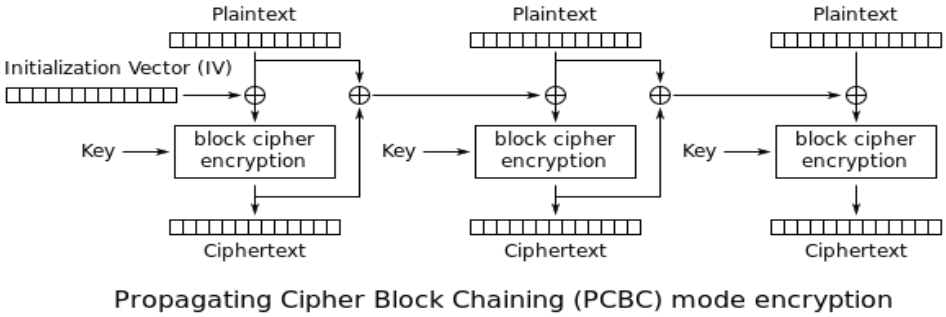
Επειδή η διαδιακασια δέν ειναι παράλληλη θα πρέπει να χρησιμοποιήσουμε κάποιο paddingετσι ώστε το μήνυμα είναι πολλαπλάσιο του μέγεθος του μπλόκ.Η χρήση του IV(initializationvector) είναι απαραίτητη σε αρκετούς τρόπους λειτουργίας καθώς ο ρόλος του είναι να μας δίνει προστασία σε περίπτωση που το ίδιο μήνυμα κρυπτογραφείται αρκετές φορές και έτσι θα παράγει το ίδιο ciphertext.Με το IVμπορούμε αν αλλάζουμε το ciphertextων μηνυμάτων που κρυπτογραφούνται πολλές φορές μέσα σε μια κρυπτογράφικη διαδικασία.Παρακάτω φαίνεται η αποκρυπτογράφηση με CBC(**Εικόνα 43**)



Εικόνα 42:Αποκρυπτογράφηση CBC

3.5.3 PCBCmode

Μοιάζει αρκετά με τον τρόπο λειτουργίαςCBC.Αρχικά σπάμε το μήνυμα μας σε μπλόκς.Στη συνέχεια το πρώτο plaintextμπλόκ περνά μαζί με το IVμέσα απο μια πράξη XORκαι το αποτέλεσμα αυτού, περνά μεσα απο έναν αλγόριθμο τμήματος όπο μέσω ενός κλειδιού k θα μας βγάλει ώς έξοδο το πρώτο ciphertextμας(**Εικόνα44**)Toplaintexttου πρώτου μπλόκ πηγαίνει σε μια πράξη XORμαζί με το ciphertextπου μόλις υπολογίσαμε.



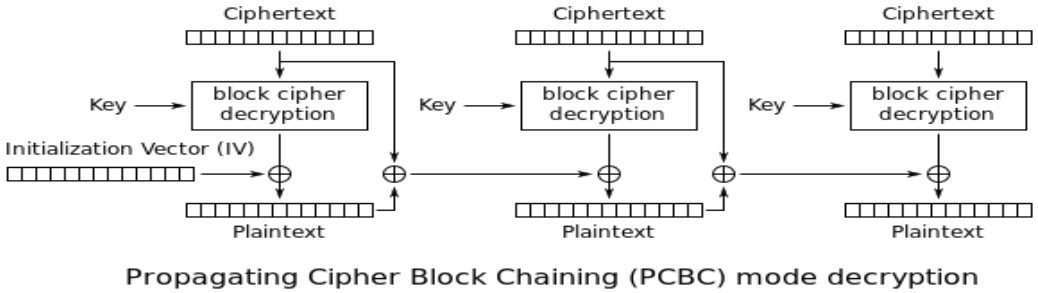
Εικόνα 43:Κρυπτογράφηση PCBC

Το αποτέλεσμα αυτής της πράξης XOR πηγαίνει μαζί με το δεύτερο plaintext μπλοκ σε μια άλλη πράξη XOR και το αποτέλεσμα αυτής περνά απο τον αλγόριθμο τμήματος και μας βγάζει ώς αποτέλεσμα το δεύτερο ciphertext μας.Η διαδικασία αυτή συνεχίζεται αλυσιδωτά μέχρι να χρησιμοποιήσουμε και το τελευταίο μπλόκ του μηνύματος μας.Με μαθηματικούς συμβολισμούς έχουμε:

$$\mathbf{C_i=E_k(\ P_i\ \oplus P_{i-1} \oplus C_i\ ),\ P_0 \oplus C_0=IV}$$

$$\mathbf{P_i=D_k(C_i) \oplus P_{i-1} \oplus C_{i-1}\ ,\ P_0 \oplus C_0=IV}$$

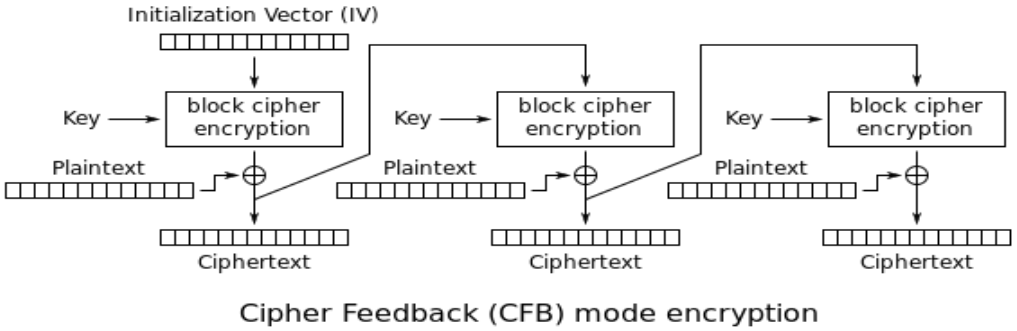
Η αποκρυπτογράφηση με PCBCφαίνεται στο παρακάτω σχήμα(**Εικόνα 45**)



Εικόνα 44:Αποκρυπτογράφηση PCBC

4.5.4 CFB mode

Ο τρόπος λειτουργίας μοιάζει αρκετά πολύ με τον CBC οποίος μετατρέπει τον αλγόριθμο σε έναν αυτοσυχρονιζόμενο αλγόριθμο ροής.Αρχικά τοποθετούμε το IVμέσα σε έναν αλγόριθμο τμήματος όπου με ένα κλειδί kθα κρυπτογραφηθεί.Το αποτέλεσμα της κρυπτογράφησης περνά μαζι με το πρώτο plaintextμπλόκ του μηνύματος μας μέσα σε μια πράξη XORγια να παράγουμε το πρώτο ciphertext(**Εικόνα 46**).Στη συνέχεια το αποτέλεσμα της πράξης XOR περνάει απευθείας στον αλγόριθμο τμήματος για να κρυπτογραφηθεί με το δέυτερο μπλόκ plaintext.Αυτη η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο τελευταίο μπλόκ



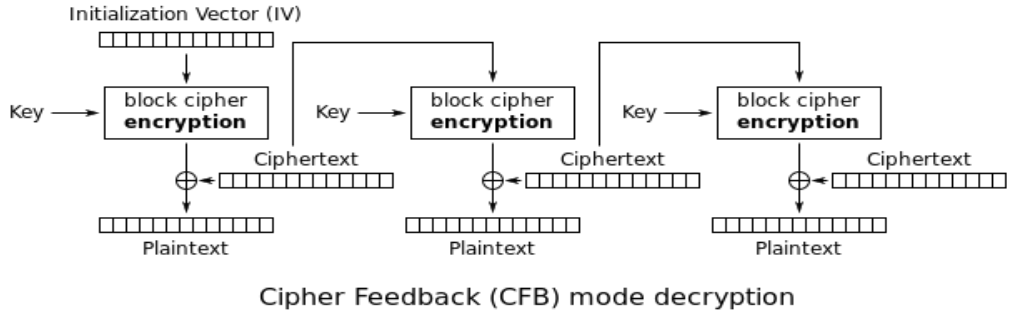
Εικόνα 45:Κρυπτογράφηση CFB

Οι εξισώσεις που διέπουν στην κρυπτογράφηση είναι οι εξής:

$$C_i=E_k(C_{i-1}) \oplus P_i$$

$$P_i=E_k(C_{i-1}) \oplus C_i \text{ και} C_0=IV$$

Εξ ορισμού εαν χαθεί ένα ciphertextσε εναν αυτοσυγχρονιζόμενο αλγόριθμο τότε ο παραλήπτης θα χάσει μόνο ένα μέρος του μηνύματος και θα μπορέσει να συνεχίσει την αποκρυπτογράφηση(**Εικόνα 47)**για τα επόμενα δεδομένα.Το μόνο αρνητικό είναι ότι η κρυπτογράφηση μπορεί να συνεχιστεί μονο εάν χαθεί ολόκληρο το μπλόκ.Εαν χαθεί μόνο ένα byteή μερικά bitstότε η κρυπτογράφηση θα χαλάσει ολοκληρωτικά.Για να είμαστε ικανοί να συγχρονιστούμε μετά απο χάσιμο ενός byteή μερικών bits θα πρέπει να κρυπτογραφείται ένα bytetη φορα(ή μερικά bits).



Εικόνα 46:Αποκρυπτογράφηση CFB

Για να συμβεί αυτό θα χρησιμοποιήσουμε εκτός απο CFB έναν **καταχωρητή ολίσθησης**(shiftregister) σαν είσοδο στον αλγόριθμο τμήματος με σκοπό να δημιουργήσομεμε εναν αυτοσυγχρονιζόμενο αλγόριθμο ροής που θα συγχρονίζεται για κάθε πολλαπλάσιο των xbitsπου ενδεχομένως τύχει να χαθούν.Αρχικά αρχικοποιούμε τον καταχωρητή ώστε να έχει μέγεθος όσο ένα μπλόκ.Αυτή η τιμή κρυπτογραφείται στη συνέχεια με έναν αλγόριθμο τμήματος και τα πιό σημαντικά bits του αποτελέσματος περνάνε σε μια πράξη XORμε τα xbitstου plaintextκαι παράγουμε τελικά xbitsciphertext.Αυτά τα xbitsτης εξόδου ολισθαίνουν μέσα σε έναν καταχωρητή ολίσθησης και η διαδικασία επαναλαμβάνεται για τα επόμενα xbitstου plaintext.

Παρόμοια είναι και η αποκρυπτογράφηση.Η διαδικασία που περιγράψαμε ονομάζεται CFB-8 ή CFB-1(αφου εξαρτάται απο το μέγεθος της ολίσθησης).Εαν θεωρήσουμε ότι S_iείναι η ικατάσταση του καταχωρητή, το head(a,x) είναι τα χμεγαλύτερα bitstου a και έαν πείναι ο αριθμός των bitstου IVέχουμε ότι:

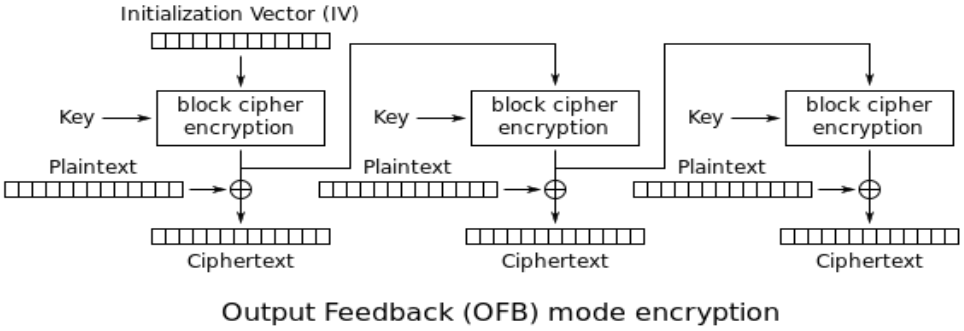
$$C_i=head(E_k(S_{i-1}), x) \oplus P_i$$

$$P_i=head(E_k(S_{i-1}), x) \oplus C_i$$

$$S_i=((S_{i-1}<<x)+C_i) \bmod 2^n \text{και} S_0=IV$$

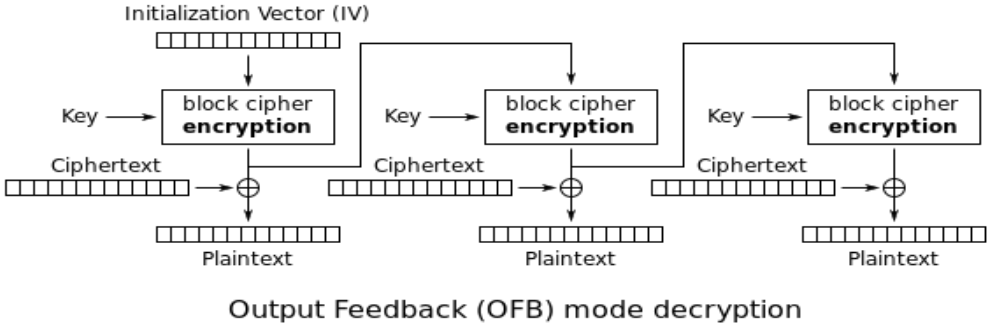
4.5.5 OFBmode

Ο τρόπος λειτουργίας OFBμετατρέπει έναν αλγόριθμο τμήματος σε έναν αλγόριθμο αυτοσυγχρονιζόμενου αλγορίθμου ροής.Δημιουργείkeystreamμπλόκς τα οποία πηγαίνουν σε μια πράξη μαζί με το plaintext για να πάρουμε το ciphertextμας.Λόγω συμμετρίας η κρυπτογράφηση και η αποκρυπτογράφηση είναι ίδια(**Εικόνα 48**)



Εικόνα 47:Η κρυπτογράφηση OFB

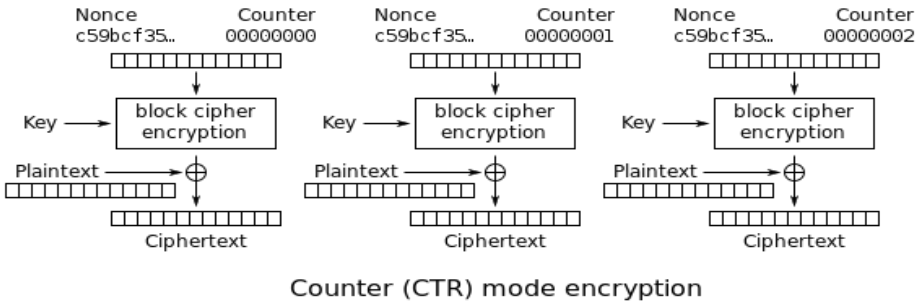
Κάθε διαδικασία βασίζεται στα προηγούμενα βήματα γιαυτό δεν γίνεται να πραγματοποιηθεί παράλληλα.Είναι πολύ έυκολο να πάρουμε OFBχρησιμοποιώντας CBCμε μια σταθερή συμβολοσειρά μηδενικών σαν είσοδο.Ηαποκρυπτογράφηση είναι συμμετρική λόγω της XOR(**Εικόνα 49**)



Εικόνα 48:Αποκρυπτογράφηση OFB

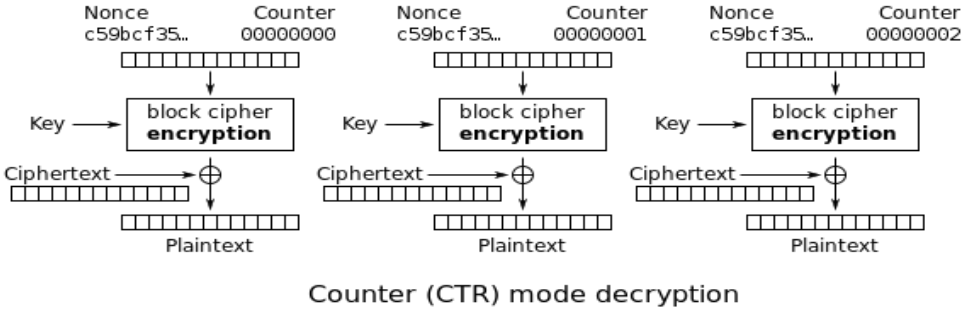
4.5.6 CTRmode

Οπως ακριβώς και ο τρόπος λειτουργίας OFBμετατρέπει έναν αλγόριθμο τμήματος σε ένα αλγόριθμο ροής.Δημιουργεί το κάθε νέο keystreamμπλοκ χρησιμοποιώντας τις τιμές ενός μετρητή.Ο μετρητής μπορεί να είναι οποιαδήποτε συνάρτηση η οποία μπορεί να παράγει μια ακολουθία η οποία δεν θα επαναλαμβάνεται για πολύ μεγάλο χρονικό διάστημα.Η κρυπτογράφηση γίνεται παράλληλα άρα και ξεχωριστά σε κάθε μπλόκ(**Εικόνα 50**)



Εικόνα 49:Κρυπτογράφηση CTR

Παίρνουμε ένα nonce(παρόμοιο με το IV) και μαζί με την τιμή του countertα κρυπτογραφούμε με εναν αλγόριθμο τμήματος.Το αποτέλεσμα γίνεται πράξη XORμαζι με το πρώτο plaintextμπλόκ του μηνύματος μας.Η ίδια διαδικασία επαναλαμβάνεται για όλα τα μπλόκς κάθε φορά ομως με διαφορετική τιμή για τον μετρητή και διαφορετικό nonce.Η αποκρυπτογράφηση(**Εικόνα 51)** γίνεται με παρόμοιο τρόπο.



Εικόνα 50:Αποκρυπτογράφηση CTR

4.6Ερωτήσειςκεφαλαίου

Ερώτηση 1

Consider the following five events:
1) Correctly guessing a random 128-bit AES key on the first try
2) Winning a lottery with 1 million contestants (the probability is $1/10^6$).
3) Winning a lottery with 1 million contestants 5 times in a row (the probability is $(1/10^6)^5$)
4) Winning a lottery with 1 million contestants 6 times in a row.
5) Winning a lottery with 1 million contestants 7 times in a row.

What is the order of these events from most likely to least likely?

a)2, 3, 1, 5, 4
b)2, 3, 4 ,1 ,5
c)2, 3, 5, 4, 1
d)2, 3, 1, 4, 5

Απάντηση

The probability of event (1) is $1/2^{\wedge 128}$.
The probability of event (5) is $1/(10^6)^7$ which is about $1/2^{139}$. Therefore, event (5) is the least likely.

The probability of event (4) is $1/(10^6)^6$ which is about $1/2^{119.5}$which is more likely than event (1).
The remaining events are all more likely than event (4).So the correct answer is **b**.

Ερώτηση2
Suppose that using commodity hardware it is possible to build a computer for about $200 that can brute force about 1 billion AES keys per second. Suppose an organization wants to run an exhaustive search for a single 128-bit AES key and was willing to spend 4 trillion dollars to buy these machines (this is more than the annual US federal budget). How long would it take the organization to brute force this single 128-bit AES key with these machines? Ignore additional costs such as power and maintenance.

a)More than a month but less than a year
b)More than a 100 years but less than a million years
c)More than a million years but less than a billion (10^9) years
d)More than a week but less than a month
e)More than a billion (10^9) years

Απάντηση
The answer is about 540 billion years.So correct answer is **e**.
Machines = $(4\cdot10^{12})/200 = 2\cdot10^{10}$.
Keys processed per sec = $10\text{^}9 \cdot (2\cdot10^{10}) = 2\cdot10^{19}$
#Seconds = $2^{128} / (2\cdot10^{19}) = 1.7\cdot10^{19}$.
This many seconds is about 540 billion years.

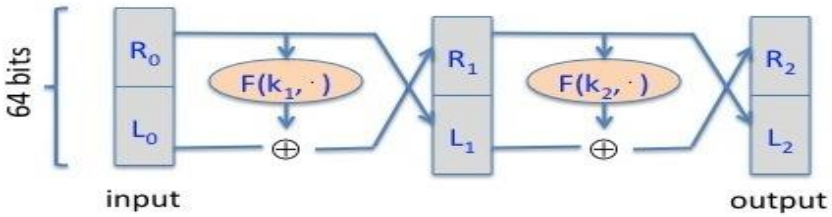
Ερώτηση3
Let$F:\{0,1\}^n\times\{0,1\}^n\rightarrow\{0,1\}^n$be a secure PRF (i.e. a PRF where the key space, input space, and output space are all $\{0,1\}^n$) and say n=128. Which of the following is a secure PRF (there is more than one correct answer)?

a)$F'((k_1,k_2),x)=\begin{cases} F(k_1,x) & when, x \neq 0^n \\ k_2 & otherwise \end{cases}$
b)$F'(k,x)=F(k,x)\|0$ (here || denotes concatenation)
c)$F'(k,x)=F(k,x) \oplus F(k,x\oplus 1^n)$
d)$F'(k,x)=F(k,x)[0,\ldots.n-2]$ (ie F′(k,x)drops the last bit of F′(k,x)
e)$F'((k_1,k_2),x)=F(k_1,x)\|F(k_2,x)$ (here || denotes concatenation)
f)$F'(k,)=\begin{cases} F(k,x) & when\ x \neq 0^n \\ 0^n \end{cases}$

Απάντηση
The answer f is not correct: This function is not secure because the adversary would submit $0^n$and guessthat that the function is not random if the output is zero.
So Adv[A,F] = |0 - 1| = 1
The answer a is correct:Despite the fact that a key is being sent, this function is secure because when the adversary submits nonzero messages,he learns only as much about F' as he would learn about F, but F is secure. When he submits 0^n, he gets a value that is chosen uniformly from K, so it is indistinguishable from random.Similarly **the answers d and e are correct**.

The answer b is not correct: A distinguisher will output not random whenever the last bit of $F(k,0^n)$ is 0.
The answer c is not correct:A distinguisher will query at $x=0^n$ and $x=1^n$ and output *not random* whenever the two responses are equal. This is unlikely to happen for a truly random function.

Ερώτηση4
Recall that the Luby-Rackoff theorem discussed in Lecture 3.2 states that applying a three round Feistel network to a secure PRF gives a secure block cipher. Let's see what goes wrong if we only use a two round Feistel. Let $F: K\times \{0,1\}^{32}\rightarrow \{0,1\}^{32}$be a secure PRF. Recall that a 2-round Feistel (**Είκονα 52**) defines the following PRP $F_2:K_2\times\{0,1\}^{64}\rightarrow\{0,1\}^{64}$:



Εικόνα51: 2 γύροιFiestel

Here R_0 is the right 32 bits of the 64-bit input and L_0 is the left 32 bits.One of the following lines is the output of this PRP F_2 using a random key, while the other three are the output of a truly random permutation $f:\{0,1\}^{64}\rightarrow\{0,1\}^{64}$. All 64-bit outputs are encoded as 16 hex characters. Can you say which is the output of the PRP?Note that since you are able to distinguish the output of F_2 from random, F_2 is not a secure block cipher, which is what we wanted to show.**Hint**: First argue that there is a detectable pattern in the xor of $F_2(\cdot,0^{64})$ and $F_2(\cdot,1^{32}0^{32})$Then try to detect this pattern in the given outputs.

a) On input 0^{64} the output is "9d1a4f78 cb28d863".On input $1^{32}0^{32}$ the output is "75e5e3ea 773ec3e6".
b) On input 0^{64} the output is "5f67abaf 5210722b".On input $1^{32}0^{32}$ the output is "bbe033c0 0bc9330e".
c) On input 064 the output is "e86d2de2 e1387ae9".On input $1^{32}0^{32}$ the output is "1792d21d b645c008".
d) On input 0^{64} the output is "7b50baab 07640c3d".On input $1^{32}0^{32}$ the output is "ac343a22 cea46d60".

Απάντηση
When the input is 0^{64}:
$R_0 = 0$, L0=0
$R_1 = F(k_1,0)$, $L_1=0$
$R_2 = F(k_2, F(k_1,0))$, $L_2 = F(k_1,0)$
When the input is $1^{32} \| 0^{32}$:
$R_0 = 0$, $L_0 = 1$
$R_1 = 1\oplus F(k_1, 0)$, $L_1 = 0$
$R_2 = F(k_2, 1\oplus F(k_1, 0))$, $L_2 = 1\oplus F(k_1, 0)$

So the first L_2 should be the inverse of the second L_2. This is the case for the pair"e86d2de2 e1387ae9" , "1792d21d b645c008" so the **correct answer is c**.

Ερώτηση5

Nonce-based CBC. Recall that in lecture 4.4 we said that if one wants to use CBC encryption with a non-random unique nonce then the nonce must first be encrypted with an independent PRP key and the result then used as the CBC IV. Let's see what goes wrong if one encrypts the nonce with the **same** PRP key as the key used for CBC encryption.

Let F:K× $\{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell}$ be a secure PRP with, say, ℓ=128. Let n be a nonce and suppose one encrypts a message m by first computing IV=F (k,n) and then using this IV in CBC encryption using F(k,·). Note that the same key k is used for computing the IV and for CBC encryption. We show that the resulting system is not nonce-based CPA secure.

The attacker begins by asking for the encryption of the two block message m=(0^{ℓ},0^{ℓ}) with nonce n=0^{ℓ}. It receives back a two block ciphertext (c_0,c_1).Observe that by definition of CBC we know that c_1=F (k,c_0). Next, the attacker asks for the encryption of the one block message m_1=$c_0 \oplus c_1$ with nonce n=c_0. It receives back a one block ciphertext c_0'.

What relation holds between c_0,c_1,c_0'? Note that this relation lets the adversary win the nonce-based CPA game with advantage 1.

a)$c_1=c'_0$
b)$c_0=c_1 \oplus c'_0$
c)$c_0=c'_0$
d)$c_1=c_0 \oplus c'_0$

Απάντηση

First, note thatc_0 = F(k, F(k, 0)) ,c_1 = F(k, F(k, F(k, 0)))and that c_1 = F(k, c_0)
In response to the attacker's request for the encryption of m_1= $(c_0)^{c_1}$withnonce c_0 the attacker receives:nonce | c'_0 so:
c_0 | F (k, F (k, c_0)^{m_1})= F (k, F(k, c_0)$^{c_0 c_1}$)=F(k, $c_1{}^{c_0 c_1}$)=F(k, c_0)=c_1

Ερώτηση6

Let m be a message consisting of ℓ AES blocks (say ℓ=100). Alice encrypts m using CBC mode and transmits the resulting ciphertext to Bob. Due to a network error, ciphertext block number ℓ/2 is corrupted during transmission. All other ciphertext blocks are transmitted and received correctly. Once Bob decrypts the received ciphertext, how many plaintext blocks will be corrupted?

a)1+ℓ/2
b)1
c)2
d)3
e) ℓ/2

Απάντηση
If block number ℓ /2 is corrupted, then that block and each block that comesafter will be corrupted, so 2 blocks of plaintext will be corrupted.So **correct answer is c**.

Ερώτηση 7
Let m be a message consisting of ℓ AES blocks (say ℓ=100). Alice encrypts m using randomized counter mode and transmits the resulting ciphertext to Bob. Due to a network error, ciphertext block number ℓ/2 is corrupted during transmission. All other ciphertext blocks are transmitted and received correctly. Once Bob decrypts the received ciphertext, how many plaintext blocks will be corrupted? :

a)3
b)1
c)1+ℓ//2
d)2
e)ℓ/2

Απάντηση
With CTR the counter value used to decrypt each block of ciphertext does notdepend on the value of any other block. Therefore with one block corrupted, only 1 block of plaintext will be corrupted.So the correct answer is **b**.

Ερώτηση8
Recall that encryption systems do not fully hide the length of transmitted messages.Leaking the length of web requests has been used to eavesdrop on encrypted HTTPS traffic to a number of web sites, such as tax preparation sites, Google searches, and healthcare sites.

Suppose an attacker intercepts a packet where he knows that the packet payload is encrypted using AES in CBC mode with a random IV. The encrypted packet payload is 128 bytes.Which of the following messages is plausibly the decryption of the payload:

a) 'To consider the resistance of an enciphering process to being broken we should assume that at same times the enemy knows everything but the key being used and to break it needs only discover the key from this information.´

b)In this letter I make some remarks on a general principle relevant to enciphering in general and my machine.'

c)'We see immediately that one needs little information to begin to break down the process.'

d)The significance of this general conjecture, assuming its truth, is easy to see. It means that it may be feasible to design ciphers that are effectively unbreakable.

Απάντηση
The correct answer is **b**.The length of the string is 107 bytes, which after padding becomes 112 bytes, and after prepending the IV becomes 128 bytes.

Ερώτηση 9

Let R:=$\{0,1\}^4$ and consider the following PRF F:$R^5 \times R \rightarrow R$ defined as follows:

$$F(k,x)=\begin{cases} t = k[0] \\ for\ i = 1\ to\ 4\ do \\ if(x[i-1] == 1) t \oplus k[i] \\ output\ t \end{cases}$$

That is, the key is k=(k[0],k[1],k[2],k[3],k[4]) in R^5 and the function at, for example, 0101 is defined as F(k,0101)=k[0]⊕k[2]⊕k[4]. For a random key k unknown to you, you learn that:

F(k,0110)=0011 and F(k, 0101)=1010 and F(k,1110)=0110.

What is the value of F(k,1101)?Note that since you are able to predict the function at a new point, this PRF is insecure.

Απάντηση

We are given to following:

$k[0]^{k[2]^{k[3]}}$=0011

$k[0]^{k[2]^{k[4]}}$=1010

$k[0]^{k[1]^{k[2]^{k[3]}}}$=0110.

We aim to find $k[0]^{k[1]^{k[2]^{k[4]}}}$=$0010^{k[3]^{k[4]}}$=$0110^{1010^{k[0]^{k[2]^{k[3]}}}}$=$0110^{1010^{0011}}$=1111

Προγραμματιστική Άσκηση 2

In this project you will implement two encryption/decryption systems, one using AES in CBC mode and another using AES in counter mode (CTR). In both cases the 16-byte encryption IV is chosen at random and is prepended to the ciphertext. For CBC encryption we use the PKCS5 padding scheme discussed in class (13:50).

While we ask that you implement both encryption and decryption, we will only test the decryption function. In the following questions you are given an AES key and a ciphertext (both are **hex encoded**) and your goal is to recover the plaintext and enter it in the input boxes provided below. For an implementation of AES you may use an existing crypto library such as **PyCrypto (Python)**, **Crypto++ (C++)**, or any other. While it is fine to use the built-in AES functions, we ask that as a learning experience you implement **CBC** and **CTR** modes yourself.

Question 1

CBC key: 140b41b22a29beb4061bda66b6747e14
CBC Ciphertext 1:
4ca00ff4c898d61e1edbf1800618fb2828a226d160dad07883d04e008a7897ee\
2e4b7465d5290d0c0e6c6822236e1daafb94ffe0c5da05d9476be028ad7c1d81

Question 2

CBC key: 140b41b22a29beb4061bda66b6747e14
CBC Ciphertext 2:
5b68629feb8606f9a6667670b75b38a5b4832d0f26e1ab7da33249de7d4afc48\
e713ac646ace36e872ad5fb8a512428a6e21364b0c374df45503473c5242a253

Question 3

CTR key: 36f18357be4dbd77f050515c73fcf9f2
CTR Ciphertext 1:

69dda8455c7dd4254bf353b773304eec0ec7702330098ce7f7520d1cbbb20fc3\
88d1b0adb5054dbd7370849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f5f51eeca32eabedd9afa9329

Question 4
CTR key: 36f18357be4dbd77f050515c73fcf9f2
CTRCiphertext 2:
770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa\
0e311bde9d4e01726d3184c34451

Απάντηση
Αρχικα θα πρέπει απο την ιστοσελίδα https://www.dlitz.net/software/pycrypto/ θα πρέπει να κατεβάσουμε την βιβλιοθήκη PyCryptokαι να την εγκαταστήσουμε στον υπολογιστή , για να μπορέσουμε να κάνουμε implementτον AES στα pythonπρογραμματα μας.Με τον κώδικα που βρίσκεται εδω θα μπορέσουμε να αποκρυπτογραφήσουμε και τα 4 ciphertextsπου μας δίνονται στην προγραμματιστική άσκηση.Σε κάθε περίπτωση έχει επιλεχτεί ένα τυχαίο μεγέθους 16 byteIVτο οποίο έχει τοποθετηθεί στα ciphertexts.

Στην **ερώτηση 1** μας ζητείται να αποκρυπτογραφήσουμε το μήνυμα με τον τρόπο λειτουργίας ECBC.Τότε αρχικά κάνουμε decodeτο κλειδί και το ciphertext ώστε να μετατραπούν αποτο δεκαεξαδικό σε συμβολοσειρές και στη συνέχεια δημιουργούμε την μεταβλητή sizeη οποία έχει μέγεθος όσο ένα AESμπλοκ,και βάζουμε μέσα στο ciphertextμας ένα IV.Το IVέχει μέγεθος όσο ένα AESμπλόκ.Κατόπιν αναθετουμε στην μεταβλητή cipherτην ποσότητα AES.new(key, AES.MODE_CBC, iv) που είνα η κρυπτογράφηση του κλειδιού kκαι του IV.Για να αποκρυπτογραφήσουμε χρησιμοποιούμε την μεταβλητή cipherγια να αποκρυπτογραφήσουμε το ciphertextδηλαδη το μήνυμα μας mgs=cipher.decrypt(ciphertext[16:]).Το plaintextπου βγαίνει είναι : Basic CBC mode encryption needs padding

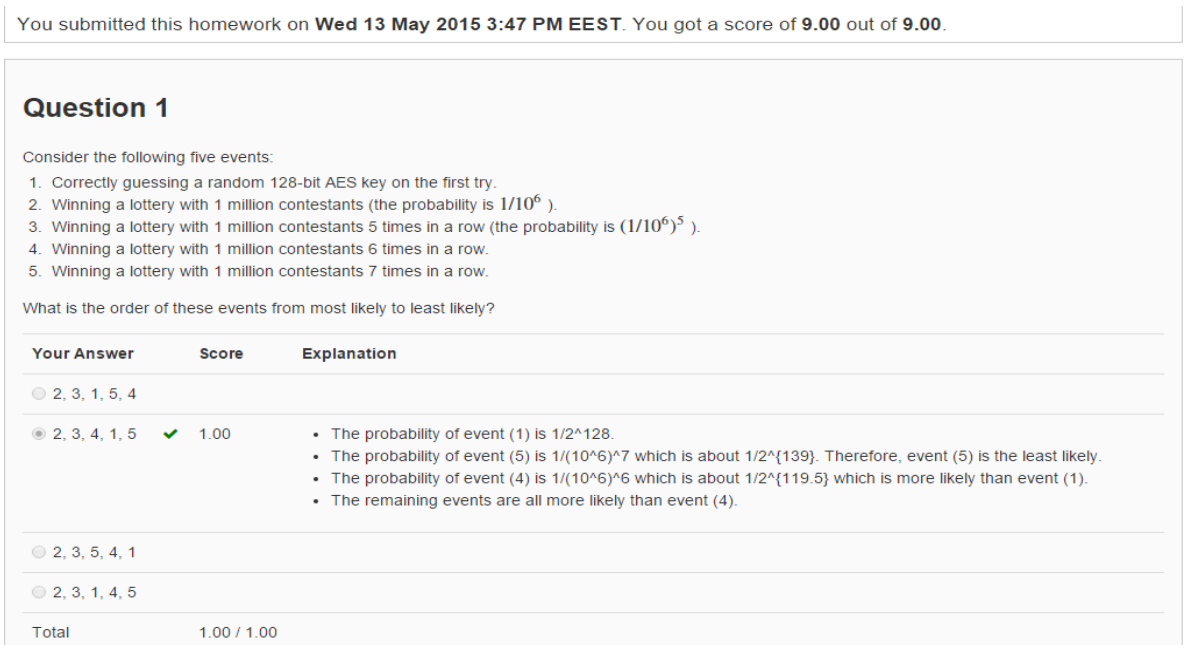
Στην **ερώτηση 2** πράττουμε ακριβώς την ίδια διαδικασίακαι το μήνυμα που παίρνουμε είναι το : Our implementation uses rand. IV.Στη συνέχεια κάνομε decodeτο κλειδί και το ciphertextγια να το πάμε απο δεξαεξαδικό σε συμβολοσειρές.Στη συνέχεια δημιουργούμε αρχικά μια συνάρτηση που την ονομάζουμεpycrypto_decryptη οποία παίρνει ώς ορίσματα το κλειδί, το iv, και μια μεταβλητή data.Η λειτουργία της είναι αφού πρώτα κρυπτογραφήσει το κλειδί και το ivμετά να αποκρυπτογραφήσει τα δεδομένα data.Η συνάρτηση αυτή θα χρησιμοποιηθεί στις ερωτήσεις 3 και 4 που θα αποκρυπτογραφήσουμε με CTR.

Στην **ερώτηση 3** όπως είπαμε χρησιμοποιήουμε μέθοδο λειτουργίας CTR.Πρώτα κατ τα γνωστά με τη συνάρτηση decodeφέρνουμε το κλειδί και το ciphertextστη μορφή συμβολοσειρών.Γνωρίζουμε ότι στη μέθοδο λειτουργίας CTRη κρυπτογράφηση και η αποκρυπτογράφηση είναι παράλληλη και σε κάθε βήμα χρησιμοποιούμε έναIV.Δημιουργούμε κάθε φορά και ενα διαφορετικό IVγια κάθε μπλόκ του μηνύματος μας.Ετσι θα χρειαστούμε συνολικά 4 IV οπου το κάθε IVθα αυξάνεται κατα 1.Το πρώτο ivθα είναι το iv+0, το δέυτερο ivθα είναι το iv+1 , το τρίτο IV θα είναι το iv+2 ενω το τέταρτο ivθα είναι το iv+3.Οσο αφορά το ciphertext, αυτό θα σπάσει σε μπλόκς.

Το πρώτο μπλόκ ct1 θα είναι τα bytes 32 εώς 64 ,το δέυτερο μπλόκ θα είναι τα bytes 64εως 96, το τρίτο μπλόκ θα είναι τα bytes 96 έως 128 καιτο τέταρτο θα περιέχει τα bytestου 128^{o} μπλόκ.Έτσι κάθε IVαντιστοιχίζεται σε διαφορετικό ciphertextμπλοκ.Τέλος θα χρησιμοποιήσουμε τη συνάρτηση που δημιουργήσαμε για

να αποκρυπτογραφήσουμε αυτά τα μπλόκς.Για παράδειγμα να αποκρυπτογραφήσουμε το πρώτο μπλόκ θα εφαρμόσουμε pycrypto_decrypt(key,v1,ct1), για το δεύτερο μπλόκ pycrypto_decrypt(key,v2,ct2) και ουτω καθ εξής για να πάρουμε το ζητούμενο μήνυμα: CTR mode lets you build a stream cipher from a block cipher.Η ερώτηση 4 λύνεται με παρόμοιο τρόπο και το ζητούμενο μήνυμα που θα πάρουμε μετά την αποκρυπτογράφηση του είναι : Always avoid the two time pad!

4.7Αναλυτικήβαθμολογία



You submitted this homework on **Wed 13 May 2015 3:47 PM EEST**. You got a score of **9.00** out of **9.00**.

Question 1

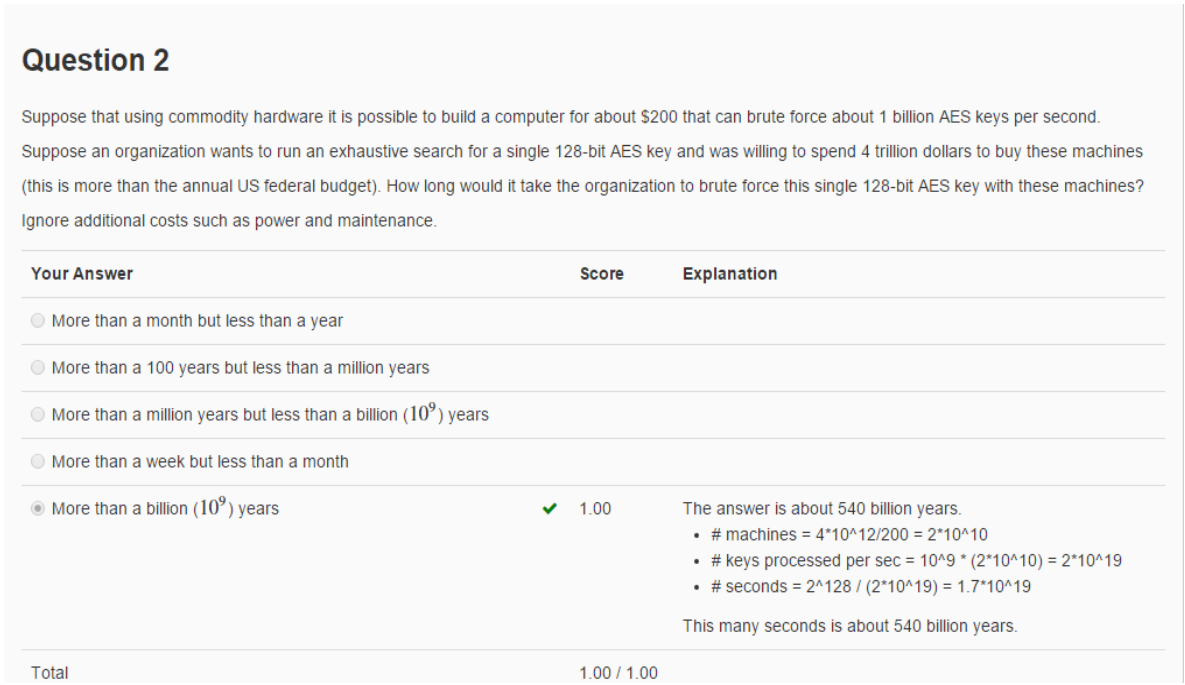
Consider the following five events:

1. Correctly guessing a random 128-bit AES key on the first try.
2. Winning a lottery with 1 million contestants (the probability is $1/10^6$).
3. Winning a lottery with 1 million contestants 5 times in a row (the probability is $(1/10^6)^5$).
4. Winning a lottery with 1 million contestants 6 times in a row.
5. Winning a lottery with 1 million contestants 7 times in a row.

What is the order of these events from most likely to least likely?

Your Answer		Score	Explanation
2, 3, 1, 5, 4			
2, 3, 4, 1, 5	✔	1.00	• The probability of event (1) is 1/2^128. • The probability of event (5) is 1/(10^6)^7 which is about 1/2^{139}. Therefore, event (5) is the least likely. • The probability of event (4) is 1/(10^6)^6 which is about 1/2^{119.5} which is more likely than event (1). • The remaining events are all more likely than event (4).
2, 3, 5, 4, 1			
2, 3, 1, 4, 5			
Total		1.00 / 1.00	

Εικόνα 52:Ερώτηση 1-Week 2



Question 2

Suppose that using commodity hardware it is possible to build a computer for about $200 that can brute force about 1 billion AES keys per second. Suppose an organization wants to run an exhaustive search for a single 128-bit AES key and was willing to spend 4 trillion dollars to buy these machines (this is more than the annual US federal budget). How long would it take the organization to brute force this single 128-bit AES key with these machines? Ignore additional costs such as power and maintenance.

Your Answer		Score	Explanation
More than a month but less than a year			
More than a 100 years but less than a million years			
More than a million years but less than a billion (10^9) years			
More than a week but less than a month			
More than a billion (10^9) years	✔	1.00	The answer is about 540 billion years. • # machines = 4*10^12/200 = 2*10^10 • # keys processed per sec = 10^9 * (2*10^10) = 2*10^19 • # seconds = 2^128 / (2*10^19) = 1.7*10^19 This many seconds is about 540 billion years.
Total		1.00 / 1.00	

Εικόνα 53:Ερώτηση 2-Week 2

Question 3

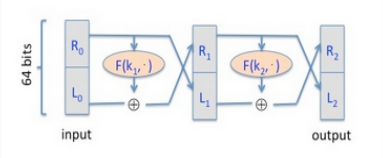
Let $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF (i.e. a PRF where the key space, input space, and output space are all $\{0,1\}^n$) and say $n = 128$. Which of the following is a secure PRF (there is more than one correct answer):

Your Answer		Score	Explanation
☑ $F'((k_1,k_2),\ x) = \begin{cases} F(k_1,x) & \text{when } x \neq 0^n \\ k_2 & \text{otherwise} \end{cases}$	✔	0.17	Correct. A distinguisher for F' gives a distinguisher for F.
☐ $F'(k,x) = F(k,x) \ \|\| \ 0$ (here $\|\|$ denotes concatenation)	✔	0.17	Not a PRF. A distinguisher will output *not random* whenever the last bit of $F(k,0^n)$ is 0.
☐ $F'(k,x) = F(k,x) \oplus F(k,\ x \oplus 1^n)$	✔	0.17	Not a PRF. A distinguisher will query at $x = 0^n$ and $x = 1^n$ and output *not random* whenever the two responses are equal. This is unlikely to happen for a truly random function.
☑ $F'(k,x) = F(k,x)[0,\ldots,n-2]$ (i.e., $F'(k,x)$ drops the last bit of $F(k,x)$)	✔	0.17	Correct. A distinguisher for F' gives a distinguisher for F.
☑ $F'((k_1,k_2),\ x) = F(k_1,x) \ \|\| \ F(k_2,x)$ (here $\|\|$ denotes concatenation)	✔	0.17	Correct. A distinguisher for F' gives a distinguisher for F.
☐ $F'(k,\ x) = \begin{cases} F(k,x) & \text{when } x \neq 0^n \\ 0^n & \text{otherwise} \end{cases}$	✔	0.17	Not a PRF. A distinguisher will query at $x = 0^n$ and output *not random* if the response is 0^n. This is unlikely to hold for a truly random function.
Total		1.00 / 1.00	

Εικόνα 54:Ερώτηση 3-Week 2

Question 4

Recall that the Luby-Rackoff theorem discussed in Lecture 3.2 states that applying a **three** round Feistel network to a secure PRF gives a secure block cipher. Let's see what goes wrong if we only use a **two** round Feistel. Let $F : K \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ be a secure PRF. Recall that a 2-round Feistel defines the following PRP $F_2 : K^2 \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$:



Here R_0 is the right 32 bits of the 64-bit input and L_0 is the left 32 bits.

One of the following lines is the output of this PRP F_2 using a random key, while the other three are the output of a truly random permutation $f : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$. All 64-bit outputs are encoded as 16 hex characters. Can you say which is the output of the PRP? Note that since you are able to distinguish the output of F_2 from random, F_2 is not a secure block cipher, which is what we wanted to show.

Hint: First argue that there is a detectable pattern in the xor of $F_2(\cdot, 0^{64})$ and $F_2(\cdot, 1^{32}0^{32})$. Then try to detect this pattern in the given outputs.

Your Answer		Score	Explanation
○ On input 0^{64} the output is "9d1a4f78 cb28d863". On input $1^{32}0^{32}$ the output is "75e5e3ea 773ec3e6".			
○ On input 0^{64} the output is "5f67abaf 5210722b". On input $1^{32}0^{32}$ the output is "bbe033c0 0bc9330e".			
◉ On input 0^{64} the output is "e86d2de2 e1387ae9". On input $1^{32}0^{32}$ the output is "1792d21d b645c008".	✔	1.00	Observe that the two round Feistel has the property that the left of $F(\cdot,0^{64}) \oplus F(\cdot,1^{32}0^{32})$ is 1^{32}. The two outputs in this answer are the only ones with this property.
○ On input 0^{64} the output is "7b50baab 07640c3d". On input $1^{32}0^{32}$ the output is "ac343a22 cea46d60".			
Total		1.00 /	

Εικόνα 55:Ερώτηση 4-Week 4

Question 5

Nonce-based CBC. Recall that in lecture 4.4 we said that if one wants to use CBC encryption with a non-random unique nonce then the nonce must first be encrypted with an **independent** PRP key and the result then used as the CBC IV. Let's see what goes wrong if one encrypts the nonce with the **same** PRP key as the key used for CBC encryption.

Let $F: K \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell}$ be a secure PRP with, say, $\ell = 128$. Let n be a nonce and suppose one encrypts a message m by first computing $IV = F(k,n)$ and then using this IV in CBC encryption using $F(k,\cdot)$. Note that the same key k is used for computing the IV and for CBC encryption. We show that the resulting system is not nonce-based CPA secure.

The attacker begins by asking for the encryption of the two block message $m = (0^{\ell}, 0^{\ell})$ with nonce $n = 0^{\ell}$. It receives back a two block ciphertext (c_0, c_1). Observe that by definition of CBC we know that $c_1 = F(k, c_0)$. Next, the attacker asks for the encryption of the one block message $m_1 = c_0 \oplus c_1$ with nonce $n = c_0$. It receives back a one block ciphertext c_0'.

What relation holds between c_0, c_1, c_0'? Note that this relation lets the adversary win the nonce-based CPA game with advantage 1.

Your Answer		Score	Explanation
◉ $c_1 = c_0'$	✔	1.00	This follows from the definition of CBC with an encrypted nonce as defined in the question.
○ $c_0 = c_1 \oplus c_0'$			
○ $c_0 = c_0'$			
○ $c_1 = c_0 \oplus c_0'$			
Total		1.00 / 1.00	

Εικόνα 56:Ερώτηση 5-Week 5

Question 6

Let m be a message consisting of ℓ AES blocks (say $\ell = 100$). Alice encrypts m using CBC mode and transmits the resulting ciphertext to Bob. Due to a network error, ciphertext block number $\ell/2$ is corrupted during transmission. All other ciphertext blocks are transmitted and received correctly. Once Bob decrypts the received ciphertext, how many plaintext blocks will be corrupted?

Your Answer		Score	Explanation
○ $1 + \ell/2$			
○ 1			
◉ 2	✔	1.00	Take a look at the CBC decryption circuit. Each ciphertext blocks affects only the current plaintext block and the next.
○ 3			
○ $\ell/2$			
Total		1.00 / 1.00	

Εικόνα 57:Ερώτηση 6-Week 2

Question 7

Let m be a message consisting of ℓ AES blocks (say $\ell = 100$). Alice encrypts m using randomized counter mode and transmits the resulting ciphertext to Bob. Due to a network error, ciphertext block number $\ell/2$ is corrupted during transmission. All other ciphertext blocks are transmitted and received correctly. Once Bob decrypts the received ciphertext, how many plaintext blocks will be corrupted?

Your Answer		Score	Explanation
○ 3			
◉ 1	✔	1.00	Take a look at the counter mode decryption circuit. Each ciphertext block affects only the current plaintext block.
○ $1 + \ell/2$			
○ 2			
○ $\ell/2$			
Total		1.00 / 1.00	

Εικόνα 58:Ερώτηση 7-Week 2

Question 8

Recall that encryption systems do not fully hide the **length** of transmitted messages. Leaking the length of web requests has been used to eavesdrop on encrypted HTTPS traffic to a number of web sites, such as tax preparation sites, Google searches, and healthcare sites. Suppose an attacker intercepts a packet where he knows that the packet payload is encrypted using AES in CBC mode with a random IV. The encrypted packet payload is 128 bytes. Which of the following messages is plausibly the decryption of the payload:

Your Answer		Score	Explanation
'To consider the resistance of an enciphering process to being broken we should assume that at same times the enemy knows everything but the key being used and to break it needs only discover the key from this information.'			
'In this letter I make some remarks on a general principle relevant to enciphering in general and my machine.'	✔	1.00	The length of the string is 107 bytes, which after padding becomes 112 bytes, and after prepending the IV becomes 128 bytes.
'We see immediately that one needs little information to begin to break down the process.'			
'The significance of this general conjecture, assuming its truth, is easy to see. It means that it may be feasible to design ciphers that are effectively unbreakable.'			
Total		1.00 / 1.00	

Εικόνα 59:Ερώτηση 8-Week 2

Question 9

Let $R := \{0,1\}^4$ and consider the following PRF $F : R^5 \times R \to R$ defined as follows:

$$F(k,x) := \begin{cases} t = k[0] \\ \text{for i=1 to 4 do} \\ \quad \text{if } (x[i-1] == 1) \quad t = t \oplus k[i] \\ \text{output } t \end{cases}$$

That is, the key is $k = (k[0], k[1], k[2], k[3], k[4])$ in R^5 and the function at, for example, 0101 is defined as $F(k,0101) = k[0] \oplus k[2] \oplus k[4]$.

For a random key k unknown to you, you learn that

$F(k,0110) = 0011$ and $F(k,0101) = 1010$ and $F(k,1110) = 0110$.

What is the value of $F(k,1101)$? Note that since you are able to predict the function at a new point, this PRF is insecure.

You entered:

1111

Your Answer		Score	Explanation
1111	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 60:Ερώτηση 9-Week 2

You submitted this homework on **Wed 13 May 2015 9:25 PM EEST**. You got a score of **4.00** out of **4.00**.

Question 1

In this project you will implement two encryption/decryption systems, one using AES in CBC mode and another using AES in counter mode (CTR). In both cases the 16-byte encryption IV is chosen at random and is *prepended* to the ciphertext. For CBC encryption we use the PKCS5 padding scheme discussed in class (13:50).

While we ask that you implement both encryption and decryption, we will only test the decryption function. In the following questions you are given an AES key and a ciphertext (both are hex encoded) and your goal is to recover the plaintext and enter it in the input boxes provided below.

For an implementation of AES you may use an existing crypto library such as PyCrypto (Python), Crypto++ (C++), or any other. While it is fine to use the built-in AES functions, we ask that as a learning experience you implement CBC and CTR modes yourself.

Question 1

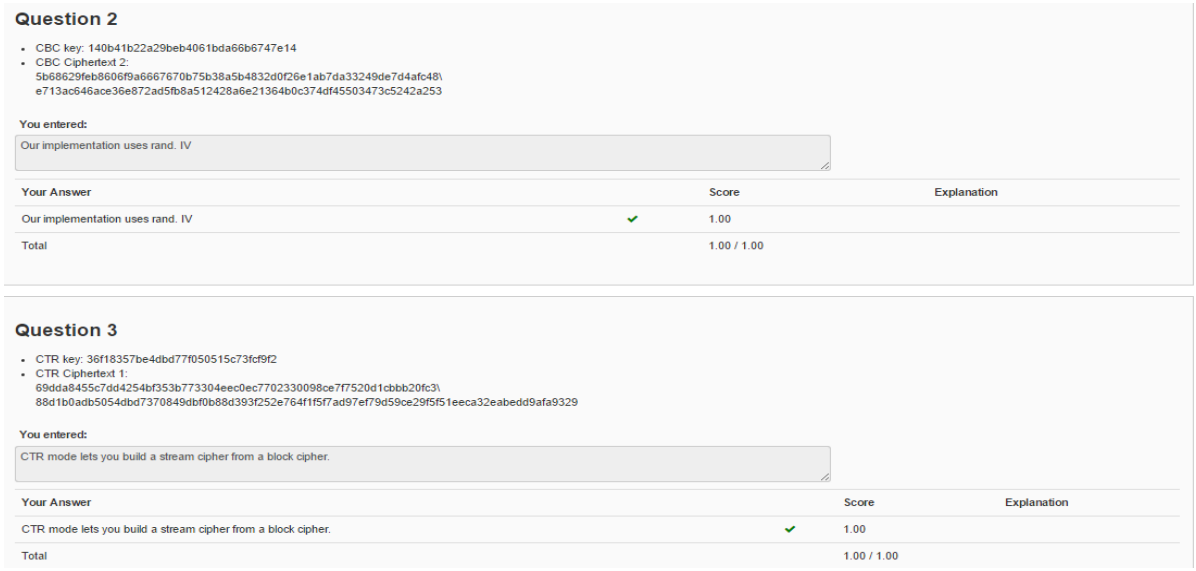
- CBC key: 140b41b22a29beb4061bda66b6747e14
- CBC Ciphertext 1:
  4ca00ff4c898d61e1edbf1800618fb2828a226d160dad07883d04e008a7897ee\
  2e4b7465d5290d0c0e6c6822236e1daafb94ffe0c5da05d9476be028ad7c1d81

You entered:

Basic CBC mode encryption needs padding.

Your Answer		Score	Explanation
Basic CBC mode encryption needs padding.	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 61:Προγραμματιστική άσκηση 2-Week 2(ερώτηση 1)



Question 2

- CBC key: 140b41b22a29beb4061bda66b6747e14
- CBC Ciphertext 2:
  5b68629feb8606f9a6667670b75b38a5b4832d0f26e1ab7da33249de7d4afc48\
  e713ac646ace36e872ad5fb8a512428a6e21364b0c374df45503473c5242a253

You entered:

Our implementation uses rand. IV

Your Answer		Score	Explanation
Our implementation uses rand. IV	✔	1.00	
Total		1.00 / 1.00	

Question 3

- CTR key: 36f18357be4dbd77f050515c73fcf9f2
- CTR Ciphertext 1:
  69dda8455c7dd4254bf353b773304eec0ec7702330098ce7f7520d1cbbb20fc3\
  88d1b0adb5054dbd7370849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f5f51eeca32eabedd9afa9329

You entered:

CTR mode lets you build a stream cipher from a block cipher.

Your Answer		Score	Explanation
CTR mode lets you build a stream cipher from a block cipher.	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 62:Προγραμματιστική άσκηση 2-Week 2(ερώτηση 2 και 3)

Question 4

- CTR key: 36f18357be4dbd77f050515c73fcf9f2
- CTR Ciphertext 2:
  770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa\
  0e311bde9d4e01726d3184c34451

You entered:

Always avoid the two time pad!

Your Answer		Score	Explanation
Always avoid the two time pad!	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 63:Προγραμματιστική άσκηση 2-Week2(ερώτηση 4)

Πίνακας 2:ΒαθμολογίεςWeek 2

	$1^{η}$ προσπάθεια	$2^{η}$ προσπάθεια	$3^{η}$ προσπάθεια	$4^{η}$ προσπάθεια
Ερώτησεις	9.00/9.00	-	-	-
Άσκηση	4.00/4.00	-	-	-

ΚΕΦΑΛΑΙΟ5
ΑΚΕΡΑΙΟΤΗΤΑ ΜΗΝΥΜΑΤΟΣ

5.1 Κώδικες πιστοποίησης ακεραιότητας μηνύματος(MAC)

Σε αυτό το κεφάλαιο ο σκοπός μας είναι να παρέχουμε ακεραιότητα μηνύματος αλλα όχι εμπιστευτικότητα.Το πώς γίνεται να μας ενδιαφέρει η ακεραιότητα και όχι η εμπιστευτικότητα θα το δείξουμε με παραδείγματα.Παραδείγματος χάρη σκεφτείτε το σύστημα αρχείων που βρίσκεται σε ένα σκληρό δίσκο.Εαν χρησιμοποιείτε Windows το σύστημα αυτό δεν έχει εμπιστευτικότηκα καθώς είναι γνωστό στο κοινό που θέλει να το χρησιμοποιήσει αλλα εσείς με κάποιο τρόπο πρέπει να είστε σίγουροι ότι τα αρχεία δεν θα τροποποιηθούν απο κάποιο ιό ή κακόβουλο λογισμικό.

Ένα άλλο παράδειγμα είναι οι διαφημίσεις τύπου bannerπου βρίσκονται σε διάφορες ιστοσελίδες.Ο πάροχος των διαφημίσεων δεν νοιάζεται άν κάποιος αντιγράψει αυτές τις διαφημίσεις για να τις δείξει στο κοινό.Αρα δέν υπάρχει θέμα εμπιστευτικότητας αλλα τον ενδιαφέρει η «τροποποηση» αυτών των διαφημίσεων με οποιόνδήποτε τρόπο.Αυτό που θα κάνει ο πάροχος είναι να μην επιτρέπει στους ανθρώπους να αλλάζουν τις διαφημίσεις.

Ένας τρόπος να παρέχουμε **ακεραιότητα** μηνύματος είναι να χρησιμοποιήσουμε ένα βασικό μηχανισμό που ονομάζεται**MAC**[45](message authentication code).Έστω ότι έχουμε την Alicekαι τον Bob που εχούν ένα κοινό κλειδί kτο οποίο δέν γνωρίζει ο επιτιθέμενος και υπάρχει επίσης και ένα μήνυμαm που θέλει η Alicevα στείλει στον Bob.Ο επιτιθέμενος δεν πρέπει να μπορεί να τροποποιήσει το μήνυμα κατα την μεταφορά του απο την Aliceστον Bob.

Για να γίνει αυτό ηAlice θα χρησιμοποιήσει τον **αλγόριθμο υπογραφής**(MACsigningalgorithm) για να παράγει μια ετικέτα που θα την τοποθετήσει μέσα στο μήνυμα.Μετά θα στείλει το μήνυμα στον Bob.Ο παραλήπτης θα χρησιμοποιήσει έναν αλγόριθμο που παίρνει ώς είσοδο ένα κλειδί k και το μήνυμα προς αποστολή για να υπολογίσει την ετικέτα.Κατόπιν θα ελέγξει για το εαν η ετικέτα που υπολόγισε είναι ίδια με την ετικέτα που έστειλε η Alice.Αν η ετικέτες είναι ίδιες το μήνυμα τότε δεν έχει πειραχθεί και έχει φτάσει ακέραιο στον Bob.Αν όμως οι ετικέτες δεν είναι ίδιες τότε το μήνυμα έχει τροποποιηθεί.

Όπως είδαμε η ακεραιότητα απαιτεί ένα κοινο κλειδί μεταξύ του Bobκαι της Alice.Τώρα θα δείξουμε με τη χρήση ενός αλγορίθμου ότι η χρήση κλειδιού είναι απαραίτητη.Έστω ο**CRC**(cyclicredundancycheck) που είναι ένας κλασσικός αλγόριθμος υπολογισμού του checksumo οποίος δημιουργήθηκε με σκοπό να ανιχνεύει τα διάφορα λάθη μέσα στο μήνυμα.Φανταστείτε ότι η Alice αντί για ένα κλειδί το οποίο θα μας δημιουργήσει μια ετικέτα απο την πλευρά της, να χρησιμοποιήσει έναν αλγόριθμο CRCο οποίος θα μας δημιουργήσει την ετικέτα χωρίς την ανάγκη του κλειδιού.

Στη συνέχεια βάζει αυτήν την ετικέτα στο μήνυμα και το στέλνει στον Bob.Ο Bobαπο την πλευρά του θα υπολογίσει το **tag=CRC(m)** και άν ο αλγόριθμος του βγάλει

---

[45] Ο μηχανισμός MAC, https://en.wikipedia.org/wiki/Message_authentication_code

«yes»τότε το μήνυμα ειναι ακέραιο ενώ αν του βγάλει «no» τότε το μήνυμα δέν έφτασε ακέραιο.

Το πρόβλημα όμως έιναι ότι το παραπάνω σχήμα δέν ειναι ασφαλές και είναι πολύ έυκολο ένας επιτιθέμενος να το σπάσειΑυτό που μπορεί να κάνει ο επιτιθέμενος είναι να ακυρώσει την ετικέτα του μηνύματος και μετά να παράγει το δικό του μήνυμα.Στη συνέχεια θα υπολογίσει το CRCτου μηνύματος του και μετά θα στείλει την αλληλουχία αυτήν στον Bob.Ο Bobθα τρέξει τον αλγόριθμο επαλήθευσης και θα δεί ότι το μήνυμα έιναι σωστό!Επειδή στο CRCδέν υπάρχει κλειδί ουσιαστικά δέν υπάρχει καμία διαφορά μεταξύ επιτιθέμενου και της Alice.

<u>Ασφάλεια στο MAC</u>

Θα προσπαθήσουμε να δείξουμε τι σημαίνει ένα σύστημα MACνα είναι ασφαλέςμε ένα παιχνίδι.Θα ορίσουμε την δύναμη την οποία έχει ο επιτιθέμενος και ποίος ο σκοπός του.Ο επιτιθέμενος έχει την δύναμη να επιλέξει σε ποιο μήνυμα θα επιτεθεί.Εφαρμόζει δηλαδή την επίθεση **επιλεγμένου μηνύματος**(chosenmessageattack).Με άλλα λόγια ο επιτιθέμενος μπορεί να δώσει στην Aliceαυθαίρετα μηνύματα της επιλογής του $m_1 \ldots m_q$και η Aliceθα υπολογίσει τις ετικέτες αυτών για τον επιτιθέμενο.Γιατί όμως η Alicena τιςυπολογίσει;

Για πάραδειγμα μπορεί ο επιτιθέμενος να στείλει στην Alice ένα email.Η Aliceθα αποθηκεύσει το μήνυμα αυτό, στον δίσκο της αλλα πρώτα θα υπολογίσει την ετικέτα του μηνύματος αυτού με σκοπό να μήν μπορέσει κανείς να το αλλάξει.Αρα τελικά θα αποθηκεύσει όχι μόνο το μήνυμα αλλα και την ετικέτα στον δίσκο της.

Λίγο αργότερα μπορεί ο επιτιθέμενος να κλέψει τον δίσκο της Alice και πλέον να διαθέτει και το μήνυμα αλλα και την ετικέτα αυτού, υπολογισμένη απο την Alice.Ο σκοπός του είναι να κάνει**υπαρξιακή πλαστογραφία**[46](existentialforgery).Με άλλα λόγια ο επιτιθέμενος προσπαθεί να παράγει μερικές νέες ετικέτες.Δηλαδή κάποιο ζέυγος μηνύματος-ετικέτας το οποίο είναι διαφορετικό απο αυτά που του δώθηκαν κατα την διάρκεια της επίθεσης επιλεγμένου μηνύματος:

$$\mathbf{(m,t) \notin \{(m_1,t_1),\ldots(m_q,m_t)\}}$$

Εάν μπορεί να το κάνει αυτό τότε το σύστημα είναι **μή ασφαλές**.Δηλαδή η επίθεση υπαρξιακής πλαστογραφίας δείχνει ότι, ακόμη και εάν ο επιτιθέμενος διαθέτει ένα ζέυγος μηνύματος και ετικέτας**,πρέπει**να μην μπορεί να παράγει νέα ετικέτα **ακόμη** και αν το μήνυμα περιγράφει ασυναρτησίες(gibberish).

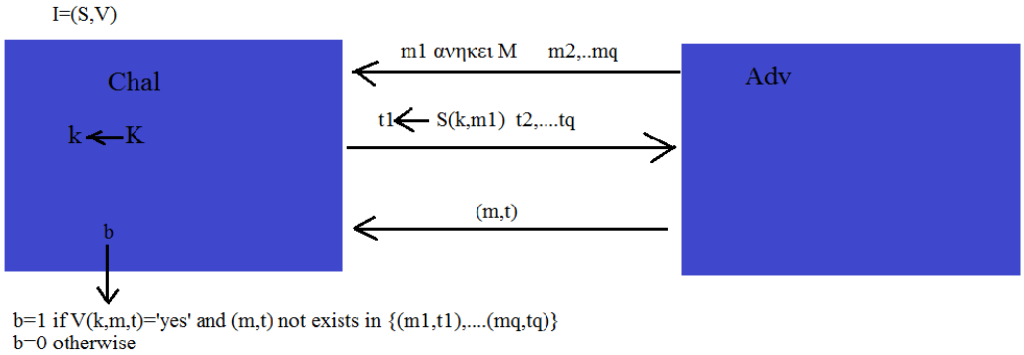
Ένα ερώτημα εδώ θα ήταν γιατι να ένοιαζε τον επιτιθέμενο το έαν μπορεί να υπολογίσει την ετικέτα ενός μηνύματος που πιθανόν να περιέγραφε ασυναρτησίες;Δέν έχει κάποια αξία για τον επιτθέμενο, απλά θέλουμε να δημιουργήσουμε MAC'Soι οποίοι είναι ασφαλείς σε κάθε περίπτωση.

Υπάρχουν περιπτώσεις για παράδειγμα όπου θέλουμε να υπολογίσουμε την ετικέτα για ένα τυχαίο μυστικό κλειδί.Σε αυτήν την περίπτωση ο επιτιθέμενος είναι ικανός να υπολογίσει την ετικέτα για ένα εντελός τυχαίο μήνυμα και ίσως να είναι ικανός να τονεξαπατήσει στο να χρησιμοποιήσει ένα λανθασμένο μυστικό κλειδί.Επομένως

[46] Υπαρξιακή πλαστογραφία, https://en.wikipedia.org/wiki/Digital_signature_forgery`

θέλουμε να είμαστε σίγουροι ότι ο επιτιθέμενος δεν θα μπορεί να παράγει κάποια έγκυρη ετικέτα για οποιδήποτε μήνυμα, είτε αυτό είναι σημαντικό είτε περιγράφει ασυναρτησίες.Επίσης να τονίσουμε ότι δεν θα πρέπει ο επιτιθέμενος ενω θα έχει ένα ζευγος μηνύματος-ετικέτας να μπορεί να παράγει και άλλη ετικέτα διαφορετική απο αυτήν που ήδη έχει για αυτό το μήνυμα.

Εστω οτι έχουμε ένα πειραμα με 2 αλγορίθμους τους S, V και έναν αντίπαλο(**Εικόνα65**).Ο διεκδικητήςχρησιμοποιεί ένα τυχαίο κλειδί για τον MACκαι ο επιτιθέμενος πράττει **επίθεση επιλεγμένου μηνύματος.**Επομένως υποβάλει το m_1στον διεκδικητή και λαμβάνει την ετικέτα του μηνύματος m_1.Μετά υποβάλει το μήνυμα m_2 στον διεκδικητή και λαμβάνει την ετικέτα m_2.Ουσιαστικά υποβάλει qστον αριθμό μηνύματα και λαμβάνει qστον αριθμό ετικέτες μηνυμάτων.



Εικόνα 64:Ασφάλεια στο MAC

Στη συνέχεια ο επιτιθέμενος προσπαθεί να κάνει επίθεση **υπαρξικης πλαστογραφίας.**Εξάγει ένα έγκυρο ζεύγος (m,t) και περιμένει την απόκριση του διεκδικητή.Θα λέμε ότι ο επιτιθέμενος κέρδισε το παιχνίδι(**b=1**) εαν το ζεύγος μηνύματος-ετικέτας εξάγει έ**γκυρη** ετικέτα και κατα δεύτερο λόγο θεωρείται**φρέσκο ζεύγος**.Δηλαδή δέν είναι είναι ζεύγος που του έχει δωθεί προτύτερα.

Σε κάθε άλλη περίπτωση θα λέμε ότι ο επιτθέμενος έχασε το παιχνίδι(**b=0**).Ως συνήθως θα ορίσουμε το πλεονέκτημα του επιτιθέμενου σαν την πιθανότητα ο διεκδικητής να βγάλει 1 ως έξοδο.Με άλλα λόγια κανείς «αποδοτικός» επιτιθέμενος δέν μπορεί να κερδίσει το παιχνίδι με μή αμελητέα πιθανότητα:

Τώρα θα δείξουμε ότι οποιαδήποτε PRFμας δίνει απευθείας ασφαλή MAC.Ας δούμε πώς θα το καταφέρουμε.Υποθέστε ότι έχουμε μια συνάτηση ψευδοτυχίας η οποία παίρνει ώς είσοδο ενα X και μας βγάζει έξοδο στο Y.Ας ορίσουμε τώρα ένα MAC.Ο τρόπος με τον οποίο υπογράφουμε ένα μήνυμα M είναι πολύ απλά χρησιμοποιώντας την συνάρτηση PRF:

$$S(k, m) := F(k, m)$$

Οπότε η ετικέτα μας ειναι η τιμή της συνάρτησης PRFστο σημείο M και ο τρόπος με τον οποίο επαληθεύουμε την ετικέτα είναι απλά το να υπολογίσουμε ξανά την τιμή της συνάρτησης στο σημείο M και να δούμε έαν η ετικέτα που υπολογίσαμε είναι ίδια με αυτή που μας δώσανε.

<u>Θεώρημα ασφάλειας MAC 5.1.1</u>

Εάν F:K×X→Y είναι μια ασφαλής PRF και το 1/|Y| είναι μη αμελητέο τότε η I_F είναι ενας ασφαλής MAC.Για κάθε MAC αντίπαλο A που επιτίθεται στην I_F, υπάρχει ένας άλλος PRF αντίπαλος B που επιτίθεται στην F και ισχύει ότι :

$Adv_{MAC}[A,I_F] \leq Adv_{PRF}[B,F]+1/|Y| \Rightarrow I_F$είναιασφαλήςέαν |Y| είναι πολύ μεγάλο συνήθως $|Y|=2^{80}$

Λήμμα 5.1.1

Υποθέστε F:K×X→$\{0,1\}^n$ είναι μια ασφαλής PRF.Τότε ισχύει τότε ότι:

$F_t(k,m)=F(k,m)[1,..t]$ για $1 \leq t \leq n$

Δηλαδή έαν έχετε nbitPRF και βγάλετε ώς έξοδο μόνο μερικά bits απο το κλειδί τότε το αποτέλεσμα θα είναι πάλι μια ασφαλής PRF.

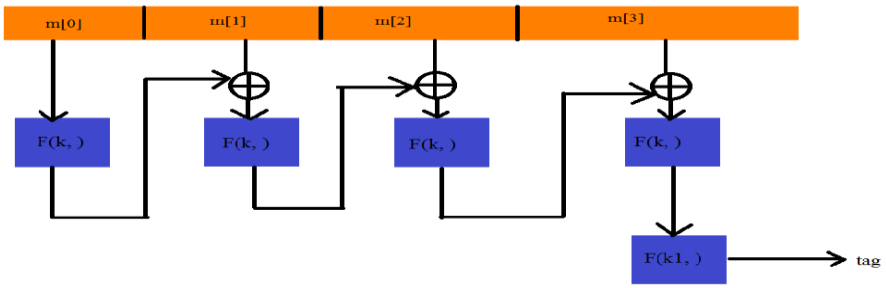
5.2.1CBC-MAC[47] και NMAC

Στην προηγούμενη ενότητα αναφέραμε ότι έαν μας δώσουν μια ασφαλή PRFτότε μπορεί να χρησιμοποιηθεί για να κατασκευάσουμε έναν ασφαλή MACαπλα και μόνο ορίζοντας S(k,m)=F(k,m).Η μόνη απαίτηση είναι η έξοδος της PRFνα είναι πολύ μεγάλη.Το ερώτημα όμως είναι μπορούμε απο μια PRFη οποία δημιουργήθηκε για μικρού μήκους μηνύματα(παραδείγματος χάρη AES) νακατασκευάσουμε μια PRFγια μηνύματα μεγάλου μεγέθους;

Η πρώτη κατασκευή που θα φτιάξουμε ονομάζεται **CBC-MAC**(ECBC).Το ECBCχρησιμοποιεί μιαPRFη οποία παίρνει μηνύματα στο διάστημα $X\{0,1\}^n$και εξάγει τα μηνύματα στο ίδιο διάστημα X.Αυτο που θα κατασκευάσουμε είναι μια PRFη οποία παίρνει ώς είσοδους ζευγάρια κλειδιών και πολύ μεγάλα μηνύματα και μας εξάγει ετικέτες στο X.Τα μηνύματα που δέχεται ώς είσοδο πρέπει να είναι μικρότερα ή ισα απο το L,οπου Lείναι το πληθος των μπλόκς του μηνύματος.

Τώρα θα δείξουμε την λειτουργία του ECBC(**Εικόνα 66**).Παίρνουμε το μήνυμα μας και το σπάμε σε μπλόκς.Το κάθε μπλόκ πρέπει να έχει μήκος όσο το μήκος του μπλόκ της PRF.Κρυπτογραφούμε το πρώτο μπλόκ και το αποτέλεσμα της κρυπτογράφησης αυτής το περνάμε μέσα απο μια πράξη XORκαι το αποτέλεσμα αυτού, περνά και πάλι από μια πράξη XORμεταξύ του δεύτερου μηνύματος.

Το αποτέλεσμα κρυπτογραφείται και η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο τέλος όπου η κρυπτογράφηση του τελευταίου μπλόκ του μηνύματος περνάει πάλι απο μία άλλη κρυπτογράφηση όχι ομως με το κλειδί καλλα με το κλειδί k_1 και μας δίνει την ζητούμενη ετικέτα.Το τελευταίο αυτό βήμα ονομάζεται **rawCBC**.

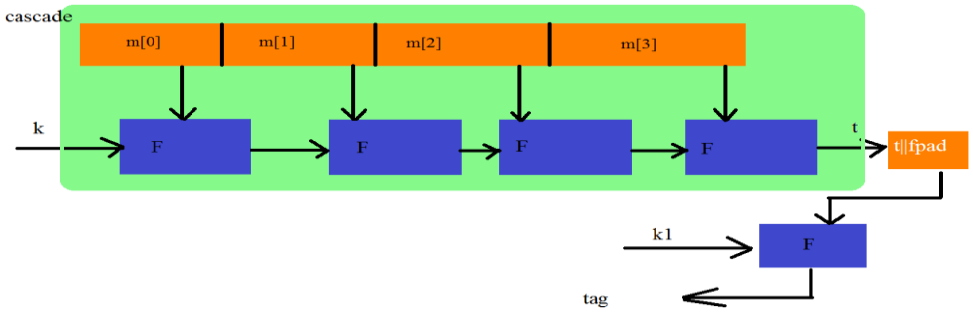


Εικόνα 65: Η λειτουργία ECBC

[47] Η τεχνική CBC-MAC, https://en.wikipedia.org/wiki/CBC-MAC

Μια άλλη κατασκευή όπου **απο μια μικρήPRF περνάμε σεμια μεγάλη PRF**ονομάζεται **NMAC**.Ας δούμε πώς λειτουργεί.Ξεκινάμε με μια PRFπου παίρνει εισόδους στο Χαλλά δέ μας βγάζει έξοδο στο Χ.Παίρνουμε ένα μήνυμα και το διασπάμε σε μπλόκς όπου κάθε μπλόκ είναι τόσο μεγάλο όσο το μπλόκ της μικρής PRF(**Εικόνα 67**)

Έπειτα χρησιμοποιούμε ένα κλειδί σαν είσοδο στην συνάρτηση Fκαι το πρώτο μπλόκ .Το αποτέλεσμα της συνάρτησης μας δίνει το κλειδί για το επόμενο μπλοκ του NMAC.Το στοιχείο που βγαίνει ώς έξοδος ανήκει στον κλειδόχωρο K.Στη συνέχεια επαναλαμβάνουμε αυτη την διαδικασία μέχρι να φτάσουμε στην τελική έξοδο.Μέχρι έδω, η διαδικασία έγινε με αυτό που ονομάζουμε **cascadefunction**.



Εικόνα 66:Η λειτουργία NMAC

Για να κάνουμε ασφαλή τον MAC, θα πρέπει να χαρτογραφήσουμε το στοιχείο tτο οποίο ανήκει στο X, στο πεδίο Χ.Αυτό που κάνουμε απλά είναι να μια απλή προσάρτησητου στοιχείου t σε ενα«fixedpad»και το αποτέλεσμα αυτής της διαδικασίας είναι το στοιχείο t, που πλέον να ανήκει στο Χ.Τέλος πράττουμε πάλι κρυπτογραφήση με το κλειδί k_1αυτή τη φορά και το αποτέλεσμα είναι η ετικέτα του μηνύματος.

Γιατί όμως είναι **απαραίτητη** η χρήση της cascadeσυνάρτησης;Έστω ότι δέν υπήρχε το βήμα της προσάρτησης του «fixedpad».Αυτό που θα μπορούσε να κάνει ο επιτιθέμενος είναι να προθέσει ακόμη ένα μπλόκ στο τέλος του μηνύματος(έστω w)και μετά θα λάμβανε την έξοδο της «cascade»συνάρτησης η οποία θα είχε τιμή t.Μετά θα μπορούσε να εφαρμόσει τη συνάρτηση Fστο στοιχείο tγια ακόμη μια φορά και αυτό που θα λάβει είναι ένα t΄ το οποίο είναι το αποτέλεσμα της της cascadeσυνάρτησης του μηνύματος σε αλληλουχία με το w, το οποίο μπορεί να χρησιμοποιηθεί σε **επίθεση υπαρξιακής πλαστογραφίας**.

Ένα άλλο ερώτημα είναι γιατί έχουμε ένα επιπλέον βήμα κρυπογράφησης στο ECBC.Ας ορίσουμε ένα χάρτη(map) ο οποίος χρησιμοποιεί «rawCBC».Θα δείξουμε ότι χωρίς το τελευταίο βήμα κρυπτογράφησης ο ECBείναι μή ασφαλής.Εστω ότι στον επιτιθέμενο δόθηκε μιαrawCBC τιμή για ένα συγκεκριμενο μήνυμα και ο επιτιθέμενος θέλει να επεκτείνει και να υπολογίσει τον MACσε ένα μήνυμα Μτο οποίο βρίσκεται σε αλληλουχία με ένα άλλο μπλοκ W.

Ένας κοινός επιτιθέμενος θα προσπαθήσει να κάνει πράξη XORμεταξύ της «rawCBC»τιμής και του μπλόκ W, όμως δεν γνωρίζει πως να λύσει την συνάρτηση $F(,\cdot)$ σε αυτό το σημείο γιατί δέν γνωρίζει το κλειδί K άρα δεν μπορεί να βρεί την έξοδο της συνάρτησης.

Μπορεί όμως να λύσει αυτήν την συνάρτηση πράττωντας «chosenmessageattack».Ο επιτιθεμενος ξεκινά αναζητώντας την ετικέτα σε ένα συγκεκριμένο μήνυμα m το οποίο έχει μήκος όσο ένα μπλόκ.Για να λάβει την ετικέτα απλα υπολογίζει το t=F(k,m).Τωρα μπορούμε να ορίσουμε ένα μήνυμα m΄ το οπόιο περίεχει 2 μπλόκς.Το μήνυμα mκαι το t⊕m.Θεωρώ ότι αυτή η ετικέτα που υπολόγισα είναι σωστή και για τα 2 αυτά μηνύματα:

$$\mathbf{RawCBC(k,(m,t\oplus m))=F(k,F(k,m)\oplus(t\oplus \boldsymbol{m}))=F(k,t\oplus(\boldsymbol{t}\oplus \boldsymbol{m}))=\boldsymbol{t}}$$

Υποθέστε ότι εφαρμόζουμε την rawCBC συνάρτηση στο m.Η διαδικασία συνεχίζεται κρυπτογραφώντας το μήνυμα χρησιμοποιώντας τη συνάρτηση F.Το αποτέλεσμα της κρυπτογράφησης το κάνουμε πράξη XORμεταξύ του δευτέρου μηνύματος (το οποίο είναι το $t\oplus m$).Απο τον ορίσμο μας F(k,m)=τάρα $F(k,t\oplus(t\oplus m))=t$.

<u>Θεωρήματα ασφάλειας ECBC-NMAC 5.2.2</u>
Για κάθε PRF αντίπαλο A που υποβάλει q στο σύνολο ερωτήματα,που επιτίθεται στο ECBC ή NMAC υπάρχει αντίπαλος B τέτοιος ώστε να ισχύει ότι:
$Adv_{PRF}[A,E_{ECBC}]\leq Adv_{PRF}[B,F]+2q^2/|X|$
$Adv_{PRF}[A,E_{NMAC}]\leq q\cdot L\cdot Adv_{PRF}[B,F]+q^2/2\cdot |K|$
ECBC είναιασφαλέςέαν $q<<|X|^{1/2}$ενώ NMAC είναιασφαλέςέαν $q<<|K|^{1/2}$

5.1.2Επικάλυψη MAC και CMAC

Στην προηγούμενη ενότητα αναφερθήκαμε στα CBC-MACκαι το NMAC στην οποία όμως θεωρούσαμε οτι το μήκος του μηνύματος που θέλαμε να εφαρμόσουμε MACήταν πολλαπλάσιο του μήκους του μπλόκ.Σε αυτήν την ενότητα θα δούμε τι πρέπει να κάνουμε όταν το μήκος του μηνύματος **δέν είναι πολλαπλάσιο** του μπλόκ.Όπως έχουμε πει και στην προηγούμενη ενότητα το ECBC-MACχρησιμοποιεί μια PRF για να υπολογίσει την CBCσυνάρτηση αλλα υποθέταμε ότι το μήνυμα έσπαγε σε έναν ακέραιο αριθμό απο μπλόκς ενός αλγορίθμου τμήματος.

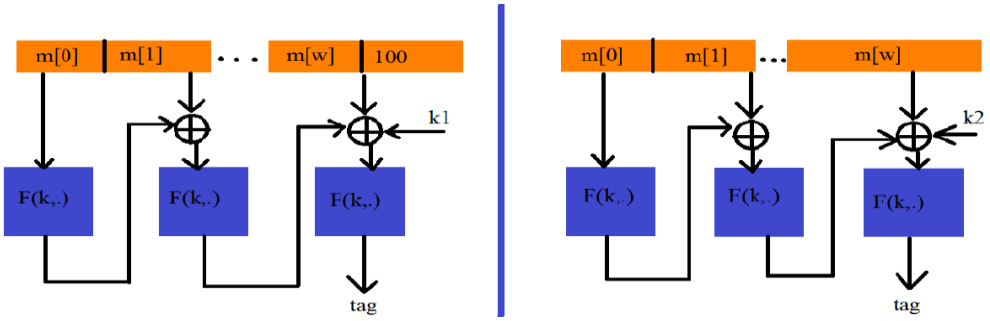
Το ερώτημα είναι τι κάνουμε όταν το μήκος του μηνύματος δέν είναι πολλαπλάσσιο.Έστω ότι έχουμε ένα μήνυμα του οποίου το τελευταίο μπλόκ είναι ουσιαστικά μικρότερο απο τα προηγούμενα μπλόκ.Αυτό που μπορούμε να κάνουμε είναι να παίρνουμε το τελευταίο μας μπλόκ και να το επικαλύπτουμε(padding) με μηδενικά ωσότου φτάσει το μήκος των άλλων μπλόκς.Η διαδικασία όμως αυτή δέν είναι ασφαλής.

Το πρόβλημα είναι ότι **pad(m)=pad(m||0)**.Εάν βάλουμε το pad(m) και το pad(m||0) σε ένα ECBCθα παρατηρήσουμε ότι έχουν ακριβώς την ίδια ετικέτα με αποτέλεσμα ένας επιτιθέμενος να εφαρμόσει πολύ έυκολα μια επίθεση υπαρξιακης πλαστογραφίας.Για τον λόγο αυτό η επικάλυψη **πρέπει να είναι αντιστρέψιμη**δηλαδή η συνάρτηση επικάλυψης να είναι 1 προς 1:

$$\mathbf{m_0\neq m_1\Rightarrow pad(m_0)\neq pad(m_1)}$$

Θα περιγραψούμε τον **αλγόριθμο αντιστροφής**.Ο αλγόριθμος αυτός ελέγχει το μήνυμα απο δεξιά προς αριστερά μέχρι να βρεί το πρώτο 1.Μόλις το βρεί θα αφαιρέσει όλα τα bitsπου βρίσκονται μετά το έναμαζί με το 1.Έτσι θα μπορέσουμε να λάβουμε το μήνυμα μασδιότι τα μπλόκ επικάλυψης ξεκινάνε πάντα με 1και τελειώνουν σε 0.Για παράδειγμα, εαν ενα συγκεκριμένο μπλόκ έχει μήκος μεγαλύτερο απο του βασικού μπλόκ απλά προσθέτουμε το 1000 σαν μπλόκ επικάλυψης(ISOpadding)

Το **CMAC**μας δείχνει ότι έαν χρησιμοποιήσουμε μια τυχαία συνάρτηση επικάλυψης θα μπορούσαμε να **αποφύγουμε** την ανάγκη να χρησιμοποιήσουμε ένα μπλόκ επικάλυψης.Το CMACχρησιμοποιεί 3 κλειδιά(**Εικόνα 68**).Το πρώτο κλειδί kχρησιμοποιείται στο κλασσικό CBCMAC, ενώ τα κλειδιά k_1,k_2 χρησιμοποιούνται για το σχήμα επικάλυψης(paddingscheme)για το τελευταίο μπλόκ του συνολικού μηνύματος που θέλουμε να αποστείλουμε.Τα κλειδιά αυτά παράγονται απο το κλειδί kσύμφωνα με μία γεννήτρια ψευδοτυχαίων αριθμών.



Εικόνα 67:Η λειτουργία του CMAC

Έαν το μήνυμα τυγχάνει να μην είναι πολλαπλάσιο του βασικού μπλόκ τότε εφαρμόζουμεISOpaddingκαι μετα κάνουμε πράξη XORστο τελευταίο αυτό μπλόκ με το κλειδί k_1.Εαν το μήνυμα είναι όμως πολλαπλάσιο του βασικού μπλόκ δεν κάνουμε απολύτως τίποτα.Απλά του κάνουμε πράξη XORμεταξύ του κλειδιού k_2.

5.1.3 Η λειτουργία τουPMAC

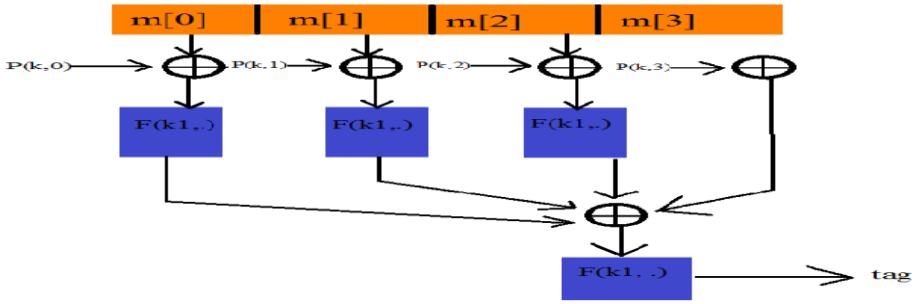
Οι κατασκευέςCBC-MACκαιNMAC έιναι ακολουθιακές με αποτέλεσμα εάν κάποιος έχει πολλαπλούς επεξεργαστες δέν θα μπορεί να κάνει αυτές τις κατασκευές να λειτουργήσουν πίο γρήγορα.Για τον λόγο αυτό θα μιλήσουμε για παράλληλα MACτα οποία επίσης μετατρέπουν μικρες PRFσε πολύ μεγάλες PRFαλλα με **παράλληλη** διαδικασία.

Θα δούμε μια παράλληλη κατασκευή η οποία ονομάζεται **PMAC**η οποία χρησιμοποιεί μια PRFF:K× X →Xγια να κατασκευάσουμε μια PRF_{PMAC}για πολύ μεγάλα μηνύματα:

PRF_{PMAC}:K^2×X→X όπου X≤L

Αρχικά παίρνουμε το μήνυμα μας και το σπάμε σε μπλόκς και μετά επεξεργαζόμαστε κάθε μπλόκ ανεξάρτητα απο το άλλο(**Εικόνα 69**). Στη συνέχεια χρησιμοποιούμε μια συνάρτηση Pκαι το αποτέλεσμα της συνάρτησης του κάνουμε πράξη XORμε το εκάστοτε μήνυμα..Στο τελικό αποτέλεσμα όμως, εφαρμόζουμε μια συνάρτηση Fχρησιμοποιώντας το κλειδί k_1.

Αυτην την διαδικασία την εφαρμόζουμε για κάθε μπλόκ του μηνύματος μας.Τέλος συλλέγουμε όλα αυτά τα μηνύματα που υπέστησαν την παρακάτω διαδικασία και τα περνάμε μέσα απο μια τελική XORπράξη και μια κρυπτογράφηση με την συνάρτηση F, πάλι με το κλειδί k_1, και αυτή είναι τελικά η ζητούμενη ετικέτα μας.



Εικόνα 68:Η λειτουργία του PMAC

Έστω ότι δέν υπήρχε η συνάρτηση Ρκαι υπήρχε όμως όλη η άλλη διαδικασία PMACόπως την ξέρουμε.Η διαδικασία όμως τώρα είναι ανασφαλής καθώς έαν εναλλάξουμε(swap) 2 μπλόκς, δέν αλλάζει η τελική τιμή της ετικέτας.Σαν αποτέλεσμα ένας επιτιθέμενος μπορεί να ζητήσει μια ετικέτα για ένα συγκεκριμένο μήνυμα και να λάβει όμως, την ετικέτα ενός μηνύματος **όπου** 2 απο τα συνολικά μπλόκς έχουν εναλλαχτεί, και έτσι μπορεί να υπάρξει επίθεση υπαρξιακής πλαστογραφίας.

Αυτό που κάνει η συνάρτηση **P**είναι να παίρνει ώς είσοδο ένα κλειδί kκαι τον αριθμό του εκάστοτε μπλόκ.Η τιμή τη συνάρτησης Ρειναι διαφορετική για κάθε μπλόκ.Έαν το μήκος του κάθε μπλόκ είναι μικρότερο απο το βασικό μπλόκ τότε ο PMACχρησιμοποιεί μια παρόμοια επικάλυψη που είναι ίδια με αυτή του CMAC άρα δέν χρειάζονται πλέον αυθαίρετα μπλόκ επικάλυψης.

Εκτός αυτού, το PMACέχει μια **πολύ ενδιαφέρουσα ιδιότητα.**Υποθέστε ότι η συνάρτηση Fη οποία χρησιμοποιείται για να κατασκευάσουμε ένα PMAC δε είναι PRFαλλα μια PRP.Υποθέστε επίσης οτι υπολογίσαμε το MACγια ένα μήνυμα και ότι ένα μπλόκ του μηνύματος m[1]άλλαξε σε m΄[1] και τα υπόλοιπα μπλόκς παρέμειναν το ίδιο.Σε άλλα MACS, παραδείγματος χάρη CBC, η αλλαγή ενός μπλόκ συνεπάγεται με υπολογισμό απο την αρχή, της ετικέτας του μηνύματος μας.Τοερωτήμα είναι με το PMACμπορούμε να υπολογίσουμε την ετικέτα ενός μηνύματος πολύ **πιο γρήγορα** έαν άλλαξε μόνο ένα μπλόκ;

Υποθέστε ότι ένα μπλόκ m[1] άλλαξε σε κάποιο άλλο μπλοκ m΄[1].Αυτο που μπορούμε να κάνουμε είναι να πάρουμε την ετικέτα και το μήνυμα πρωτού υποστεί την αλλαγή και να εφαρμόσουμε την αντιστρέψιμη συνάρτηση F^{-1}με σκοπό να λάβουμε την τιμή πρωτού εφαρμοστεί η F.Τώρα επειδή έχουμε κάνει πράξη XOR μεταξύ των μπλόκς, μπορούμε να κάνουμε XORμεταξύ της τιμής αυτής και του αποτελέσματος της πράξηςXORμεταξύ μηνύματος m[1] και P[k,1] και τέλος με το αποτέλεσμα αυτού XOR πράξη στο αποτέλεσμα της συνάρτησης που κρυπτογραφεί όμως το m΄[1] και όχι το m[1] και τέλος κρυπτογραφούμε το αποτέλεσμαF($k_1, \cdot$) δηλαδή:

$$\mathbf{F^{-1}(k_1,tag)\oplus F(k_1,m[1]\oplus P(k_1,1))\oplus F(k1,m'[1]\oplus P(k_1,1))}$$

<u>Θεώρημα ασφάλειας PMAC 5.1.3</u>

Έαν υπάρχει αντίπαλος A που επιτίθεται στο PMAC, τότε μπορούμε να δημιουργήσουμε έναν επιτιθέμενο B ο οποίος επιτίθεται στηνPRF σύν έναν όρο λάθους(errorterm):

$$\mathbf{Adv_{PRF}[A,F_{PMAC}]\leq Adv_{PRF}[B,F]+2q^2L^2/|X|}$$

Επειδή η PRF είναι ασφαλής τότε και το errorterm θα είναι αμελητέο.Το PMAC είναι ασφαλές έαν $\mathbf{qL<<|X|^{1/2}}$, οπου L το μήκος του μεγαλύτερου μπλόκ του μηνύματος.

5.2Αντοχή στις συγκρούσεις

5.2.1 Η κατασκευή Merkl-Damgard[48]

Πρώτα θα περιγράψουμε τι είναι μια συνάρτηση κατακερματισμού και στη συνέχεια θα προχωρήσουμε στην κατασκευή**Merkl-Damgard**. Η συνάρτηση κατατεμαχισμού, γνωστή και ως συνάρτηση κατακερματισμού, είναι μια μαθηματική συνάρτηση που δέχεται ως είσοδο κάποιο δεδομένο τυχαίου μεγέθους και επιστρέφει ένα ακέραιο σταθερού μεγέθους αναπαράστασης.

Το μέγεθος αυτό μπορεί να είναι από 32bit μέχρι 256bit ή περισσότερα, ανάλογα με το λόγο χρήσης της συνάρτησης. Οι τιμές που επιστρέφει η συνάρτηση κατατεμαχισμού ονομάζονται τιμές κατατεμαχισμού (hash values), κώδικες κατατεμαχισμού (hash codes), αθροίσματα κατατεμαχισμού (hash sums) ή απλά τιμές κατατεμαχισμού (hashes).Οι τιμές αυτές θα πρέπει να είναι διαφορετικές για διαφορετική είσοδο, καθώς η κύρια χρησιμότητα αυτών των συναρτήσεων είναι να ταυτοποιούν τα δεδομένα.

Μια συνάρτηση κατατεμαχισμού μπορεί να αντιστοιχίζει δύο ή περισσότερους εισόδους στην ίδια τιμή κατατεμαχισμού. Στις περισσότερες εφαρμογές είναι επιθυμητή η ελαχιστοποίηση αυτών των συγκρούσεων. Αυτό σημαίνει ότι η συνάρτηση κατατεμαχισμού θα πρέπει να αντιστοιχίζει κάθε είσοδο σε διαφορετική τιμή κατατεμαχισμού.Μερικές απο τις εφαρμογές τους είναι οι ψηφιακές υπογραφές και η **αυθεντικότητα μηνύματος**(MAC)

Θα λέμε ότι μια συνάρτηση κατακερματισμού έχει αντοχή στις συγκρούσεις εάν είναι πολύ δύσκολο να βρούμε δύο εισόδους που να μας δίνουν την ίδια κατακερματισμένη έξοδο.Η κατασκευή Merkl-Damgardπαρέχει αντοχή στις συγκρούσεις.Θεωρείστε μια συνάρτηση κατακερματισμού $H:M\rightarrow T$και $|M|>>|T|$.Θεωρείστε επίσης δύο μηνύματα m_0,m_1που ανήκουν στο M και ισχύει ότι $H(m_0)=H(m_1)$ και $m_0 \neq m_1$.

Θα δείξουμε γιατι η αντοχή στις συγκρούσεις είναι πολύ χρήσιμη μ παραδείγματα.Το πρώτο παραάδειγμα αφορά για το πώς μπορούμε να κατασκευάσουμε έναν MACμε μια δοσμένη συνάρτηση κατακερματισμού που έχει αντοχή στις συγκρούσεις.Έστω ένας MACο οποίος χρησιμοποιείται για μικρού μήκους μηνύματα(πχ AES) και μια συνάρτηση κατακερματισμού η οποία παίρνει πολύ μεγάλα μηνύματα ώς είσοδο(συνήθως μερικά gigabytes) και μας δίνει μικρά μηνύματα δηλαδή $H:M^{big}\rightarrow M$.

Ας ορίσουμε τονMACως $I_{big}=(S^{big},V^{big})$στο (K,M^{big},T) όπου:

$$\mathbf{S^{big}(k,m)=S(k,H(m))\ και V^{big}(k,m,t)=V(k,H(m),t)}$$

Αρα εφαρμόζοντας στην έξοδο της συνάρτησης κατακερματισμού από το «μικρό»MACπαίρνουμε τα αντίστοιχα S^{big},V^{big}.Για να επαληθεύσουμε το MACπρέπει όταν μας δώσουν μια ετικέτα, να υπολογίσουμε το hashτου μηνύματος που μας έδωσαν και να δούμε έαν η ετικέτα είναι ίδια με αυτή που μας δώσαν.

---

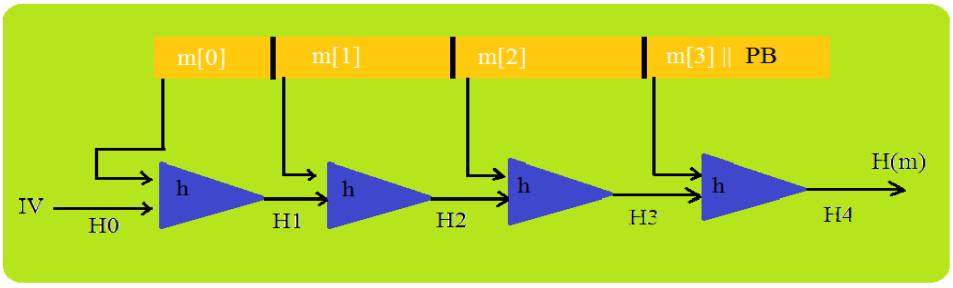
[48] Η κατασκευή Merkl-Damgard,
https://en.wikipedia.org/wiki/Merkle%E2%80%93Damg%C3%A5rd_construction

Θα δείξουμε ένα ακομή παράδειγμα.Έστω ότι εχουμε ένα πλήθος αρχείων που θέλουμε να προστατεύσουμε και ότι αυτά τα αρχεία είναι πακέτα λογισμικού που θέλουμε να κάνουμε εγκατάσταση στο σύστημα μας.Εμείς θέλουμε να είμαστε σίγουροι ότι τα πακέτα που κατεβάσαμε είναι έγκυρα, και ότι δέν έχουν τροποποιηθεί απο κάποιο επιτιθέμενο.

Ο δημιουργός των πακέτων απλά υπολογίζει το hash των πακέτων πρίν τα ανεβάσει σε κάποιον serverγια να τα κατεβάσουμε εμεις.Όμως η συνάρτηση κατακερματισμού που χρησιμοποιήσε πρέπει να έχει αντοχή στις συγκρούσεις που αυτό συνεπάγεται ότι ένας επιτιθέμενος δεν μπορεί να αλλάξει το περιεχόμενο των πακέτων.Αυτό που κάνουμε εμείς είναι μόλις κατεβάσουμε τα πακέτα, να υπολογίσουμε το hashαυτών.Έαν το hashπου υπολογίσαμε εμείς είναι το ίδιο με αυτό που υπολόγισε ο δημιουργός των πακέτων, τότε τα πακέτα είναι έγκυρα δηλαδή μη τροποποιημένα απο κάποιον τρίτο.

Τώρα μπορούμε να αναφερθούμε στην κατασκευή Merkl-Damgard.Έστω μια συνάρτηση Hη οποία θα θεωρήσουμε οτι είναι μια συνάρτηση κατακερματισμου η οποία έχει αντοχή στις συγκρούσεις για μικρου μεγέθους εισόδους.Θα χρησιμοποιήσουμε επίσηςμια συνάρτηση hπου ονομάζεται**συνάρτηση συμπίεσης**(compressionfunction) και ένα μεγάλο μήνυμα M.

Αρχικά σπάμε το μήνυμα M σε μπλόκς.Στη συνέχεια το πρώτο μπλόκ του μηνύματος μας m[0] μαζί με μια τιμή IVπερνά μέσα απο την συνάτηση hκαι μας βγάζει ένα αποτέλεσμα που ονομάζεται μεταβλητή αλυσίδας.Το αποτέλεσμα που βρήκαμε θα περάσει απο μια hμαζί με το δέυτερο μπλόκ του μηνύματος m[1] και η διαδικασία θα συνεχιστεί μέχρι το τελευταίο μπλόκ στο οποίο όμως θα προσθέσουμε και ένα πλόκ επικάλυψης.Το μπλόκ επικάλυψης και τελευταίο μήνυμα περνάνε απο μια συνάρτηση h μαζί με το προτελευταίο αποτέλεσμα αυτής(**Εικόνα 70**)



Εικόνα 69:Η λειτουργία της κατασκευής Mercle-Damgad

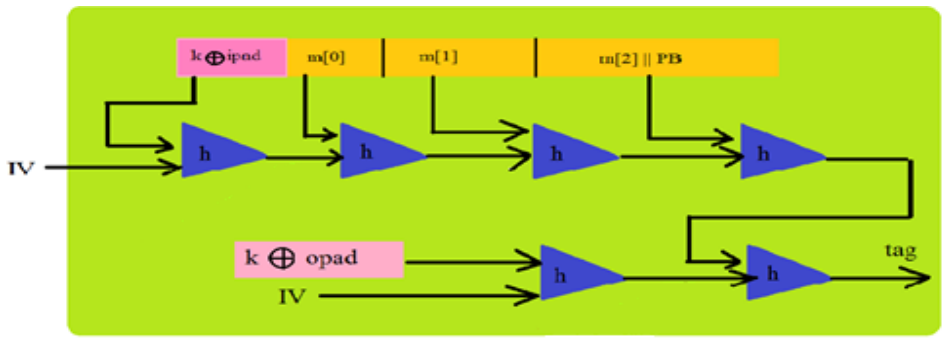
***Θεώρημα Merkle-Damgard* 5.2.1**

Εάν μια συνάρτηση κατακερματισμού h έχει αντοχη στις συγκρούσεις τότε η «μεγάλη» συνάρτηση κατακερματισμούHτουMerkle-Damgad έχει επίσης αντοχή στις συγκρούσεις.Με άλλα λόγια όταν προσπαθούμε να κατασκευάσουμε μια συνάρτηση κατακερματισμού η οποία έχει αντοχή στις συγκρούσεις, για πολύ μεγάλες εισόδους τότε το μόνο που έχουμε να κάνουμε είναι απλά να κατασκευάσουμε μια συνάρτηση συμπίεσης η οποία έχει αντοχή στις συγκρούσεις.

5.2.2HMAC[49]

Τώρα που γνωρίζουμε τι είναι οι συναρτήσεις κατακερματισμού με αντοχή στις συγκρούσεις και πώς μπορούμε να τις κατασκευάσουμε, μπορούμε να παρουσιάσουμε έναν πολύ γνωστό MACο οποίος ονομάζεται **HMAC**.Θα προσπαθήσουμε να χρησιμοποιήσουμε την συνάρτηση H για να κατασκευάσουμε έναν MACχωρίς να βασιστούμε σε μια PRF.Αυτό μπορεί να γίνει εαν χρησιμοποιήσουμε την συνάρτηση κατακερματισμού **SHA-256**[50].Η έξοδος θα είναι 256 bits.Επειδή το HMACείναι μια συνάρτηση ψευδοτυχίας τότε αφού έχουμε έξοδο 256 bitsαπο την SHA-256, τότε θα έχουμε μια συνάρτηση ψευδοτυχίας με έξοδο 256 bits.

Αρχικα σπάμε το μήνυμα μας σε μπλόκς.Στη συνέχεια εισάγουμε ένα **k ⊕ipad**μπλόκ μαζί με μια τιμή IVκαι τα περνάμε μέσα απο μια συνάρτηση συμπίεσης.Η έξοδος της συνάρτησης hχρησιμέυει ώς είσοδος στην επόμενη συνάρτηση hπου θα πάρει και ώς είσοδο επίσης το μπλόκ 0 του μηνύματος μας.Η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο τελευταίο μας μπλόκ το οποίο περιέχει και ένα μπλόκ επικάλυψης.Η τελευταία έξοδος της συνάρτησης hδέν μας δίνει το hashή την ετικέτα όσο αφορά τον HMAC, αλλα μας δίνει μια έξοδο που θα τηβάλουμε ώς είσοδο σε μια άλλη συνάρτηση h(**Εικόνα 71**)



Εικόνα 70:Η λειτουργία του HMAC

Ομως αυτή η τελευταία συνάρτηση hπαίρνει δύο εισόδους αντί για δύο.Την έξοδο της τελευταίας συνάρτησης hπου χρησιμοποιήθηκε για το τελευταίο μπλόκ του μηνύματος και ένα αποτέλεσμα μιας άλλης συνάρτησης hη οποία έχει και αυτή δύο εισόδους.Ενα μπλόκ **k ⊕opad**και μια τιμή IV.Τα k⊕ipad , k⊕opad οι οποίες είναι σταθερές 512 bitσταθερές που ορίζονται απο το HMAC και δέν αλλάζουν ποτέ.

5.3 Οι επιθέσεις πάνω στους κώδικες MAC

5.3.1 Επίθεση γενεθλίων[51]

Οταν μιλήσαμε για αλγόριθμους τμήματος είδαμε την επίθεση εξαντλητικής αναζήτησης που έκανε επίθεση στο κλειδί το αλγορίθμου.Παρόμοια υπάρχει μια επίθεση στις συναρτήσεις που έχουν αντοχή στις συγκρούσεις που ονομάζεται **επίθεση γενεθλίων**.Εστω ότι έχουμε μια συνάρτηση κατακερματισμού H η οποία βγάζει ώς έξοδονbitsκαι ο χώρος των μηνυμάτων Μείναι πολύ μεγαλύτερος απο 2^n.Θα

[49] Η λειτουργία του HMAC, http://en.wikipedia.org/wiki/Hash-based_message_authentication_code
[50] Οι συναρτήσεις κατακερματισμού SHA, http://en.wikipedia.org/wiki/SHA-2
[51] Η επίθεση γενεθλίων, http://en.wikipedia.org/wiki/Birthday_attack

δείξουμε έναν αλγόριθμο που μας βρίσκει σύγκρουση στην συνάρτηση H σε χρόνο $2^{n/2}$.

Αλγόριθμος

Αρχικά επιλέγουμε $2^{n/2}$τυχαία μηνύματα απο το Μτα οποία είναι διαφορετικά μεταξύ τους.Για ένα i=1,…$2^{n/2}$πρέπει να υπολογίσουμε το $t_i=H(m_i)$ και μετά να κοιτάξουμε για κάθε i και ένα jεαν υπάρχουν μηνύματα που να ισχύει $t_i=t_j$άρα θα βρούμε σύγκρουση.Αλλίως επαναλαμβάνουμε την διαδικασία απο την αρχή.Για να αναλύσουμε αυτή την επίθεση θα πρέπει να αναφερθούμε στο **παράδοξο τωνγενεθλίων**[52]**.**

Θεώρημα παραδόξου γενεθλίων

Έστω ότι έχουμε N τυχαίες μεταβλητές $r_1,…r_n \in \{1,…,B\}$ οι οποίες είναι ανεξάρτητες μεταξύ τους και έχουν διανεμηθεί με τον ίδιο τρόπο.Τυχαίνει οτι όταν $n=1.2\times B^{1/2}$ τότε η πιθανότητα ότι δύο απο όλα αυτά τα δείγματα να είναι ίδια έιναι μεγαλύτερη ή και ίση απο 1/2 .

Απο το θεώρημα βλέπουμε σε κάθε επανάληψη το να βρούμε σύγκρουση είναι περιπου 1/2 τότε, χρειαζόμαστε μόνο 2 φορές να επαναλάβουμε τον αλγόριθμο πρωτού βρούμε μια σύγκρουση.Έαν για παράδειγμα έχουμε 128bitώς έξοδο τότε μπορούμε να βρούμε μια σύγκρουση σε χρόνο $2^{n/2}=2^{128/2}=2^{64}$το οποίο δέν θεωρειται ασφαλές.Για αυτόν τον λόγο οι συναρτήσεις κατακερματισμού που έχουν αντοχή στις συγκρούσεις δέν θα έχουν ποτέ ώς έξοδο 128 bits.

5.4Ερωτήσειςκεφαλαίου

Ερώτηση 1

Suppose a MAC system (S,V) is used to protect files in a file system by appending a MAC tag to each file. The MAC signing algorithm S is applied to the file contents and nothing else. What tampering attacks are not prevented by this system?

a) Replacing the contents of a file with the concatenation of two files on the file system.
b) Erasing the last byte of the file contents.
c) Swapping two files in the file system.
d) Replacing the tag and contents of one file with the tag and contents of a file from
e)Another computer protected by the same MAC system, but a different key.

Απάντηση

The correct answer is **c**. Both files contain a valid tag and will be accepted at verification time.

Ερώτηση 2

Let (S,V) be a secure MAC defined over (K,M,T) where M={0,1}n and $T=\{0,1\}^{128}$ (i.e. the key space is K, message space is {0,1}n, and tag space is $\{0,1\}^{128}$). Which of the following is a secure MAC: (as usual, we use ‖ to denote string concatenation)

a) $S'(k,m)=S(k,m)[0,…,126]$και $V'(k,m,t)=[V(k, m, t‖0)$ ή $V(k, m, t‖1)$]

[52] Το παράδοξο των γενεθλίων, http://en.wikipedia.org/wiki/Birthday_problem

b) $S'(k,m)=S(k,m)$ και $V'(k,m,t)=[V(k, m, t)$ ή $V(k, m\oplus 1^n, t)]$

c)$S'(k, m)=[t\leftarrow S(k,m)$, έξοδος$(t_1,t_2)$)] καί$V'(k,m,(t_1,t_2))=\begin{cases} V(k,m,t1)\ \varepsilon\alpha\nu\ \ t1=t2 \\ \text{"0"}\ otherwise \end{cases}$

d) $S'(k,m)=S(k,\ m[0,\ldots,n-2]\|0)$ και $V'(k,m,t)=V(k,\ m[0,\ldots,n-2]\|\|0,\ t)$

e)$S'(k,m)=\begin{cases} S(k,1^n)\varepsilon\acute{\alpha}\nu\ \ m=0^n \\ S(k,m)otherwise \end{cases}$ και$V'(k,m)=\begin{cases} V(k,1^n\ ,t)\ \varepsilon\acute{\alpha}\nu\ m=0^n \\ V(k,m,t)\ otherwise \end{cases}$

f) $S'(k,m)=S(k,\ m\|m)$και $V'(k,m,t)=V(k,\ m\|m,\ t)$.

<u>Απάντηση</u>

Thecorrectanswersare**a**, **c**, **f** because a forger for (S',V') gives a forger for (S,V).

For the answer a we have:

If an attacker can forge a message pair on (S',V') with probability P, then we can forge a message pair on (S,V) with probability P/2.Since P/2 is negligible, P is also negligible.

The answer b is not correct:

This construction is insecure because a valid tag on $m=0^n$ is also a valid tag on $m=1^n$. Consequently, the attacker can request the tag on $m=0^n$ and output an existential forgery for $m=1^n$.

For example in cwe have:

If an attacker could generate a forgery $(m', (t_1', t_2'))$ on (S',V') then that implies that $V'(k, m, (t_1', t_2')) = 1$, so $t_1'=t_2'$ and $V(k,m',t_1') = 1$.Hence the attacker can forge the pair (m',t_1') on implying (S, V) is not secure.Therefore the MAC is secure

The answer d is not correct.This construction is insecure because the tags on $m=0^n$ and $m=0^{n-1}$are the same. Consequently, the attacker can request the tag on $m=0^n$ and output an existential forgery for $m=0^{n-1}$.**Second solution**: This MAC is not securebecause can attacker can ask for a message pair (m, t) with m[n-1] != 0 and produce the forgery (m[0,...,n-1] || 0, t)

The answer e is not correct.This construction is insecure because an adversary can request the tag for the message 0^n and output the result as a valid forgery for the message 1.In similar way answer **f is correct**.

<u>Ερώτηση 3</u>

Recall that the ECBC-MAC uses a fixed IV (in the lecture we simply set the IV to 0). Suppose instead we chose a random IV for every message being signed and include the IV in the tag. In other words, S(k,m):=(r, ECBCr(k,m)) where ECBCr(k,m) refers to the ECBC function using r as the IV. The verification algorithm V given key k, message m, and tag (r,t) outputs ``1" if t=ECBCr(k,m) and outputs ``0" otherwise.

The resulting MAC system is insecure. An attacker can query for the tag of the 1-block message m and obtain the tag (r,t). He can then generate the following existential forgery: (we assume that the underlying block cipher operates on n-bit blocks)

a)The tag (r⊕t, m) is a valid tag for the 1-block message 0^n.
b)The tag (r⊕m, t) is a valid tag for the 1-block message 0^n.
c)The tag (r, t⊕r) is a valid tag for the 1-block message 0^n.
d)The tag (m⊕t, r) is a valid tag for the 1-block message 0^n.

Απάντηση
The correct answer is **b**. The CBC chain initiated with the IV r⊕m and applied to the message 0^nwill produce exactly the same output as the CBC chain initiated with the IV r and applied to the message m. Therefore, the tag (r⊕m, t) is a valid existential forgery for the message 0.
Second solution:
The attacker found that m → (r,t). In other words, t=F(k_1, F(k, (m ⊕r))).The attacker can forge _(m ⊕ 1^n, (r ⊕ 1^n, t) because ECBC checks that t = F(k_1, F(k, (m' ⊕ r'))) = F(k1, F(k, (m ⊕ 1^n xor r ⊕ 1^n))).We see from the above that this is true and ECBC outputs "1"

Ερώτηση 4
Suppose Alice is broadcasting packets to 6 recipients B_1…B6. Privacy is not important but integrity is. In other words, each of B_1…B6 should be assured that the packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and B_1... B6 all share a secret key k. Alice computes a tag for every packet she sends using key k. Each user B_i verifies the tag when receiving the packet and drops the packet if the tag is invalid. Alice notices that this scheme is insecure because user B_1 can use the key k to send packets with a valid tag to users B_2… B6 and they will all be fooled into thinking that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys S={k_1,…,k_4}. She gives each user Bi some subset S_i⊆S of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user Bi receives a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys {k_1,k_2} he will accept an incoming packet only if the first and second tags are valid. Note that B1 cannot validate the 3rd and 4th tags because he does not have k3 or k4.

How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user?

α)S_1={k_1,k_2},S_2={k_2,k_3},S_3={k_3,k_4},S_4={k_1,k_3},S_5={k_1},
β)S_1={k_1,k_2},S_2={k_1},S_3={k_1,k_4}, S_4={k_2,k_3},S_5={k_2,k_4}
γ)S_1={k_2,k_4},S_2={k_1,k_3},S_3={k_1,k_4},S_4={k_2, k_3},S_5={k_2}
δ)S_1={k_1,k_2}S_2={k_1,k_3,,k_4}, S_3={k_1,k_4}, S_4={k_2,k_3 }

Απάντηση
The correct answer is **c**. Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set Sj, no user i can fool a user j into accepting a message sent by i.

Ερώτηση 5

Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message m comprising of n AES blocks. Let m′ be the n-block message obtained from m by flipping the last bit of m (i.e. if the last bit of m is b then the last bit of m′ is b⊕1). How many calls to AES would it take to compute the tag for m′ from the tag for m and the MAC key? (In this question please ignore message padding and simply assume that the message length is always a multiple of the AES block size)

A)n+1
B)3
Γ)4
Δ)6

Απάντηση

The correct answer is **c**. You would decrypt the final CBC MAC encryption step done using k_2, the decrypt the last CBC MAC encryption step done using k_1, flip the last bit of the result, and re-apply the two encryptions.

Ερώτηση 6

Let H: M→T be a collision resistant hash function. Which of the following is collision resistant: (as usual, we use ‖ to denote string concatenation)

a) H′(m)=H(m[0,..|m|-2])
b) H′(m)=H(H(m))
c) H′(m)=H(|m|) (i.e hash the length of m)
d) H′(m)=H(m) ⊕H(m⊕$1^{|m|}$) (where m⊕$1^{|m}$is the complement of m)
e) H′(m)=H(0)
f) H′(m)=H(m||m)
g) H′(m)=H(H(H(m)))

Απάντηση

For answer b:
Suppose an attacker finds m1 and m_2 so that H(H(m_1)) == H(H(m_2)). Then the attacker could simply let H(m_1) = m_1' and H(m_2) = m_2' and show that H(m_1') = H(m2'). Since H is collision resistant, no such (m_1', m_2') can be found so the hash is collision resistant
For the answer g:
The hash is collision resistant from the same argument as for H(H(m)).
For the answer c:
The hash clearly _is not collision resistant_ because an attacker could produce a collision with any (m_1,m_2) where |m_1| == |m_2|
For the answer d:This hash is not collision resistant. An attacker can produce any pair (m, ~m) → (H(m) ⊕ H(~m), H(~m) ⊕ H(m)) which is a collision.
So the correct answers are c f, g

Ερώτηση 7

Suppose H_1 and H_2 are collision resistant hash functions mapping inputs in a set M to $\{0,1\}^{256}$. Our goal is to show that the function $H_2(H_1(m))$ is also collision resistant. We prove the contra-positive: suppose $H_2(H1(\cdot))$ is not collision resistant, that is, we are given $x \neq y$ such that $H_2(H_1(x))=H_2(H_1(y))$. We build a collision for either H_1 or for H2. This will prove that if H_1 and H_2 are collision resistant then so is $H_2(H_1(\cdot))$. Which of the following must be true?

a) Either x,y are a collision for H_1 or $H_1(x)$,H1(y) are a collision for H_2
b) Either $H_2(x)$,$H_2(y)$ are a collision for H1 or x,y are a collision for H_2.
c) Either x,y are a collision for H_1 or x,y are a collision for H_2.
d) Either x,y are a collision for H_1 or x,y are a collision for H_2.

Απάντηση
The correct answer is a because:
If $H_2(H_1(x))=H_2(H_1(y))$ then either $H_1(x)=H_1(y)$ and $x \neq y$, thereby giving us a collision on H_1. Or $H_1(x) \neq H_1(y)$ but $H_2(H_1(x))=H_2(H_1(y))$ giving us a collision on H_2. Either way we obtain a collision on H_1 or H_2 as required.

Ερώτησεις 8-9
In this question and the next, you are asked to find collisions on two compression functions:

- $f1(x,y)=AES(y,x) \oplus y$
- $f2(x,y)=AES(x,x) \oplus y$

where AES(x,y) is the AES-128 encryption of y under key x.

We provide an AES function for you to play with. The function takes as input a key k and an x value and outputs AES (k,x) once you press the "encrypt" button. It takes as input a key k and a y value and outputs $AES^{-1}(k,y)$ once you press the "decrypt" button. All three values k,x,y are assumed to be hex values (i.e. using only characters 0-9 and a-f) and the function zero-pads them as needed.Your goal is to find four distinct pairs (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4) such that $f_1(x_1,y_1)=f_1(x_2,y_2)$ and $f_2(x_3,y_3)=f_2(x_4,y_4)$. In other words, the first two pairs are a collision for f_1 and the last two pairs are a collision for f_2. Once you find all four pairs, please enter them below and check your answer using the "check" button.

Απάντηση
$f_1(x,y) = AES(y,x) \oplus y$.
We aim to find x_1,y_1,x_2,y_2 such that:

$$AES(y_1, x_1) \oplus y_1 = AES(y_2, x_2) \oplus y_2$$
$$AES(y_1,x_1) = AES(y_2,x_2) \oplus y_2 \oplus y_1$$

So first just let $x_2,y_2 = (0^n, 1 \,||\, 0^{n-1})$ and $y_1 = 11 \,||\, 0^{n-2}$.

Then $AES(y_2,x_2)$ = f5569b3ab6a6d11efde1bf0a64c6854a so $AES(y_1,x_1)$ = f5569b3ab6a6d11efde1bf0a64c6854a $\oplus$ $0^n \oplus$ 11 || 0^{n-2}= e4569b3ab6a6d11efde1bf0a64c6854a $\Rightarrow$ x1 = bc042352a96d8509fd1722c082c85c0c
For x_3 we must do : $f_2(x,y) = AES(x,x) \oplus y$

Ερώτηση 10

Let H:M→T be a random hash function where |M|≫|T| (i.e. the size of M is much larger than the size of T). In lecture we showed that finding a collision on H can be done with $O(|T|^{1/2})$ random samples of H. How many random samples would it take until we obtain a three way collision, namely distinct strings x,y,z in M such that H(x)=H(y)=H(z)?

Απάντηση

The correct answer isa. An informal argument for this is as follows: suppose we collect n random samples. The number of triples among the n samples is n choose 3 which is $O(n^3)$. For a particular triple x,y,z to be a 3-way collision we need H(x)=H(y) and H(x)=H(z).

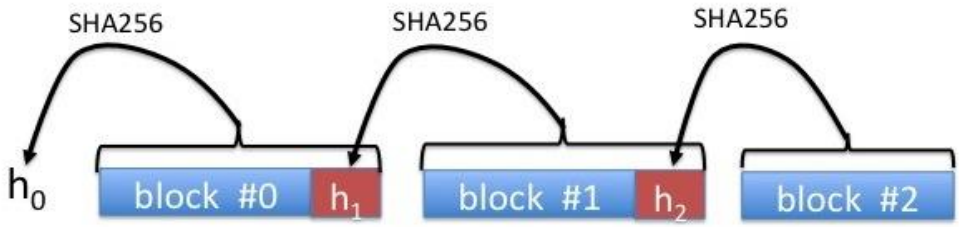
Since each one of these two events happens with probability 1/|T| (assuming H behaves like a random function) the probability that a particular triple is a 3-way collision is O(1/|T|2). Using the union bound, the probability that some triple is a 3-way collision is O(n3/|T|2) and since we want this probability to be close to 1, the bound on n follows.

Προγραμματιστική άσκηση 3

Suppose a web site hosts large video file F that anyone can download. Browsers who download the file need to make sure the file is authentic before displaying the content to the user. One approach is to have the web site hash the contents of F using a collision resistant hash and then distribute the resulting short hash value h=H(F) to users via some authenticated channel (later on we will use digital signatures for this). Browsers would download the entire file F, check that H(F) is equal to the authentic hash value h and if so, display the video to the user.

Unfortunately, this means that the video will only begin playing after the *entire* file F has been downloaded. Our goal in this project is to build a file authentication system that lets browsers authenticate and play video chunks as they are downloaded without having to wait for the entire file.

Instead of computing a hash of the entire file, the web site breaks the file into 1KB blocks (1024 bytes). It computes the hash of the last block and appends the value to the second to last block. It then computes the hash of this augmented second to last block and appends the resulting hash to the third block from the end. This process continues from the last block to the first as in the following diagram(**Εικόνα2**).



Εικόνα 71:Τοποθέτηση μιας τιμής hash στο αμέσως προηγούμενο μπλόκ

The final hash value h0 – a hash of the first block with its appended hash – is distributed to users via the authenticated channel as above.Now, a browser downloads

the file F one block at a time, where each block includes the appended hash value from the diagram above.

When the first block ($B_0 \| h_1$) is received the browser checks that $H(B_0 \| h_1)$ is equal to h0 and if so it begins playing the first video block. When the second block ($B_1 \| h_2$) is received the browser checks that $H(B_1 \| h_2)$ is equal to h1 and if so it plays this second block.

This process continues until the very last block. This way each block is authenticated and played as it is received and there is no need to wait until the entire file is downloaded.

It is not difficult to argue that if the hash function H is collision resistant then an attacker cannot modify any of the video blocks without being detected by the browser. Indeed, since $h_0=H(B_0 \| h_1)$ an attacker cannot find a pair $(B'_0,h'_1)\neq(B_0,h_1)$ such that $h_0=H(B_0 \| h_1)$ since this would break collision resistance of H.

Therefore after the first hash check the browser is convinced that both B_0 and h_1 are authentic. Exactly the same argument proves that after the second hash check the browser is convinced that both B_1 and h_2 are authentic, and so on for the remaining blocks.

In this project we will be using SHA256 as the hash function. For an implementation of SHA256 use an existing crypto library such as PyCrypto (Python), Crypto++ (C++), or any other.

When appending the hash value to each block, please append it as binary data, that is, as 32 unencoded bytes (which is 256 bits). If the file size is not a multiple of 1KB then the very last block will be shorter than 1KB, but all other blocks will be exactly 1KB.

Your task is to write code to compute the hash h0 of a given file F and to verify blocks of F as they are received by the client. In the box below please enter the (hex encoded) hash h0 for this video file.

Απάντηση

Ο κώδικας της προγραμματιστικής άσκησης 3 βρίσκεται εδώ.Αυτό που κάνουμε είναι μέσω της openανοίγουμε το mp4 αρχείο μας και ενώ τα dataδεν είναι κενά αρχίζουμε να διαβάζουμε μπλόκ των 1024KB.Μετά για όλα τα μπλόκς δίνουμε το hashενός μπλόκ στο αμέσως προηγούμενο μπλόκ απο αυτό και μετα εφαρμόζουμε hashστο προηγούμένο αυτό μπλοκ(με το hashπου πήρε ώς τιμη).

Ολη διαδικασία γίνεται μέχρι να φτάσουμε στο πρώτο μπλόκ του αρχείου για να δούμε έαν το hashτο αρχείου που μας δώθηκε έαν είναι ίδιο με αυτό που υπολογίσαμε και εμεις.Στη προκειμένη περίπτωση το βίντεο είναι έγκυρο και το πρόγραμμα βγάζει ώς έξοδο το hashσε μορφή δεκαξαδικού:

sample: 5b96aece304a1422224f9a41b228416028f9ba26b0d1058f400200f06a589949

5.5 Αναλυτική βαθμολογία

You submitted this homework on **Sat 23 May 2015 8:38 PM EEST**. You got a score of **7.38** out of **10.00**. You can attempt again in 10 minutes.

Question 1

Suppose a MAC system (S, V) is used to protect files in a file system by appending a MAC tag to each file. The MAC signing algorithm S is applied to the file contents and nothing else. What tampering attacks are not prevented by this system?

Your Answer		Score	Explanation
Replacing the contents of a file with the concatenation of two files on the file system.			
Erasing the last byte of the file contents.			
Swapping two files in the file system.	✔	1.00	Both files contain a valid tag and will be accepted at verification time.
Replacing the tag and contents of one file with the tag and contents of a file from another computer protected by the same MAC system, but a different key.			
Total		1.00 / 1.00	

Εικόνα 72:Ερώτηση 1-Week 3

Question 2

Let (S,V) be a secure MAC defined over (K,M,T) where $M = \{0,1\}^n$ and $T = \{0,1\}^{128}$ (i.e. the key space is K, message space is $\{0,1\}^n$, and tag space is $\{0,1\}^{128}$). Which of the following is a secure MAC: (as usual, we use $\|$ to denote string concatenation)

Your Answer		Score	Explanation
$S'(k,m) = S(k,m)[0,\ldots,126]$ and $V'(k,m,t) = \left[V(k, m, t\|0) \text{ or } V(k, m, t\|1)\right]$ (i.e., $V'(k,m,t)$ outputs "1" if either $t\|0$ or $t\|1$ is a valid tag for m)	✔	0.17	a forger for (S', V') gives a forger for (S, V).
$S'(k,m) = S(k,m)$ and $V'(k,m,t) = \left[V(k, m, t) \text{ or } V(k, m \oplus 1^n, t)\right]$ (i.e., $V'(k,m,t)$ outputs "1" if t is a valid tag for either m or $m \oplus 1^n$)	✔	0.17	This construction is insecure because a valid tag on $m = 0^n$ is also a valid tag on $m = 1^n$. Consequently, the attacker can request the tag on $m = 0^n$ and output an existential forgery for $m = 1^n$.

Εικόνα 73:Ερώτηση 2a-Week 3

✔ $S'(k,m) = \left[t \leftarrow S(k,m),\ \text{output } (t,t)\right)$ and $V'\left(k,m,(t_1,t_2)\right) = \begin{cases} V(k,m,t_1) & \text{if } t_1 = t_2 \\ \text{"0"} & \text{otherwise} \end{cases}$ (i.e., $V'\left(k,m,(t_1,t_2)\right)$ only outputs "1" if t_1 and t_2 are equal and valid)	0.17	a forger for (S',V') gives a forger for (S,V).
✔ $S'(k,m) = S(k,\ m[0,\ldots,n-2]\|0)$ and $V'(k,m,t) = V(k,\ m[0,\ldots,n-2]\|0,\ t)$	✘ 0.00	This construction is insecure because the tags on $m = 0^n$ and $m = 0^{n-1}1$ are the same. Consequently, the attacker can request the tag on $m = 0^n$ and output an existential forgery for $m = 0^{n-1}1$.
✔ $S'(k,m) = \begin{cases} S(k,1^n) & \text{if } m = 0^n \\ S(k,m) & \text{otherwise} \end{cases}$ and $V'(k,m) = \begin{cases} V(k,1^n,t) & \text{if } m = 0^n \\ V(k,m,t) & \text{otherwise} \end{cases}$	✔ 0.17	This construction is insecure because an adversary can request the tag for the message 0^n and output the result as a valid forgery for the message 1^n.
✔ $S'(k,m) = S(k,\ m\|m)$ and $V'(k,m,t) = V(k,\ m\|m,\ t)$.	✘ 0.00	a forger for (S',V') gives a forger for (S,V).
Total	0.67 / 1.00	

Εικόνα 74:Ερώτηση 2b-Week 3

Question 3

Recall that the ECBC-MAC uses a fixed IV (in the lecture we simply set the IV to 0). Suppose instead we chose a random IV for every message being signed and include the IV in the tag. In other words, $S(k,m) := \left(r,\ \text{ECBC}_r(k,m)\right)$ where $\text{ECBC}_r(k,m)$ refers to the ECBC function using r as the IV. The verification algorithm V given key k, message m, and tag (r,t) outputs ``1" if $t = \text{ECBC}_r(k,m)$ and outputs ``0" otherwise.

The resulting MAC system is insecure. An attacker can query for the tag of the 1-block message m and obtain the tag (r,t). He can then generate the following existential forgery: (we assume that the underlying block cipher operates on n-bit blocks)

Your Answer	Score	Explanation
The tag $(r \oplus t,\ m)$ is a valid tag for the 1-block message 0^n.		
◉ The tag $(r \oplus m,\ t)$ is a valid tag for the 1-block message 0^n.	✔ 1.00	The CBC chain initiated with the IV $r \oplus m$ and applied to the message 0^n will produce exactly the same output as the CBC chain initiated with the IV r and applied to the message m. Therefore, the tag $(r \oplus m,\ t)$ is a valid existential forgery for the message 0.
The tag $(r,\ t \oplus r)$ is a valid tag for the 1-block message 0^n.		
The tag $(m \oplus t,\ r)$ is a valid tag for the 1-block message 0^n.		
Total	1.00 / 1.00	

Εικόνα 75:Ερώτηση 3-Week 3

Question 4

Suppose Alice is broadcasting packets to 6 recipients $B_1, \ldots, B_6$. Privacy is not important but integrity is. In other words, each of $B_1, \ldots, B_6$ should be assured that the packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and $B_1, \ldots, B_6$ all share a secret key k. Alice computes a tag for every packet she sends using key k. Each user B_i verifies the tag when receiving the packet and drops the packet if the tag is invalid. Alice notices that this scheme is insecure because user B_1 can use the key k to send packets with a valid tag to users $B_2, \ldots, B_6$ and they will all be fooled into thinking that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys $S = \{k_1, \ldots, k_4\}$. She gives each user B_i some subset $S_i \subseteq S$ of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user B_i receives a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4.

How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user?

Your Answer	Score	Explanation
$S_1 = \{k_1, k_2\},\ S_2 = \{k_2, k_3\},\ S_3 = \{k_3, k_4\},\ S_4 = \{k_1, k_3\},\ S_5 = \{k_1,$		
$S_1 = \{k_1, k_2\},\ S_2 = \{k_1\},\ S_3 = \{k_1, k_4\},\ S_4 = \{k_2, k_3\},\ S_5 = \{k_2, k_4$		
$S_1 = \{k_1, k_2\},\ S_2 = \{k_1, k_3\},\ S_3 = \{k_1, k_4\},\ S_4 = \{k_2, k_3\},\ S_5 = \{k_2,$ ✔	1.00	Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j, no user i can fool a user j into accepting a message sent by i.
$S_1 = \{k_1, k_2\},\ S_2 = \{k_1, k_3, k_4\},\ S_3 = \{k_1, k_4\},\ S_4 = \{k_2, k_3\},\ S_5 = \{$		
Total	1.00 / 1.00	

Εικόνα 76: Ερώτηση 4-Week 3

Question 5

Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message m comprising of n AES blocks. Let m' be the n-block message obtained from m by flipping the last bit of m (i.e. if the last bit of m is b then the last bit of m' is $b \oplus 1$). How many calls to AES would it take to compute the tag for m' from the tag for m and the MAC key? (in this question please ignore message padding and simply assume that the message length is always a multiple of the AES block size)

Your Answer	Score	Explanation
$n+1$		
3		
4 ✔	1.00	You would decrypt the final CBC MAC encryption step done using k_2, the decrypt the last CBC MAC encryption step done using k_1, flip the last bit of the result, and re-apply the two encryptions.
6		
Total	1.00 / 1.00	

Εικόνα 77:Ερώτηση 5-Week 3

Question 6

Let $H : M \to T$ be a collision resistant hash function. Which of the following is collision resistant: (as usual, we use $\|$ to denote string concatenation)

Your Answer		Score	Explanation
☑ $H'(m) = H(m[0,\ldots,\|m\|-2])$ (i.e. hash m without its last bit)	✘	0.00	This construction is not collision resistant because $H(00) = H(01)$.
☑ $H'(m) = H(H(m))$	✔	0.14	a collision finder for H' gives a collision finder for H.
☐ $H'(m) = H(\|m\|)$ (i.e. hash the length of m)	✔	0.14	This construction is not collision resistant because $H(000) = H(111)$.
☐ $H'(m) = H(m) \bigoplus H(m \oplus 1^{\|m\|})$ (where $m \oplus 1^{\|m\|}$ is the complement of m)	✔	0.14	This construction is not collision resistant because $H(000) = H(111)$.
☐ $H'(m) = H(0)$	✔	0.14	This construction is not collision resistant because $H(0) = H(1)$.
☐ $H'(m) = H(m\|\|m)$	✘	0.00	a collision finder for H' gives a collision finder for H.
☑ $H'(m) = H(H(H(m)))$	✔	0.14	a collision finder for H' gives a collision finder for H.
Total		0.71 / 1.00	

Εικόνα 78:Ερώτηση 6-Week 3

Question 7

Suppose H_1 and H_2 are collision resistant hash functions mapping inputs in a set M to $\{0,1\}^{256}$. Our goal is to show that the function $H_2(H_1(m))$ is also collision resistant. We prove the contra-positive: suppose $H_2(H_1(\cdot))$ is not collision resistant, that is, we are given $x \neq y$ such that $H_2(H_1(x)) = H_2(H_1(y))$. We build a collision for either H_1 or for H_2. This will prove that if H_1 and H_2 are collision resistant then so is $H_2(H_1(\cdot))$. Which of the following must be true:

Your Answer		Score	Explanation
◉ Either x, y are a collision for H_1 or $H_1(x), H_1(y)$ are a collision for H_2.	✔	1.00	If $H_2(H_1(x)) = H_2(H_1(y))$ then either $H_1(x) = H_1(y)$ and $x \neq y$, thereby giving us a collision on H_1. Or $H_1(x) \neq H_1(y)$ but $H_2(H_1(x)) = H_2(H_1(y))$ giving us a collision on H_2. Either way we obtain a collision on H_1 or H_2 as required.
○ Either $H_2(x), H_2(y)$ are a collision for H_1 or x, y are a collision for H_2.			
○ Either x, y are a collision for H_1 or x, y are a collision for H_2.			
○ Either $x, H_1(y)$ are a collision for H_2 or $H_2(x), y$ are a collision for H_1.			
Total		1.00 / 1.00	

Εικόνα 79:Ερώτηση 7-Week 3

Question 8

In this question and the next, you are asked to find collisions on two compression functions:

- $f_1(x,y) = \text{AES}(y,x) \oplus y$, and
- $f_2(x,y) = \text{AES}(x,x) \oplus y$,

where $\text{AES}(x,y)$ is the AES-128 encryption of y under key x.

We provide an AES function for you to play with. The function takes as input a key k and an x value and outputs $\text{AES}(k,x)$ once you press the "encrypt" button. It takes as input a key k and a y value and outputs $\text{AES}^{-1}(k,y)$ once you press the "decrypt" button. All three values k,x,y are assumed to be hex values (i.e. using only characters 0-9 and a-f) and the function zero-pads them as needed.

Your goal is to find four distinct pairs (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4) such that $f_1(x_1,y_1) = f_1(x_2,y_2)$ and $f_2(x_3,y_3) = f_2(x_4,y_4)$. In other words, the first two pairs are a collision for f_1 and the last two pairs are a collision for f_2. Once you find all four pairs, please enter them below and check your answer using the "check" button.

Note for those using the NoScript browser extension: for the buttons to function correctly please allow Javascript from class.coursera.org and cloudfront.net to run in your browser. Note also that the "save answers" button does not function for this question and the next.

You entered:

Your Answer		Score	Explanation
x1 = 00000000000000000000000000000000 y1 = 00000000000000000000000000000000 x2 = 00000000000000000000000000000000 y2 = 00000000000000000000000000000000	✖	0.00	Don't cheat, you must enter distinct pairs !
Total		0.00 / 1.00	

Εικόνα 80:Ερώτηση 8-Week 3

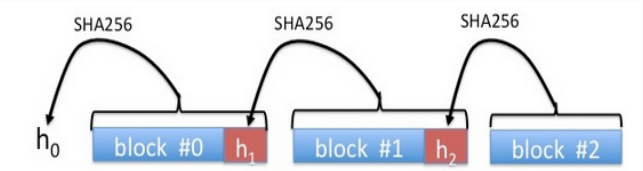
You submitted this homework on **Sat 23 May 2015 10:04 PM EEST**. You got a score of **1.00** out of **1.00**.

Question 1

Suppose a web site hosts large video file F that anyone can download. Browsers who download the file need to make sure the file is authentic before displaying the content to the user. One approach is to have the web site hash the contents of F using a collision resistant hash and then distribute the resulting short hash value $h = H(F)$ to users via some authenticated channel (later on we will use digital signatures for this). Browsers would download the entire file F, check that $H(F)$ is equal to the authentic hash value h and if so, display the video to the user.

Unfortunately, this means that the video will only begin playing after the *entire* file F has been downloaded. Our goal in this project is to build a file authentication system that lets browsers authenticate and play video chunks as they are downloaded without having to wait for the entire file.

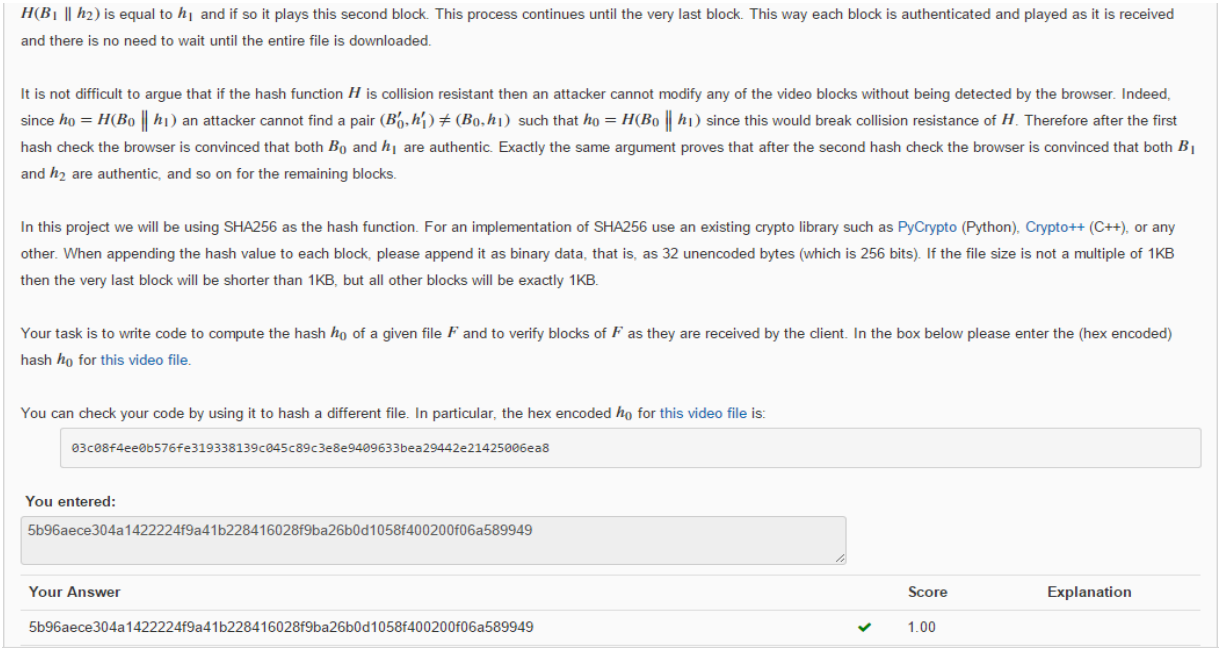
Instead of computing a hash of the entire file, the web site breaks the file into 1KB blocks (1024 bytes). It computes the hash of the last block and appends the value to the second to last block. It then computes the hash of this augmented second to last block and appends the resulting hash to the third block from the end. This process continues from the last block to the first as in the following diagram:



The final hash value h_0 – a hash of the first block with its appended hash – is distributed to users via the authenticated channel as above.

Now, a browser downloads the file F one block at a time, where each block includes the appended hash value from the diagram above. When the first block $(B_0 \,\|\, h_1)$ is received the browser checks that $H(B_0 \,\|\, h_1)$ is equal to h_0 and if so it begins playing the first video block. When the second block $(B_1 \,\|\, h_2)$ is received the browser checks that

Εικόνα 81:Προγραμματιστική άσκηση 3-a


$H(B_1 \| h_2)$ is equal to h_1 and if so it plays this second block. This process continues until the very last block. This way each block is authenticated and played as it is received and there is no need to wait until the entire file is downloaded.

It is not difficult to argue that if the hash function H is collision resistant then an attacker cannot modify any of the video blocks without being detected by the browser. Indeed, since $h_0 = H(B_0 \| h_1)$ an attacker cannot find a pair $(B_0', h_1') \neq (B_0, h_1)$ such that $h_0 = H(B_0 \| h_1)$ since this would break collision resistance of H. Therefore after the first hash check the browser is convinced that both B_0 and h_1 are authentic. Exactly the same argument proves that after the second hash check the browser is convinced that both B_1 and h_2 are authentic, and so on for the remaining blocks.

In this project we will be using SHA256 as the hash function. For an implementation of SHA256 use an existing crypto library such as PyCrypto (Python), Crypto++ (C++), or any other. When appending the hash value to each block, please append it as binary data, that is, as 32 unencoded bytes (which is 256 bits). If the file size is not a multiple of 1KB then the very last block will be shorter than 1KB, but all other blocks will be exactly 1KB.

Your task is to write code to compute the hash h_0 of a given file F and to verify blocks of F as they are received by the client. In the box below please enter the (hex encoded) hash h_0 for this video file.

You can check your code by using it to hash a different file. In particular, the hex encoded h_0 for this video file is:

03c08f4ee0b576fe319338139c045c89c3e8e9409633bea29442e21425006ea8

You entered:

5b96aece304a1422224f9a41b228416028f9ba26b0d1058f400200f06a589949

Your Answer		Score	Explanation
5b96aece304a1422224f9a41b228416028f9ba26b0d1058f400200f06a589949	✔	1.00	

Εικόνα 82:Προγραμματιστική άσκηση 3-b

Πίνακας 3:Βαθμολογίες-Week 3

	1η προσπάθεια	2η προσπάθεια	3η προσπάθεια	4η προσπάθεια
Ερώτησεις	7.38/10.00	-	-	-
Άσκηση	1.00/1.00	-	-	-

ΚΕΦΑΛΑΙΟ 6
ΑΥΘΕΝΤΙΚΗ ΚΡΥΠΤΟΓΡΑΦΗΣΗ

6.1 Επιθέσεις CPA και ασφάλεια κρυπτογράφησης

Σε αυτή την ενότητα θα προσπαθήσουμε να συνδυάσουμε την εμπιστευτικότητα και την ακεραιότητα για να πάρουμε ασφαλή σχήματα τα οποία είναι ανεκτικά σε injecting και tamperingεπιθέσεις.Ας δούμε πρώτα ένα παράδειγμα για το πώς ο επιτιθέμενος μπορεί να «πειράξει»το δίκτυο.

Παράδειγμα με TCP/IP

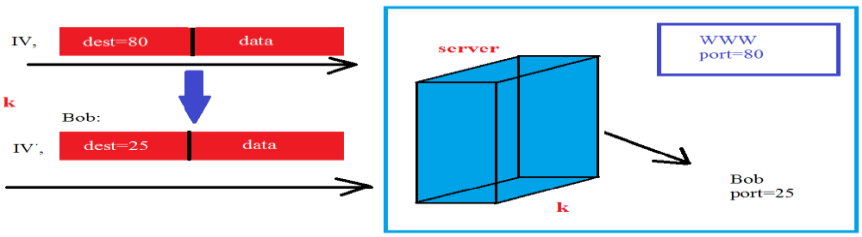
Έστω 2 μηχανές που επικοινωνούν μέσω του πρωτοκόλλου TCP/IP[53].Η μια μηχανή χρησιμοποιείται από τον χρήστη ο οποίος βρίσκεται σπίτι του, ενώ η άλλη μηχανή χρησιμεύει ως server.Ο serverέχει μια στοίβα(stack) η οποία δέχεται τα πακέτα απο τον χρήστη.Ο χρήστης βάζει σε κάθε πακέτο ένα πεδίο προορισμού το οποίο κοιτά ο serverκαι έτσι τα προωθεί στο κατάλληλο μέρος.Έστω επίσης ότι ο serverακούει στην θύρα 80 ενώ ο Bobαπο την πλευρά του serverακούει στην θύρα 25.

Οταν ένα πακέτο έρχεται απο τον χρήστη στον server,η στοίβα κοιτάει στο πεδίο προορισμού και την θύρα προορισμού του πακέτου, η οποία σε αυτήν την περίπτωση είναι η 80 και συνέχεια το προωθεί σε έναν webserver.Εαν το πακέτο είχε ώς θύρα προορισμού την 25, τότε η στοίβα θα προωθούσε το πακέτο στον Bob.Έστω το πρωτόκολλο ασφάλειας που ονομάζεται IPsec[54], το οποίο κρυπτογραφεί αυτα τα IPπακέτα μεταξύ του αποστολέα και του παραλήπτη.Ο αποστολέας και ο παραλήπτης

[53] Το πρωτόκολλο TCP/IP , http://el.wikipedia.org/wiki/TCP/IP
[54] Το πρωτόκολλο IPsec, http://en.wikipedia.org/wiki/IPsec

μοιράζονται το ίδιο κλειδί κα όταν ο αποστολέας στέλνει τα πακέτα αυτά κρυπτογραφούνται με αυτό το κλειδί.

Οταν το πακέτο φτάνει στον προορισμό δηλαδή στον server, η στοίβα θα αποκρυπτογραφήσει το πακέτο και μετά θα κοιτάξει το πεδίο προορισμού και θα προωθήσει το πακέτο κατάλληλα.Τώρα θα σας δείξω ότι χωρίς ακεραιότητα δεν θα μπορέσουμε να λάβουμε κανένα είδος εμπιστευτικότητας.Έστω ότι ένας επιτιθέμενος που διακόπτειένα πακέτο που προορίζεται για τον webserver άρα και για την θύρα 80(**Εικόνα 84)**



Εικόνα 83:Αλλαγή προορισμού ενός πακέτου με χρήση tampering

Ουσιαστικά ο επιτιθέμενος μπορεί να λάβει την κρυπτογράφηση οποιουδήποτε πακέτου το οποίο προορίζεται για την θύρα 25 διότι η στοίβα θα αποκρυπτογραφήσει τα πακέτα για την θύρα 25 και θα τα στείλει στον Bob(επιτιθέμενος).

Αυτό που θα κάνει ο Bobείναι να διακόπτει αυτό πακέτο πρωτού αυτό φτάσει στον webserverκαι θα προσπαθήσει να το αλλοιώσει ώστε αντί το πακέτο να έχει τον προορισμό την θύρα 80, τώρα θα έχει προορισμό την θύρα 25.Αρα ο Bobαπλά αλλάζοντας τον προορισμό ενός πακέτου, μπορεί να διαβάσει δεδομένα τα οποία **δένπροορίζονταν για αυτόν**.

Για παράδειγμα έαν το μήνυμα κρυπτογραφήθηκε με CBCκαι ένα τυχαίο IV(CPAsecurescheme) θα σας δείξουμε ότι για τον επιτιθέμενο είναι πολύ εύκολο να αλλάξει το ciphertextκαι να φτιάξει ένα νέο ciphertextόπου ο προορισμός του θα είναι η θύρα 25.Το μόνο πράγμα που θα αλλαξει ο επιτιθέμενος είναι το πεδίο IV και όλα τα υπόλοιπα θα παραμείνουν τα ίδια.Σύμφωνα με το CBCτο πρώτο plaintextμπλόκ του μηνύματος που θέλει να τροποποιήσει ο επιτιθέμενος είναι απλά η αποκρυπτογράφηση με το κλειδί k, το πρώτου ciphertextκαι το αποτέλεσμα αυτού μια XORπράξη μεταξύ του IV.Με συμβολισμούς έχουμε :

$$\mathbf{m[0]=D(k,c]0])\oplus IV=\text{¨}dest=80\text{¨}}$$

Πώς θα αλλάξαμε το IVώστε το μήνυμα να προορίζεται για την θύρα 25;Αυτο που θα μπορούσαμε να κάνουμε είναι να αντικαταστήσουμε το IVμε ένα νέο $IV'=IV\oplus(..80..)\oplus(..25...)$έτσι θα ακυρώσουμε τον προορισμό για την θύρα 80 και θα δωθεί ως νέος προορισμός η θύρα 25.**Η κρυπτογράφηση CPAπαρέχει εμπιστευτικότητα μόνο έαν ο επιτιθέμενος χρησιμοποιεί επιθέσεις eavesdroppingκαι δεν μπορεί να αλλάξει το ciphertextενός μηνύματος.**

6.2 Επιθέσεις σε επιλεγμένα κρυπτογραφήματα

Το σύστημα **αυθεντικής κρυπτογράφησης** (authenticatedencryption)αποτελείται απο έναν αλγόριθμο κρυπτογράφησης που παίρνει ως όρισματα ένα κλειδί, ένα μήνυμα

και προαιρετικά ένα noncekαι μας βγάζει ώς έξοδο ένα ciphertextκαιαπό εναν αλγόριθμο αποκρυπτογράφησης που βγάζει ως έξοδο το μήνυμα και ένα σύμβλο το οποίο ονομάζεται «bottom».Οταν ο αλγόριθμος αποκρυπτογράφησης βγάζει ως έξοδο το σύμβολο «bottom»τότε το ciphertextείναι λανθασμένο και αγνοείται.Με συμβολισμούς έχουμε :

$$\mathbf{E: K\times M\times N\rightarrow C}$$
$$\mathbf{D: K\times C\times N\rightarrow M\cup\{\perp\},\ \perp\notin M}$$

Ένα σύστημα αυθεντικής κρυπτογράφησης για να είναι είναι ασφαλές, πρέπει το σύστημα να ικανοποιεί δύο ιδιότητες.Η **πρώτη** ιδιότητα μας λέει ότι πρέπει το σύστημα να σημασιολογικά ασφαλές κάτω απο επιθέσεις CPA ενώ η **δεύτερη** ιδιότητα μας λέει ότι ο επιτιθέμενος δεν θα πρέπει να μπορεί να δημιουργήσει νέα ciphertextsτα οποία μπορούν να χρησιμοποιηθούν για την αποκρυπτογράφηση άλλων

Έστω ο (E,D) να είναι ενας αλγόριθμος με χώρο μηνυμάτων το M.Ο διεκδικητής διαλέγει ένα τυχαίο κλειδί K και ο αντίπαλος υποβάλλει μηνύματα της επιλογής του για να πάρει την κρυπτογράφηση των μηνυμάτων αυτών

Το c_1είναι κρυπτογράφηση του μηνύματος m_1, οπου το m_1επιλέχτηκε απο τον αντίπαλο.Ο αντίπαλος συνεχίζει την ίδια ακριβώς διαδικασία μέχρι να πάρει την κρυπτογράφηση όλων των μηνυμάτων(qστο σύνολο).Ο σκοπός του τώρα είναι **να παράγει μερικά νέα ciphertextsτα οποία θα είναι έγκυρα**.Θα λέμε ότι ο αντίπαλος κερδίζει ο παιχνίδι εάν το νέο ciphertextπου δημιούργησε θα αποκρυπτογραφεί σωστά, κάτι διαφορετικό απο το «bottom».

Τώρα θα **ορίσουμε** το πλεονέκτημα του αντιπάλου και την**ακεραιότητα** του ciphertextσαν την πιθανότητα του διεκδικητή να βγάλει ώς έξοδο 1 στο τέλος του παιχνιδιούκαι μετά θα δώσουμε **τον ορισμό της αυθεντικής κρυπτογράφησης**:

Ορισμός

Ένα (E,D) έχει ciphertext ακεραιότητα αν για όλους τους A ισχύει ότι:

$$\mathbf{Adv_{CL}[A,E]=P[Chal.outputs\ 1]\ είναι\ αμελητέο.}$$

Ορισμός αυθεντικής κρυπτογράφησης

Ένας αλγόριθμος (E,D) μας παρέχει αυθεντική κρυπτογράφηση(AE) έαν έχει σημασιολογική ασφάλεια κάτω απο επιθέσεις CPA και έχει ciphertext ακεραιότητα

Θα εισάγουμε και 2 ακόμη έννοιες.Η πρώτη έννοια ονομάζεται **αυθεντικότητα**(authenticity) η οποία σημαίνει ότι **ένας επιτιθέμενος δεν μπορεί να κοροιδέψει τον παραλήπτη Bobμε το να νομίζει ότι ένα μήνυμα στάλθηκε απο την Alice, το οποίο δεν το έστειλε όμως αυτή**.Για να συμβεί αυτό ένας επιτιθέμενος προσπαθεί να επικοινωνήσει με την Aliceαπλά στέλνοντας μηνύματα σε αυτήν και ζητώντας να κρυπτογραφήσει τα μηνύματα αυτά.Στη συνέχεια ο σκοπός του επιτιθέμενο είναι να παράγει κάποια νέα ciphertextsτα οποία ουσιαστικά δεν δημιουργήθηκαν απο την Alice.Αυτό όμως δεν μπορεί να συμβεί.

Οταν ο Bobλαμβάνει τα μηνύματα και τα αποκρυπτογραφεί επιτυχώς χρησιμοποιώντας έναν αλγόριθμο αποκρυπτογράφησης, γνωρίζει ότι τα μηνύματα ήρθαν απο κάποιον ο οποίος γνωρίζει το μυστικό κλειδί.Συγκεκριμένα εάν η Alice μόνο γνωρίζει το μυστικό κλειδί, τότε ο Bobξέρει ότι το ciphertextπου έλαβε ήρθε

πραγματικά απο την Alicekαι δέν είναι καποιο τροποποιημένο μήνυμα που στάλθηκε απο τον επιτιθέμενο.

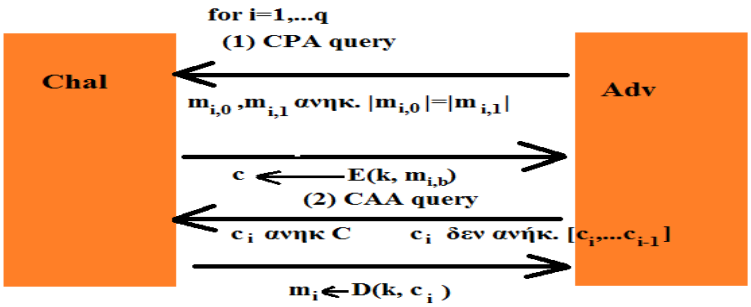
Το μόνο κακό είναι ότι η αυθεντικήκρυπτογράφηση δέν μας παρέχει ασφάλεια πάνω στίς **επιθέσεις επανάληψης**[55](replayattacks).Για παράδειγμα η Alicεμπορεί να στείλει ένα μήνυμα στον Bobλέγοντας «Μετέφερε 100$ στον Charlie» και στην συνέχεια ο Charlie θα μπορούσε να επαναλάβει(replay) το ciphertextμε αποτέλεσμα ο Bobστείλει άλλα 100$ στον Charlie

Η **δεύτερη έννοια** αφορά, την άμυνα ενάντια στην**επίθεση επιλεγμένων ciphertexts**[56](chosenciphertextattacks).Για να την περιγράψουμε θα χρησιμοποιήσουμε ένα πείραμα.Ορίζουμε την δύναμη του επιτιθέμενου στο να μπορεί να πράττει **CPA**και **CCA**επιθέσεις.Με άλλα λόγια ο επιτιθέμενος μπορεί να πάρει την κρυπτογράφηση οποιαδήποτε μηνύματος θέλει και μπορεί να κρυπτογραφήσει οποιδήποτε ciphertextης επιλογής του.**Ο σκοπός του είναι να σπάσει την σημασιολογική ασφάλεια**

Έστω ένας αλγόριθμος (E,D) που ορίζεται στο (K,M,C).Θα ορίσουμε και δύο πειράματα το πείραμα 1 και το πείραμα 0.Ο διεκδικητής(challenger) θα επιλέξει ένα τυχαίο κλειδί και ο αντίπαλος θα αρχίσει αν υποβάλλει ερωτήματα σε αυτόν.Τα ερωτήματα που μπορεί να υποβάλλει είναι δύο ειδών.Θα είναι είναι ένα CPAερώτημα ή ένα CCAερώτημα.

Οσο αφορά το CPAο επιτιθέμενος υποβάλλει δύο μηνύματα(**Εικόνα85**) m_0,m_1τα οποία έχουν το ίδιο μέγεθος και λαμβάνει την κρυπτογράφηση του m_0 (έαν είμαστε στο πείραμα 0)είτε την κρυπτογράφηση του m_1(εάν είμαστε στο πειραμα 1)

Το δεύτερο είδος ερωτήματος είναι πιό ενδιαφέρον.Ο επιτιθέμενος υποβάλλει ένα ciphertextης επιλογής του και αυτού που παίρνει πίσω, είναι η κρυπτογράφηση αυτού του ciphertext.Επομένως ο επιτιθέμενος μπορεί να αποκρυπτογραφήσει όσα ciphertextθέλει αυτός με την προυπόθεση τα ciphertextsνα μήν είναι αυτά που έλαβε απο τα CPAερωτήματα.



Εικόνα 84:CPA και CCA επιθέσεις

Εάν ο επιτιθέμενος μπορούσε να χρησιμοποιήσει τα ciphertextsπου έλαβε απο τα CPAερωτήματα στα CCAερωτήματα τότε αυτό δέν θα ήταν δίκαιο.Εαν ο επιτιθέμενος έπαιρνε ένα ciphertextτο οποίο το έλαβε κάνονταςCPAερώτημα τότε αυτό θα ήταν η

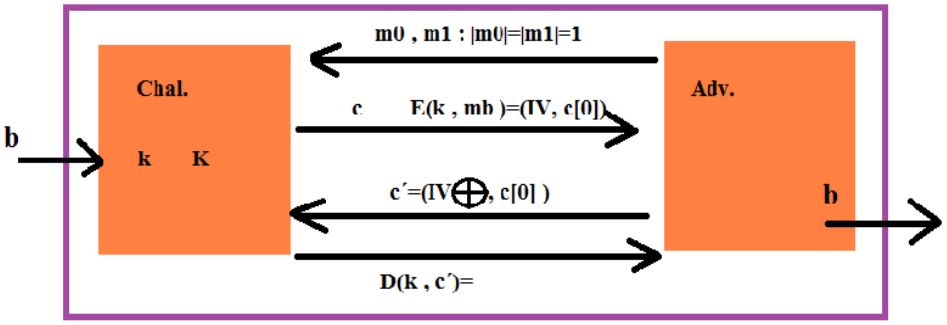
[55] Επιθέσεις επανάληψης, http://en.wikipedia.org/wiki/Replay_attack
[56] Επίθεσεις επιλεγμένων ciphertexts, http://en.wikipedia.org/wiki/Chosen-ciphertext_attack

κρυπτογράφηση του m_0ή του m_1 και έαν υποβάλλει CCAερώτημα για το συγκεκριμένο ciphertexttότε θα πάρει ώς απάντηση την κρυπτογράφηση m_0είτε του m_1, άρα θα είναι σε θέση να γνωρίζει απο ποίο πείραμα πάρθηκε το ciphertext.Επομένως ο σκοπός του είναι να μάθει έαν βρίσκεται στο πείραμα ή στο πείραμα 0.

Ο αλγόριθμος CCAείναι ασφαλής έαν ο αντίπαλος συμπεριφέρεται με τον ίδιο τρόπο στο πείραμα 1 και στο πειραμα 0.Με άλλα λόγια δέν μπορεί να διακρίνει το πείραμα 1 απο το πείραμα 0.Με συμβολισμούς έχουμε:

$$\mathbf{Adv_{CCA}[A,E]=|P[EXP(0)=1-P[EXP(1)]=1|\ είναι\ αμελητέο}$$

Για παράδειγμα,**ενα CBC με τυχαίο IV δεν είναι ασφαλές σε επιθέσειςCCA**.Για να το αποδείξουμε αυτό θα θεωρήσουμε το παρακάτω παράδειγμα.Ο επιτιθέμενος υποβάλλει τα μηνύματα m_0και m_1που έστω ότι έχουν μέγεθος όσο ένα μπλόκ και είναι διαφορετικά μεταξύ τους(**Εικόνα 86**).Αυτο που θα λάβει πίσω είναι η CBCκρυπτογράφηση του m_0 ή του m_1.Ο επιτιθέμενος τώρα θα προσπαθήσει να τροποποιήσει αυτό το ciphertextc, που έλαβε αλλάζοντας το IV κάνοντας πράξηXOR μεταξύ της τιμής IVκαι του 1.Οπότε το νέο ciphertextc΄ είναι διαφορετικό απο το ciphertextc.Στη ο επιτιθέμενος υποβάλλει το ciphertextc΄ με σκοπό να λάβει την κρυπτογράφηση του.



Εικόνα 85:CBCμε τυχαίο IV δεν είναι ασφαλές σε επιθέσεις CCA

Αυτό που θα λάβει πίσω είναι $\mathbf{D(k,c')=m_b\oplus 1}$.Επομένως έλαβε ουσιαστικά $m_0\oplus 1$ ή $m_1\oplus 1$ αρα μπορεί να διακρίνει σε ποιό πείραμα αυτός βρίσκεται άρα το πλεονέκτημα του είναι 1 .Το ζήτημα είναι πώς μπορούμε να σχεδιάσουμε κρυπτοσυστήματα τα οποία να είναι ασφαλή κάτω απο επιθέσεις CCA.

Θεώρημα CCA ασφάλειας

Έαν μας ένας αλγόριθμος (E,D) ο οποίος μας παρέχει authenticated κρυπτογράφηση τότε ο αλγόριθμος αυτός είναι ασφαλής σε επιθέσεις CCA.Εαν δηλαδή υπάρχει αντίπαλος ο οποίος υποβάλλει qστο σύνολο ερωτήματα τότε υπάρχουν δύο αντίπαλοι B_1B_2 που ικανοποιούν την παρακάτω ανισότητα:

$$\mathbf{Adv_{CCA}[A,E]\leq 2\cdot q\cdot Adv_{CL}[B_1,\ E]+Adv_{CPA}[B_2,E]}$$

Η ποσότητα $Adv_{CPA}[B_2,E]$ είναι αμελητέα επειδή είναι ασφαλής σε CPAεπιθέσεις.Επίσης η ποσότητα $Adv_{CL}[B_1,E]$ είναι αμελητέα επειδή έχει ciphertextακεραιότητα.Αρα και οι δύο όροι είναι αμελητέοι.Επομένως το πλεονέκτημα του επιτιθέμενου που πράττει CCAείναι επίσης αμελητέο.

6.3 Αυθεντικά συστήματα κρυπτογράφησης(MAC-PRP)

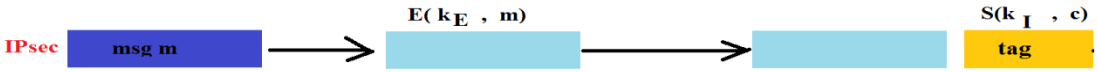
Σε αυτήν την ενότητα θα προσπαθήσουμε να κατασκευάσουμε αυθεντικά(authenticated)συστήματα κρυπτογράφησης.Μιας και έχουμε ασφαλή κρυπτογράφηση για επιθέσεις CPAκαι ασφαλή MAC, γεννιέται ένα πολύ καλό ερώτημα.Το ερώτημα αυτό αφορά το πώς θα συνδιάσουμε αυτές τις δύο έννοιες με σκοπό να κατασκευάσουμε μια αυθεντικήκρυπτογράφηση.Ας δούμε μερικούς συνδυασμούς που έχουν ασφάλεια CPA-MACοι οποίοι εισήχθησαν απο διαφορετικά πρότζεκτ.

Θα αναφερθούμε σε 3 απο όλα αυτά τα πρότζεκτ και επιπρόσθετα και στα 3 αυτά παραδείγματα που θα παρουσιάσουμε παρακάτω υπάρχειξεχωριστο κλειδί k_Eγια την κρυπτογράφηση των κάθε ένα απο αυτα και τέλος ένα ξεχωριστό κλειδί για MACing.Τα δύο αυτά κλειδιά είναι ανεξάρτητα μεταξύ τους.

Το πρώτο μας παράδειγμα είναι το πρωτόκολλο **SSL**[57].Ο τρόπος με τον οποίο το SSLσυνδυάζει κρυπτογράφηση και MACγια να πετύχει αυθεντικότητα είναι η παρακάτω.Αρχικά παίρνουμε ένα plaintextmκαι υπολογίζουμε την ετικέτα του μηνύματος χρησιμοποιώντας το κλειδί k_Iγια τον υπολογισμό της ετικέτας.Έπειτα μπορώ να θέσω την ετικέτα σε μια αλληλουχία με mκαι στη συνέχεια να εφαρμόσω κρυπτογράφηση και θα πάρω το ciphertext(**Εικόνα 87**)

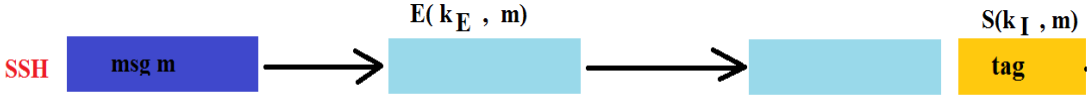
Εικόνα 86:Κρυπτογράφηση με SSL

Η δεύτερη επιλογή είναι το πρωτόκολλο **IPsec**.Αυτό που κάνει είναι να κρυπτογραφούμε πρώτα το μήνυμα με το κλειδί k_E και μετά υπολογίζω την ετικετα στο ciphertextπου έφτιαξα(**Εικόνα 88**)



Εικόνα 87:Κρυπτογράφηση με IPsec

Η τρίτη επιλογή ονομάζεται πρωτόκολλο **SSH**.Αρχικά κρυπτογραφώ το μήνυμα m χρησιμοποιώντας ένα ασφαλές CPAσχήμα κρυπτογράφησης και μετά εφαρμόζω σε αλληλουχία με την κρυπτογράφηση την ετικέτα στο μήνυμα μου.Η διαφορά με το IPsec είναι ότι η ετικέτα υπολογίζεται σε όλο το ciphertextενώ στο SSHη ετικέτα υπολογίζεται σε όλο το μήνυμα (**Εικόνα 89**)



Εικόνα 88:Κρυπτογράφηση με SSH

[57] Το πρωτόκολλο SSL, http://el.wikipedia.org/wiki/SSL

Αυτές οι τρείς επιλογές συνδυάζουν κρυπτογράφηση και MACκαι το ερώτημα εδω είναι ποιές απο τις παραπάνω επιλογές είναι ασφαλής;Ας ξεκινήσουμε με την επιλογή SSH.Οπώς παρατηρούμε η ετικέτα υπολογίζεται πάνω στο μήνυμα και το ciphertextέρχεται σε αλληλουχία με την ετικέτα.Αυτό όμως θεωρείται πρόβλημα διότι τα MACδέν παρέχουν εμπιστευτικότητα.Τα MACσχεδιάστηκαν για να παρέχουν ακεραιότητα.

Δεν είναι κακό οτι το MACσαν μέρος της ετικέτας εμφανίζει μερικά bitsαπο το plaintext.Το πρόβλημα είναι ότι αυτά τα bitsλογω της αλληλουχίας ετικέτας-ciphertextμπορούν να διαρρέυσουν πρός το ciphertextμε αποτέλεσμα το ασχήμα να μην έχει ασφάλεια σε CCAεπιθέσεις.Παρόλο θεωρείται γενικά ασφαλές καλύτερα να μήν το χρησιμοποιήσουμε.

Ας δούμε την επιλογή SSLκαι IPsec.Η προτιμώμενη μέθοδος είναι η IPsecεπειδή βγαίνει ώς συμπέρασμα ότι δέν μας αφορά τι CPAσύστημα και MACχρησιμοποιούμε αλλα ο συνδυασμός CPAασφαλούς συστήματος και MACμας παρέχει αυθεντική κρυπτογράφηση.Μόλις κρυπτογραφήσουμε το μήνυμα, θα υπολογίσουμε την ετικέτα του ciphertext με αποτέλεσμα ουσιαστικά κανείς να μήν μπορεί να τροποποιήσει το ciphertext!

Επομένως κανείς δεν θα μπορεί να παράγει ένα διαφορετικό ciphertextο οποίο να μοιάζει έγκυρο.Οσο αφορά το SSLχρησιμοποιούμε CCAασφαλή κρυπτογράφηση με MAC και έτσι είναι ευάλωτο σε επιθέσεις επιλεγμένων ciphertexts και δέν παρέχει αυθεντική κρυπτογράφηση.

<u>Διαφορετικές ονομασίες</u>

Η SSLμέθοδος συνήθως ονομάζεται και «MAC-then-encrypt».Ημέθοδος IPsecονομάζεται και «encrypt-then-MAC» ενώ η μέθοδος SSHονομάζεται «encrypt-and-MAC».

ΈστωέναςασφαλήςCPAαλγόριθμοςκαιένας (S,V)MAC.Τότεηδιαδικασία «**πρώτακρυπτογραφώκαιμεταχρησιμοποιώMAC**»(Encrypt-then-MAC) πάνταπαρέχει αυθεντική κρυπτογράφηση.Ηδιαδικασία «**πρώταχρησιμοποιώMACκαιμετάκρυπτογραφώ**»(MAC-then-encrypt) ίσωςναμηνείναιασφαλήςσεεπιθέσειςCCA.Έανόμωςένα (E,D) είναι ένα τυχαίοCTRήένατυχαίοCBCτότεηδιαδικασίαMAC-then-encryptμαςπαρέχειαυθεντικήκρυπτογράφηση.

Καθώς η αυθεντικήκρυπτογράφηση άρχισε να γίνεται γνωστή, ένας αριθμός προσεγγίσεων οι οποίες συνδίαζαν κρυπτογράφηση και MAC άρχισαν να εμφανίζονται καιέγιναν standards απο την NIST.Εδώ θα αναφέρουμε 3 προσεγγίσεις εκ των οποίων τα δύο έγιναν standardsστην NISTκαι ονομάζονται**GCM**(galoiscountermode) και **CCM**(CBCcountermode).

Ο GCMχρησιμοποιεί έναν μετρήτη για την διαδικασία της κρυπτογράφησης μαζί με τον Cartel-WegmanMAC.Ο τρόπος με τον οποίο ο Cartel-Wegmanδουλεύει στο CBCείναι πολύ απλός.Ουσιαστικά ο Cartel-Wegmanείναι μια συνάρτηση κατακερματισμου πάνω στο μήνυμα.Το αποτέλεσμα αυτου κρυπτογραφείται με τη χρήση μιας PRF.

Το**CCSM**είναιέναακόμηNISTstandardπου χρησιμοποιείCBCMACκαιμετάcountermodeκρυπτογράφηση.Τέλος υπάρχει το **EAX**το οποίο χρησιμοποιεί countermodeκρυπτογράφηση και μετά CMAC.Επίσης όλα αυτα τα standardsβασίζονται σε noncesάρα τα ζεύγη κλειδί-nonceδεν θα πρέπει να επαναλαμβάνονται.

Ολα τα παραπάνω modes-προσεγγίσειςονομάζονται «**authenticatedencryptionwithassociateddata»**(AEAD)και είναι μια επέκταση της αυθεντικής κρυπτογράφησης.Η ιδέα πίσω απο το AEADείναι το γεγονός ότι η διαδικασία κρυπτογράφησης δέν θα κρυπτογραφεί ολόκληρο το μήνυμα μας αλλα μόνο ένα μέρος του και φυσικά θα παρέχει αυθεντικότητα στο μήνυμα μας.

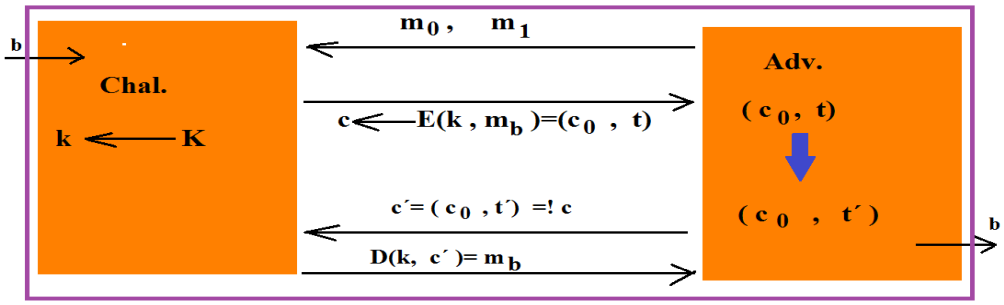
Ένα καλό παράδειγμα είναι ένα πακέτο δικτύου το οποίο έχει μια επικεφαλίδα(header) και ένα ωφέλιμο φορτίο(payload).Τυπικά η επικεφαλίδαδεν θα κρυπτογραφηθεί διότι συνήθως περίεχει το προορισμό ένος πακέτου και τα ρούτερ δέν θα γνωρίζουν πού να στείλουν το συγκεκριμένο πακέτο.Μπορεί όμως η επικεφαλίδα να έχει αυθεντικότητα;Αυτό ακριβώς κανουν τα modestou AEAD.

Παρέχουν αυθεντικότητα στην επικεφαλίδακαι κρυπτογράφηση στο ωφέλιμο φορτίο του πακέτου.Τώρα που πλέον έχουμε εξηγήσει τί είναι το «encrypt-then-MAC» μπορούμε να γυρίσουμε πίσω στον ορισμό της ασφάλειας ενός MACγια να εξηγήσουμε ακόμη μία έννοια.

Μια απο τις απαιτήσεις που είχαμε απο τον ορισμό του MAC ήταν ότι για ένα δοσμένο ζευγάρι μηνύματος-MACο επιτιθέμενος να μήν μπορεί να παράγει και άλλη ετικέτα για το ίδιο μήνυμα m.Ακόμη και ο επιτιθέμενος να είχε την ετικέτα για το μήνυμα m, να μήν είναι ικανός να παράγει νέα ετικέτα για το ίδιο μήνυμα m.Γιατί όμως αυτό είναι τόσο σημαντικό;Είναι σημαντικό διότι αλλίως δέν θα είχαμε ακεραιότητα ciphertext.Ας δούμε τι μπορεί να πάει στραβά έαν δέν ίσχυε η απαίτηση αυτή απο τον ορισμό του MAC.

Ο επιτιθέμενος ξεκινάει στέλοντας δύο μηνύματα(**Εικόνα 90**) και λαμβάνει ώς συνήθως την κρυπτογράφηση m_0 ή την κρυπτογράφηση του m_1μέσω του c.Μιας και χρησιμοποίουμε την μέθοδο «encrypt-then-MAC» ο επιτιθέμενος λαμβάνει το ciphertext (έστω c_0) και ένα MACπάνω στο ciphertextc_0.Όπως είπαμε πρίν έαν μας δώσουν ένα MACενός μηνύματος ο επιτιθέμενος μπορεί να παράγει άλλο MAC πάνω στο ίδιο μήνυμα.

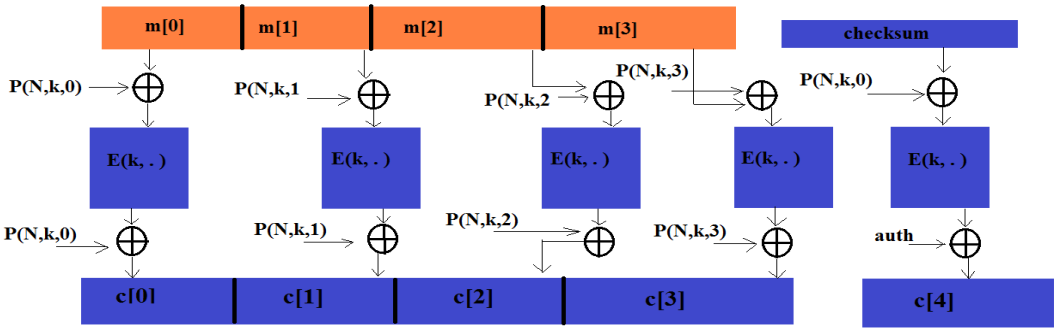
Αρα θα παράγει ένα νέο MACπάνω στο ciphertextc_0.Τώρα ο επιτιθέμενος θα υποβάλλει ένα ερώτημα επιλεγμένου ciphertextμέσω ενός c΄ το οποίο είναι έγκυρο ciphertext, επειδή είναι διαφορετικό απο το c.Επομένως ο διεκδικητής θα πρέπει να αποκρυπτογραφήσει αυτό το ciphertextκαι θα στείλει πίσω στον επιτιθέμενο την κρυπτογράφηση του c΄ που είναι το μήνυμα m_b.



Εικόνα 89:Η εξήγηση μιας απαίτησης στον ορισμό του MAC

Πρωτού εμφανιστεί η έννοια της αυθεντικής κρυπτογράφησης ο κάθε ένας συνδύαζε MACκαι κρυπτογράφηση με πολλούς και διάφορους τρόπους με σκοπό να πετύχει κατα κάποιον τρόπο αυθεντική κρυπτογράφηση.Μετα την εμφάνιση της αυθεντικής κρυπτογράφησης σκέφτηκαν ότι για να πάρουν αυθεντική κρυπτογράφηση πιό αποδοτικά, θα πρέπει να συνδυάσουν έναν MACμε ένα σχήμα κρυπτογράφησης.Για αν σκεφτούμε πώς αυτό το γεγονός λειτουργούσε ας πούμε ότι συνδυάζουμε countermodeμε CMAC, μετα για κάθε μπλόκ απο το plaintextπρέπει να χρησιμοποιήσουμε έναν αλγόριθμο τμήματος για το countermodeκαι μετα να χρησιμοποιήσουμε τον αλγόριθμο ξανά, για το CBC-MAC.

Αυτό σημαίνει οτι έαν συνδυάσουμε ασφαλή CPAκρυπτογράφηση με έναν MAC, τότε για κάθε μπλόκ απο το plaintextθα πρέπει να εφαρμόσουμε τον αλγόριθμο τμήματος δύο φορές.Μπορούμε άραγε να κατασκευάσουμε αυθεντική κρυπτογράφηση απο μια PRP.Αυτο μπορεί να γίνει με την κατασκευή που ονομάζεται **OCB**[58]**(Εικονα 91)**.



Εικόνα 90:Η λειτουργία του OCBmode

Κάθε μπλόκ κρυπτογραφείται ανεξάρτητα απο τα άλλα μπλόκ του μηνύματος.Τέλος το αποτέλεσμα της τελευταίας PRPμαζί με το checksumτου μηνύματος πηγαίνει σε μια XORκαι μετά το αποτέλεσμα αυτής κρυπτογραφείται και μας δίνει την ετικέτα του μηνύματος μας.

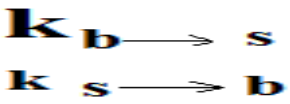
6.4 Το πρωτόκολλο TLSrecord

Σε αυτήν την ενότητα θα δούμε το πώς η αυθεντική κρυπτογράφηση εφαρμόζεται στον πραγματικό κόσμο.Θα αναφερθούμε στο **TLS**πρωτόκολλο.Θα δούμε αρχικά ένα παράδειγμα λειτουγίας του.Σε αυτό το πρωτόκολλο κάθε TLSrecordξεκινάει με μια

[58] Η λειτουργία του OCBmode, http://en.wikipedia.org/wiki/OCB_mode

επικεφαλίδα(HDR) ακολουθούμενη απο κρυπτογραφημένη πληροφορία η οποία στάλθηκε απο την μία πλευρά στην άλλη.

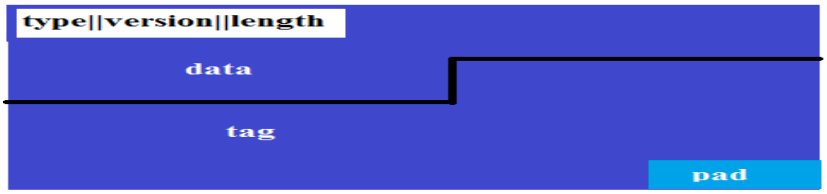
Συνήθως τα recordsείναι μεγέθους 16 KB.Το TLSχρησιμοποιεί αυτα που ονομάζουμε **κλειδιά μονής κατευθύνσης(**unidirectionalkeys) δηλαδή υπάρχει ένα κλειδί απο τον φυλλομετρητή(browser) στον σέρβερ και ένα διαφορετικό κλειδί απο τον σέρβερ στον φυλλομετρητή.Αυτά τα κλειδια γεννιούνται μέσω του **TLSexchange**πρωτοκόλλου.Άς θεωρήσουμε οτι ο σερβερ και ο φυλλομετρητής έχουν έτοιμα τα κλειδία τους.Ο συμβολισμός των κλειδίων έχει ώς εξής(**Εικόνα 92**).



Εικόνα 91:Κλειδιά μονής κατεύθυνσης

Το πρωτόκολλο TLSχρησιμοποιεί αυτό που λέμε statefullκρυπτογράφηση που αυτό σημαίνει ότι η κρυπτογράφηση του κάθε πακέτου γίνεται χρησιμοποιώντας συγκεκριμένη κατάσταση(state) η οποία διατηρείται μέσα στον φυλλομετρητή και τον σέρβερ.Για εμάς η κατάσταση(state) είναι 2 μετρητές μεγέθους 64 bits.Ο ενας για για το trafficαπο τον φυλλομετρητή προς τον σέρβερ και ένας μετρητής για το trafficαπο τον σέρβερ προς τον φυλλομετρητή.

Αρχικά αυτοι οι μετρητές αρχίζουν απο την τιμή 0 και σε κάθε recordπου στέλνεται αυξάνονται κατα 1.Ο σκοπός τους είναι να αποτρέπουν τις επιθέσεις επανάληψης(replayattacks).Τώρα θα δούμε λεπτομερώς το πώς λειτουργέι το TLSrecordπρωτόκολλο το οποίο χρησιμοποιεί κρυπτογράφηση με την χρήση του AES 128-CBCκαι δημιουργία ετικέτας με τη χρήση HMAC-SHA1.Ας δούμε το πώς ο φυλλομετρητής στέλνει δεδομένα στον σέρβερ(**Εικόνα 93**).



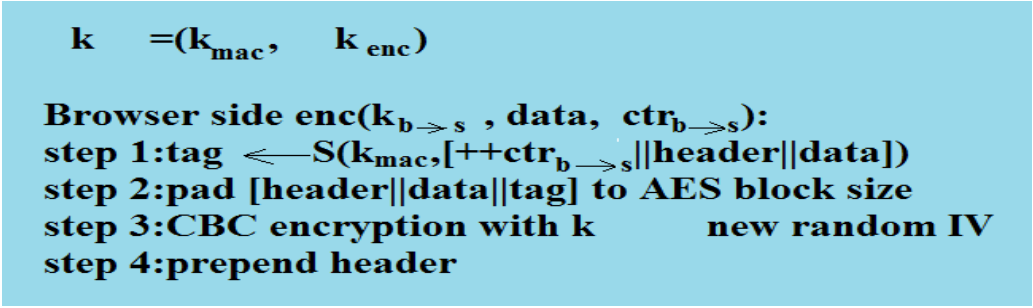
Εικόνα 92:TLSrecord

Το κλειδί φυλλομετρητή προς σέρβερ, έιναι φτιαγμένο απο ένα MACκλειδί και ένα κλειδί κρυπτογράφησης.Το κάθε recordδιαθέτει μια επικεφαλίδα η οποία περιέχει τον τύπο του πακέτου, την έκδοση του πρωτοκόλλου και το μήκος του πακέτου.Όταν κρυπτογραφούμε τα δεδομένα ενός record απο την πλευρά του φυλλομετρήτη τότε η κρυπτογράφηση γίνεται**en(k_{bs}, data, ctr_{bs})** σε 4 βήματα.Το πρώτο βήμα που κάνουμε είναι κάνουμε MACingτην ποσότητα :

$$tag \leftarrow S(k_{mac}, [++ctr_{bs} \,||\, header \,||data])$$

Το επόμενο βήμα είναι η ετικέτα να μπεί σε αλληλουχία με τα δεδομένα με σκοπό να ταιριάζουν σε μέγεθος ενός AESμπλόκ.Άρα η επικεφαλίδα, τα δεδομένα και η ετικέτα γίνονται paddingσε AESμπλόκ.Το αμέσως επόμενο βήμα είναι να εφαρμόσουμε CBCκρυπτογράφηση χρησιμοποιώντας το κλειδί k_{enc}και ένα νέο τυχαίο IV.Στο τέταρτο και τελευταίο βήμα στο οποίο απλα τοποθετούμε στο recordτην επικεφαλίδα, τον τύπο του TLSκαι το μέγεθος.

Αυτό το recordτώρα μπορούμε να το στείλουμε στον σέρβερ.Όπως μπορούμε να δούμε το TLSείναι ένα implementationτης μεθόδου «MAC-then-encrypt».Συνοπτικά τα βήματα κρυπτογράφησης μέσω του πρωτοκόλλου TLSφαίνονται στην παρακάτω εικόνα(**Εικόνα 94**)



Εικόνα 93:Κρυπτογράφηση με TLSrecord

Ας δούμε το πώς απο την πλευρά του σέρβερ το recordπρωτόκολλο αποκρυπτογραφεί το εισερχόμενο record.Με συμβολισμούς:

$$\mathbf{dec(k_{bs}, record, ctr_{bs})}$$

Τα βήματα αποκρυπτογράφησης είναι 4 οπώς και πρίν.Αρχικά αποκρυπτογραφούμε το recordχρησιμοποιώντας το k_{enc} .Μετα απο αυτό θα τσεκάρει το padformat και έαν δέν είναι σωστό απλα θα στείλει ενα μήνυμα προειδοποιήσης και θα τερματίσει τη σύνδεση του.Έαν το padformatείναι σωστό τοτε, το τελευταίο βήμα είναι να κάνει extractτην ετικέτα απο το recordκαι μετά θα επαληθεύσει topad για την επικεφαλίδα, τα δεδομένα και την τιμή του μετρητή c_{bs}

Εαν το MACδεν ειναι σωστό θα σταλθεί ένα μήνυμα προειδοποιήσης και θα τερματιστεί η σύνεδεση.Έαν όμως το padεπαληθευτεί τότε ο παραλήπτης θα αφαιρέσει την ετικέτα,την επικεφαλίδα και ότι μείνει είναι το plaintext.Έαν ένας επιτιθέμενος μπορούσε να πάρει αυτό το recordκαι το υποβάλλει στον σέρβερ μια άλλη χρονική στιγμή τότε η τιμή του μετρητή c_{bs} θα αλλάξει αρα και η ετικέτα του recordθα αλλάξει άρα δέν θα επαληθευθεί απο τον σέρβερ.

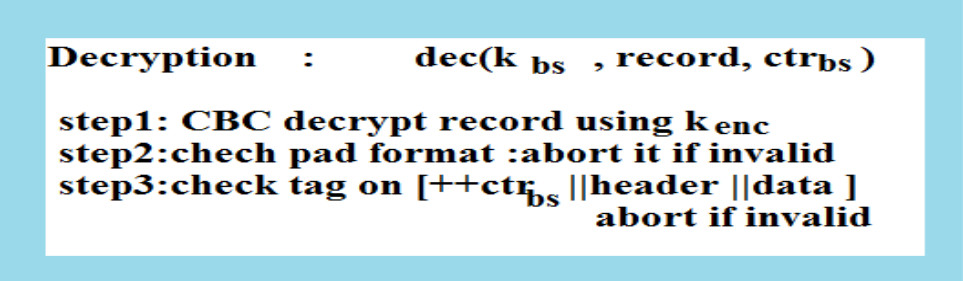
Αυτοί οι μετρητές είναι απαραίτητοι γιατί απλά αποτρέπουν τις επιθέσεις επανάληψης.Επίσης και οι δύο πλευρές γνωρίζουν τις τιμές των μετρητών με αποτέλεσμα να μήν χρειαστεί να σταλθεί η τιμή των μετρητών μέσα στο record.Το όλο σχήμα μας παρέχει αυθεντικότητα εφόσον δέν διαρέυσει πληροφορία κατα την αποκρυπτογράφηση.

6.5 Επιθέσεις CBCpadding

6.5.1 Επιθέση στο TLS πρωτόκολλο(CBC κρυπτογράφηση)

Στο TLSμπορούν να συμβούν δύο λάθη κατα την αποκρυπτογράφηση ενός record.Είτε paddingerroreίτε MACerror.Είναι πολύ σημαντικό για τον επιτιθέμενο να μήν ξέρει **ποιά απο τα δύο λάθη συνέβησαν**.Και θα εξηγήσουμε το γιατί.Υποθέστε ότι ο επιτιθέμενος μπορεί να ξεχωρίσει το ένα erroraπο το άλλο.Το αποτέλεσμα είναι αυτό που ονομάζουμε **paddingoracle**.Έστω οτι ο επιτιθέμενος έχει ένα ciphertextο οποίο το πήρε κατα την διάρκεια της μετάδοσης και θέλει να το αποκρυπτογραφήσει.

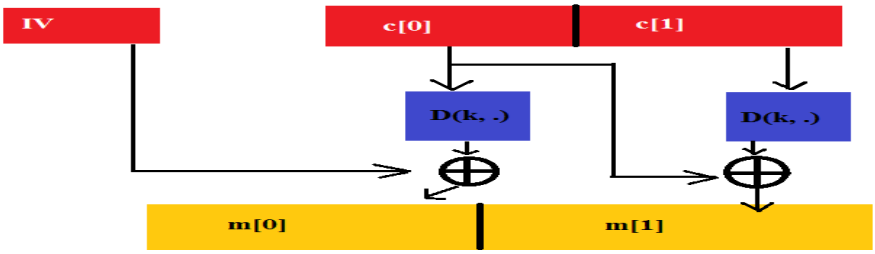
Αυτό που μπορεί να κάνει είναι να πάρει αυτό το ciphertextκαι να το υποβάλλει στον σέρβερ.Ο σέρβερ θα προσπαθήσει να αποκρυπτογραφήσει και θα κοιτάξει έαν το pad έχει correctformatpad.Έαν δέν έχει σωστό correctformatθα στείλει ένα είδος λάθους ενώ έαν είναι σωστό θα στείλει MACerror(**Εικόνα 95**).Σάν αποτέλεσμα ο επιτιθέμενος που υποβάλλει το ciphertextθα μπορεί να καταλάβει έαν τα τελευταία bytestου αποκρυπτογραφημένου ciphertexttέχουν έγκυρη ετικέτα ή όχι.Αυτό είναι ένα παράδειγμα που αφορά επιθέσεις επιλεγμένων ciphertexts.Και τώρα μένει να δούμε έαν μπορούμε να αποκρυπτογραφήσουμε **ολόκληρο το ciphertext**.



Εικόνα 94:Αποκρυπτογράφηση TLS

Έστω ότι έχουμε το ciphertext **c=(c[0],c[1], c[2])** και θέλουμε την αποκρυπτογράφηση του m[1](**Εικόνα 96)**.Αρχικά θα πετάξει το c[2] μιας και δέν το χρειάζεται.Έστω ότι έχει μια guess G για το τελευταίο byte του m[1].Ατό που θα κάνει ο επιτιθέμενος είναι να κάνει XORπράξη μεταξύ της τιμής g, της τιμής 0x001 και με το τελευταίο bytetου c[0]του προηγούμενου μπλόκ.

Τι θα γίνει έαν αυτα τα δύο μπλόκς c[0] και c[1] αποκρυπτογραφούνταν;Τοc[0] έαν είχε υποστεί αποκρυπτογράφηση θα μας έβγαζε κάτι που δέν θα μας ενδιέφερε, αλλά η αποκρυπτογράφηση του c[1] μας ενδιαφέρει.Όταν το c[1] αποκρυπτογραφείται το τελευταίο bytetου θα τροποποιηθεί απο το c[0](λόγω της XORπράξης που υπάρχει κάτω απο την αποκρυπτογράφηση) και σάν αποτέλεσμα το τελευταίο bytetου plaintextθα υποστεί XOR με την τιμήg⊕0x001.



Εικόνα 95:Η λειτουργία paddingoracle

Επομένως έαν η τιμή guessGείναι σωστή, τότε η τιμή last-bytekαι gθα αλληλοακυρωθούν με αποτέλεσμα να πάρουμε το τελευταίο bytetου plaintextπο είναι η τιμή 0x001.Εάν το guessμας για το τελευταίο byteείναι σωστό, τότε θα πάρουμε ένα padπου είναι wellformed άρα και έγκυρο.Εάν δέν μαντέψουμε σωστά τότε θα πάρουμε λανθασμένο padάρα και μή έγκυρο.Αυτό που θα κάνει ο επιτιθέμενος είναι να δημιουργήσει ένα τροποποιημένο ciphertext (IV,c΄[0], c[1]) στο οποίο τροποποιήσε το δέυτερο μπλόκ του.Μετα θα στείλουμε αυτο το ciphertextσε ένα paddingoracle και μετά απο τα αποτελέσμα του θα μάθει έαν το geίναι σωστό ή όχι.Αυτήν την διαδικασία μπορούμε να την επαναλάβουμε απο 0 εως 255 φορές για να μάθουμε το τελευταίο bytetου m[1].

Με αυτόν τον τρόπο μπορούμε να μάθουμε το πρότελευταιο bytetου m[1].Τι padθα χρησιμοποιήσουμε για να μάθουμε αυτό το πρότελευταιο byte;Αντί να χρησιμοποιήσουμε το padπου περιέχει το byte 1(0x001) θα χρησιμοποιήσουμε ένα padπου έχει μέγεθος 2 byte που περιέχει τις τιμές 2-2 και όχι 1 όπως πρίν.Μπορούμε να μάθουμε το προτελευταίο byte απλά δοκιμάζοντας πολλές τιμές gμέχρι να βρούμε αυτήν που ταιριάζει στο padχρησιμοποιώντας το 0202.Μπορούμε να επαναλάβουμε αυτήν την διαδικασία μέχρι να αποκρυπτογραφήσουμε ολόκληρο το ciphertext.Αφού το μέγεθος του μπλόκ είναι 16 bytes τότε μετα απο 16×256(queries) θα μάθουμε ολόκληρο το ciphertext.

6.5.2 IMAPoverTLS

Το **IMAP**[59]είναι ένα πολύ γνωστό πρωτόκολλο για διάβασμα εμαιλ απο ένα IMAPserverκαι είναι πολύ συνηθισμένο να το προστατεύουμε χρησιμοποιώντας TLSπρωτόκολλο.Κάθε 5 λεπτά ο IMAPclientθα συνδεθεί στον IMAPserverκαι θα ελέγχει έαν έχει έρθει κάποιο νέο εμαιλ.

"

Στη συνέχεια αφού συνδεθεί ελέγχει έαν έχει έρθει κάποιο εμαιλ.Αυτό σημαίνει ότι κάθε 5 λεπτά ο επιτιθέμενος παίρνει την κρυπτογράφηση ενός συγκεκριμένου μηνύματος παραδείγματος χάρη την κρυπτογράφηση του password επομένως κάθε 5 λεπτά μπορεί να εφαρμόσει ένα guessστο μπλόκ που περιέχει το password.Εάν το password σας είναι 8 χαρακτήρες σε μέγεθος θα τον αποκαλύψει ένα ένα bytetην φορά κάνοντας guessμία φορα ανα 5 λεπτά.Σε μόλις μερικές ώρες θα έχει βρεί το password.Η άμυνα σε αυτήν την επίθεση είναι πάντα να ελέγχουμε το MACελέγχοντας έαν το padείναι έγκυρο ή μη έγκυρο.

6.5.3 Επιθέσεις μη ατομικής-κρυπτογράφησης

Αυτό που θα δείξουμε εδώ είναι μια συγκεκριμένη επίθεση στο **SSHbinaryπρωτόκολο**.Το SSHείναι ένα **standardsecureremoteshellapplication**το οποίο χρησιμοποιεί το μοντέλο πελάτη-διαφημιστή.Διαθέτει έναν μηχανισμό ανταλλαγής κλειδίων και απο τις δύο πλευρές και χρησιμοποιεί αυτό που ονομάζεται binarypacketπρωτόκολλο για να στείλει μηνύματα μεταξύ του clientκαι του server.

Η λειτουργία του SSHχρησιμοποιεί την μέθοδο «encrypt-and-MAC».Αυτό που τεχνικά συμβαίνει είναι κάθε SSHπακέτο ξεκινάει με έναν ακολουθιακό αριθμό, και το πακέτο περιέχει το μέγεθος του πακέτου, το μέγεθος του CBCpad, το ωφέλιμο φορτίο και τέλος το CBCpad(**Εικόνα 97**)

Εικόνα 96:Το πρωτόκολλο SSH

[59] Το IMAPπρωτόκολλο, http://el.wikipedia.org/wiki/IMAP

Ολόκληρο το CBCπακέτο κρυπτογραφείται χρησιμοποιώντας CBCκρυπτογράφηση.ΤοMAC υπολογίζεται βασίζομενο στο plaintextκαι στέλνεται ξεχωριστά απο το πακέτο που περιέχει CBCκρυπτογράφηση.Έχουμε πεί ότι αυτός δέν είναι σωστός τρόπος επειδή τα MACδέν παρέχουν εμπιστευτικότητα και στέλοντας το MAC που να περιέχει το καθαρό plaintext, πολύ πιθανόν να διαρρεύσουν δεδομένα που δέν θέλουμε.Αλλά δεν θα δείξουμε επίθεση τέτοιο είδους σε αυτήν την ενότητα, αλλα θα δείξουμε μια επίθεση πολύ πιο έξυπνη.Για να το κάνουμε αυτό ας δούμε πρώτα το πώς ο σέρβερ αποκρυπτογραφεί το πακέτο.

Αρχικά ο σέρβερ αποκρυπτογραφεί το κρυπτογραφημένο πεδίο που περιέχει το μέγεθος το πακέτου.Μετά θα διαβάσει στο δίκτυο τόσα bytesόσο του δείχνει το πεδίο του μεγέθους του πακέτου που μόλις έμαθε.Μετά θα αποκρυπτογραφήσει τα εναπομείνατα μέρη του ciphertextχρησιμοποιώντας CBCαποκρυπτογράφηση.Μόλις αποκαλύψει όλα τα δεδομένα του SSHpacket, θα ελέγξει το MACτου plaintextκαι θα δείξει erroreαν το MACείναι λανθασμένο.

Το πρόβλημα εδώ είναι ότι το το μέγεθος του πακέτου αποκρυπτογραφείται, και μετα χρησιμοποιείται για να αποφασίσει για το μέγεθος του πακέτου πρωτού κάποιο είδος αυθεντικότητας να εμφανιστεί.Δηλαδή το πρωτόκολλο χρησιμοποιεί το μέγεθος του πακέτου πρωτού επαληθευτεί το MAC.Επομένως μπορεί να υπάρξει επίθεση λόγω αυτού του τρόπου αποκρυπτογράφησης.

<u>Επίθεση στο κρυπτογραφημένο πεδίο που περιέχει το μέγεθος του πακέτου</u>
Έστω ότι ο επιτιθέμενος διακόπτει ένα συγκεκριμένο ciphertext**c=(AES,m)**και ότι θέλει να βρεί το μήνυμα m.Έστω ότι τα 32 πίο σημαντικά bitstου plaintexttου μηνύματος m τυγχάνει να είναι ο αριθμός 5.Ο σέρβερ θα προσπαθήσει να αποκρυπτογραφήσει το μπλόκ και θα πάρει τον αριθμό τον αριθμό 5 σαν μέγεθος.

Ο επιτιθέμενος θα «ταίσει» τον σέρβερ ένα bytetην φορά και μόλις τον «ταίσει» για 5 bytesοσέρβερ θα καταλάβει ότι αποκάλυψε το lengthπεδίο το πακέτου και έτσι θα ξεκινήσει την επαλήθευση MAC.Η επαλήθευση του MACθα βγεί λανθασμένη και θα στείλει στον επιτιθέμενο ένα error.Αρα μετά απο το διάβασμα των 5 bytesοεπιτιθέμενος θα δεί ένα MACerrorκαι θα μάθει ότι τα πίο σημαντικά 32 bitstου πακέτου είναι ο αριθμός 5.

6.6 Παραγωγή κλειδιών

Σε αυτή την ενότητα θα δούμε πως παράγουμε κλειδιά απο ένα και μόνο κλειδί.Έστω ότι έχουμε ένα **πηγαίο κλειδί**(sourcekey) που δημιουργήθηκε με διάφορους μεθόδους παραδείγματος χάρη μια μέθοδος θα ήταν μια hardwarePRG ή ένα πρωτόκολλο ανταλλαγής κλειδιών.Πολλές φορές χρειαζόμαστε αρκετά κλειδιά για να ασφαλίσουμε μια sessionκαι όχι μόνο ένα κλειδί.Ο μηχανισμός που μας παράγει κλειδία απο ένα και μόνο κλειδί ονομάζεται **KDF**(keyderivationfunction).Θα δούμε το πώς λειτουργούν οι κατασκευές KDF.

Υποθέστε ότι έχουμε μια ασφαλή PRFη οποία έχει ώς κλειδόχωρο(keyspace) το K και μας βγάζει αποτελέσματα στο $\{0,1\}^n$.Επίσης υποθέστε ότι το sourcekeyμας είναι κατανεμημένο στο K.Η συνάρτηση KDFπαίρνει σαν όρισματα το sourcekey, ένα πλαίσιο παραμέτρων(parametercontext) και το μέγεθος της εισόδου.Αυτό που θα κάνουμε είναι να εφαρμόσουμε την συνάρτηση για τιμή 0, για τιμή 1, για τιμή 2 μέχρι την τιμή L(**Εικόνα 98**)

Εικόνα 97:Η λειτουργία KDF

Το μόνο ερώτημα ειναι τι είναι το contextstring.Είναι είναι ένα μοναδικό string το οποίο ορίζει την εφαρμογή μας.Πιθανόν να έχετε πολλές εφαρμογές στο σύστημα σας που προσπαθούν να δημιουργήσουν μοναδικά κλειδιά.Παραδείγματος χάρη μπορεί να έχετε την διεργασία SSH, έναν webservero οποίος τρέχει, IPsecδιεργασία και όλες οι τρείς αυτές διεργασίες χρειάζονται 3 μυστικά κλειδιά για να λειτουργήσουν.

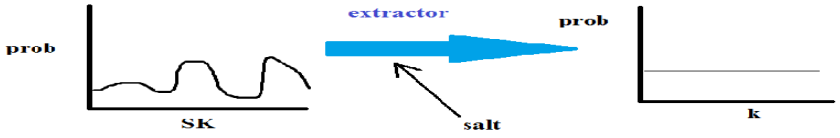
Το contextstringπροσπαθεί να ξεχωρίσει αυτά τα 3 κλειδιά μεταξύ τους τοCTXδηλαδή διαπραγματεύεται το ότι ακομή και έαν δύο εφαρμογές έχουν το ίδιο sourcekey, έφοσον χρειαστούν κλειδιά θα είναι ανεξάρτητα το ένα από το άλλο.

Τι κάνουμε στην περίπτωση που το sourcekeyδεν είναι(uniform);Εαν το sourcekeyδέν είναι uniform στο Κτότε δέν θα μπορούμε να ισχυριστούμε έαν η έξοδος της ψευδοτυχαίας συνάρτησης PRFείναι διακρίσιμη απο μια πραγματικά τυχαία έξοδο.Αρα η έξοδος της PRFδεν θα μοιάζει και τόσο τυχαία.

Extract-thenexpand

Το πρώτο βήμα του KDFείναι να εξαγάγει ένα ψευδοτυχαίο κλειδί απο ένα πηγαίο κλειδί.Ο κάτω άξονας(**Εικόνα 99**) μας δείχνει τις πιθανές τιμές του sourcekeyκαι ο κάθετος άξονας μας δείχνει την πιθανότητα της κάθε τιμής.Οπως μπορείτε να δείτε είναι ένα είδος συνάρτησης που συμπεριφέρεται ανώμαλα(bumpy) και όχι ομοιόμορφα(uniform).

Επομένως θα μπορούσαμε να πούμε ότι το κλειδί δεν είναι uniform. Αυτό που κάνουμε σε αυτή τη περίπτωση είναι αυτό που ονομάζεται **extractor**.Ο extractoreίναι κατι πο παίρνει μια ανώμαλη κατανομή και μας την μετατρέπει σε ομοιόμορφη κατανομή πάνω στον κλειδόχωρο K.



Εικόνα 98:Η λειτουργία του extractor

Στην δική μας περίπτωση θα χρησιμοποιήσουμε τους **computationalextractors**οι οποίοι δέν παράγουν απαραίτητα ομοιόμορφη κατανομή στο τέλος, αλλα δημιουργούν μια κατανομή που είναι μη διακριτή απο μια ομοιόμορφη κατανομή.Οι extractorsπαίρνουν σαν είσοδο αυτο που ονομάζουμε salt και αυτό που κάνειείναι να ανακατέυει τα bits και χώρις να μας αφορά τι είδους κατανομή ειχαμε ώς είσοδο, πάντα η έξοδος θα ειναι μή διακριτή απο μια πραγματικά τυχαία έξοδο.

Ουσιαστικά το saltίναι ένα μή μυστικό string το οποίο δέν είναι κρυμμένο απο κανέναν.Δέν μας αφορά έαν ο επιτιθέμενος το γνωρίζει .Η μόνη απαίτηση είναι ότι όταν το επιλέξουμε πρέπει να το επιλέξουμε τυχαία απο άλλα salts.Το δεύτερο βήμα είναι να επεκτείνουμε το κλειδί για όσα bitsχρειαζόμαστε για το session.

Ένας τρόπος να δημιουργήσουμε KDF σύναρτηση είναι απο το HMAC.Ο τρόπος αυτός ονομάζεται **HKDF**.Το HMACχρησιμοποιείται και ώς PRFγια την επέκταση του κλειδιού αλλα και ώς extractorγια την εξαγωγή το αρχικού ψευδοτυχαίου κλειδιού.Ο τρόπος λειτουργίας του είναι αρκετά απλός.Στο βήμα της εξαγωγής θα χρησιμοποιήσουμε το salt που όπως είπαμε είναι μια δημόσια τιμή που γεννίεται τυχαίακαι θα το χρησιμοποιήσουμε ώς HMAC κλειδί , ενώ το sourcekeyθα χρησιμοποιηθεί σαν HMACδεδομένα.Μετά εφαρμόζουμε HMACσε αυτά τα δύο και σαν αποτέλεσμα θα έχουμε ένα κλειδί το οποίο θα είναι μή διακρισιμο απο ένα πραγματικά τυχαίο κλειδί:

k←HMAC(salt, SK)

Στη συνέχεια θα χρησιμοποιήσουμε το HMACώς PRFγια να παράγουμε ένα sessionκλειδι χρησιμοποιώντας το κλειδί k.Ποτέ δε χρησιμοποιούμε το sourcekeyαπευθείας σαν sessionkey.Αυτό που θα πρέπει να κάνουμε είναι τροφοδοτήσουμε το sourcekeyμέσα απο μία KDF και αυτή με τη σειρά της θα μας δώσει όλα τα κλειδία που θα χρειαστούμε.

Ένα άλλο ζήτημα είναι το πώς θα εξάγουμε κλειδιά απο τα passwords.Η κατασκευή αυτή ονομάζεται **PBKDF**.Δέν θα πρέπει να παράγουμε κλειδιά απο τα passwordsέαν χρησιμοποιούμε HDF διότι τα passwordsδέν έχουν αρκετή εντροπία.Επίσης τα παραγόμενα κλειδία θα είναι ευάλωτα σε επιθέσεις εξαντλητικής αναζήτησης.Οπότε αυτο που θα κάνουμε είναι να χρησιμοποιήσουμε PBKDF.

Αρχικά χρησιμοποιούμε ένα saltκαι μια αργή συνάρτηση κατακερματισμού και τώρα θα δούμε πως θα παράγουμε κλειδιά απο τα passwords. Ο τρόπος ονομάζεται **PCS#5**.Αυτό που κάνει είναι να κατακερματίζει την αλληλουχία passwordκαι saltκαι αυτή την διαδικασία την κάνουμε πάρα πολλές φορές.Το αποτέλεσμα αυτής της διαδικασίας είναι το κλειδί που χρειαζόμαστε.

6.7 Ντετερμινιστική κρυπτογράφηση

6.7.1 Η εννοια της ντετερμινιστικής κρυπτογράφησης

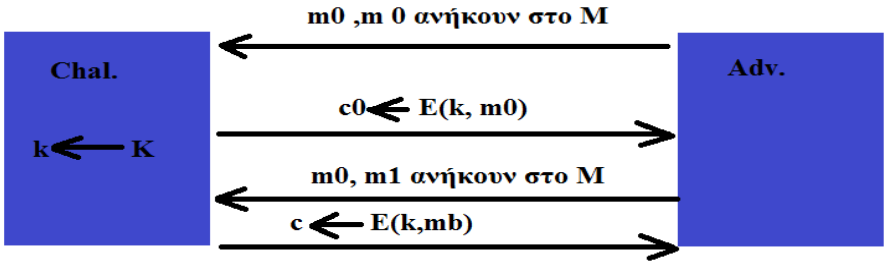
Οταν λέμε για ένα ντετερμινιστικό σύστημα κρυπτογράφησης εννούμε ένα σύστημα κρυπτογράφησης στο οποίο πάντα ένα συγκεκριμένο μήνυμα μας δίνει συγκεκριμένο ciphertext.Έαν κρυπτογραφήσουμε το μήνυμα μας 3 φορές τότε και τις 3 φορές θα πάρουμε το ίδιο ciphertext.Έστω ότι έχουμε έναν servero οποίος θα αποθηκέυσει πληροφορία μέσα σε μια κρυπτογραφημένη βάση δεδομένων.Αυτό που θα αποθηκέυσει είναι records.

Κάθε recordέχει έναν δείκτη και κάποια δεδομένα πο αποθηκέυτονται μέσα στο record.Αυτό που θα κάνει ο σέρβερ είναι να κρυπτογραφήσει αυτό το record.Ο indexκρυπτογραφείται με ένα κλειδί k_1και τα δεδομένα κρυπτογραφούνται με ένα κλειδί k_2 και το κρυπτογραφημένο recordθα σταλθεί στη βάση δεδομένων.Το ίδιο γίνεται για αρκέτα records.

Έαν η κρυπτογράφηση είναι ντετερμινιστική, σε μια στιγμή αργότερα όταν ο σέρβερ θέλει να λάβει ένα recordαπο την βάση το μόνο που θα κάνει είναι να στείλει στη βάση είναι η κρυπτογράφηση του index.

Μετά αυτό θα αποκρυπτογραφηθεί και θα δεί η βάση ότι το ciphertextείναι ίδιο με αυτό που έιχε γραφτεί την στιγμή που είχε γραφτεί στη βάση το συγκεκριμένο record και μετά θα ψάξει να το βρεί στη βάση και να θα στείλει το recordσε αυτόν που το ζητάει.

Το πρόβλημα είναι ότι ο μηχανισμός αυτός είναι ευάλωτος σε επιθέσεις CPA.Εαν ο επιτιθέμενος δεί ένα συγκεκριμένο ciphertext 2 φορές θα είναι σίγουρος ότι τα μηνύματα αυτών θα είναι τα ίδια.Αυτο δηλαδή οδηγεί σε επιθέσεις ειδικά άν ο κλειδόχωρος των μηνυμάτων M είναι πολύ μικρός.Θα δείξουμε ότι η ντετερμινιστική κρυπτογράφηση είναι ευάλωτη σε επιθέσεις CPAμε ένα απλό παιχνίδι(**Εικόνα 100**)



Εικόνα 99:Ντετερμινιστική κρυπτογράφηση

Το παιχνίδι ξεκινάει με τον επιτιθέμενο να στέλνει 2 μηνύματα m_0, m_1Επειδή αυτά τα μηνύματα είναι τα ίδια τότε θα πάρει πίσω την κρυπτογράφηση του m_0.Στη συνέχεια θα στείλει παλι τα μηνύματαm_0, και m_1 άρα θα πάρει πίσω είτε την κρυπτογράφηση του m_0είτε το mκαι ο σκοπός του είναι ωα καταλάβει ποιά απο τις δύο αποκρυπτογραφήσεις πήρε.Επειδή όμως η κρυπτογράφηση είναι ντετερμινιστική αυτό που θα κάνει είναι να ελέγξει έαν το ceίναι ίδιο με το c_0και μετά θα καταλάβει έαν έλαβε την κρυπτογράφηση τουc$_0$ή την κρυπτογράφηση του c_1.

Αυτό που θα μπορούσαμε να κάνουμε είναι να περιορίσουμε τον αριθμό των μηνυμάτων τα οποία κρυπτογραφούνται κάτω απο το ίδιο κλειδί.Θα υποθέσουμε ότι ο encryptor ποτέ δεν κρυπτογραφεί το ίδιο μήνυμα με το ίδιο κλειδί.**Με άλλα λόγια το ζέυγος μήνυμα-κλειδί πρέπει να είναι μοναδικό.**

6.7.2 SIV και widePRP

Τώρα θα προσπαθήσουμε να κατασκευάσουμε **συστήματα** τα οποία μας παρέχουν ασφάλεια ενάντια στην ντετερμινιστική κρυπτογράφηση.Αρχικά θα αναφερθούμε στην κατασκευή που ονομάζεται **syntheticIV**(SIV).Φανταστείτε ότι έχουμε ένα γενικο σχήμα (E, D) που μας παρέχει CPAκρυπτογράφηση άρα η κρυπτογράφηση γίνεται ώς $E(k, m, r) \to c$.Το r για παράδειγμα εαν αναφερόμαστε σε countermode μπορεί να είναι το τυχαίο IV.Έστω $F:K \times M \to R$ να είναι μια ασφαλής PRF.Το r είναι μια μικρή τιμή που ανήκει στο R.

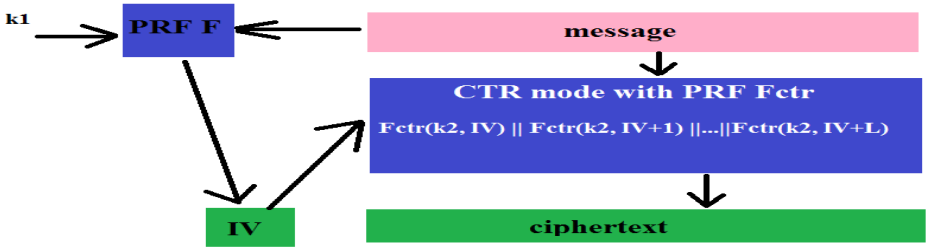
Το πρώτο βήμα είναι να εφαρμόσουμε την ψευδοτυχαία συνάρτηση F στο μήνυμα για να δώσουμε την δυνατότητα της «τυχαιότητας» στο ασφαλές σύστημα CPA.Κατόπιν

θα κρυπτογραφήσουμε το μήνυμα mχρησιμοποιώντας τορπου μόλις υπολογίσαμε δηλαδή:

$$r \leftarrow F(k_1, m)$$
$$c \leftarrow E(k_2, m, r)$$

Μια πολύ καλή ιδιότητα του SIVείναι ότι μπορούμε να λάβουμε ciphertextακεραιότητα χωρίς την χρηση MAC.Δέν χρειάζεται να χρησιμοποιήσουμε MACμιας και το SIVέχει σχεδιαστεί με έναν μηχανισμό ακεραιότητας.Αυτό θα το δουμε αμέσως.

Ο σκοπός μας είναι να να κατασκευάσουμε αυτό που ονομάζεται **ντετερμινιστικηαυθεντική κρυπτογράφηση**(DAE).Ας δούμε το **SIV-CTR**οπου το SIV έχει χρησιμοποιήσει countermodeμε τυχαίο IV.Όπως είπαμε και πρίν το SIV παίρνει το μήνυμα μας και εφαρμόζει μια PRFσε αυτό(μαζί με το κλειδί k_1) και αυτή η διαδικασία μας παράγει ένα τυχαίο IV και αυτό το IVθα χρησιμοποιηθεί για να κρυπτογραφήσουμε το μήνυμα χρησιμοποιώντας CTR(**Εικόνα 101**)



Εικόνα 100:Η λειτουργία SIV-CTR

Το SIVείναι πολύ αποδοτικό στην περίπτωση που έχουμε να κάνουμε με πολύ μεγάλα μηνύματα.Τι γίνεται όμως έαν το μήνυμα μας είναι πολύ μικρό παραδείγματος χάρη μικρότερο απο 16bytes;Το μόνο που έχουμε να κάνουμε είναι να χρησιμοποιήσουμε μια PRP.Υποθέστε ότι (E, D) να είναι μια ασφαλής PRP και E:K×X→X.Εαν την χρησιμοποιήσουμε απευθείας μας δίνει ασφάλεια ενάντια σε ντετερμινιστικες επιθέσεις CPA.Θα δείξουμε γιατι συμβαίνει αυτό.

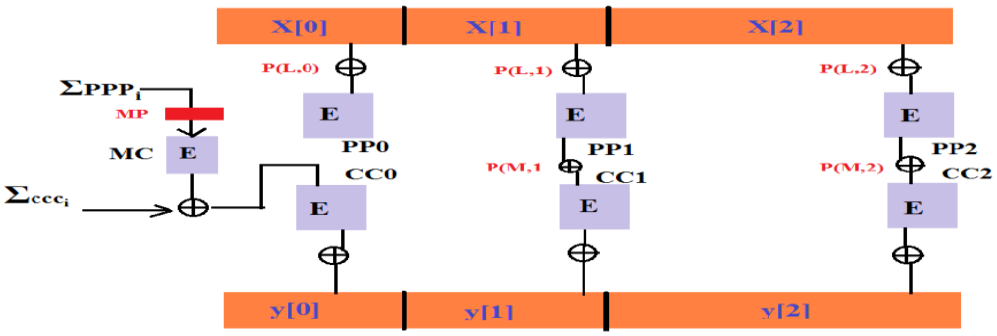
Υποθέστε ότι η f:X→Xείναι μια πραγματικά τυχαία και αντιστρέψιμη συνάρτηση στο πείραα EXP(0).Θυμηθείτε ότι μια PRP είναι μή διακρίσιμη απο μια τυχαία συνάρτηση.Για να το δείξουμε αυτό θα θεωρήσουμε όπως πάντα δύο πειράματα.Στο πείραμα 0 που ο επιτιθέμενος υποβάλλει μηνύματα θα δεί είναι τις τιμές $\mathbf{f(m_{1,0}…m_{q,0})}$ και αυτά τα μηνύματα είναι διαφορετικά μεταξύ τους.

Αν κοιτάξουμε στο πείραμα 1 βλέπει τα μηνύματα $\mathbf{f(m_{1,1}…f(m_{q,1})}$ οπου και πάλι αυτά τ μηνύματα είναι διαφορετικά μεταξύ τους.Οι δύο αυτές κατανομές είναι είναι ίδιες που σάν αποτέλεσμα ο επιτιθέμενος δέν μπορεί να διακρίνει το πείραμα 0 απο το πειραμα 1 και αφού δέν μπορεί για μια πραγματικά τυχαία συνάρτηση δέν θα μπορεί και για την PRP.Όμως αυτό δέν μας παρέχει ακεραιότητα.

Τι κάνουμε έαν έχουμε μηνύματα μεγαλύτερα απο 16 bytes;Θα μπορούσαμε να χρησιμοποιήσουμε SIV.Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε PRPτιθεται το ερώτημα το έαν μπορούμε να δημιουργήσουμε PRP'Σπου έχουν messagespaceμεγαλύτερο απο 16 bytes.Σε προηγούμενη ενότητα είχαμε κατασκευάσει PRF'Σπου έιχαν μεγάλο messagespaceαπο PRF'Σπου έιχαν μικρό messagespace.Έδω

θα κατασκευάσουμε PRP'Sμε μεγάλο messagespaceαπο PRF'Sπου έχουν μικρό messagespace.

Έστω ότι (E, D) είναι ασφαλής PRP η οποία λειτουργεί πάνω σε Νμπλόκς.Υπάρχει μια μέθοδος που ονομάζεται **EME**η οποία κασκευάσει PRP'Soι οποίες λειτουργούν σε Νμπλόκς οπου N>>n.Αυτό μας επιτρέπει να κάνουμε ντετερμινιστική κρυπτογράφηση σε μηνύματα μεγαλύτερα απο 16 bytes.Ας δούμε πώς λειτουργεί.ToEME παίρνει 2 εισόδους τα κλειδία Κκαι Λόπου το Λπαράγεται απο το Κ αλλα για τους δικούς μας σκοπούς ας υποθέσουμε ότι είναι δύο διαφορετικά κλειδιά.Αυτό που κάνουμε αρχικά ειναι να παίρνουμε το μήνυμα μας Χ και να το σπάμε σε μπλόκς(**Εικονα 102**)



Εικόνα 101:Η λειτουργία του EME

Στη συνέχεια θα κάνουμε πράξη XORσε κάθε μπλόκ με μια συγκεκριμένη paddingσυνάρτηση.Χρησιμοποιούμε το κλειδί Λγια να διαμοιράσουμε ένα διαφορετικό padγια κάθε μπλόκ και μετά θα εφαρμόσουμε PRPσε κάθε μπλόκ χρησιμοποιώντας το κλειδί Κ.Θα ονομάσουμε το κάθε αποτέλεσμα αυτής της διαδικασίας PP0, PP1, PP2. Το επόμενο πράγμα που θα κάνουμε είναι να κάνουμε πράξη XORμεταξύ όλων αυτών των PPP και το αποτέλεσμα που θα προκύψει το ονομάζουμε MP.Κατόπιν κρυπτογραφούμε το αποτέλεσμα αυτό χρησιμοποιώντας το Εκαι το Κ.

Θα ονομάσουμε την έξοδο της κρυπτογράφησης MC και στη συνέχεια κάνουμε πράξη XORμεταξύ του MCκαι του MP το οποίο μας βοηθά να παράγουμε ακόμη ένα padπου το χρησιμοποιούμε σε πράξη XORμεταξύ των PPPγια να πάρουμε τα CCCκαι μετά πάλι XORγια να πάρουμε την τιμή CCC0, το οποίο θα το κρυπτογραφήσουμε όπως και πρίν με όλα αυτά τα Ε..και μας δίνει τελικά την έξοδο τουEME.

6.8 Tweakableencryption

Ας δούμε το πρόβλημα κρυπτογράφησης ενός δίσκου.Εστω ότι θέλουμε να κρυπτογραφήσουμε τομείς(sectors) του δίσκου και έστω ότι ο κάθε τομέας είναι 4 kilobytesσε μέγεθος.Το πρόβλημα είναι ότι δέν έχουμε χώρο για επέκταση.Με άλλα λόγια έαν το μέγεθος ενός τομέαείναι 4 kilobytestότε το ciphertextπρέπει να είναι ακριβώς 4 kilobytesεπειδή δέν υπάρχει χώρος για έξτρα bitsεάν το ciphertextήταν μεγαλύτερο απο το plaintext.Ο σκοπός μας είναι να κατασκευάσουμε μια κρυπτογράφηση που δέν θα επεκτείνει τον χώρο.Δηλαδή ο messagessearch είναι ίδιος με τον ciphertextspace.

Σε αυτήν την περίπτωση πρέπει να χρησιμοποιήσουμε ντετερμινιστική κρυπτογράφηση επειδή έαν η κρυπτογράφηση ήταν τυχαία δέν θα έιχαμε χώρο να

αποθηκέυσουμε αυτήν την «τυχαιότητα».Επίσης δέν έχουμε χώρο για την ακεραιότητα επειδή δέν μπορουμε να επεκτείνουμε το ciphertextπροσθέτωντας επιπλέον bitακεραιότητας.Υπάρχει και ένα **λημμα 6.8.1** που μας λέει ότι έαν μας δώσουν έναν ντετερμινιστικό και ασφαλη CPA αλγόριθμο όπου το messagespaceείναι ίσο με το ciphertextspaceτότε ο αλγόριθμος είναι RPR.Επομένως κάθε sector του δίσκου μας πρέπει να κρυπτογραφηθεί με PRP.

Αρχικά παίρνουμε τον δίσκο μας και τον σπάμε σε τομείςs και έαν κρυπτογραφούμε κάθε τομέαςχρησιμοποιώντας PRPστον ίδιο χρόνο, θα βρούμε ένα **πρόβλημα** στην ντετερμινιστική μας κρυπτογράφηση διότι έαν τυγχάνει παραδείγματος χάρη, ο τομέας 1 και ο τομέας 3 να έχουν το ίδιο plaintextτότε ο κρυπτογραφημένος τομέας 1 θα είναι ίδιος με τον κρυπτογραφημένο τομέας 3 και ο επιτιθέμενος θα καταλάβαινε ότι τα plaintexttους είναι ίδια.

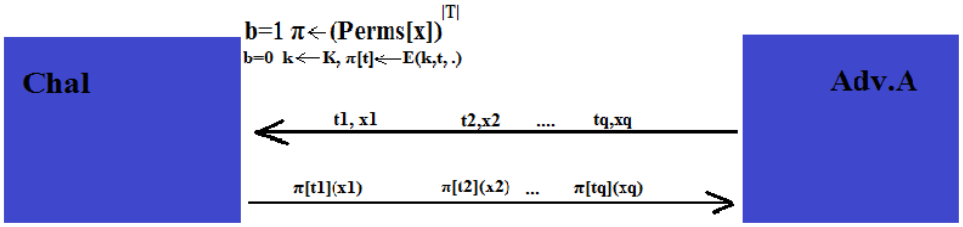
Μια λύση στο πρόβλημα είναι να χρησιμοποιούμε διαφορετικό κλειδί για τον κάθε τομέας.Δεν ειναι τέλεια λύση δίοτι υπάρχει πρόβλημα διαρροής πληροφορίας**.**Για παράδειγμα έαν ο χρήσης θέλει να αλλάξει το λιγότερο 1 bit απο εναν τομέας και στη συνέχεια αυτός θα κρυπτογραφηθεί και θα μας δώσει τελείως διαφορετικό ciphertext.Όμως έαν ο χρήστης κανεί undo την αλλαγή και αναστρέψει την ενέργεια τότε ο τομέας θα επιστρέψει στην κατάσταση που ήταν πρίν κρυπτογραφημένος και ο επιτιθέμενος θα μπορούσε να παρατηρήσει την αλλαγή που συνέβη.

Απο την άλλη πλευρά, οι δίσκοι συνεχώς γίνονται όλο και μεγαλύτεροι σε μέγεθος και αυτό σημαίνει ότι θα δημιουργούμε παρα πολλά κλειδιά.Για να γίνει αυτό, εφόσον δέν χρειάζεται να αποθηκέυσουμε τα κλειδία κάπου είναι να χρησιμοποιήσουμε ένα master key, και έναν αριθμό, τα οποία θα περάσουν μέσα απο μια PRF για να μας παράγουν sector keys.Μπορούμε όμως να βρούμε μια καλύτερη λύση;

Αυτό μας οδηγεί στο να εισάγουμε έναν **tweakable**αλγόριθμο τμήματος ο οποίος μας δημιουργεί απο ένα masterkeyπολλές PRP.Για να γίνει αυτό απλά κρυπτογραφουμε με αυτό το κλειδί και με έναν αριθμο RPP.Ο αριθμός αυτός ονομάζεται**tweak**.Σε έναν tweakableαλγόριθμο τμήματος ο αλγόριθμος κρυπτογράφησης και αποκρυπτογράφησης παίρνουν ένα κλειδί ώς είσοδο και ένα tweak δηλαδή **E, D:K×T×X→X**.Η εφαρμογή μας θα χρησιμοποιεί τον αριθμό του κάθε τομέαςσαν tweakγια να παράγουμε ανεξάρτητες και διαφορετικές μεταξύ τους PRP για κάθε τομέαςξεχωριστά.

Ας δούμε πιο αναλυτικά τι είναι ο **twekable αλγόριθμος τμήματος**(tweakableblockcipher).Οπώς είπαμε υπάρχει ένα tweakspaceκαι ώς συνήθως θα ορίσουμε τα δύο πειράματα μας.Στο πείραμα 1 θα επιλέξουμε τόσα permutations όσος είναι ο αριθμός των tweaks και θα δημιουργήσουμε ένα κλειδί K.Ο επιτιθέμενος θα υποβάλει κάθε φορα ένα tweakκαι ένα μήνυμα X(**Εικόνα 103**).

Θα λάβει την τιμή της PRPγια το tweak_1 και αυτή την διαδικασία θα πράξει για q στο σύνολο υποβολές μηνυμάτων και tweaks.Ο σκοπός του είναι να καταλάβει έαν επιδρά με μια πραγματικά τυχαία permutationή με μια ψευδοτυχαία permutation.Εαν δεν μπορεί να το κάνει τότε λέμε ότι ο tweakableαλγόριθμος είναι ασφαλής.



Εικόνα 102:Tweakableblockcipher

Οπότε το E είναι μια ασφαλής tweakablePRP εάν για κάθε επιτιθέμενο A ισχύει ότι

$$\mathbf{Adv_{Tprp}[A, E]=|P[EXP(0)=1]-P[EXP(1)=1]|}$$

Ένα παράδειγμα tweakableαλγορίθμου είναι η **trivialcontstruction**.Εστω (E,D) να είναι μια ασφαλής PRP όπου E: K×X→X.Υποθέστε επίσης ότι K=X.Ένας αλγόριθμος ορίζεται ώς εξής:

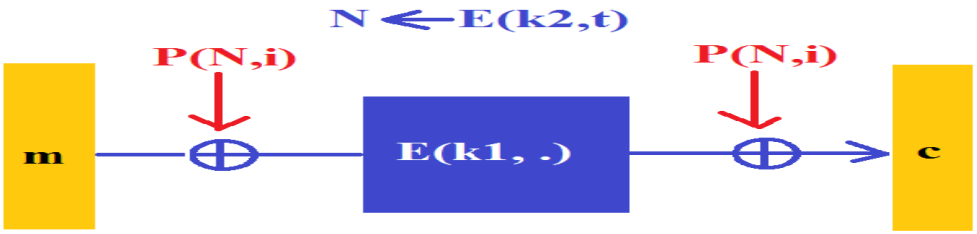
$$\mathbf{E_{tweak}(k, t, x)=E(E(k,t), x)}$$

Οπου κρυπτογραφούμε το tweakμας χρησιμοποιώντας το masterkeyκαι στη συνέχεια κρυπτογραφούμε τα δεδομένα χρησιμοποιώντας το αποτέλεσμα του E(k,t).Μια άλλη κατασκευή καλυτερη απο την trivialconstructionείναιη **XTSconstruction**.Έστω (E,D) μια ασφαλής PRP όπου E: $K\times\{0,1\}^n \rightarrow\{0,1\}^n$.Αντί για έναν απλό αλγόριθμο τμήματος θα ορίσουμε επίσης έναν tweakableαλγόριθμο:

$$\mathbf{E_{tweak}((k_1,k_2), (t,i), x)}$$

Το πρώτο βήμα(**Εικόνα 104**) είναι να κρυπτογραφήσουμε το αριστερό tweak που ονομάζεται t που γίνεται πολύ απλά:

$$\mathbf{N\leftarrow E(k_2,t)}$$



Εικόνα 103:XTSconstruction

Μετά θα κάνουμε πράξη XORτο μήνυμα μας με μια paddingσυνάρτηση P(N,i).Το αποτέλεσμα της XORπράξης θα κρυπτογραφηθεί με το κλειδί k_1και μετά θα εφαρμόσουμε πάλι XORχρησιμοποιώντας το ίδιο padP(N,i) και το αποτέλεσμα αυτής της πράξης θα είναι το ciphertextμας.

6.9 Ερωτήσειςκεφαλαίου

Ερώτηση 1

An attacker intercepts the following ciphertext (hex encoded): 20814804c1767293b99f1d9cab3bc3e7ac1e37bfb15599e5f40eef805488281d.He knows that the plaintext is the ASCII encoding of the message "Pay Bob 100$" (excluding the quotes). He also knows that the cipher used is CBC encryption with a random IV using AES as the underlying block cipher. Show that the attacker can change the ciphertext so that it will decrypt to "Pay Bob 500$". What is the resulting ciphertext (hex encoded)? This shows that CBC provides no integrity.

Απάντηση

We receive the following ciphertext:
20814804c1767293b99f1d9cab3bc3e7 ac1e37bfb15599e5f40eef805488281d.The first 16 bytes are the IV. We see that on decryption, the IV will beXORed with the decryption of the first message block. We simply identify theposition of the number '1' in "Pay Bob 100$":
20814804c1767293b99f1d9cab3bc3e7
P A Y B O B 1 0 0 $
Then we replace the '1' with a '5' by XORing the correct positon with ord('1')^ ord('5') = 0x04, where ord(x) is the ASCII encoding of x :

```
                20814804c1767293b99f1d9cab3bc3e7
                00000000000000004000000000000000
20814804c1767293bd9f1d9cab3bc3e7
```

So the ciphertext is:
20814804c1767293bd9f1d9cab3bc3e7 ac1e37bfb15599e5f40eef805488281d

Ερώτηση 2

Let (E,D) be an encryption system with key space K, message space $\{0,1\}^n$ and ciphertext space $\{0,1\}^s$. Suppose (E,D) provides authenticated encryption. Which of the following systems provide authenticated encryption: (as usual, we use ‖ to denote string concatenation)

a)E′(k,m)= [c←E(k,m) , output (c,c) and D′(k, (c_1,c_2))=$\begin{cases} D(k,c_1) \ \ if \ c_1 = c_2 \\ \perp \ otherwise \end{cases}$

b)E′(k,m)=E(k,m) and D′(k, c)=$\begin{cases} D(k,c) \ if \ D(k,c) \neq \perp \\ 0^n \ otherwise \end{cases}$

c)E′(k,m)=(E(k, m), E(k,m)) and D′(k, (c_1,c_2))=D(k,c_1)

d)E′((k_1,k_2), m)=E(k_2, E(k_1,m)) and D′((k_1,k_2), c)=$\begin{cases} D(k_1, D(k_2,c)) \ if \ \ D(k_2,c) \neq 1 \\ \perp \ otherwise \end{cases}$

Απάντηση

The answer a is correct:
The (E′,D′) provides authenticated encryption because an attack on (E′,D′) directly gives an attack on (E,D).
The answer b is not correct:
This system does not provide ciphertext integrity since an attacker can simply output the ciphertext 0s and win the ciphertext integrity game.
The answer c is not correct:
This system does not provide ciphertext integrity. The attacker can query for E′(k,0^n) to obtain (c_1,c_2). It then outputs (c_1,0^s) and wins the ciphertext integrity game.
The answer d is correct:
The (E′,D′) provides authenticated encryption because an attack on (E′,D′) gives an attack on (E,D). It's an interesting exercise to work out the ciphertext integrity attack on (E,D) given a ciphertext integrity attacker on (E′,D′).

Ερώτηση 3

If you need to build an application that needs to encrypt multiple messages using a single key, what encryption method should you use? (for now, we ignore the question of key generation and management)

a)Implement Encrypt-and-MAC yourself
b)Use a standard implementation of one of the authenticated encryption modes GCM, CCM, EAX or OCB.
c)Use a standard implementation of CBC encryption with a random IV.
d)Use a standard implementation of randomized counter mode.

Απάντηση
The correct answer is **b**. Also, just using CBC encryption does not providem essage integrity. Therefore the correct answer is to _use a standard implementation of one of the authenticated encryption modes GCM, CCM, EAX, or OCB

Ερώτηση 4
Let (E,D) be a symmetric encryption system with message space M (think of M as only consisting for short messages, say 32 bytes). Define the following MAC (S,V) for messages in M:

S(k,m):=E(k,m);V(k,m,t):=$\begin{cases} 1 \; if \; D(k,t) = m \\ 0 \; othwerwise \end{cases}$

a)Semantic security under a chosen plaintext attack
b)Ciphertext integrity
c)Chosen ciphertext security
d)Semantic security

Απάντηση
The correct answer is **b**. Indeed, ciphertext integrity prevents existential forgery under a chosen message attack.

Ερώτηση 5
In lecture 8.1 we discussed how to derive session keys from a shared secret. The problem is what to do when the shared secret is non-uniform. In this question we show that using a PRF with a non-uniform key may result in non-uniform values. This shows that session keys cannot be derived by directly using a non-uniform secret as a key in a PRF. Instead, one has to use a key derivation function like HKDF.

Suppose k is a non-uniform secret key sampled from the key space $\{0,1\}^{256}$. In particular, k is sampled uniformly from the set of all keys whose most significant 128 bits are all 0. In other words, k is chosen uniformly from a small subset of the key space. More precisely,

for all c$\in\{0,1\}^{256}$: P[k=c]=$\begin{cases} 1/2^{128} \; if \; MSB(c) = 0^{128} \\ 0 \; otherwise \end{cases}$

Let F(k,x) be a secure PRF with input space $\{0,1\}^{256}$. Which of the following is a secure PRF when the key k is uniform in the key space $\{0,1\}^{256}$, but is insecure when the key is sampled from the non-uniform distribution described above?

a)F′(k,x)=$\begin{cases} F(k,x)\ if MSB_{128}(k) = 0^{128} \\ 0^{256}\ otherwise \end{cases}$

b)F′(k,x)=F(k,x)

c)F′(k,x)=$\begin{cases} F(k,x)\ if\ MSB_{128} \neq 0^{128} \\ 0^{256}\ otherwise \end{cases}$

d)F′(k,x)=$\begin{cases} F(k,x)\ if\ MSB_{128} \neq 1^{128} \\ 1^{256}\ otherwise \end{cases}$

Απάντηση

The correct answer is **c**. F′(k,x) is a secure PRF because for a uniform key k the probability that $MSB_{128}(k)=0^{128}$ is negligible. However, for the *non-uniform* key k this PRF always outputs 0 and is therefore completely insecure. This PRF cannot be used as a key derivation function for the distribution of keys described in the problem.

Ερώτηση 6

In what settings is it acceptable to use deterministic authenticated encryption (DAE) like SIV?

a)When messages have sufficient structure to guarantee that all messages to be encrypted are unique.

b)To individually encrypt many packets in a voice conversation with a single key.

c)When a fixed message is repeatedly encrypted using a single key.

d)To encrypt many records in a database with a single key when the same record may repeat multiple times

Απάντηση

The correct answer is **a**. DAE can be used whenever messages are very likely to be unique. It shouldn't be used if there is a chance that identical messages will be encrypted.

Ερώτηση 7

Let E(k,x) be a secure block cipher. Consider the following tweakable block cipher: $E'((k_1,k_2),t,x)=E(k_1,x)\oplus E(k_2,t)$. Is this tweakable block cipher secure?

a)No because for $x\neq x'$ we have:
$E'((k_1,k_2), t, x)\oplus E'((k_1,k_2),t',x)=E'((k_1,k_2),t,x') \oplus E'((k_1,k_2,),t')$

b)No because for $t\neq t'$we have:
$E'((k_1,k_2), t, 0) \oplus E'((k_1,k_2),t',1)=E'((k_1,k_2),t',1) \oplus E'((k_1,k_2,),t')$

c)No because for x≠x we have :
$E'((k1, k2), 0, x) \oplus E'((k1, k2), 0, x)=E'((k1, k2), 0, x')\oplus E'((k1, k2), 0)$

d)No because we have:
$E'((k1, k2), t, x) \oplus E'((k1, k2), t, 1) = E'((k1, k2), t', 0) \oplus E'((k1, k2), t')$

e)Yes, it is secure assuming E is a secure block cipher

Απάντηση
The correct answer is **d**.Since this relation holds, an attacker can make 4 queries to E′ and distinguish E′ from a random collection of one-to-one functions.

Ερώτηση8
In lecture 8.5 we discussed format preserving encryption which is a PRP on a domain {0,…,s−1} for some pre-specified value of s. Recall that the construction we presented worked in two steps, where the second step worked by iterating the PRP until the output fell into the set {0,…,s−1}.

Suppose we try to build a format preserving credit card encryption system from AES using *only* the second step. That is, we start with a PRP with domain {0,1}128 from which we want to build a PRP with domain 1016. If we only used step (2), how many iterations of AES would be needed in expectation for each evaluation of the PRP with domain 1016?

a)4
b)$2^{128}/10^{16} \approx 3.4 \times 10^{22}$
c)$10^{16}/2^{128}$
d)10^{16}

Απάντηση
The correct aswer is **b**. On every iteration we have a probability of $10^{16}/2^{128}$ of falling into the set $\{0,\ldots,10^{16}\}$ and therefore in expectation we will need $2^{128}/10^{16}$ iterations. This should explain why step (1) is needed.

Ερώτηση 9
Let (E,D) be a secure tweakable block cipher. Define the following MAC (S,V):
S(k,m)=E(k,m,0) : V(k,m,tag)=$\begin{cases} 1 \; if \; E(k,m,0) = tag \\ 0 \; otherwise \end{cases}$

In other words, the message m is used as the tweak and the plaintext given to E is always set to 0. Is this MAC secure?

a) it depends on the tweakable block cipher.
b)yes
c)no

Απάντηση
The correct answer is **b**.The MAC is secure. Suppose an adversary could forge a message tag pair (m, t). He could then mount attack (E,D) by making the requests needed to find (m,t), then asking for the encryption of [m](0). If E(k, m, 0) == t, then the adversary knows that the challenger is using E instead of a random function.

Ερώτηση 10

In Lecture 7.6 we discussed padding oracle attacks. These chosen-ciphertext attacks can break poor implementations of MAC-then-encrypt. Consider a system that implements MAC-then-encrypt where encryption is done using CBC with a random IV using AES as the block cipher. Suppose the system is vulnerable to a padding oracle attack.

An attacker intercepts a 64-byte ciphertext c (the first 16 bytes of c are the IV and the remaining 48 bytes are the encrypted payload). How many chosen ciphertext queries would the attacker need in the worst case in order to decrypt the entire 48 byte payload? Recall that padding oracle attacks decrypt the payload one byte at a time.

a) 12240
b) 1024
c) 12288
d) 48

Απάντηση

The correct answer is **c**. orrect. Padding oracle attacks decrypt the payload one byte at a time. For each byte the attacker needs no more than 256 guesses in the worst case. Since there are 48 bytes total, the number queries needed is 256×48=12288.

Προγραμματιστικήάσκηση 4

In this project you will experiment with a padding oracle attack against a toy web site hosted at crypto-class.appspot.com. Padding oracle vulnerabilities affect a wide variety of products, including secure tokens. This project will show how they can be exploited. We discussed CBC padding oracle attacks in Lecture 7.6, but if you want to read more about them, please see Vaudenay's paper.

Now to business. Suppose an attacker wishes to steal secret information from our target web site crypto-class.appspot.com. The attacker suspects that the web site embeds encrypted customer data in URL parameters such as this:

http://crypto-class.appspot.com/po?er=f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4

That is, when customer Alice interacts with the site, the site embeds a URL like this in web pages it sends to Alice. The attacker intercepts the URL listed above and guesses that the ciphertextfollowing the "po?er=" is a hex encoded AES CBC encryption with a random IV of some secret data about Alice's session.

After some experimentation the attacker discovers that the web site is vulnerable to a CBC padding oracle attack. In particular, when a decrypted CBC ciphertext ends in an invalid pad the web server returns a 403 error code (forbidden request). When the CBC padding is valid, but the message is malformed, the web server returns a 404 error code (URL not found).

Armed with this information your goal is to decrypt the ciphertext listed above. To do so you can send arbitrary HTTP requests to the web site of the form

http://crypto-class.appspot.com/po?er="your ciphertext here"

and observe the resulting error code. The padding oracle will let you decrypt the given ciphertext one byte at a time. To decrypt a single byte you will need to send up to 256 HTTP requests to the site. Keep in mind that the first ciphertext block is the random IV. The decrypted message is ASCII encoded.

To get you started here is a short Python script that sends a ciphertext supplied on the command line to the site and prints the resulting error code. You can extend this script (or write one from scratch) to implement the padding oracle attack. Once you decrypt the given ciphertext, please enter the decrypted message in the box below.

This project shows that when using encryption you must prevent padding oracle attacks by either using encrypt-then-MAC as in EAX or GCM, or if you must use MAC-then-encrypt then ensures that the site treats padding errors the same way it treats MAC errors.

Απάντηση

Ο κώδικας της προγραμματιστικής άσκησης 4 βρίσκεται εδώ.Σε αυτήν την άσκηση θα δούμε την επίθεση paddingoracleattackστο σίτε crypto-class.appspot.com.Υποτίθεται ότι ένας επιτιθέμενος θέλει να υποκλέψει κάποιες προσωπικές πληροφορίες απο το σίτε αυτό.Το σίτε όμως κρυπτογραφεί το μέρος του URLπου περιέχει τις ευαίσθητες πληροφορίες του πελάτη.Όταν η Alice(πελάτης) επικοινωνεί με το σίτε όπως είπαμε κρυπτογραφείται το URL.

Όμως ο επιτιθέμενος διακόπτει το URL και μαντέυει ότι η τιμή «po?er=»είναι κωδικοποιημένη σε δεκαξεαδική μορφή και έχει υποστεί κρυπτογράφηση με AESCBC μαζί με ένα τυχαίο IV.Μετά απο πολύ προσπάθεια ο επιτιθέμενος καταλαβαίνει ότι είναι ευαίσθητο σε επιθέσεις CBCpadding.Τότε όταν ένα CBCciphertextοδηγεί σε λανθασμένο padστέλνει μήνυμα 403(forbiddenrequest) ενώ αν το CBCpaddingείναι σωστό τότε ο σέρβερ επιστρέφει 404 error(URLnotfound)

Αυτό που θα κάνουμε εδώ είναι να αποκρυπτογραφήσουμε το URLχρησιμοποιώντας HTTPrequestsστο σίτε και θα παρατηρούμε τι λάθος μας επιστρέφει.Επίσης το paddingoracleμας επιτρέπει να αποκρυπτογραφούμε ένα byteαπο το ciphertextτην φορά υποβάλλοντας 256 HTTPrequestsγια κάθε byte.Τέλος το αποκρυπτογραφημένο μήνυμα μας θα έχει τη μορφή δεκαεξαδικού

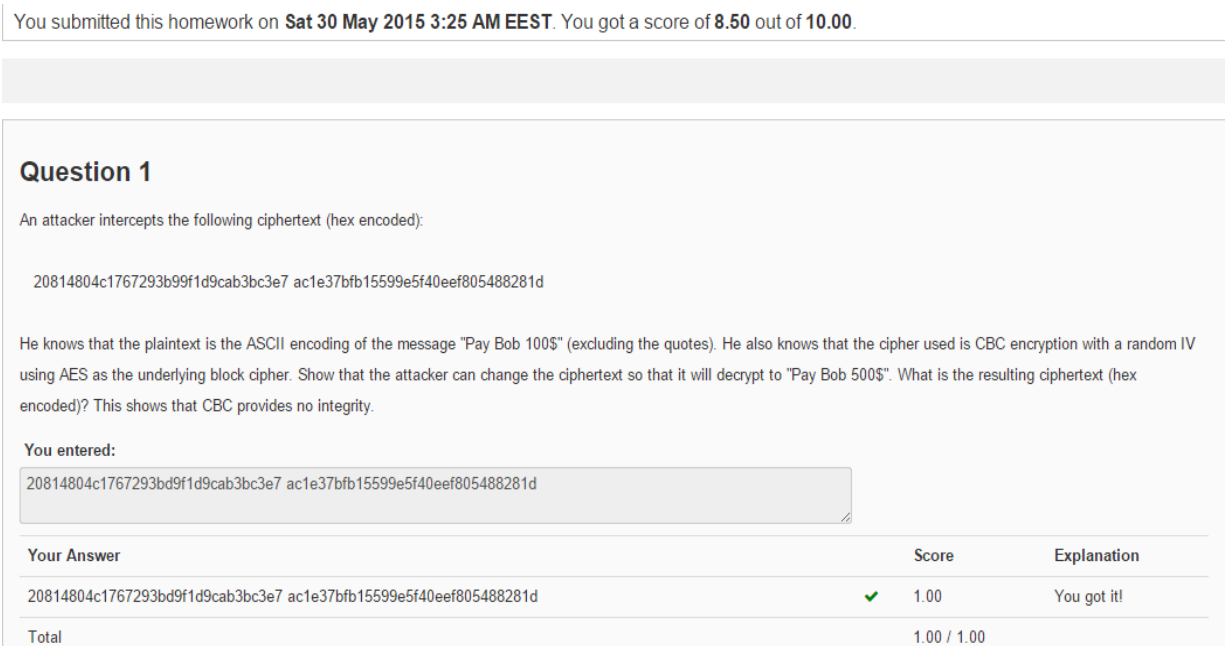
Αρχικά δημιουργούμε την συνάτηση queryη οποία παίρνει ενα ερώτημα qκαι αυτό που κάνει πρώτα είναι να στέλνει ένα HTTPrequestστον σέρβερ.Στη συνέχεια περιμένει για απάντηση.Εαν μας στείλιε λάθος θα πρέπει να δούμε ποιό λάθος μας έστειλε.Αν μας στείλει το λάθος 404 τότε το padding είναι σωστό και προχωράμε.

Ορίζουμε ώς stringτομέρος του URLπου είναι κρυπτογραφημένο και στη συνέχειαΟρίζουμε μια δέυτερη συνάρτηση που την ονομάζουμεint2hex(i) η οποία επιστρέφει την ακέραια μορφή του i, όπουιείναι μια συμβολοσειρά στο δεκαεξαδικό.Μετά δημιουργούμε τις συναρτήσεις exor_pad(i), exor_g(g, pos),

rellenar_zero(s) για να μας βοηθήσουνε με το padding ενώ δημιουργούμε την συνάρτηση strxor(a,b) για να μπορέσουμε να κάνουμε πράξη XORμεταξύ ΄ δύο συμβολοσειρών με διαφορετικό μήκος και την συνάρτηση hexexor(s1, s2) που χρησιμοποιεί την συνάρτηση hexexor(s1, s2) για να τις μετατρέψει σε συμβολοσειρές.

Κατόπιν με διαδικασίες παρόμοιες για κάθε μπλόκ προσπαθούμε να αποκρυπτογραφήσουμε το μήνυμα χρησιμοποιώντας τις παραπάνω συναρτήσεις σύμφωνα με τον ορισμό του CBCoraclepadding πχ το ivγια το μπλόκ 3 είναι το μπλόκ 2, και με ανάλογες επαναλήψεις βρίσκουμε ένα –ένα τα ζητούμενα bytes.

6.10 Αναλυτική βαθμολογία



Εικόνα 104:Ερώτηση 1-Week 4

Question 2

Let (E,D) be an encryption system with key space K, message space $\{0,1\}^n$ and ciphertext space $\{0,1\}^s$. Suppose (E,D) provides authenticated encryption. Which of the following systems provide authenticated encryption: (as usual, we use $\|$ to denote string concatenation)

Your Answer		Score	Explanation
☑ $E'(k,m) = \left[c \leftarrow E(k,m),\ \text{output } (c,c)\right]$ and $D'(k,(c_1,c_2)) = \begin{cases} D(k,c_1) & \text{if } c_1 = c_2 \\ \perp & \text{otherwise} \end{cases}$	✔	0.25	(E',D') provides authenticated encryption because an attack on (E',D') directly gives an attack on (E,D).
☐ $E'(k,m) = E(k,m)$ and $D'(k,c) = \begin{cases} D(k,c) & \text{if } D(k,c) \neq \perp \\ 0^n & \text{otherwise} \end{cases}$	✔	0.25	This system does not provide ciphertext integrity since an attacker can simply output the ciphertext 0^s and win the ciphertext integrity game.
☑ $E'(k,m) = \big(E(k,m),\ E(k,m)\big)$ and $D'(k,(c_1,c_2)) = D(k,c_1)$	✘	0.00	This system does not provide ciphertext integrity. The attacker can query for $E'(k,0^n)$ to obtain (c_1,c_2). It then outputs $(c_1,0^s)$ and wins the ciphertext integrity game.
☐ $E'\big((k_1,k_2),m\big) = E(k_2,\ E(k_1,m))$ and $D'\big((k_1,k_2),\ c\big) = \begin{cases} D(k_1,\ D(k_2,c)) & \text{if } D(k_2,c) \neq \perp \\ \perp & \text{otherwise} \end{cases}$	✘	0.00	(E',D') provides authenticated encryption because an attack on (E',D') gives an attack on (E,D). It's an interesting exercise to work out the ciphertext integrity attack on (E,D) given a ciphertext integrity attacker on (E',D').
Total		0.50 / 1.00	

Εικόνα 105:Ερώτηση 2-Week 4

Question 3

If you need to build an application that needs to encrypt multiple messages using a single key, what encryption method should you use? (for now, we ignore the question of key generation and management)

Your Answer		Score	Explanation
○ implement Encrypt-and-MAC yourself			
◉ use a standard implementation of one of the authenticated encryption modes GCM, CCM, EAX or OCB.	✔	1.00	
○ use a standard implementation of CBC encryption with a random IV.			
○ use a standard implementation of randomized counter mode.			
Total		1.00 / 1.00	

Εικόνα 106:Ερώτηση 3-Week 4

Question 5

In lecture 8.1 we discussed how to derive session keys from a shared secret. The problem is what to do when the shared secret is non-uniform. In this question we show that using a PRF with a *non-uniform* key may result in non-uniform values. This shows that session keys cannot be derived by directly using a *non-uniform* secret as a key in a PRF. Instead, one has to use a key derivation function like HKDF.

Suppose k is a *non-uniform* secret key sampled from the key space $\{0,1\}^{256}$. In particular, k is sampled uniformly from the set of all keys whose most significant 128 bits are all 0. In other words, k is chosen uniformly from a small subset of the key space. More precisely,

$$\text{for all } c \in \{0,1\}^{256}: \quad \Pr[k=c] = \begin{cases} 1/2^{128} & \text{if } \mathrm{MSB}_{128}(c) = 0^{128} \\ 0 & \text{otherwise} \end{cases}$$

Let $F(k,x)$ be a secure PRF with input space $\{0,1\}^{256}$. Which of the following is a secure PRF when the key k is uniform in the key space $\{0,1\}^{256}$, but is insecure when the key is sampled from the *non-uniform* distribution described above?

Your Answer	Score	Explanation
$F'(k,x) = \begin{cases} F(k,x) & \text{if } \mathrm{MSB}_{128}(k) = 0^{128} \\ 0^{256} & \text{otherwise} \end{cases}$		
$F'(k,x) = F(k,x)$		
$F'(k,x) = \begin{cases} F(k,x) & \text{if } \mathrm{MSB}_{128}(k) \neq 0^{128} \\ 0^{256} & \text{otherwise} \end{cases}$	✔ 1.00	$F'(k,x)$ is a secure PRF because for a uniform key k the probability that $\mathrm{MSB}_{128}(k) = 0^{128}$ is negligible. However, for the *non-uniform* key k this PRF always outputs 0 and is therefore completely insecure. This PRF cannot be used as a key derivation function for the distribution of keys described in the problem.
$F'(k,x) = \begin{cases} F(k,x) & \text{if } \mathrm{MSB}_{128}(k) \neq 1^{128} \\ 1^{256} & \text{otherwise} \end{cases}$		
Total	1.00 / 1.00	

Εικόνα 107:Ερώτηση 5-Week 4

Question 6

In what settings is it acceptable to use *deterministic* authenticated encryption (DAE) like SIV?

Your Answer	Score	Explanation
when messages have sufficient structure to guarantee that all messages to be encrypted are unique.		
to individually encrypt many packets in a voice conversation with a single key.		
when a fixed message is repeatedly encrypted using a single key.	✘ 0.00	This would be insecure because an attacker can tell that all the resulting ciphertexts are an encryption of the same message.
to encrypt many records in a database with a single key when the same record may repeat multiple times.		
Total	0.00 / 1.00	

Εικόνα 108:Ερώτηση 6-Week 4

Question 7

Let $E(k,x)$ be a secure block cipher. Consider the following tweakable block cipher:

$$E'\big((k_1,k_2),t,x\big) = E(k_1,x) \bigoplus E(k_2,t)$$

Is this tweakable block cipher secure?

Your Answer	Score	Explanation
no because for $x \neq x'$ and $t \neq t'$ we have $E'((k_1,k_2),t,x) \bigoplus E'((k_1,k_2),t',x) = E'((k_1,k_2),t,x') \bigoplus E'((k_1,k_2),t',$		
no because for $t \neq t'$ we have $E'((k_1,k_2),t,0) \bigoplus E'((k_1,k_2),t',1) = E'((k_1,k_2),t',1) \bigoplus E'((k_1,k_2),t'$		
no because for $x \neq x'$ we have $E'((k_1,k_2),0,x) \bigoplus E'((k_1,k_2),0,x) = E'((k_1,k_2),0,x') \bigoplus E'((k_1,k_2),0,$		
✔ no because for $t \neq t'$ we have $E'((k_1,k_2),t,0) \bigoplus E'((k_1,k_2),t,1) = E'((k_1,k_2),t',0) \bigoplus E'((k_1,k_2),t',$	1.00	since this relation holds, an attacker can make 4 queries to E' and distinguish E' from a random collection of one-to-one functions.
yes, it is secure assuming E is a secure block cipher.		
Total	1.00 / 1.00	

Εικόνα 109:Ερώτηση 7-Week 4

Question 8

In lecture 8.5 we discussed format preserving encryption which is a PRP on a domain $\{0,\ldots,s-1\}$ for some pre-specified value of s. Recall that the construction we presented worked in two steps, where the second step worked by iterating the PRP until the output fell into the set $\{0,\ldots,s-1\}$.

Suppose we try to build a format preserving credit card encryption system from AES using *only* the second step. That is, we start with a PRP with domain $\{0,1\}^{128}$ from which we want to build a PRP with domain 10^{16}. If we only used step (2), how many iterations of AES would be needed in expectation for each evaluation of the PRP with domain 10^{16}?

Your Answer	Score	Explanation
4		
✔ $2^{128}/10^{16} \approx 3.4 \times 10^{22}$	1.00	On every iteration we have a probability of $10^{16}/2^{128}$ of falling into the set $\{0,\ldots,10^{16}\}$ and therefore in expectation we will need $2^{128}/10^{16}$ iterations. This should explain why step (1) is needed.
$10^{16}/2^{128}$		
10^{16}		
Total	1.00 / 1.00	

Εικόνα 110:Ερώτηση 8-Week 4

Question 9

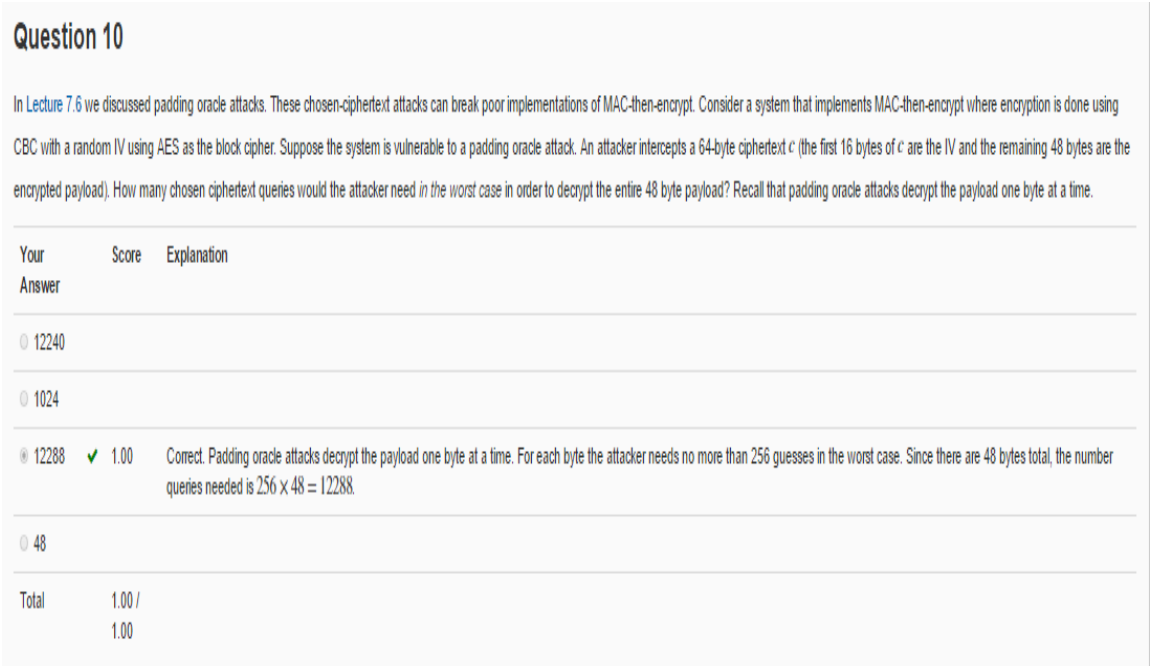
Let (E,D) be a secure tweakable block cipher. Define the following MAC (S,V):

$$S(k,m) := E(k,m,0) \quad ; \quad V(k,m,\text{tag}) := \begin{cases} 1 & \text{if } E(k,m,0) = \text{tag} \\ 0 & \text{otherwise} \end{cases}$$

In other words, the message m is used as the tweak and the plaintext given to E is always set to 0. Is this MAC secure?

Your Answer	Score	Explanation
it depends on the tweakable block cipher.		
✔ yes	1.00	A tweakable block cipher is indistinguishable from a collection of random permutations. The chosen message attack on the MAC gives the attacker the image of 0 under a number of the permutations in the family. But that tells the attacker nothing about the image of 0 under some other member of the family.
no		
Total	1.00 / 1.00	

Εικόνα 111:Ερώτηση 9-Week 4

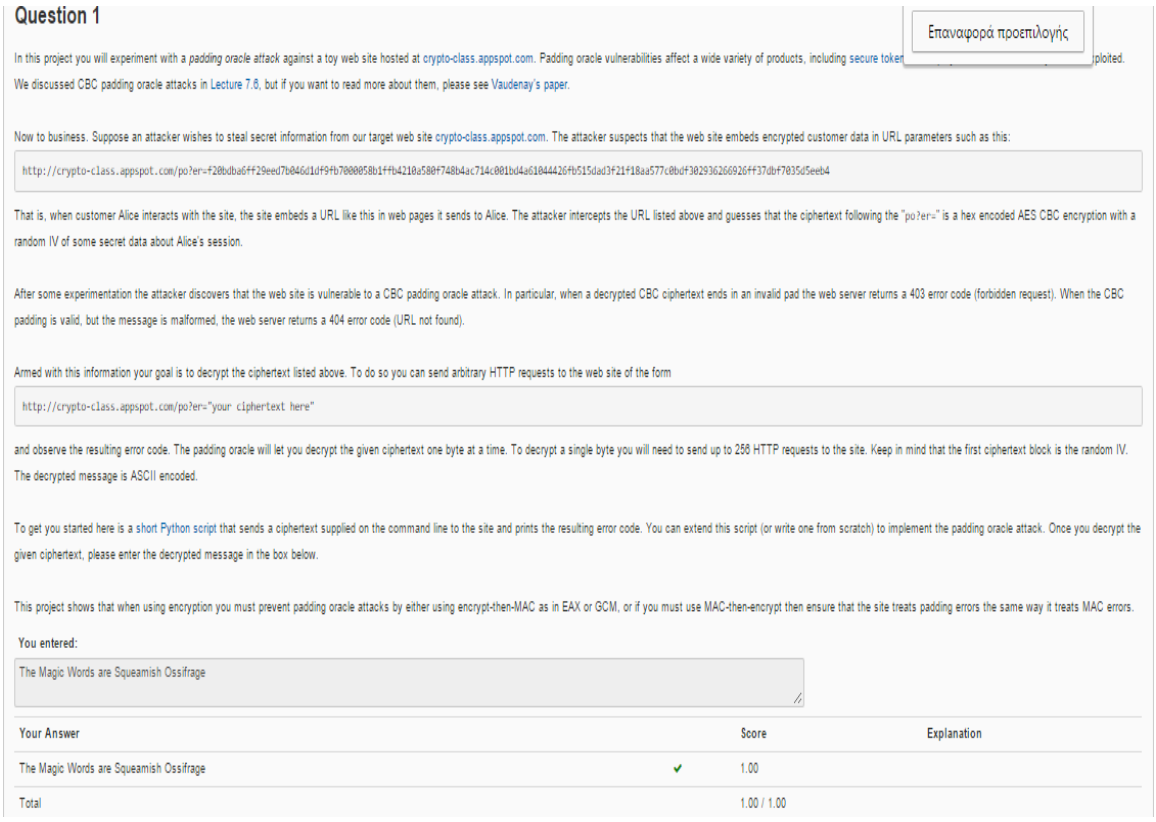


Question 10

In Lecture 7.6 we discussed padding oracle attacks. These chosen-ciphertext attacks can break poor implementations of MAC-then-encrypt. Consider a system that implements MAC-then-encrypt where encryption is done using CBC with a random IV using AES as the block cipher. Suppose the system is vulnerable to a padding oracle attack. An attacker intercepts a 64-byte ciphertext c (the first 16 bytes of c are the IV and the remaining 48 bytes are the encrypted payload). How many chosen ciphertext queries would the attacker need *in the worst case* in order to decrypt the entire 48 byte payload? Recall that padding oracle attacks decrypt the payload one byte at a time.

Your Answer	Score	Explanation
12240		
1024		
12288	✓ 1.00	Correct. Padding oracle attacks decrypt the payload one byte at a time. For each byte the attacker needs no more than 256 guesses in the worst case. Since there are 48 bytes total, the number queries needed is 256 x 48 = 12288.
48		
Total	1.00 / 1.00	

Εικόνα 112:Ερώτηση 10-Week 4



Question 1

Επαναφορά προεπιλογής

In this project you will experiment with a *padding oracle attack* against a toy web site hosted at crypto-class.appspot.com. Padding oracle vulnerabilities affect a wide variety of products, including secure token ploited. We discussed CBC padding oracle attacks in Lecture 7.6, but if you want to read more about them, please see Vaudenay's paper.

Now to business. Suppose an attacker wishes to steal secret information from our target web site crypto-class.appspot.com. The attacker suspects that the web site embeds encrypted customer data in URL parameters such as this:

```
http://crypto-class.appspot.com/po?er=f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4
```

That is, when customer Alice interacts with the site, the site embeds a URL like this in web pages it sends to Alice. The attacker intercepts the URL listed above and guesses that the ciphertext following the "po?er=" is a hex encoded AES CBC encryption with a random IV of some secret data about Alice's session.

After some experimentation the attacker discovers that the web site is vulnerable to a CBC padding oracle attack. In particular, when a decrypted CBC ciphertext ends in an invalid pad the web server returns a 403 error code (forbidden request). When the CBC padding is valid, but the message is malformed, the web server returns a 404 error code (URL not found).

Armed with this information your goal is to decrypt the ciphertext listed above. To do so you can send arbitrary HTTP requests to the web site of the form

```
http://crypto-class.appspot.com/po?er="your ciphertext here"
```

and observe the resulting error code. The padding oracle will let you decrypt the given ciphertext one byte at a time. To decrypt a single byte you will need to send up to 256 HTTP requests to the site. Keep in mind that the first ciphertext block is the random IV. The decrypted message is ASCII encoded.

To get you started here is a short Python script that sends a ciphertext supplied on the command line to the site and prints the resulting error code. You can extend this script (or write one from scratch) to implement the padding oracle attack. Once you decrypt the given ciphertext, please enter the decrypted message in the box below.

This project shows that when using encryption you must prevent padding oracle attacks by either using encrypt-then-MAC as in EAX or GCM, or if you must use MAC-then-encrypt then ensure that the site treats padding errors the same way it treats MAC errors.

You entered:

The Magic Words are Squeamish Ossifrage

Your Answer		Score	Explanation
The Magic Words are Squeamish Ossifrage	✓	1.00	
Total		1.00 / 1.00	

Εικόνα 113:Προγραμματιστική άσκηση -Week 4

Πίνακας 4:Βαθμολογίες-Week 4

	1η προσπάθεια	2η προσπάθεια	3η προσπάθεια	4η προσπάθεια
Ερώτησεις	5.00/10.00	7.00/10.00	8.50/10.00	7.00/10.00
Άσκηση	0.00/1.00	0.00/1.00	0.00/1.00	1.00/1.00

Μαθηματική και αλγοριθμική ανάλυση κρυπτογραφικών τεχνικών

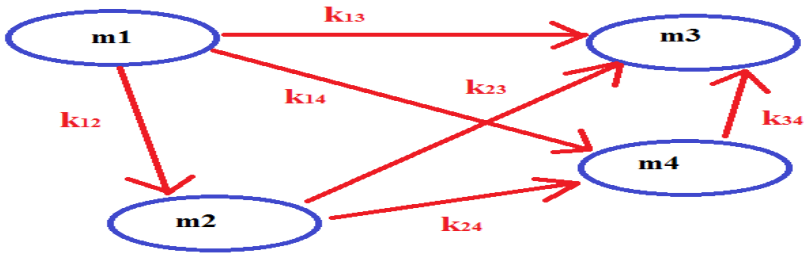
ΚΕΦΑΛΑΙΟ 7
ΑΝΤΑΛΛΑΓΗΚΛΕΙΔΙΩΝ

7.1Trusted3dpartiesκαιmerkle-puzzles

7.1.1 Trusted 3d parties [60]και toy protocol

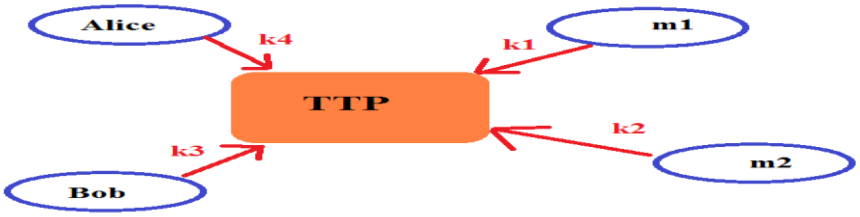
Τώρα που γνωρίζουμε το πώς δύο χρήστες μπορούν να προστατέψουν τα δεδομένα τους χρησιμοποιώντας ένα μυστικό κλειδί το οποίο το μοιράζονται, το επόμενο ερώτημα είναιτο πώς αυτοί οι δύο χρήστες δημιουργούν αυτό το κλειδί..Σε αυτήν την ενότητα θα δούμε μερικά πρωτόκολλα ανταλλαγής κλειδιών τα οποία θα μας βοηθήσουν να συλλάβουμε την κεντρική ιδέα της κρυπτογραφίας δημόσιου κλειδιού.

Φανταστείτε ότι υπάρχουν N χρήστες στον κόσμο.Το ερώτημα είναι πως αυτοί οι χρήστες διαχειρίζονται αυτά τα μυστικά κλειδιά τα οποία θα χρησιμοποιήσουν για να επικοινωνήσουν μεταξύ τους.Έστω ότι υπάρχουν 4 χρήστες στον κόσμο(**Εικόνα 115**).Για παράδειγμα ο χρήστης 1 και ο χρήστης 3 μοιράζονται το ίδιο κλειδί, ο χρήστης 1 και χρήστης 2 μοιράζονται το ίδιο κλειδί όμοίως για τους χρήστες 2 και 4 και τους χρήστες 4 και 3...Το πρόβλημα είναι ότι έχουμε πάρα πολλά κλειδιά να διαχειριστούμε και ο κάθε χρήστης πρέπει να αποθηκέυσει N κλειδιά στο σύνολο με σκοπό να επικοινωνήσει με τους N χρήστες.Μπορούμε να κάνουμε **κάτι καλύτερο;**



Εικόνα 114:Προσπάθεια επικοινωνίας με Νκλειδιά

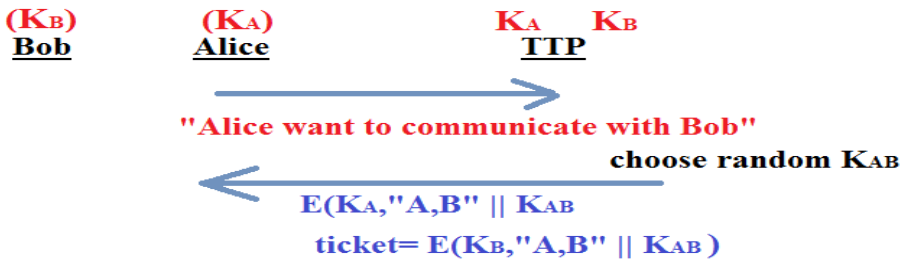
Αυτό που θα κάνουμε ονομάζεται **onlinetrusted 3dparty**(TTP).Ο κάθε χρήστης θα μοιράζεται ένα μοναδικό κλειδί με την TTP.Για παράδειγμα (**Εικονα 116**) ο χρήστης m_1θα μοιράζεται με την TPP το κλειδί k_1, ο χρήστης 2 θα μοιράζεται με την TPPτο κλειδί k_2, η Aliceθα μοιράζεται το κλειδί k_Aκαι ο Bobτο κλειδί k_b.Ο κάθε χρήστης χρειάζεται να θυμάται **ένα μόνο** μυστικό κλειδί.Εαν η Aliceθα θέλει να επικοινωνήσει με τον Bobθα πρέπει τα κλειδιά τους να χρησιμοποιηθούν με κάποιο τρόπο για να δημιουργήσουν ένα κλειδί k_{AB} .Θα χρησιμοποιήσουμε ένα πρωτόκολλο που θα το κάνει αυτό.



Εικόνα 115:Το πρωτόκολλο TPP

[60] Εκτενής αναφορά για τα trusted 3 parties, https://en.wikipedia.org/wiki/Trusted_third_party

Το πρωτόκολλο αυτό ονομάζεται **toyprotocol**που είναι ασφαλές μόνο σε επιθέσεις eavesdropping.Θα δούμε αμέσως το πώς λειτουργεί.Η Aliceξεκινάει στέλνοντας ένα μήνυμα στηνTPP λέγοντας ότι θέλει ένα μυστικό κλειδί το οποίο θα το μοιραστεί με τον Bobγια να επικοινωνήσουν.Η TPPθα επιλέξει ένα τυχαίο κλειδί k_{AB}.Θα στείλει ένα μήνυμα πίσω στην Aliceτο οποίο αποτελείται απο δύο μέρη.Τοπρώτο μέρος είναι η κρυπτογράφηση με το κλειδί της Alice(k_A) του κλειδιού k_{AB}και το γεγονός ότι το κλειδί k_{AB}θα χρησιμοποιηθεί απο την Aliceκαι τον Bob(**Εικόνα 117**)



Εικόνα 116:Το πρωτόκολλο toy

Το δέυτερο μέρος του μηνύματος είναι αυτό που ονομάζουμε **ticket**είναι ένα μήνυμα το οποίο κρυπτογραφήθηκεμε το κλειδί του για τον Bob(k_B) που περιέχει το γεγονός ότι το κλειδί k_{AB}θα χρησιμοποιηθεί απο την Aliceκαι τον Bob και το κλειδί k_{AB}.Με λίγα λόγια η Alicelαμβάνει ένα μήνυμα που κρυπτογραφήθηκε για αυτήν και ένα μήνυμα που κρυπτογραφήθηκε για τον Bob.Οταν θέλησει να επικοινωνήσει με τον Bobα απλα θα αποκρυπτογραφήσει το μέρος που πρόκειται για αυτήν ενώ στον Bobθα στείλει το ticket.

Το TTPχρειάζεται για κάθε ανταλλαγή κλειδιών άρα η Aliceκαι ο Bobδεν μπορούν να κάνουν ανταλλαγή κλειδιών έαν το TTP**είναιoffline**και επίσης το πρωτόκολλο γνωρίζει όλες τα sessionkeys.Έαν για κάποιο λόγο το TTPδιακοπεί τότε ένας επιτιθέμενος μπορεί πολύ εύκολα να κλέψει όλα τα μυστικά κλειδιά που ανταλλάχθηκαν μεταξύ χρήστη και συστήματος.Για τον λόγο αυτό, ονομάστηκε trusted 3dpartyεπειδή γνώριζει όλα τα sessionkeys.Το πρόβλημα είναι έαν χρησιμοποιούμε το TTPπαγκοσμίως, τότε ποιός θα είναι η onlinetrustedparty.

7.1.2Το πρωτόκολλο merkle-puzzles[61]

Σε αυτήν την ενότητα θα δούμε ένα πρωτόκολλο ανταλλαγής κλειδιών **που δεν** χρησιμοποιεί trusted 3dparty.Έστω ότι έχουμε την Aliceκαι τον Boboι οποίοι δέν έχουν επικοινωνήσει παλαιότερα και θέλουν με κάποιο τρόπο να δημιουργήσουν ένα κοινό κλειδί.Θα μπορούσαμε να χρησιμοποιήσουμε αλγόριθμους τμήματος ή και συναρτήσεις κατακερματισμού αλλα τα πρωτόκολλα που αφορουν την ανταλλαγή κλειδιών μέσω αυτών, δέν είναι αποδοτικά και δεν χρησιμοποιούνται στην πράξη.

Ένα απλό πρωτόκολλο που πράττει αυτά που θέλουμε όνομάζεται **merkle-puzzles**το οποίο εφευρέθηκε απο τον RalphMerkleτο 1974.Το κύριο εργαλείο του πρωτοκόλλου ονομάζεται puzzle.Υποθέστε ότι έχουμε έναν συμμετρικό αλγόριθμο ο οποίος χρησιμοποιεί κλειδί μήκους 128bits(πχ AES)και ότι επιλέγω ένα AESκλειδί έτσι ώστε

[61] Το πρωτόκολλο merkle-puzzles, http://en.wikipedia.org/wiki/Merkle%27s_Puzzles

τα πρώτα 96 bitsείναι μηδέν και τα εναπομείναντα 32 bitsνα μήν είναι μηδέν αλλα τυχαία.

Στη συνέχεια θα κρυπτογραφήσω το μήνυμα «message» χρησιμοποιώντας το κλειδί των 128 bit.Το αποτέλεσμα το ονομάζω puzzle.Το ονομάσαμε puzzleεπειδή δεν είναι πολύ δύσκολο να βρούμε το μυστικό κλειδί Pτο οποίο βρίσκεται ένα δοκιμάσουμε 2^{32} πιθανές τιμές.Για κάθε κλειδί θα προσπαθήσουμε να το αποκρυπτογραφήσουμε για να δούμε έαν θα πάρουμε το μήνυμα «message».Ο τρόπος με τον οποίο θα γίνει η διαδικασία είναι αρκέτα απλή.

Η Aliceθα αρχίσει να δημιουργεί ένα μεγάλο αριθμό απο puzzles.Ουσιαστικά θα δημιουργήσει 2^{32}puzzles και θα επιλέξει 32bitτυχαία puzzlesP_i και δύο άλλες τιμές x_iκαι k_iοι οποίες θα έχουν μέγεθος 128 bit.Στη συνέχεια θα χρησιμοποιήσει το puzzleP_iσαν AESμυστικό κλειδί και θα δημιουργήσει 128 bitκλειδί εκ των οποίων τα 96 bit είναι μηδέν.Το plaintextπου θα προσπαθήσει να αποκρυπτογραφήσει είναι το μήνυμα **" puzzle # x_i " || k_i "** και μετά θα στείλει όλα αυτά τα διαφορετικά puzzlesστον Bob.

Ο Bobθα λάβει όλα αυτα τα διαφορετικά 2^{32}puzzlesκαι θα επιλέξει ένα απο αυτά πχ το puzzlej.Στη συνέχεια αφιερώνει 2^{32} σε χρόνο για να λύσει το puzzle.Το να λύσει ένα puzzlesημαίνει ότι θα δοκιμάσει όλες τις πιθανές τιμές P_iκαι θα αποκρυπτογραφήσει το puzzleπου επέλεξε και θα ελεγξεί έαν το πρώτο μέρος του plaintext ξεκινάει με την λέξη puzzle.

Εαν κατάλαβει ότι έλυσε το puzzleθα λάβει τα τούpuzzle(x_j,k_j).Κατόπιν θα στείλει πίσω στην Aliceτο x_jη οποία με τη σειρά της κοιτώντας στη βάση που κρατάει τα puzzlestης θα καταλάβει ότι ο Bobχρησιμοποιήσε το το κλειδί k_j.Τοτε το κλειδί k_jθα είναι τελικά το **κοινό κλειδί με το οποίο θα επικοινωνήσουν**

Απο ότι βλέπουμε ο επιτιθέμενος χρειάζεται χρόνο n^2για να σπάσει το πρωτόκολλο μιας και πρέπει να λύσει nστον αριθμό puzzlesκαι χρειάζεται nχρόνο για να λύσει κάθε puzzle.Τυπικά ο χρόνος που χρειάζεται για να σπάσει το πρωτόκολλο είναι 2^{64} πράγμα που καθιστά το πρωτόκολλο **μή ασφαλές** και για τον λόγο αυτό δέν χρησιμοποιείται στην πράξη.

7.3 Το πρωτόκολλο Diffie-Hellman[62]

Σε αυτήν την ενότητα θα μιλήσουμε για το **πρωτόκολλοDiffieHellman**το οποίο είναι ένας βασικός μηχανισμός ανταλλαγής κλειδιών.Όπως έχουμε συνηθίσει η Aliceκαι ο Bobδεν έχουν συναντηθεί και θέλουν να ανταλλάξουν ένα μυστικό κοινό κλειδί με σκοπό να επικοινωνήσουν με ασφάλεια μεταξύ τους.Για την καλύτερη κατανόηση του πρωτοκόλλου οι επιτιθέμενοι **δεν επιτρέπεται να έχουν ενεργητική δράση**.

Επιτρέπονται μόνο οι επιθέσεις eavesdropping.Στην προηγούμενη ενότητα είδαμε ένα πρωτόκολλο ανταλλαγής κλειδιών που ονομάζεται merkle-puzzles στο οποίο ο επιτιθέμενος μπορεί να επιτεθεί στο πρωτόκολλο και να το σπάσει σε χρόνο n^2 και

[62] Το πρωτόκολλο DiffieHellman, http://el.wikipedia.org/wiki/%CE%A0%CF%81%CF%89%CF%84%CF%8C%CE%BA%CE%BF%CE%BB%CE%BB%CE%BF_Diffie-Hellman

αυτό σαν αποτέλεσμα το πρωτόκολλο να θεωρείται μη ασφαλές πρακτικά.Έαν η Alicekαι ο Bobεργάζονται στο πρωτόκολλο σε χρόνο n, ο επιτιθέμενος θα πρέπει να να μπορεί να σπάσει το πρωτόκολλο σε χρόνο εκθετικό τουn.

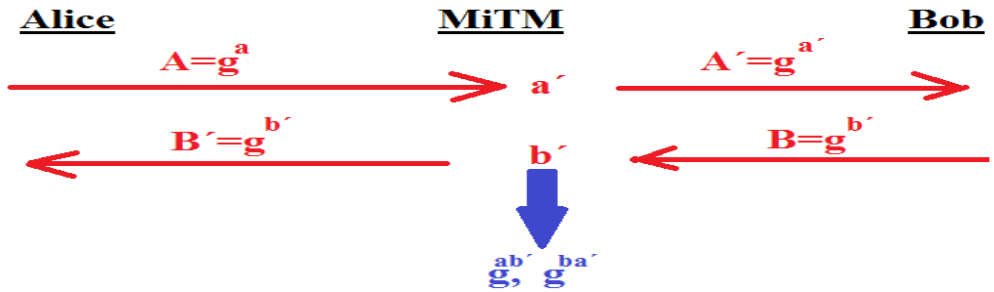
Ας δούμε πώς λειτουργεί το πρωτόκολλο DiffieHellman.Θα θεωρήσουμεέναν πρώτο αριθμό(prime) το οποίο έχει μέγεθος 600 ψηφίων που θα τον ονομάσουμε p.Θα θεωρήσουμε επίσης έναν ακέραιο αριθμο τον g οποίος θα ανήκει στο διάστημα {1,…p}.Αυτές οι δύο τιμές είναι παράμετροι του πρωτοκόλλου.Έστω επίσης η Alicekαι ο Bobπου θέλουν να επικοινωνήσουν.HAliceθα επιλέξει έναν αριθμό απου μπορεί να έχει τιμή μεταξύ {1,..p-1} και μετά θα υπολογίσει το $\mathbf{g^a mod p}$.Το αποτέλεσμα της παράστασης θα το ονομάσουμε A.Στη συνέχεια θα στείλει το A στον Bob.

Ο Bobμε τη σειρά του θα κάνει την ίδια διαδικασία.Θα επιλέξει έναν τυχαίο αριθμο bπου θα ανήκει {1,..p-1} και θα υπολογίσει στη συνέχεια $\mathbf{g^b mod p}$και το αποτέλεσμα θα το ονομάσουμε B.Στη συνέχεια θα στείλει το Bστην Alice.Τώρα ισχυρίζονται ότι δημιούργησαν ένα κοινό κλειδι.Ποίο είναι όμως αυτό το κλειδί;

Τομυστικό κλειδί θα το ονομάσουμε K_{AB}, και θα έχει τιμή $\mathbf{g^{ab} mod p}$.Μια παρατήρηση στο πρωτόκολλο είναι ότι η τιμή gμπορεί να υπολογιστεί και από τα δύο μέρη της επικοινωνίας.Για παράδειγμα η Alicelαμβάνει την τιμή B που στάλθηκε από τον Bob.Δηλαδή $\mathbf{B^a mod p = (g^b)^a}$.Ομοίως ο Bobμπορεί να υπολογίσει το $\mathbf{(g^a)^b = A^b}$

Όμως το πρωτόκολλο αυτό είναι μή ασφαλές σε ενεργητικές επιθέσεις όπως επίθεση τύπου **maninthemiddle**και θα δείξουμε το γιατί.Έστω ότι υπάρχει ένας maninthemiddle(MiTM) ο οποίος προσπαθεί να ακούσει την επικοινωνία μεταξύ Alicekαι Bob.

Η Aliceξεκινάει στέλνοντας στον Bob το $\mathbf{A=g^a}$.Την τιμή Aθα την κλέψει ο MiTMκαι θα την αντικαταστήσει(**Εικόνα 118**)με το δικό του μήνυμα $\mathbf{a'=g^{a'}}$.OBobδέν γνωρίζει ότι ο MiTMέκανε κάτι στο δίκτυο και το μόνο που βλέπει είναι η τιμή A΄ που νομίζει ότι στάλθηκε απο την Alice.



Εικόνα117:Επίθεςman in the middle στοDiffie Hellman

Αυτό που θα κάνει ο Bobείναι να στείλει πίσω την δική του τιμή B, πίσω στην Alice($\mathbf{B=g^b}$).Οπως και πριν ο MiTMθα κλέψει την τιμή B αυτήν την φορά, και θα δημιουργήσει το b΄ που θα το στείλει στην Alice η οποία ουσιαστικά θα πάρει το $\mathbf{B'=g^{b'}}$ και θα υπολογίσει το $\mathbf{g^{ab}}$.OBobμε τη σειρά του θα υπολογίσει το $\mathbf{g^{ba'}}$.Αν παρατηρήσετε δεν είναι τα ίδια κλειδιά αλλα ο MiTMμπορεί να υπολογίσει τις τιμές $\mathbf{g^{ab'}}$, $\mathbf{g^{ba'}}$και όταν η Alicestείλει ένα κρυπτογραφημένο μήνυμα στο Bobο MiTM πολύ

απλα θα το αποκρυπτογραφεί, και μετα θα το επανακρυπτογραφεί με το κλειδί του Bobκαι μετά θα το στέλνει σε αυτόν.

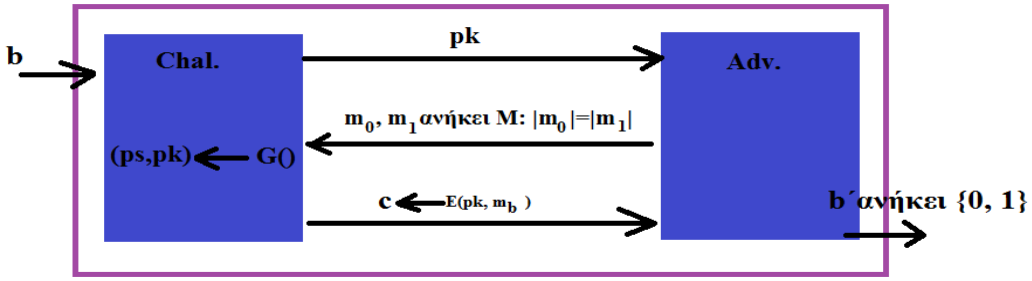
7.4 Κρυπτογραφία δημόσιου κλειδιού[63]

Θα προσπαθήσουμε να δούμε μια διαφορετική προσέγγιση που αφορά την ανταλλαγή κλειδιών που βασίζεται στην δημόσια κρυπτογραφία.Τί είναι όμως η **κρυπτογραφία δημόσιου κλειδιού**;Είναι ενας συμμετρικός τρόπος κρυπτογράφησης στον οποίον χρησιμοποιούμε έναν συμμετρικό αλγόριθμο κρυπτογράφησης και αποκρυπτογράφησης.

Εστω όπως πάντα ότι η Aliceκαι ο Bob θέλουν να επικοινωνήσουν.Ο Bobτώρα διαθέτει δύο κλειδιά.Ένα κλειδί που είναι δημόσιο και ένα κλειδί που είναι μυστικό και το γνωρίζει μόνο αυτός.Πιό αναλυτικά στην κρυπτογραφία δημόσιου κλειδιού υπάρχουν 3 αλγόριθμοι (G, E, D)

Ο αλγόριθμος Gονομάζεται γεννήτρια κλειδιών και όταν «τρέξουμε» τον αλγόριθμο Gη έξοδος του θα είναι 2 κλειδιά , το δημόσιο και το ιδιωτικό κλειδί(pk, sk).Ο αλγόριθμος Επαίρνει ώς είσοδο το μήνυμα μας και το κλειδί pkκαι μας βγάζει ένα ciphertext.Τέλος ο αλγόριθμος D, παίρνει ώς είσοδους το κλειδί skκαι το ciphertextc, και μας δίνει ώς αποτέλεσμα το μήνυμα m ή $\perp$ σε περίπτωση λάθους.Τι σημαίνει ότι η κρυπτογραφία δημόσιου κλειδιού να **είναι ασφαλής**;Ας δούμε την σημασιολογική ασφάλεια της κρυπτογραφίας του δημόσιου κλειδιού.

Έστω ότι ο challengerτρέχει την γεννήτριαGγια να παράγει τα κλειδιά sk, pk(**Εικόνα 119**) και στέλνει το δημόσιο κλειδί του στον αντίπαλο ενω κρατάει το μυστικό κλειδί για τον εαυτό του.Ο αντίπαλος απο την πλευρά του θα βγάλει ώς έξοδο δύο μηνύματα m_0,m_1 που έχουν ίδιο μέγεθος ενώ ο challengerθα του στείλει είτε την κρυπτογράφηση το m_0είτε την κρυπτογράφηση του m_1.



Εικόνα 118:Σημασιολογική ασφάλεια στην κρυπτογραφία δημόσιου κλειδιού

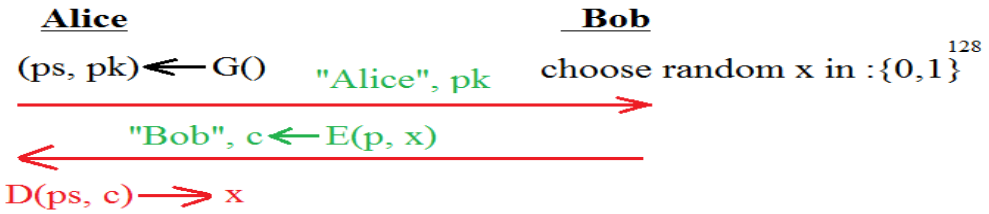
Δηλαδή έχουμε ορίσει δύο πειραματα, το πείραμα 0 και το πείραμα 1 , και στο πείραμα 0 θα λάβει την κρυπτογράφηση του m_0ενώ στο πείραμα 1 θα λάβει την κρυπτογράφηση του m_1 και ο σκοπός του επιτιθέμενου είναι να καταλάβει **ποιά κρυπτογράφηση του δώθηκε**.Στη δημόσια κρυπτογραφία δεν χρειάζεται να δώσουμε στον επιτιθέμενο την δυνατότητα να χρησιμοποιήσει επίθεση chosenplaintextattack» επειδή στην περίπτωση συμμετρικού συστήματος ο επιτιθέμενος θα πρέπει να επιλέγει αυτόςτην κρυπτογράφηση του μηνύματος που θέλει να λάβει.

[63] Κρυπτογραφία δημόσιου κλειδιού, http://en.wikipedia.org/wiki/Public-key_cryptography

Στην περίπτωση δημόσιου συστήματος ομως ο επιτιθέμενος έχει το δημόσιο κλειδί με αποτέλεσμα να κρυπτογραφεί όποιο μήνυμα αυτός θέλει.Δέν χρειαζεται την βοήθεια του challengerγια να δημιουργήσει κρυπτογραφήσεις της επιλογής του.Τώρα μπορούμε να πούμε ότι το σύστημα GEDείναι σημασιολογικά ασφαλές έαν ο επιτιθέμενος δέν μπορεί να διακρίνει το πείραμα 0 απο το πείραμα 1.Δηλαδή:

$$\mathbf{Adv_{SS}[A, E]=| P[EXP(0)]=1 – P[EXP(1)]=1 | <αμελητέο}$$

Τώρα θα δόυμε πώς χρησιμοποιείται η κρυπτογραφία δημόσιου κλειδιού.Έστω η Aliceκαι ο Bob που θέλουν να επικοινωνήσουν.HAliceθα δημιουργήσει ένα τυχαίο ζεύγος κλειδιών **(ps, pk)**για τον εαυτό της χρησιμοποιώντας τον αλγόριθμο G.και θα στείλει στον Bobεκτός απο το δημόσιο κλειδί της pkκαι ένα μήνυμα λέγοντας πως είναι η Alice.Ο Bobμε τη σειρά του θα δημιουργήσει μια τιμή $x \in \{0,1\}^{128}$ όπου θα την κρυπτογραφήσει με το δημόσιο κλειδί της Aliceκαι στη συνέχεια θα την στείλει σε αυτήν μαζί με ένα μήνυμα ότι η κρυπτογραφημένη τιμή x, έρχεται απο τον Bob(**Εικόνα 120**).



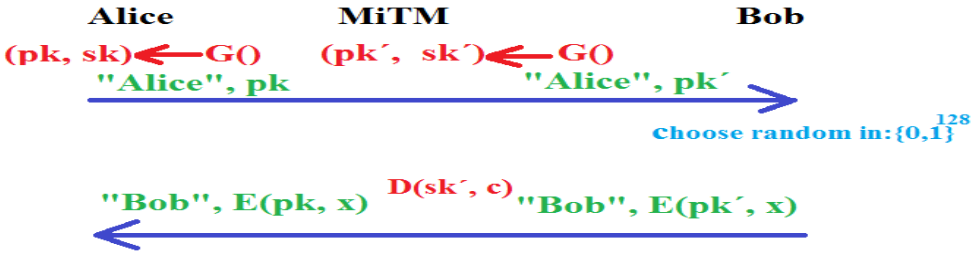
Εικόνα 119:Λειτουργία πρωτοκόλλου δημόσιου κλειδιού

HAliceπου θα λάβει το ciphertextθα αποκρυπτογραφήσει το μήνυμα που έλαβε με το μυστικό κλειδί της psγια να πάρει την τιμή x.Αυτή η τιμή μπορεί να χρησιμοποιηθεί σαν ένα κοινό μυστικό(sharedsecret) μεταξύ τους.Τώρα που έιδαμε πώς δουλεύει θα θα πρέπει να δούμε γιατι είναι ασφαλές σαν πρωτόκολλο σε επιθέσεις eavesdropping..Σε αυτό το πρωτόκολλο ο επιτιθέμενος βλέπει το δημόσιο κλειδί και την κρυπτογράφηση του x και αυτό **που θέλει** είναι η τιμή x.Μιας και γνωρίζουμε οτι το σύστημα δημόσιο κλειδιού είναι σημασιολογικά ασφαλές, αυτό μας δείχνει ότι ο επιτιθέμενος δεν μπορεί να διακρίνει την κρυπτογράφηση του x, απο την κρυπτογράφηση κάτι άλλου εντελώς τυχαίου.

Δοσμένης της κρυπτογράφησης του xδέν μπορεί να διακρίνει έαν πρόκειται για το plaintextτης τιμής x, ή κατι άλλο.Παρόλο που αυτό το πρωτόκολλο είναι ασφαλές σε επιθέσεις eavesdroppingδεν μας δίνει ασφάλεια πάνω σε επιθέσεις **maninthemiddle**.

Επίθεση man in the middle στοπρωτόκολλο

Έστω οτι η Aliceδημιουργεί το δικό της ζέυγος κλειδιών (ps, pk) ενώ ο MiTMδημιουργεί και αυτός το δικό του ζέυγος (ps΄, pk΄).HAliceστέλνει το δημόσιο κλειδί της στον Bob, όμως ο MiTδιακόπτει το μήνυμα που πάει να στείλει η Alicestον Bob(**Εικόνα 121**).Ο Bobστη συνέχεια αντί να λάβει το δημόσιο κλειδί της Aliceθα λάβει **το δημόσιο κλειδί του MiTM** νομίζοντας πως αυτό στάλθηκε απο την Alice.Στη συνέχεια όπως και πρίν, θα επιλέξει πάλι μια τυχαία τιμή x, η οποία θα κρυπτογραφηθεί με το δημόσιο κλειδί της υποτιθέμενης Alice(δεν ειναι η Aliceαλλα ο MiT) και θα την στείλει στην Alice.



Εικόνα 120:Maninthemiddleστη κρυπτογραφία δημόσιου κλειδιού

Ουσιαστικά ο MiTMθα διακόψει το μήνυμα που θα πήγαινε απο τον Bobστην Alicekαι θα το αποκρυπτογραφήσει με το δικό του μυστικό κλειδί άρα D(sk´, c) και θα μάθει την μυστική τιμή x.Τέλος θα το ξανακρυπτογραφήσει με το δημόσιο κλειδι της Alicekαι θα της το στείλει.

7.5 Στοιχεία θεωρίας αριθμών

7.5.1 Βασικές ιδιότητες των ακεραίων

Θα χρησιμοποιήσουμε την θεωρία αριθμών για να κατασκευάσουμε πρωτόκολλα ανταλλαγής κλειδιών, ψηφιακές υπογραφές και κρυπτογραφία δημόσιων κλειδιών.Πιο αναλυτικά,σε αυτήν την υποενότητα θα αναφερθούμε στις βασικές ιδιότητες των ακεραίων αριθμών και σε έννοιες όπως η διαιρετότητα, τα γινόμενα πρώτων αριθμών, κοινοί και μέγιστοι κοινοί διαιρέτες και ελάχιστο κοινό πολλαπλάσιο δύο αριθμών.

Έστω οι ακέραιοι $\mathbb{Z}=\{..., -2, -1, 0, 1, 2,...\}$.Για κάθε $a, b\in \mathbb{Z}$θα λέμε ότι ο a**διαιρεί** τον b έαν $az=b$για κάποιο $z\in \mathbb{Z}$.Έαν το αδιαιρεί το b θα γράφουμε $a|b$και θα λέμε ότι το αείναι **διαιρέτης** του bή ότι το βείναι **πολλαπλάσιο** του aή το b**διαιρείται** απο το a.Έαν το αδεν διαιρεί το bθα γράφουμε $a\nmid b$.Ας δούμε μερικά αποτελέσματα που βγαίνουν σχετικά με την διαιρετότητα

Θεώρημα 7.5.1

Για όλους τους $a, b, c\in \mathbb{Z}$ έχουμε :

- $a|a$, $1|a$, $a|0$
- $0|a$ εάν και μόνο άν $a=0$
- $a|b$ εαν και μόνο αν $-a|b$ αν και μόνο αν $a|-b$
- $a|b$ και $a|c$ τότε $a|(b+c)$
- $a|b$ και $b|c$ τότε $a|C$

Επίσης παρατηρούμε ότι έαν $a|b$και $b\neq 0$ τότε $1\leq|a|\leq|b|$.Εάν $az=b\neq 0$ για κάποιο ακέραιο zτότε $a\neq 0$ και $z\neq 0$ τότε $|a|\geq 1$ και $|z|\geq 1$ άρα $|a|\leq|a||z|=|b|$.

Θεώρημα 7.5.2

Για όλους τα $a, b\in \mathbb{Z}$ έχουμε $a|b$και $b|a$ αν και μόνο αν $a=\pm b$.Πιο συγκεκριμένα για κάθε $a\in \mathbb{Z}$ έχουμε $a|1$ αν και μόνο αν $a=\pm 1$

Απόδειξη

Εάν $a=\pm b$τότε $a|b$και $b|a$.Ας υποθέσουμε ότι $a|b$και $b|a$και θα αποδείξουμε ότι $a=\pm b$.Εάν το $a=0$ τότε $b=0$ και έαν $b=0$ τότε $a=0$.Θα υποθέσουμε ότι κανένα απο τα δύο δέν είναι μηδέν.Με αυτήν την παρατήρηση απο το $a|b$συνεπάγεται ότι $|a|\leq |b|$ και το $b|a$μας δίνει $|b|\leq|a|$ άρα $a=\pm b$.Για το δεύτερο μέρος της απόδειξης θέτουμε $b:=1$ ■

Επίσης να θυμίσουμε ότι το γινόμενο δύο μη μηδενικών ακεραίων είναι μηδέν.Αυτό φαίνεται απο τον νόμο της **απαλοιφής των στοιχείων**: εάν a,bκαι ceίναι ακέραιοι τέτοιοι ώστε a≠0 και ab=actότε πρέπει να έχουμε b=c.Αλλιώςa(b-c)=0⇒b-c=0⇒b=c

Τώραθα πρέπει να ορίσουμε τους πρώτους και τους σύνθετους **αριθμούς**.Έστω ο nνα είναι ένας θετικός ακέραιοςκαι θεωρούμε ότι το 1 και το nδιαιρούν το n.Εάν n>1 και κανένας θετικός ακέραιος εκτός απο το 1 και το n δεν διαιρούν το nτότε θα λέμε ότι το neίναι **πρώτος αριθμός**.Εάν n>1 και το nδέν είναι πρώτος τότε θα λέμε ότι είναι **σύνθετος**.Ο αριθμός 1 θεωρείται ότι δεν είναι ούτε πρώτος αλλα όυτε και σύνθετος αριθμός.Γενικά ο neίναι σύνθετος αριθμός αν και μόνο άν n=abγια κάποιους ακεραίους a,bμε 1<a<nκαι 1<b<n.Μερικοί πρώτοι αριθμοί είναι:

2, 3, 5, 7, 11, 13, 17…

Θεώρημα 7.5.3(θεμελιώδες θεώρημα της αριθμητικής)

Κάθε μη μηδενικός ακέραιος nμπορεί να εκφραστεί ώς:

$$n=\pm {p_1}^{e_1}\ldots..{p_r}^{e_r}$$

όπου $p_1,\ldots,p_r$είναι διαφορετικοί πρώτοι αριθμοί και $e_1,\ldots,e_r$είναι θετικοί ακέραιοι αριθμοί.Παρατηρήστε ότι έαν $n=\pm 1$ τότε r=0 και το γινόμενο μηδενικών όρων μας κάνει 1.Το θεώρημα μας λέει ότι μπορούμε να εκφράσουμε έναν μή μηδενικό ακέραιο ώς γινόμενο πρώτων αριθμών.Στη συνέχεια θα ξεκινήσουμε την απόδειξη του θεωρήματος αλλα και θα εισάγουμε κάποια εργαλεία που θα χρειαστούμε για να ολοκληρώσουμε τελικά την απόδειξη του.

Απόδειξη (ύπαρξη)

Η απόδειξη της ύπαρξης του θεωρήματος είναι έυκολη υπόθεση.Θα χρησιμοποιήσουμε τη μέθοδο της **επαγωγής**[64].Εαν n=1 τότε η δήλωση είναι αληθής μιας και το neίναι γινόμενο μηδενικών παραγόντων.Τώρα n>1 και υποθέτουμε ότι κάθε θετικός ακέραιος μικρότερος απο το nθα μπορεί να εκραστεί ώς γινόμενο πρώτων.Εάν ο neίναι πρώτος τότε η δήλωση είναι αληθής καθώς το neίναι είναι γινόμενο πρώτων.Υποθέστε τότε ότι το neίναι σύνθετος ακέραιος άρα θα υπάρχουν τα a,b∈ ℤμε 1<a<n , 1<b<n και n=ab.Απο επαγωγή πρέπει και το ακαι το bνα μπορούν να εκφραστούν σαν γινόμενο πρώτων όπως και το n ■

Η ύπαρξη της **μοναδικότητας** είναι το δύσκολο σημείο.Ένα σημαντικο συστατικό της απόδειξης είναι το παρακάτω θεώρημα:

Θεώρημα 7.5.4

Έστω τα a, b∈ ℤμε b>0.Τότε υπάρχουν μοναδικά q, r∈ ℤ τέτοια ώστε a=bq+rκαι 0≤r<b

Απόδειξη

Θεωρήστε το σέτ Sμη μηδενικών ακεραίων της μορφής a-btμε t ∈ ℤ.Αυτό το σετ είναι καθαρά μή κενο αφού έαν a≥0 τότεθέτουμε t:=0 και έαν a<0 τότε θέτουμε t:=a.Μιας και κάθε μη μηδενικό σέτ μη μηδενικών ακεραίων διαθέτει ένα ελάχιστο θα ορίσουμε το rσαν το μικρότερο στοιχείο του S.Απο τον ορισμό το reίναι της μορφής r=a-bqγια κάποιο q ∈ ℤ και r≥0.Επίσης πρέπει να έχουμε r<bαφού αλλίως το r-bθα

[64] Η μέθοδος της επαγωγής, https://en.wikipedia.org/wiki/Mathematical_induction

ήταν στοιχείο μικρότερο απο το r.Αν $r \geq b$θα μπορούσαμε να έχουμε $0 \leq r-b=a-b(q+1)$.Αυτο αποδεικνύει την ύπαρξη των rκαι q.

Για την μοναδικότητα υποθέστε ότι $a=bq+r$και $a=bq'+r'$ όπου $0 \leq r < b$και $0 \leq r' < b$.Τότε με αφαίρεση έχουμε $r'-r=b(q-q')$.Τότε $r'-r$είναι πολλαπλάσιο του b.Όμως $0 \leq r' < b$άρα $|r'-r| < b$.Για να γίνει αυτό πρέπει $r'-r=0 \Rightarrow 0=b(q-q')$ και $b \neq 0$ που μας δίνει ώς αποτέλεσμα $q-q'=0$ ■

Θεώρημα 7.5.5

Έστω $a,b \in \mathbb{Z}$με $b>0$ και θέστε ένα $x \in \mathbb{R}$.Τότε υπάρχουν μοναδικά $q,r \in \mathbb{Z}$ τέτοια ώστε $a=bq+r$και $r \in [x,x+b)$

Για να συνεχίσουμε με την απόδειξη του **θεωρήματος7.5.3** θα πρέπει να ορίσουμε την έννοια του**ιδεώδους** του$\mathbb{Z}$το οποίο είναι ένα μη κενό σύνολο ακεραίων το οποίο είναι κλειστό ως προς την πρόσθεση και επίσης κλειστό ώς προς τον πολλαπλασιασμό με έναν αυθαίρετο ακέραιο.Το μή κενό σετ $I \subseteq \mathbb{Z}$είναι **ιδεώδες** του $\mathbb{Z}$ αν και μόνο αν για όλα τα a, $b \in I$ και όλα τα $z \in \mathbb{Z}$ ισχύει:

$$\mathbf{a+b \in I \text{ και } a \cdot z \in I}$$

Είναι ένυκολο να δούμε ότι εάν ενα ιδεώδες υποσύνολο I περιέχει έναν ακέραιο ατότε θα περιέχει και τον –αμιας και $-a=a \cdot (-1) \in \mathbb{Z}$.Έτσι έαν περιέχει τα a,bτότε θα εμπεριέχει και το a-b.Το {0} λέγεται **τετριμμένο ιδεώδες**.Πιο συγκεκριμένα ενα ιδεώδες υποσύνολο Iείναι ίσο με το $\mathbb{Z}$ αν και μόνο αν το $1 \in I$που αυτό μας δείχνει ότι για κάθε $z \in \mathbb{Z}$ έχουμε $z=1 \cdot z \in I$και άρα $I=\mathbb{Z}$.Τότε το I λέγεται **μή γνήσιο ιδεώδες** του $\mathbb{Z}$.

Για όλα τα $a \in \mathbb{Z}$ ορίζουμε το $a\mathbb{Z}:=\{az: z \in \mathbb{Z}\}$.Αρα το $a\mathbb{Z}$ είναι ένα σετ που περιέχει όλα τα πολλαπλάσια του a.Εάν a=0 τότε $a\mathbb{Z}=\{0\}$ αλλίως το $a\mathbb{Z}$ περιέχει διαφορετικούς ακεραίους :

$$\mathbf{\ldots,-3a, -2a, -a, 0, a, 2a, 3a,...}$$

Είναιεύκολο να δούμε ότι το $a\mathbb{Z}$ είναι ιδεώδες αν :για όλα τα $az,az' \in \mathbb{Z}$ και $z'' \in \mathbb{Z}$ έχουμε ότι $az+az'=a(z+z') \in a\mathbb{Z}$ και $(az)z''=a(zz'') \in a\mathbb{Z}$.Το ιδεώδες $a\mathbb{Z}$ ονομάζεται ιδεώδες υποσύνολο που γεννήθηκε απο το a.

Παρατηρήστε ότι για όλα τα a, $b \in \mathbb{Z}$ έχουμε $b \in a\mathbb{Z}$αν και μόνο αν $a|b$.Παρατηρήστε ακόμη ότι για κάθε ιδεώδες I έχουμε $b \in I$αν και μόνο αν $b\mathbb{Z} \subseteq I$.Εάν συνδυάσουμε αυτές τις δύο παρατηρήσεις θα δούμε ότι $b\mathbb{Z} \subseteq a\mathbb{Z}$αν $a|b$.Υποθέστε τώρα ότι I_1, I_2είναι ιδεώδη.Μπορούμε να δούμε ότι :

$$\mathbf{I_1+I_2=\{a_1+a_2 : a_1 \in I_1,\ a_2 \in I_2\}}$$

Τοοποίοείναιεπίσηςιδεώδες.Υποθέστεεπίσηςότι$a_1+a_2 \in I_1+I_2$ και$b_1+b_2 \in I_1+I_2$.Τότεέχουμεότι $(a_1+a_2)+(b_1+b_2)=(a_1+b_1)+(a_2+b_2) \in I_1+I_2$ καιγιακάθε$z \in \mathbb{Z}$ έχουμεότι $(a_1+a_2)z=a_1z+a_2z \in I_1+I_2$

Θεώρημα 7.5.6

Έστω το Ινα είναι ένα ιδεώδες του $\mathbb{Z}$.Τότε υπάρχει ένας μή μηδενικός ακέραιος dτέτοιος ώστε $I=d\mathbb{Z}$

Απόδειξη

Πρώτα θα αποδείξουμε την ύπαρξη του θεωρήματος.Εαν το $I=\{0\}$ τότε d=0.Αρα θα υποθέσουμε ότι $I \neq 0$.Αφου το I περιέχει μη μηδενικούς ακεραίους τότε θα πρέπει να

περιέχει θετικούς ακεραίους μιας και έαν a∈ Ιτο ίδιο θα πρέπει να ισχύει και για το –a.Έστω ότι deίναι το μικρότερο στοιχείο του συνόλο στο I θα δείξουμε ότι I=d ℤ.

Στη συνέχεια θα δείξουμε ότι I ⊆ ℤ.Για να γίνει αυτό θα θέσουμε το ανα είναι οποιαδήποτε στοιχείο του I.Τοτε θ.δ.ο d|a.Χρησιμοποιώντας το **θεώρημα 7.5.4**έχουμε ότι a=dq+rόπου 0≤r<dκαι λόγω των ιδιοτήτων της κλειστότητας των ιδεωδών βλεπουμε ότι το r=a-dq είναι επίσης στοιχείο του I και πρέπει να έχουμε r=0 δηλαδή d|a.

Αυτό απέδειξε την ύπαρξη του θεωρήματος.Τώρα μένει να δείξουμε την μοναδικότητα.Προσέξετε ότι εάν d ℤ=e ℤγια κάποιον μή μηδενικό ακέραιο ετότε d|εκαι e|d όπου απο το **θεώρημα 7.5.2**, d=±e.Αφου τα d,eείναι μη μηδενικά πρέπει d=e■

Πολυ σημαντικές είναι οι έννοιες που αφορούν τους **κοινούς** και τον **μέγιστο κοινόδιαιρέτη** δύο ακεραίων αριθμών.Για a,b∈ ℤ θα λέμε ότι το d∈ ℤ είναι **κοινόςδιαιρέτης** των ακαι bαν d|ακαι d|b.Επίσης ο d καλείται **μέγιστος κοινός διαιρέτης** έαν το deίναι μή αρνητικό και αν όλοι οι κοινοί διαιρέτες των a, b**διαιρούν** το d.

Θεώρημα 7.5.7
Για όλα τα a, b∈ ℤυπάρχει ο μέγιστος κοινός διαιρέτης dτων a,bκαι ισχυει ότι a ℤ+b ℤ=d ℤ

Απόδειξη
Θα εφαρμόσουμε το θ**εώρημα 7.5.6** στο I:= a ℤ+b ℤ.Έστω d ∈ ℤμε I=d ℤ.Θέλουμε να δείξουμε ότι ο deίναι ο μέγιστος κοινός διαιρέτης των a, b.Σημειώστε ότι τα a, b, d ∈ Ιείναι μη αρνητικά.Μιας και a ∈ I=d ℤ βλέπουμε ότι d|a.Ομοίως d|b.Τότε βλέπουμε οτι το deίναι ο κοινός διαιρέτης των ακαι b.Επειδή d ∈ I = a ℤ+b ℤ τότε υπάρχουν s, t τέτοια ώστε s, t ∈ ℤτέτοια ώστε as+bt=d.Υποθέτουμε τώρα ότι a=a´d´ και b=b´d´ για κάποια a´,b´, d´ ∈ ℤ.

Τότε η εξίσωση as+bt=d μας έχει σαν αποτέλεσμα ότι d´(a´s+b´t)=dτο οποίο μας λέει ότι d´|d.Έτσι κάθε κοινός διαιρέτης d´ διαιρεί τα ακαι b.Αυτο μας αποδεικνύει ότι ο deίναι ο μέγιστος κοινός διαιρέτης των a, b.Για μοναδικότητα παρατηρήστε ότι έαν το eείναι ο μέγιστος κοινός διαιρέτης των a,bτότε d|εκαι e|d άρα d=±εκαι αφού d,eμή αρνητικοί τότε d=e■

Για a, b∈ ℤ γράφουμε **gcd(a, b)** για τον μέγιστο κοινό διαιρέτη των a ,b.Θα λέμε ότι τα a, b∈ ℤείναι **σχετικά πρώτοι** μεταξύ τους έαν gcd(a, b)=1 το οποίο είναι το ίδιο σαν να λέγαμε ότι ο μοναδικός κοινός διαιρέτης των a, bείναι ±1

Μια πολύ σημαντική έννοια που απορρέει απο το **θεώρημα 7.5.7** είνα ότι:Για όλους τους ακεραίους x, yυπάρχουν ακέραιοι a,bτέτοιοι ώστε **a·x+b·y=gcd(x , y)**.Τα a,bμπορούν να υπολογιστούν πολύ έυκολα με τη χρήση του αλγορίθμου του Ευκλείδη[65]

[65] Ο αλγόριθμος του Ευκλείδη, https://en.wikipedia.org/?title=Euclidean_algorithm

Θεώρημα 7.5.8
Έστωa, b, r ∈ ℤ καιd:=gcd(a, b).Τότε υπάρχουν s, t ∈ ℤτέτοια ω΄στε as+bt=rαν και μόνο αν d|r.Πιο συγκεκριμένα τα aκαι bείναι σχετικα πρώτοι μεταξύ τους έαν και μόνο αν υπάρχουν ακέραιοι τέτοιοι ώστε as+bt=1.

Απόδειξη
as+bt=rγιακάποιας,t∈ ℤ ⇔ r ∈a ℤ+b ℤ ⇔r ∈ d ℤ ⇔d|r.Εάνθέσω r:=1 τότε αποδεικνύωκαι την μοναδικότητα.

Θεώρημα 7.5.9
Έστω a,b , c∈ ℤτέτοια ώστε c|abκαι gcd(a, c)=1.Τότε c|b.

Απόδειξη
Υποθέστε ότι c|abκαι gcd(a, c)=1.Αφού gcd(a, c)=1απο το **θεώρημα 7.5.8**έχουμε ότι as+ct=1 για κάποια s, t∈ ℤ.Πολλαπλασιάζοντας την εξίσωση με bέχουμε ότι abs+cbt=b.Μιας και το cδιαιρεί το abαπο την υπόθεση τότε το cδιαιρεί και καθαρά το cbtάρα διαιρεί το αριστερό μέλος της εξίσωσης και τότε το cδιαρεί το aκαι b■

Παρατήρηση
Υποθέστε ότι ο ρείναι πρώτος και και ο aείναι οποισδήποτε ακέραιος.Εαν οι μοναδικοί διαιρέτες του ρείναι οι ± 1 και $\pm$ρτότε έχουμε ότι :
p|a ⇒gcd(a, p)=p

pła ⇒gcd(a, p)=1

Θεώρημα 7.5.10
Έστω οτι το ρείναι πρώτος και ας θέσουμε a, b∈ ℤ.Τότε p|abμας δίνει p|aή p|b.

Απόδειξη
Υποθέστε ότι p|ab.Έαν p|ατελειώσαμε.Τότε θα υποθέσουμε ότι pła.Απο την παραπάνω παρατήρηση gcd(a, p)=1 και τότε απο το **θεώρημα 7.5.9** έχουμε p|b■

Τώρα μπορούμε να συνεχίσουμε την απόδειξη του **θεωρήματος 7.3**.Αποδείξαμε την **ύπαρξη** του θεωρήματος σε προηγούμενες σελίδες και τωρα μένει να δείξουμε την μοναδικότητα :

Απόδειξη θεωρ 7.5.3(μοναδικότητα)
Θα αποδέιξουμε την μοναδικότητα του θεωρήματος που αυτό σημαίνει εάν τα $p_1,\ldots p_r$είναι πρώτοι αριθμοί και όχι απαραίτητα διαφορετικοί μεταξύ τους και $q_1,\ldots,q_r$είναι επίσης πρώτοι τέτοιοι ώστε:

$$p_1 \cdots p_r = q_1 \cdots q_s$$

τοτε $(p_1,\ldots p_r)$ είναι απλά μια αναδιάταξη των $(q_1,\ldots q_s)$.Θα το αποδέιξουμε με επαγωγή.Έαν r=0 θα πρέπει να έχουμε s=0 και τελειώσαμε.Τώρα υποθέστε ότι r>0 και η δήλωση στέκεται για r-1.Μιας και r>0 πρέπει να έχουμε καθαρά s>0.Αφου τοp_1διαιρεί το αριστερό μέλος της σχέσης τοτε το q_jθα διαιρεί το δεξί μέλος της σχέσης.Απο το **θεώρημα 7.5.10** έχουμε ότι $p_1|q_j$για κάποιο j=1,…s και έαν το q_jείναι πρώτος αριθμός τότε $p_1=q_j$.Ετσι μπορούμε να αφαιρέσουμε το p_1 απο το αριστερό μέλος και το q_jαπο το δεξί μέλος.Αυτο αποδυκνείει την μοναδικότητα του θεωρήματος 7.3 ■

Τώρα θα αναφερθούμε στο **ελάχιστο κοινό πολλαπλασίο**δύο ακεραίων αριθμών.Για κάποια a, b∈ ℤ το **κοινό πολλαπλασιο** των a, bείναι ένας ακέραιος mτέτοιος ώστε a|mκαι b|m.Πιο ειδικά το mλέγεται ελάχιστο κοινό πολλαπλάσιο των a,b έαν ο meίναι μή μηδενικός ακέραιος και διαιρεί όλα τα κοινά πολλαπλάσια των a, b.Είναι έυκολο να δούμε ότι το κοινό πολλαπλάσιο υπάρχει και είναι μοναδικό και συμβολίζεται ώς **lcm(a ,b)**.Για όλα τα a, b ∈ ℤ εάν ή το aή το bείναι μηδέν τότε το μοναδικό πολλαπλάσιο των a, bείναι το μηδέν αλλίως :

$$\text{lcm}(a, b) = \prod_p p^{\max(\nu_p(a),\ \nu_p(b))}$$

Ακόμη καλύτερα μπορεί να θεωρηθεί ως τον μικρότερο ακέραιο που μπορεί να διαιρεθεί απο τα ακαι b.

7.5.2Ισοτιμιες και επίλυση γραμμικών εξισώσεων

Πρώτα θα αναφερθούμε στις ιδιότητες των σχέσεων της ισοδυναμίας και μετά θα αναφερθούμε στις ισοτιμίες.Έστω το Sνα είναι ένα σετ.Μια διαδική σχέση ~ καλείται ισοδύναμη σχέση έαν ισχύει ότι:

- **Ανακλαστική**: a~a για όλα τα a∈ S
- **Συμμετρική**: a~a μας δίνει b~a για όλα τα a, b∈ S
- **Μεταβατική**: a~b και b~c τότε a~c για όλα τα a, b, c ∈ S

Εάν ~ είναι μια σχέση ισοδυναμίας στο S, τότε για a ∈ Sορίζουμε την **κλάσηισοδυναμίας** να έιναι το σύνολο {x ∈ S : x~a}

Θεώρημα 7.5.11

Έστω n να είναι ένας θετικός ακέραιος.Για όλα τα a, b, c∈ ℤ έχουμε:

- a≡b (mod n)
- a≡b (mod n) μαςδίνει b≡a (mod n)
- a≡b (mod n) και b≡c(mod n) μαςδίνει a≡c (mod n)

Απόδειξη

Για το a≡b (modn) απλά παρατηρούμε ότι 0=a-a, για το δέυτερο παρατηρούμε ότι έαν το nδιαιρεί τα a-bτότε θα διαιρεί και το -(a-b)=b-a.Για το τρίτο έαν το nδιαιρεί το a-bκαι το b-cτότε διαιρεί και το (a-b)+(b-c)=a-c■

Ας δούμε μερικές **βασικές ιδιότητες**:

- Ισοτιμιες με το ίδιο μέτρο μπορούν να προστεθούν, να αφαιρεθούν ή να πολλαπλασιαστούν κατα μέλη
- Τα δύο μέλη μιας ισοτιμίας μπορούν να πολλαπλασιαστούν με τον ίδιο αριθμό.
- Αν $f(x_1,..x_n)$ είναι μια πολυωνυμική παράσταση με ακεραίους συντελεστές $a_i \equiv b_j \bmod(m)$ για i=1,…n τότε $f(a_1,\ldots,a_n)=f(b_1,\ldots,b_n) \bmod(m)$
- Τα δύο μέλη μιας ισοτιμίας μπορούν να πολλαπλασιαστούν με τον ίδιο αριθμό.
- Τα μέλη της ισοτιμίας μπορούν να διαιρεθούν με έναν κοινό διαιρέτη των δύο μελών της ισοτιμίας, αρκεί αυτός ο διαιρέτης να είναι πρώτος προς το μέτρο.
- Αν a≡b (modm) και d≥2 να είναι ο διαιρέτης του m, τότε a≡b (modd)
- Αν a≡b (mod m) τότε (a, m)= (b, m)

Θεώρημα 7.5.12

Έστωταa,a΄,b,b΄, ημεn ∈ ℤ μεn>0.Έανa≡a΄ (modn) καιb≡b΄ (mod n) τότεθαισχύειότια+b≡a΄+b΄ (modn) καια·b=a΄·b΄ (modn)

Έστω ότιτο γράμμα Nθα χρησιμοποιείται για να αναφερθουμε σε θετικούςακεραίους, και το γράμμα ργια αναφερθούμε σε έναν θετικό πρώτο αριθμόκαι το σύνολο $\mathbb{Z}_N=\{0,1,2...N-1\}$ το οποίο θα ορίζει έναν δακτύλιο όπου γίνονται προσθέσεις και πολλαπλασιασμοί modulo και είναι πολύ χρήσιμο στην κρυπογραφία γενικότερα.

Ορισμός

Η **αντιστροφή** ενός στοχείου xστο $\mathbb{Z}_N$είναι ένα άλλο στοιχείο στο $\mathbb{Z}_N$ τέτοιο ώστε να ισχύειx·y=1 στο $\mathbb{Z}_N$και τοσυμβολίζεται ώςx^{-1}.Με άλλα λόγια x·y= 1 modN.

Για παράδειγμα έστω Nνα είναι ένας περιττός αριθμός.Ποιός είναι ο αντίστροφος του αριθμού 2 στο$\mathbb{Z}_N$;Είναι ο N+1/2.Για να το επαληθεύσουμε μπορούμε απλά να κάνουμε 2·(N+1/2)=N+1=1 στο $\mathbb{Z}_N$.

Λήμμα: το οποίο μας λέει ότι έαν, το xπου ανήκει στο $\mathbb{Z}_N$έχει αντίστροφο εάν και μόνο άν **gcd(x, N)=1**

Θα πρέπει να ορίσουμε ένα νέο σύνολο το οποίο θα μας είναι πολύ χρήσιμο στη συνέχεια.Το σύνολο αυτό **συμβολίζεται ως**$\mathbb{Z}_N^*$ και περιέχει όλα τα αντιστρέψιμα στοιχεία του συνόλου $\mathbb{Z}_N$.Με συμβολισμούς έχουμε $\mathbb{Z}_N^*=\{x\in \mathbb{Z}_N : gcd(x, N)=1 \}$

Εαν μας δώσουν μια εξίσωση της μορφής a·x+b=0 στο $\mathbb{Z}_N$τότε η λύση της είναι η$\mathbf{x=-b\cdot a^{-1}}$στο $\mathbb{Z}_N$.Το a^{-1}μπορεί να βρεθεί πολύ έυκολα χρησιμοποιώντας τον αλγόριθμο του Ευκλείδη σε χρόνο O(**$\log^2$N**).

7.5.3Fermat[66]και Euler[67]

Σε αυτήν την ενότητα θα χρησιμοποιήσουμε θεώρηματα του Fermatκαι του Euler για να εξάγουμε κάποια σημαντικά αποτελέσματα που θα μας χρειαστούν στη συνέχεια.Θα μιλήσουμε για την δομή του Z_p*,τον βαθμό της γεννήτριας gπάνω στο $\mathbb{Z}_p$* ,την γενίκευση το θεωρήματος Fermataπο τον Eulerμε χρήση της συνάρτησης φ και πώς βρίσκουμε τον αντίστροφο.

ΘέώρημαFermat 7.5.14

Υποθέστε ότι σας δίνουν έναν περιττό αριθμό p.Τότε για κάθε στοιχείο xπου ανήκει στο $\mathbb{Z}_N$* ισχύει ότι $\mathbf{x^{p-1}=1}$ **στο** $\mathbf{Z_p}$.Έχουμε ότι $\mathbf{x\in(\mathbb{Z}_p)\Rightarrow x\cdot x^{p-2}=1\Rightarrow x^{-1}=x^{p-2}}$**στο** $\mathbf{Z_p}$.Αυτός είναι ένας τρόπος υπολογισμού του αντιστρόφου αλλα είναι λιγότερο αποδοτικός απο τον αλγόριθμο του Ευκλείδη αφού παίρνει O(**$\log^3$p**).

Ας προχωρήσουμε σε αυτό που ονομάζουμε**κατασκευή του (Zp)***που είναι θεώρημα του **Euler**.Το (Z_p)* (αναφέρται και ώς**κυκλική ομάδα**).Τι σημαίνει όμως ότι το (Z_p)* είναι μια κυκλική ομάδα;Σημαίνει ότι:

$$\mathbf{\exists\, g\in(Z_p)^* : \{1, g, g^2, g^3,\ldots,g^{p-2}\}=(Z_p)^*}$$

[66] Ο Πιερ ντε Φερμάτ, https://en.wikipedia.org/wiki/Pierre_de_Fermat
[67] Ο λεοναρντ Οιλερ, https://en.wikipedia.org/?title=Leonhard_Euler

Οπου το gονομάζεται γεννήτρια(generator) του $(Z_p)^*$.Οι δυνάμεις του gμας δίνουν όλα τα στοιχεία του συνόλου $(Z_p)^*$.Ας δούμε ένα ένα παράδειγμα πάνω σε αυτό:

Υποθέστε ότι p=7.Τότε $\{1, 3, 3^2, 3^3, 3^4, 3^5\}=\{1, 2, 6, 4, 5\}=(Z_7)^*$.Όμως δεν είναι όλα τα στοιχεία γεννήτριες.Για παράδειγμα έαν παίρναμε τις δυνάμεις του 2 τότε θα είχαμε $\{1, 2^2, 2^3, 2^4, 2^5\}=\{1, 2, 4\}$.Έπισης αξίζει να θυμόμαστε ότι για κάθε $g\in(Z_p)^*$ το σετ $\{1, g^2, g^3...\}$ που γεννάται απο το g**συμβολίζεται με** <g>.Ονομάζουμε **βαθμό**(order) του $g\in(Z_p)^*$ (που είναιουσιαστικά είναι το μέγεθος του <g>):

$$\mathbf{ord_p(g)=|<g>|=(smallest\ a>0\ s.t\ g^a=1\ in\ Z_p)}$$

Ας δούμε μερικά παραδείγματα:

- **$ord_7(3)=6$**.Εδώ ζητάμε πιό είναι το order του 3mod7.Ουσιαστικά ρωτάμε ποιό είναι το μέγεθος της ομάδας ώστε το 3 να γεννάει το mudulo 7.Είναι το 3 η γεννήτρια του 7;Το 3 γεννάει όλα τα στοιχεία που υπάρχουν στον $(Z_7)^*$.Υπάρχουν 6 στοιχεία(elements) στο $(Z_7)^*$ και βλέπουμε ότι $ord_7(3)=6$.
- **$ord_7(2)=3$.**Εάν κοιτάξουμε στο σετ των δυνάμεων 2 mod 7 τότε έχουμε οτι $\{1, 2^1, 2^2\}=\{1, 2, 4\}$.Οπότε $|\{1, 2, 4\}|=3$
- **$ord_7(1)=1$.**Το αποτέλεσμα μας είναι 1 διότι έαν κοιτάξουμε στην ομάδα τότε θα δούμε ότι υπάρχει μόνο ένα στοιχείο εκεί που είναι το 1 δηλαδή {1} άρα έχουμε ότι $|\{1\}|=1$.

Ενα πολύ σημαντικό θεώρημα(**θεώρημα 7.5.15**) είναι αυτο τουLangrage[68]που μας λέει ότι

$$\mathbf{\forall g \in (Z_p)^*\ :ord_p(g)=\ διαιρεί\ το\ p-1.}$$

Εάν κάνουμε εφαρμογή του **θεωρημάτος 7.5.15** στα προηγούμενα παραδείγματα θα δούμε ότι το 6 διαιρεί το 7-1 καθως το 6 διαιρεί το 6, το 3 διαιρεί το 7-1 αφού το 3 διαιρεί το 6 και το 1 διαιρεί το7-1.

Θεώρημα Euler7.5.16

Για έναν ακέραιο αριθμό N, **$\varphi(N)=|(Z_N)^*|$ όπου φ(N) η συνάρτηση Euler.**Για παράδειγμα φ(12)=|{1, 5, 7, 11| οπότε **$\varphi(p)=p-1$**.Γενικά έαν τύχει N=p·qτότε θα ισχύει φ(N)=N-p-q+1=(p-1)(q-1).Ισχύει για κάθε χανήκει στο $(Z_N)^*$οτι $\mathbf{x^{\varphi(N)}=1}$ στο Z_N.

7.5.4 Επίλυση τετραγωνικώνεξισώσεων

Έστω ρνα είναι ένας περιττός πρώτος αριθμόςκαι cνα ανήκει το Z_p.Μπορούμε να λύσουμε εξισώσεις της μορφής $x^2-c=0$, $y^3-c=0$, $z^{37}-c=0$ στο Z_p.Για παράδειγμα στο πρώτο πολυώνυμο πρέπει να υπολογίσουμε $\sqrt[2]{x}$, στο δέυτερο την $\sqrt[3]{y}$, ενώ στο τρίτο θέλουμε να υπολογίσουμε την $\sqrt[37]{z}$.

Ορισμός

Οπως και πρίν ρείναι περιττός αριθμόςκαι $c \in Z_p$ τότε υπάρχει $x \in Z_p$τέτοιο : $\mathbf{x^e=c}$στο Z_pτο οποίο ονομάζεται**modulare'throot**τους(modularεκθετική ρίζα του c).

Ας δούμε μερικά παραδείγματα που βασίζονται στον ορισμό:

---

[68]OLangrage, https://en.wikipedia.org/wiki/Joseph-Louis_Lagrange

- $\mathbf{7^{1/3}=6}$ στο Z_{11}.Επειδή $6^3=216=7$ στο Z_{11}.Οπότε η κυβική ρίζα 7mod11 ισούται με 6
- $\mathbf{3^{1/2}= 5}$ στο Z_{11} διότι $5^2=25$ το οποίο είναι 3 mod 11.Αφού 25mod11=7
- $\mathbf{1^{1/3}=1}$ στο Z_{11} διότι ψάχνω έναν αριθμό που άμα τον υψώσω στο τετράγωνο και κάνω την πράξη αυτού του αποτελέσματος με το 11 θα πρέπει να έχω αποτέλεσμα 1.

Να σημειώσουμε επίσης ότι αυτές οι ρίζες **δεν υπάρχουν πάντα**.Παραδείγματος χάρη δέν υπάρχει το $2^{1/2}$ στο Z_{11}.Ενα πολύ σημαντικό ερώτημα είναι εκτός απο το εαν υπάρχουν αυτές οι ρίζες, ειναι έαν μπορούν να υπολογιστούν «αποδοτικά».Υποθέστε ότι θέλουμε να υπολογίσουμε το **gcd(e, p-1)=1**.

Τότε για όλα τα c στο (Zp)* : c1/e∃ στο Zp και είναι πολύ έυκολο να υπολογιστεί.Μιας και το e είναι σχετικά πρώτος με το p-1 γνωρίζουμε ότι το e έχει αντίστροφο που τον ονομάζουμε d και ισχύει d=e-1 στο $Z_{p-1}(1)$.Βλέπουμε ότι $c_1/e=c^d$ στο Zp(2)Η τελευταία εξίσωση μας δείχνει ότι για όλα τα c που ανήκουν στο Zp η **εκθετικες ρίζες αυτών υπάρχουν**.

Δείξαμε τελικά ότι η εκθετική ρίζα υπάρχει όταν το e είναι σχετικά πρώτο με το p-1 και είναι πολύ έυκολο να υπολογιστεί χρησιμοποιώντας την εξίσωση (2).Τώρα θα δούμε τι θα γίνει στην περίπτωση που το e**δέν είναι σχετικά πρώτο με το p-1**.Για παράδειγμα e=2.Πρώτα όμως θα δώσουμε τον ορισμό των **τετραγωνικών ισοϋπολοίπων**(quadraticresidues)

Ορισμός
Έστω ο ακέραιος m>1 και ο απρώτος ώς προς τον m.Αν η ισοτιμία $x^2\equiv a$ (modm) έχει λύση τότε αχαρακτηρίζεται ώς **τετραγωνικό ισοϋπόλοιπο** μέτρο m.Διαφορετικά **τετραγωνικό ανισοϋπόλοιπο** μέτρο m.Αν a≡b (modm) είναι προφανές ότι το bείναι ισοϋπόλοιπο μέτρο m.

Θεώρημα 7.5.17
Έστω περιττός πρώτος p τότε έχουμε :

- Ένα περιορισμένο σύστημα υπολοίπων μέτρο p περιέχει ακριβώς p-1/2 το πλήθος τετραγωνικά ισοϋπόλοιπα τα οποία είναι ισότιμα με τους αριθμούς 1^2, 2^2,…$(p-1/2)^2$(1)
- Έστω (a, p)=1.Αν ο a είναι τετραγωνικό ισοϋπόλοιπο p τότε $a^{p-1/2}\equiv 1$ (mod p) (2) ενώ έαν ο aείναι τετραγωνικό ανισοϋπόλοιπο τοτε $a^{p-1/2}\equiv$ (modp) (3)

Για έμας ένα $x \in \mathbb{Z}_p$είναι τετραγωνικό ισοϋπόλοιπο(quadraticresidueQ.R) έαν έχει τετραγωνική ρίζα στο $\mathbb{Z}_p$.Αν ρένας περιττός πρώτος αριθμός , τότε ο αριθμός των τετραγωνικών ισοϋπολοίπων είναι p-1/2+1 επειδή γνωρίζουμε ότι ακριβώς τα μισά στοιχεία του $\mathbb{Z}_p$* έχουν τετραγωνικά ισοϋπόλοιπα λόγω της squareσυνάρτησης που είναι 2 προς 1.Ομώς στο $\mathbb{Z}_p$έχουμε και το μηδέν μιας και αυτό έχει πάντα τετραγωνικό ισοϋπόλοιπο επειδή $0^2=0$, όποτε προσθέτουμε +1.Έαν μας δώσουν ένα στοιχείο x το οποίο θα ανήκει στο $\mathbb{Z}_p$θα μπορούμε να πούμε έαν έχει τετραγωνική ρίζα ή όχι;Θα το δούμε αυτό στο παρακάτω θεώρημα

Θεώρημα 7.5.18

Εάν $x\in (\mathbb{Z}_p)^*$ έχει τετραγωνική ρίζα(είναι τετραγωνικό ισοϋπόλοιπο) έαν $x^{(p-1)/2}=1$ στο $\mathbb{Z}_p$

Σημείωση: $x\neq 0 \Rightarrow x^{(p-1)/2}=(x^{p-1})^{1/2}=1^{1/2}\in\{1, -1\}$ στο Z_p.Το σύμβολο $x^{(p-1)/2}$ονομάζεται σύμβολο του Legendre[69](1798)

Ένα ακόμη θέμα είναι πως θα υπολογίσουμε αυτές τις ρίζες.Υπάρχουν δύο περιπτώσεις.Πρώτα θα αναφερθούμε στη περίπτωση που p=3 (mod 4).

Λήμμα: εάνc ∈ Z_pκαι είναι ένα τετραγωνικό ισοϋπόλοιπο τότε $\sqrt{c}=c^{(p+1)/4}$στο Z_p.

Απόδειξη
$[c^{(p+1)/4}]^2=c^{(p+1)/2}=c^{p-1/1}\cdot c=1\cdot c=c$στο Z_p■

7.6Ερωτήσειςκεφαλαίου

Ερώτηση 1

Consider the toy key exchange protocol using an online trusted 3rd party (TTP) discussed in Lecture 9.1. Suppose Alice, Bob, and Carol are three users of this system (among many others) and each have a secret key with the TTP denoted k_a,k_b,k_c respectively.

They wish to generate a group session key k_{ABC} that will be known to Alice, Bob, and Carol but unknown to an eavesdropper. How would you modify the protocol in the lecture to accomodate a group key exchange of this type? (note that all these protocols are insecure against active attack.

a) Alice contacts the TTP. TTP generates a random k_{ABC} and sends to Alice
$E(k_a,k_{ABC}),ticket_1\leftarrow E(k_b,k_{ABC}),ticket_2\leftarrow E(k_c,k_{ABC})$.
Alice sends k_{ABC} to Bob and k_{ABC} to Carol.

b) Alice contacts the TTP. TTP generates a random k_{ABC} and sends to Alice
$E(k_a,k_{ABC}),ticket_1\leftarrow E(k_c,E(k_b,k_{ABC})),ticket_2\leftarrow E(k_b,E(k_c,k_{ABC}))$.
Alice sends k_{ABC} to Bob and k_{ABC} to Carol.

c) Bob contacts the TTP. TTP generates random k_{ABC} and sends to Bob
$E(k_b,k_{ABC}),ticket_1\leftarrow E(k_a,k_{ABC}),ticket_2\leftarrow E(k_c,k_{ABC})$.
Bob sends $ticket_1$ to Alice and $ticket_2$ to Carol.

d) Bob contacts the TTP. TTP generates a random k_{AB} and a random k_{BC}.
It sends to Bob
$E(k_a,k_{AB}),ticket_1\leftarrow E(k_a,k_{AB}),ticket_2\leftarrow E(k_c,k_{BC})$.
Bob sends $ticket_1$ to Alice and $ticket_2$ to Carol.

Απάντηση

The protocol works because it lets Alice, Bob, and Carol obtain k_{ABC} but an eavesdropper only sees encryptions of k_{ABC} under keys he does not have.

---

[69]OLegendre, https://en.wikipedia.org/wiki/Adrien-Marie_Legendre

Ερώτηση 2

Let G be a finite cyclic group (e.g. $G=\mathbb{Z}_p^*$) with generator g. Suppose the Diffie-Hellman function $DH_g(g^x,g^y)=g^{xy}$ is difficult to compute in G. Which of the following functions is also difficult to compute?

d)$f(g^x, g^y)=(\sqrt{g})^{x+y}$
b)$f(g^x, g^y)=g^{xy+x+y+1}$
c)$f(g^x, g^y)=g^{x(y+1)}$
d)$f(g^x, g^y)=g^{x-y}$

Απάντηση

$$\mathbf{f := g^{(xy + x + y + 1)}}$$

This function is difficult to compute. Suppose an attacker can compute f.
He could then compute DH with the equation
$DH(g^x, g^y) = g^{xy} = f(g^x, g^y)/(g \cdot g^x \cdot g^y)$

$$\mathbf{f := g^{x(y+1)}}$$

This function is difficult to compute. Suppose an attacker can compute f.
He could then compute DH with the equation
$DH(g^x, g^y) = g^{xy} = f(g^x, g^y)/g^x$

$$\mathbf{f := g^{(x-y)}}$$

This function is not difficult to compute. Specifically, an attacker can compute f by simply computing $g^x/g^y = g^{(x-y)}$

$$\mathbf{f:=} (\sqrt{g})^{x+y}$$

This function is not difficult to compute. Specifically, an attacker could compute f from $sqrt(g^x \cdot g^y) = f(g^x, g^y)$

So the correct answers are **b** and **c**

Ερώτηση 3

Suppose we modify the Diffie-Hellman protocol so that Alice operates as usual, namely chooses a random a in {1,…,p−1} and sends to Bob $A \leftarrow g^a$. Bob, however, chooses a random b in {1,…,p−1} and sends to Alice $B \leftarrow g^{1/b}$. What shared secret can they generate and how would they do it?

a)Secret=$g^{a/b}$. Alice computes the secret as $B^a$and Bob computes $A^{1/b}$.
b)Secret=g^{ab}. Alice computes the secret as $B^{1/a}$and Bob computes A^b.
c)Secret=$g^{b/a}$. Alice computes the secret as B^a and Bob computes $A^{1/b}$.
d)Secret=$g^{a/b}$. Alice computes the secret as $B^{1/a}$ and Bob computes A^b.

Απάντηση

The correct answer is **a**.Since Bob sends Alice $g^{1/b}$, she has no way of computing g^{ab}.The secret key must therefore be $g^{a/b}$. They generate this key as B^a and $A^{1/b}$. In other words: secret = $g^{a/b}$. Alice computes the secret as B^a and Bob computes $A^{1/b}$.

Ερώτηση 4

Consider the toy key exchange protocol using public key encryption described in Lecture 9.4. Suppose that when sending his reply $c \leftarrow E(pk,x)$ to Alice, Bob appends a MAC t:=S(x,c) to the ciphertext so that what is sent to Alice is the pair (c,t).

Alice verifies the tag t and rejects the message from Bob if the tag does not verify. Will this additional step prevent the man in the middle attack described in the lecture?

a)It depends on what MAC system is used.
b)It depends on what public key encryption system is used.
c) Noandyes

Απάντηση
The additional step does not prevent the MitM attack. An attacker canintercept the initial message from Alice to Bob, and replace pk with pk'. He sends pk' to Bob, who responds with (E(pk', x), S(x, E(pk', x)). The attacker then decrypts E(pk', x) to determine x, computes E(pk, x) and sends (E(pk,x),S(x,E(pk,x)) to Alice. Alice sees that the MAC is correct and communicates insecurely with Bob.

Ερώτηση 5
The numbers 7 and 23 are relatively prime and therefore there must exist integers a and b such that 7a+23b=1. Find such a pair of integers (a,b) with the smallest possible a>0. Given this pair, can you determine the inverse of 7 in $\mathbb{Z}_{23}$?

Απάντηση
$7\times10+23\times(-3)=1$. Therefore $7\times10=1$ in $\mathbb{Z}_{23}$ implying that $7^{-1}=10$ in $\mathbb{Z}_{23}$.

Ερώτηση 6
Solve the equation $3x+2=7$ in $\mathbb{Z}_{19}$.

Απάντηση
$3x + 2 = 7 \Rightarrow \quad 3x = 5 \Rightarrow x = 5 \cdot (3^{-1}) = 5 \cdot 13 = 65 \bmod 19 = 8$

Ερώτηση 7
How many elements are there in $_{35}{}^{*}$?

Απάντηση
We aim to find $|\mathbb{Z}_{35}^{*}|$ .Since $35 = 7\cdot5$, $|\mathbb{Z}_{35}^{*}| = 35 - 7 - 5 + 1 = 24$

Ερώτηση 8
How much is $2^{10001} \bmod 11$? (Please do not use a calculator for this)
Hint: use Fermat's theorem.

Απάντηση
We aim to find $2^{10001} \bmod 11$.Femat's theorem says that $x^{(p-1)} = 1$ in Z_p.
So consider the ring Z11, so $x^{(10)} = 1$ in $\mathbb{Z}_{11}$
We notice that $2^{(10001)} = 2\cdot2^{(10000)} = 2\cdot((((2^{10})^{10})^{10})^{10}) = 2\cdot1 = 2$ in $\mathbb{Z}_{11}$

Ερώτηση 9
While we are at it, how much is $2^{245} \bmod 35$?
Hint: use Euler's theorem (you should not need a calculator)

Απάντηση
We aim to compute $2^{245} \bmod 35$.Since $35 = 7\cdot5$, phi (35) = 24.

2^{245}= =$2^5 \cdot 2^{240} = 2^5 \cdot (2^{24})^{10}$.From Euler's theorem, we see that $2^{24} = 2^{phi(35)} = 1$ in$\mathbb{Z}_{35}$
So we have $2^{245} = 2^5 \cdot 1 = 32$ in $\mathbb{Z}_{35}$

Ερώτηση 10
What is the order of 2 in $\mathbb{Z}_{35}^*$?

Απάντηση
We aim to compute $ord_{35}(2)$.By Lagranges theorem, $ord_5(2) \mid 4$ and $ord_7(2) \mid 6$..So $ord_{35}(2) \mid 24$.Clearly $ord_2(35) > 5$, so we try 6, 8, and 12.
$2^6 = 64 = 29$ in $\mathbb{Z}_{35}^*$, $2^8 = 256 = 11$ in $\mathbb{Z}_{35}^*$, $2^{12} = 4096 = 1$ in $\mathbb{Z}_{35}^*$
So $ord_2(35) = 12$

Ερώτηση 11
Which of the following numbers is a generator of $\mathbb{Z}_{13}^*$?
a) 2,⟨2⟩,{1,2,4,8,3,6,12,11,9,5,10,7}
b) 10, ⟨10⟩,{1,10,9,12,3,4}
c)4,⟨4⟩,{1,4,3,12,9,10}
d)6,⟨6⟩,{1,6,10,8,9,2,12,7,3,5,4,11}
e) 8,⟨8⟩,{1,8,12, 5}

Απάντηση
The generated groups shown are not trickery. Since 13 is prime, all positive integers less than 13 are in $\mathbb{Z}_{13}^*$. So the generators are 2, 6, and 7.So the correct answers are **a** and **d**.

Ερώτηση 12
Solve the equation $x^2+4x+1=0$ in $\mathbb{Z}_{23}$. Use the method described in lecture 9.3 using the quadratic formula.

Απάντηση
We aim to find the solution to $x^2 + 4x + 1 = 0$ in $\mathbb{Z}_{23}$.
$x=(-b\pm\sqrt{b^2-4ac})/2a =(-b\pm\sqrt{b^2-4ac}) \cdot (2a)^{-1}=(-4\pm\sqrt{16-4}) \cdot 2=(-4\pm\sqrt{12}) \cdot (2a)^{-1}$
The inverse of 2 in $\mathbb{Z}_{23}$can be found easily since 2^{-1} in Z_N for any odd N is just (N+1)/2. Here N=23, so $2^{-1} = 12$. Thus we have $(-4\pm\sqrt{12}) \cdot (2a)^{-1}$
Now since 23 = 3 mod 4, $\sqrt{12}$ can be computed using the simple relation
$\sqrt{c}= c^{p+1/4}$ in Zp where p is 23.
So $\sqrt{12}= 12^{24/4} \bmod 23 = 12^6 \bmod 23 = 9$.So now we have:
$x = (-4 \pm 9) \cdot 12 = \{5 \cdot (12, -13 \cdot (12\} = \{60, -156\} = \{14, 5\}$ in 23
We can verify easily:
$14^2 + 4 \cdot 14 + 1 = 253 = 23 \cdot 11$
$5^2 + 4 \cdot 5 + 1 = 46 = 23 \cdot 2$

Ερώτηση 13
What is the 11th root of 2 in $\mathbb{Z}_{19}$? (i.e. what is $2^{1/11}$ in $\mathbb{Z}_{19}$)
Hint: observe that $11-1=5$ in $\mathbb{Z}_{18}$.

Απάντηση
We aim to find $2^{1/11}$ in $\mathbb{Z}_{19}$
We have a theorem that says when gcd(e, p-1) = 1, then $c^{1/e}$ is $c^{e^{-1}}$

Fortunately, 11 and 18 are coprime. That is, gcd(11, 18) = 1.
So $2^{1/11} = 2^d$, where d = the inverse of 11 in $\mathbb{Z}_{18}$.The inverse of 11 in $\mathbb{Z}_{18}$can be found efficiently using euclid's algorithm.We compute $11^{-1} = 5$, so the 11th root of 2 in $\mathbb{Z}_{19}$is $2^5 = 13$ in $\mathbb{Z}_{19}$

Ερώτηση 14
What is the discete log of 5 base 2 in $\mathbb{Z}_{13}$? (i.e. what is Dlog2(5))
Recall that the powers of 2 in $\mathbb{Z}_{13}$ are ⟨2⟩={1,2,4,8,3,6,12,11,9,5,10,7}

Απάντηση
We aim to find $Dlog_2(5)$ in $\mathbb{Z}_{13}$.That is, we aim to find x so that $5 = 2^x$ in $\mathbb{Z}_{13}$.
From inspection of the generated group <2>, we see that $2^9 = 5$ in $\mathbb{Z}_{13}$, so $Dlog_2(5) = 9$ in $\mathbb{Z}_{13}$.

Ερώτηση 15
If p is a prime, how many generators are there in $\mathbb{Z}_p^*$?

Απάντηση
Let p be a prime number.An element g in $\mathbb{Z}_p^*$is a generator if $|\langle g\rangle| = |\mathbb{Z}_p^*|$.The set of generators of $\mathbb{Z}_p^*$is precisely the set $\{g : |\langle g\rangle| = |\mathbb{Z}_p^*| = p-1\}$

Προγραμματιστικήάσκηση 5
Your goal this week is to write a program to compute discrete log modulo a prime p. Let g be some element in $\mathbb{Z}_p^*$and suppose you are given h in $\mathbb{Z}_p^*$such that $h=g^x$ where $1 \le x \le 2^{40}$. Your goal is to find x. More precisely, the input to your program is p,g,h and the output is x.The trivial algorithm for this problem is to try all 2^{40} possible values of x until the correct one is found, that is until we find an x satisfying $h=g^x$ in $\mathbb{Z}_p$. This requires 2^{40} multiplications. In this project you will implement an algorithm that runs in time roughly $\sqrt{2^{40}} = 2^{20}$ using a meet in the middle attack.

Let $B=2^{20}$. Since x is less than B^2 we can write the unknown x base B as $x=x_0B+x1$ where x_0, x_1 are in the range [0, B−1]. Then:

$$\mathbf{h=g^x=g^{x_0B+x_1}=(g^b)^{x_0} \cdot g^{x_1} \sigma\tau o \mathbb{Z}_p^*}$$

By moving the term gx1 to the other side we obtain:

$$h/g^{x_1}=(g^b)^{x_0} \text{ στο } \mathbb{Z}_p^*$$

The variables in this equation are x_0, x_1 and everything else is known: you are given g,h and $B=2^{20}$. Since the variables x_0 and x_1 are now on different sides of the equation we can find a solution using meet in the middle (Lecture 3.3):

- First build a hash table of all possible values of the left hand side $h/g^{x_1}$for x1=0,1,…,220.
- Then for each value x0=0, 1, 2,…220 chech if the right side $(g^b)^{x_0}$ is in this hashtable .If so then you found a solution (x0,x1) from witch you can compute the required x as x=x0B+x1.

Now that we have an algorithm, here is the problem to solve:
p=

13407807929942597099574024998205846127479365820592393 3
\77723561443721764030073546976801874298166903427690031
\85818648605085375388281194656994643364900608417

g=
11717829880366207009516117596335367088558084999998952205
\59997945906392949973658374667057217647146031292859482967
\54282794665665271152127484675898946019655 68

h=
23947510405045044356526437872806578864909752095244
\95278347924529719819761432925580738569379585531805 3
\28789280014947060973941085775857324523076734440203 33

Each of these three numbers is about 153 digits. Find x such that $h=g^x$ in $\mathbb{Z}p$.To solve this assignment it is best to use an environment that supports multi-precision and modular arithmetic. In Python you could use the gmpy2 or numbthy modules. Both can be used for modular inversion and exponentiation. In C you can use GMP. In Java use a BigInteger class which can perform mod, modPow and modInverse operations.

Απάντηση

Ο κώδικας της προγραμματιστικής άσκησης βρίσκεται εδώ.Ο σκοπός μας είναι να υπολογίσουμε τον διακριτό λογάριθμο ενός περιττού αριθμού p.Υποθέτουμε ότι geίναι ένα στοιχείο στο $\mathbb{Z}_p$* και μας δίνεται ενα hπου βρίσκεται στο ίδιο σύνολο τέτοιο ώστε $h=g^x$.Ο σκοπός μας είναι να βρούμε το x.Θα χρειαστούμε πάνω απο 2^{40} πιθανές τιμές του xμέχρι να βρούμε την σωστή τιμή ετσι ώστε $h=g^x$.Θα θεωρήσουμε ένα $B=2^{20}$ και θα γράψουμε το $x=x_0B+x_1$όπου x_0,x_1 ανήκουν στο [0,B-1].Τότε θα δημιουργήσουμε ένα hashtable με όλες τις πιθανές τιμές h/g^{x_1}.Μετά για κάθε τιμή $x_0=0,1,..2^{20}$θα δούμε έαν το δεξί μέλος g^{Bx_0}είναι σε αυτό το hashtable.Εάν ναι τότε βρήκαμε μια λύση (x_0,x_1) όπου μπορούμε να υπολογίσουμε μετά την παράσταση $x=x_0B+x_1$.

Αρχικά δημιουργούμε μια συνάρτηση που την ονομάζουμε compute_x0s(p, h,gB) η οποία για κάθε τιμή του i στο Bκαι της παράστασης g^{Bx_0} μας επιστρέφει τα x_0. Επίσης με τη συνάρτηση discrete_log(p, h, g, maxExp=40) υπολογίζουμε το τέτοιο ώστε $h=g^x$modpαφού θέσουμε ότι το $B=2^{20}$.Στη συνέχεια υπολογίζουμε τα x_1και προσπαθούμε να ελέγξουμε για ισότητα στα δύο μέλη της εξίωσης .Αν η ισότητα ισχυει επιστρέφεται η τιμή $x=x_0B+x_1$.Η συνάρτηση def_test() χρησιμοποιήθηκε για να αρχικοποιήσουμε τις τιμές p,g, h.Τελος πράττουμε 2 τέστ.Οταν το xτην ελάχιστη δυνατή τιμή, και όταν το xέχει την μεγιστη δυνατή τιμή.Για την επιλυση χρησιμοποιούμε την μέγιστη δυνατή τιμή του x.Η έξοδος του προγράμματος φαίνεται παρακάτω:

```
Running tiny test
Computing x1s...
Checking for equality...
Found values!
x0 = 0
x1 = 3
x == 3
Tiny test passed!
Running short test
Computing x1s...
Checking for equality...
Found values!
x0 = 90
x1 = 192
```

```
x == 23232
Short test passed!
Running long test
Computing x1s...
Checking for equality...
Found values!
x0 = 357984
x1 = 787046
x == 375374217830
Long test passed!
```

7.7Αναλυτική βαθμολογία

You submitted this homework on **Fri 5 Jun 2015 2:33 AM EEST**. You got a score of **14.00** out of **15.00**.

Question 1

Consider the toy key exchange protocol using an online trusted 3rd party (TTP) discussed in Lecture 9.1. Suppose Alice, Bob, and Carol are three users of this system (among many others) and each have a secret key with the TTP denoted k_a, k_b, k_c respectively. They wish to generate a group session key k_{ABC} that will be known to Alice, Bob, and Carol but unknown to an eavesdropper. How would you modify the protocol in the lecture to accomodate a group key exchange of this type? (note that all these protocols are insecure against active attacks)

Your Answer		Score	Explanation
Alice contacts the TTP. TTP generates a random k_{ABC} and sends to Alice $E(k_a, k_{ABC})$, $\text{ticket}_1 \leftarrow E(k_b, k_{ABC})$, $\text{ticket}_2 \leftarrow E(k_c, k_{ABC})$. Alice sends k_{ABC} to Bob and k_{ABC} to Carol.			
Alice contacts the TTP. TTP generates a random k_{ABC} and sends to Alice $E(k_a, k_{ABC})$, $\text{ticket}_1 \leftarrow E(k_c, E(k_b, k_{ABC}))$, $\text{ticket}_2 \leftarrow E(k_b, E(k_c, k_{ABC}$. Alice sends k_{ABC} to Bob and k_{ABC} to Carol.			
Bob contacts the TTP. TTP generates random k_{ABC} and sends to Bob $E(k_b, k_{ABC})$, $\text{ticket}_1 \leftarrow E(k_a, k_{ABC})$, $\text{ticket}_2 \leftarrow E(k_c, k_{ABC})$. Bob sends ticket_1 to Alice and ticket_2 to Carol.	✔	1.00	The protocol works because it lets Alice, Bob, and Carol obtain k_{ABC} but an eaesdropper only sees encryptions of k_{ABC} under keys he does not have.
Bob contacts the TTP. TTP generates a random k_{AB} and a random k_{BC}. It sends to Bob $E(k_a, k_{AB})$, $\text{ticket}_1 \leftarrow E(k_a, k_{AB})$, $\text{ticket}_2 \leftarrow E(k_c, k_{BC})$. Bob sends ticket_1 to Alice and ticket_2 to Carol.			
Total		1.00 /	

Εικόνα121:Ερώτηση1-Week5

Question 2

Let G be a finite cyclic group (e.g. $G = \mathbb{Z}_p^*$) with generator g. Suppose the Diffie-Hellman function $\mathrm{DH}_g(g^x, g^y) = g^{xy}$ is difficult to compute in G. Which of the following functions is also difficult to compute:

As usual, identify the f below for which the contra-positive holds: if $f(\cdot,\cdot)$ is easy to compute then so is $\mathrm{DH}_g(\cdot,\cdot)$. If you can show that then it will follow that if DH_g is hard to compute in G then so must be f.

Your Answer		Score	Explanation
$f(g^x, g^y) = (\sqrt{g})^{x+y}$	✔	0.25	It is easy to compute f as $f(g^x, g^y) = \sqrt{g^x \cdot g^y}$.
$f(g^x, g^y) = g^{xy+x+y+1}$	✔	0.25	an algorithm for calculating $f(g^x, g^y)$ can easily be converted into an algorithm for calculating $\mathrm{DH}(\cdot,\cdot)$. Therefore, if f were easy to compute then so would DH, contrading the assumption.
$f(g^x, g^y) = g^{x(y+1)}$	✔	0.25	an algorithm for calculating $f(g^x, g^y)$ can easily be converted into an algorithm for calculating $\mathrm{DH}(\cdot,\cdot)$. Therefore, if f were easy to compute then so would DH, contrading the assumption.
$f(g^x, g^y) = g^{x-y}$	✔	0.25	It is easy to compute f as $f(g^x, g^y) = g^x / g^y$.
Total		1.00 / 1.00	

Εικόνα122: Ερώτηση 2-Week 5

Question 3

Suppose we modify the Diffie-Hellman protocol so that Alice operates as usual, namely chooses a random a in $\{1, \ldots, p-1\}$ and sends to Bob $A \leftarrow g^a$. Bob, however, chooses a random b in $\{1, \ldots, p-1\}$ and sends to Alice $B \leftarrow g^{1/b}$. What shared secret can they generate and how would they do it?

Your Answer		Score	Explanation
secret = $g^{a/b}$. Alice computes the secret as B^a and Bob computes $A^{1/b}$.	✔	1.00	This is correct since it is not difficult to see that both will obtain $g^{a/b}$
secret = g^{ab}. Alice computes the secret as $B^{1/a}$ and Bob computes A^b.			
secret = $g^{b/a}$. Alice computes the secret as B^a and Bob computes $A^{1/b}$.			
secret = $g^{a/b}$. Alice computes the secret as $B^{1/a}$ and Bob computes A^b.			
Total		1.00 / 1.00	

Εικόνα123:Ερώτηση 3-Week 5

Question 4

Consider the toy key exchange protocol using public key encryption described in Lecture 9.4. Suppose that when sending his reply $c \leftarrow E(pk, x)$ to Alice, Bob appends a MAC $t := S(x, c)$ to the ciphertext so that what is sent to Alice is the pair (c, t). Alice verifies the tag t and rejects the message from Bob if the tag does not verify. Will this additional step prevent the man in the middle attack described in the lecture?

Your Answer		Score	Explanation
○ it depends on what MAC system is used.			
○ it depends on what public key encryption system is used.			
◉ no	✔	1.00	an active attacker can still decrypt $E(pk', x)$ to recover x and then replace (c, t) by (c', t') where $c' \leftarrow E(pk, x)$ and $t \leftarrow S(x, c')$.
○ yes			
Total		1.00 / 1.00	

Εικόνα124:Ερώτηση 4-Week 5

Question 5

The numbers 7 and 23 are relatively prime and therefore there must exist integers a and b such that $7a + 23b = 1$. Find such a pair of integers (a, b) with the smallest possible $a > 0$. Given this pair, can you determine the inverse of 7 in $\mathbb{Z}_{23}$?

Enter below comma separated values for a, b, and for 7^{-1} in $\mathbb{Z}_{23}$.

You entered:

10,-3

Your Answer		Score	Explanation
10,-3	✖	0.00	
Total		0.00 / 1.00	

Question Explanation

$7 \times 10 + 23 \times (-3) = 1$. Therefore $7 \times 10 = 1$ in $\mathbb{Z}_{23}$ implying that $7^{-1} = 10$ in $\mathbb{Z}_{23}$.

Εικόνα 125:Ερώτηση 5-Week5

Question 6

Solve the equation $3x + 2 = 7$ in $\mathbb{Z}_{19}$.

You entered:

8

Your Answer		Score	Explanation
8	✔	1.00	
Total		1.00 / 1.00	

Question Explanation

$x = (7 - 2) \times 3^{-1} \in \mathbb{Z}_{19}$

Εικόνα 126:Ερωτήση 6-Week 5

Question 7

How many elements are there in $\mathbb{Z}_{35}^*$?

You entered:

24

Your Answer		Score	Explanation
24	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 127:Ερώτηση 7-Week 5

Question 8

How much is $2^{10001} \bmod 11$? (please do not use a calculator for this)

Hint: use Fermat's theorem.

You entered:

2

Your Answer		Score	Explanation
2	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 128:Ερώτηση 8-Week 5

Question 9

While we are at it, how much is $2^{245} \bmod 35$?

Hint: use Euler's theorem (you should not need a calculator)

You entered:

32

Your Answer		Score	Explanation
32	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 129:Ερώτηση 9-Week 5

Question 10

What is the order of 2 in $\mathbb{Z}_{35}^*$?

You entered:

12

Your Answer		Score	Explanation
12	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 130:Ερώτηση 10-Week 5

Question 11

Which of the following numbers is a generator of $\mathbb{Z}_{13}^*$?

Your Answer			Score	Explanation
☑ 2,	$\langle 2 \rangle = \{1,2,4,8,3,6,12,11,9,5,10,7\}$	✔	0.20	correct, 2 generates the entire group $\mathbb{Z}_{13}^*$
☐ 10,	$\langle 10 \rangle = \{1,10,9,12,3,4\}$	✔	0.20	No, 10 only generates six elements in $\mathbb{Z}_{13}^*$.
☐ 4,	$\langle 4 \rangle = \{1,4,3,12,9,10\}$	✔	0.20	No, 4 only generates six elements in $\mathbb{Z}_{13}^*$.
☑ 6,	$\langle 6 \rangle = \{1,6,10,8,9,2,12,7,3,5,4,11\}$	✔	0.20	correct, 6 generates the entire group $\mathbb{Z}_{13}^*$
☐ 8,	$\langle 8 \rangle = \{1,8,12,5\}$	✔	0.20	No, 8 only generates four elements in $\mathbb{Z}_{13}^*$.
Total			1.00 / 1.00	

Εικόνα 131:Ερώτηση 11-Week 5

Question 12

Solve the equation $x^2+4x+1=0$ in $\mathbb{Z}_{23}$. Use the method described in lecture 9.3 using the quadratic formula.

You entered:

{14,5}

Your Answer		Score	Explanation
{14,5}	✓	1.00	
Total		1.00 / 1.00	

Εικόνα 132:Ερώτηση 12-Week 5

Question 13

What is the 11th root of 2 in $\mathbb{Z}_{19}$? (i.e. what is $2^{1/11}$ in $\mathbb{Z}_{19}$)

Hint: observe that $11^{-1}=5$ in $\mathbb{Z}_{18}$.

You entered:

13

Your Answer		Score	Explanation
13	✓	1.00	
Total		1.00 / 1.00	

Εικόνα 133:Ερώτηση 13-Week5

Question 14

What is the discete log of 5 base 2 in $\mathbb{Z}_{13}$? (i.e. what is $\text{Dlog}_2(5)$)

Recall that the powers of 2 in $\mathbb{Z}_{13}$ are $\langle 2 \rangle = \{1,2,4,8,3,6,12,11,9,5,10,7\}$

You entered:

9

Your Answer		Score	Explanation
9	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 134:Ερώτηση 14-Week 5

Question 15

If p is a prime, how many generators are there in $\mathbb{Z}_p^*$?

Your Answer		Score	Explanation
$(p-1)/2$			
$(p+1)/2$			
$\sqrt{p}$			
$\varphi(p-1)$	✔	1.00	The answer is $\varphi(p-1)$. Here is why. Let g be some generator of $\mathbb{Z}_p^*$ and let $h = g^x$ for some x. It is not difficult to see that h is a generator exactly when we can write g as $g = h^y$ for some integer y (h is a generator because if $g = h^y$ then any power of g can also be written as a power of h). Since $y = x^{-1} \bmod p-1$ this y exists exactly when x is relatively prime to $p-1$. The number of such x is the size of $\mathbb{Z}_{p-1}$ which is precisely $\varphi(p-1)$.
Total		1.00 / 1.00	

Εικόνα 135:Ερώτηση 15-Week 5

You submitted this homework on **Thu 4 Jun 2015 10:51 PM EEST**. You got a score of **1.00** out of **1.00**.

Question 1

Your goal this week is to write a program to compute discrete log modulo a prime p. Let g be some element in $\mathbb{Z}_p^*$ and suppose you are given h in $\mathbb{Z}_p^*$ such that $h = g^x$ where $1 \le x \le 2^{40}$. Your goal is to find x. More precisely, the input to your program is p, g, h and the output is x.

The trivial algorithm for this problem is to try all 2^{40} possible values of x until the correct one is found, that is until we find an x satisfying $h = g^x$ in $\mathbb{Z}_p$. This requires 2^{40} multiplications. In this project you will implement an algorithm that runs in time roughly $\sqrt{2^{40}} = 2^{20}$ using a meet in the middle attack.

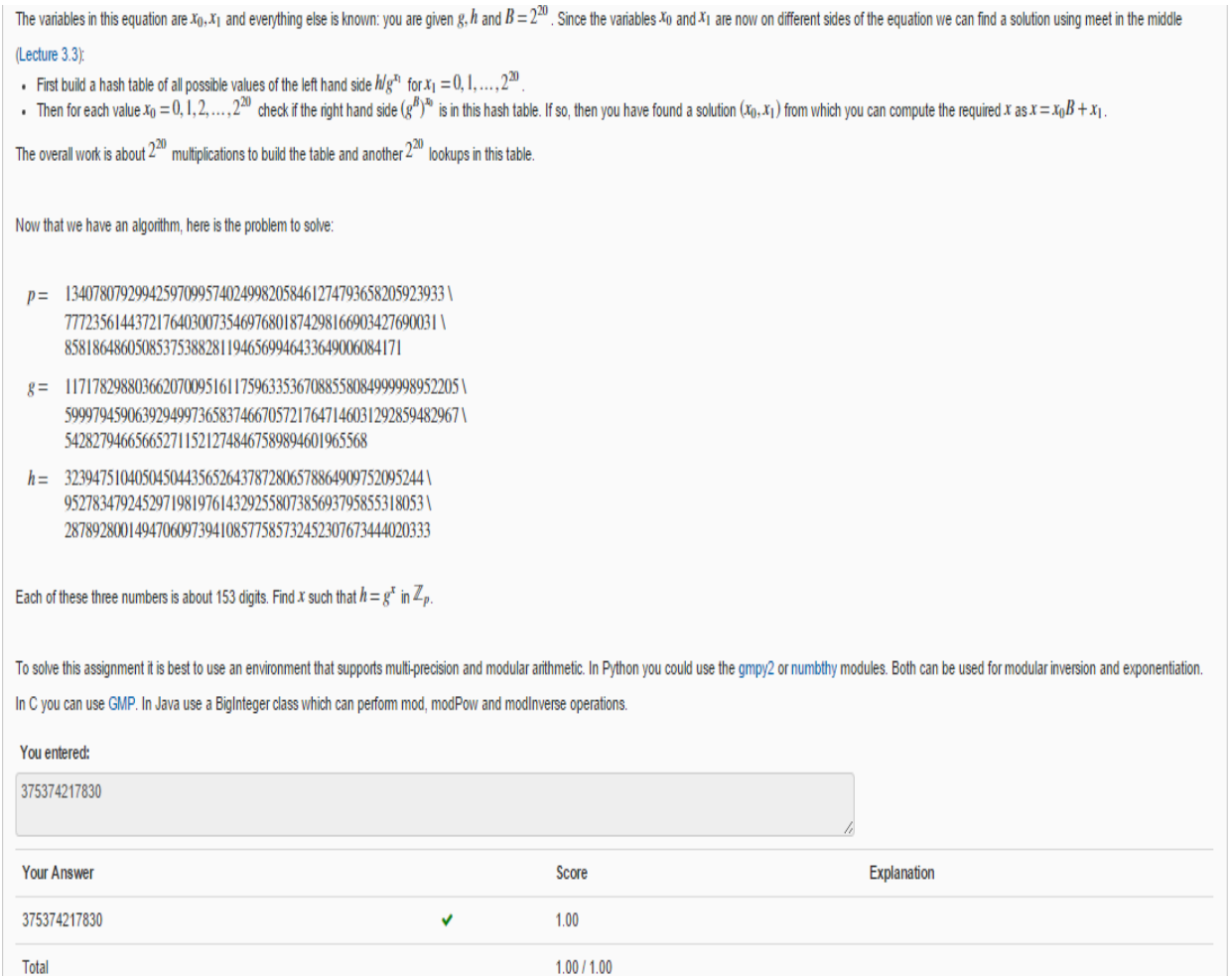
Let $B = 2^{20}$. Since x is less than B^2 we can write the unknown x base B as $x = x_0 B + x_1$ where x_0, x_1 are in the range $[0, B-1]$. Then

$$h = g^x = g^{x_0 B + x_1} = (g^B)^{x_0} \cdot g^{x_1} \text{ in } \mathbb{Z}_p.$$

By moving the term g^{x_1} to the other side we obtain

$$\boxed{h/g^{x_1} = (g^B)^{x_0}} \text{ in } \mathbb{Z}_p.$$

Εικόνα 136:Προγραμματιστική άσκηση-Week 5-a



The variables in this equation are x_0, x_1 and everything else is known: you are given g, h and $B = 2^{20}$. Since the variables x_0 and x_1 are now on different sides of the equation we can find a solution using meet in the middle (Lecture 3.3):

- First build a hash table of all possible values of the left hand side h/g^{x_1} for $x_1 = 0, 1, \ldots, 2^{20}$.
- Then for each value $x_0 = 0, 1, 2, \ldots, 2^{20}$ check if the right hand side $(g^B)^{x_0}$ is in this hash table. If so, then you have found a solution (x_0, x_1) from which you can compute the required x as $x = x_0 B + x_1$.

The overall work is about 2^{20} multiplications to build the table and another 2^{20} lookups in this table.

Now that we have an algorithm, here is the problem to solve:

$p =$ 13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031 \
85818648605085375388281194656994643364900608417l

$g =$ 11717829880366207009516117596335367088558084999998952205 \
59997945906392949973658374667057217647146031292859482967 \
54282794665665271152127484675898946019655 68

$h =$ 32394751040504504435652643787280657886490975209524 4 \
95278347924529719819761432925580738569379585531805 3 \
28789280014947060973941085775857324523076734440203 33

Each of these three numbers is about 153 digits. Find x such that $h = g^x$ in $\mathbb{Z}_p$.

To solve this assignment it is best to use an environment that supports multi-precision and modular arithmetic. In Python you could use the gmpy2 or numbthy modules. Both can be used for modular inversion and exponentiation. In C you can use GMP. In Java use a BigInteger class which can perform mod, modPow and modInverse operations.

You entered:

375374217830

Your Answer		Score	Explanation
375374217830	✔	1.00	
Total		1.00 / 1.00	

Εικόνα 137:Προγραμματιστική άσκηση Week 5-b

Πίνακας 5:Βαθμολογίες-Week 5

	1η προσπάθεια	2η προσπάθεια	3η προσπάθεια	4η προσπάθεια
Ερώτησεις	13.75/15.00	14.00/15.00	14.00/15.00	14.00/15.00
Άσκηση	1.00/1.00	-	-	-

ΚΕΦΑΛΑΙΟ 8
PUBLIC KEY ENCRYPTION FROM TRAPDOOR PERMUTATIONS–D.HELLMAN

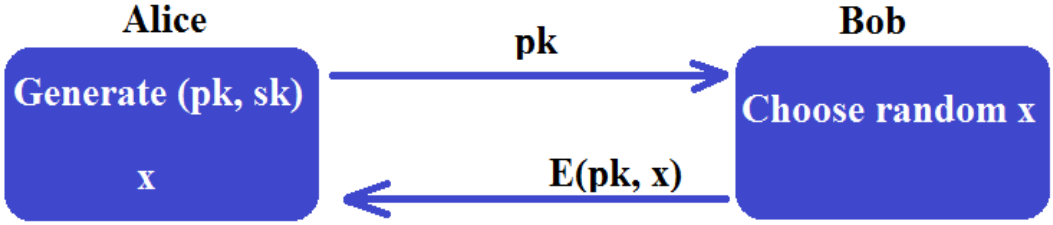
8.1 Ορισμοί και ασφάλεια

Σε αυτη την ενότητα θα χρησιμοποιήσουμε την θεωρία αριθμών για να κατασκευάσουμε **σχήματα κρυπτογράφησης δημόσιου κλειδιού** (publickeyencryptionschemes).Αρχικά θα δούμε τι ειναι η κρυπτογράφηση αυτή ενώ έπειτα θα δούμε το πόσο ασφαλής είναι.Πρωτα θα θυμήσουμε τι είναι ένα σχήμα κρυπτογράφησης δημόσιου κλειδιού.Υπάρχει ένας αλγόριθμος κρυπτογράφησης(E) και ένας αλγόριθμος αποκρυπτογράφησης(D).

Ο αλγόριθμος κρυπτογράφησης παίρνει ένα δημόσιο κλειδί (pk) ενώ ο αλγόριθμος αποκρυπτογράφησης ένα μυστικό κλειδί(sk).Το μήνυμα κρυπτογραφείται χρησιμοποιώντας ένα δημόσιο κλειδί με αποτέλεσμα ένα ciphertext.

Δηλαδή E(pk, m)=c ενώ χρησιμοποιώντας το μυστικο κλειδί πάνω σε αυτό το ciphertextλαμβάνουμε το μήνυμα μας.Η κρυπτογράφηση δημοσίου κλειδιού έχει μια πληθώρα εφαρμογών.Στο προηγούμενο κεφάλαιο είδαμε μια κλασική εφαρμογή που περιλάμβανε την ανταλλαγή κλειδιών.Τωρα θα δούμε μια ανταλλαγή κλειδιών που είναι ασφαλής μόνο σε eavesdroppingεπιθέσεις.

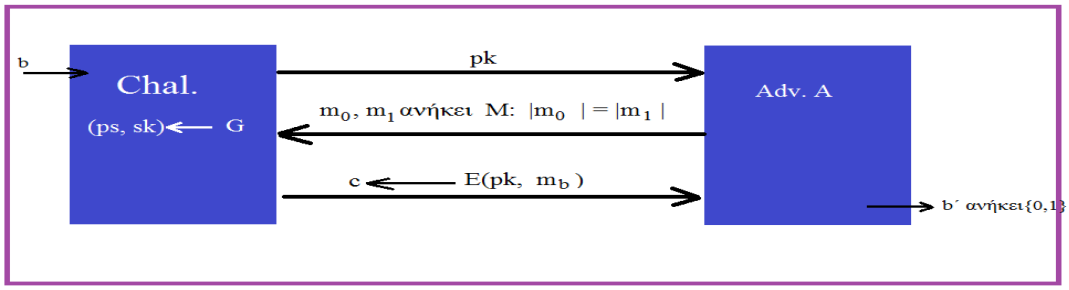
Οπως έχουμε πει το πρωτόκολλο λειτουργεί ώς εξής:Η Aliceδημιουργεί ενα ζέυγος μυστικού και δημόσιου κλειδιου (pk, sk) και στέλνει στον Bobτο δημόσιο κλειδί της.Ο Bobαπο την πλευρά του θα δημιουργήσει ένα τυχαίο χτο οποίο θα θεωρηθεί σαν μυστικό κοινο κλειδί μεταξύ της Aliceκαι του Bob.Στη συνέχεια ο Bobστέλνει αυτήν την τιμή χκρυπτογραφημένη με το δημόσιο κλειδί της Alice.Η Aliceαπο την πλευρά της αποκρυπτογραφεί το μήνυμα που έλαβε με το μυστικό της κλειδί και τώρα μπορούν να επικοινωνήσουν και δύο πλευρές(**Εικόνα 139**)



Εικόνα 138:Επικοινωνία με κρυπτογράφηση δημόσιου κλειδιού.

Ο επιτιθέμενος θα δει την κρυπτογράφηση του χπου έγινε με το δημόσιο κλειδί .Δεν θα μπορεί να βρεί καμια πληροφορία σχετικά με το μήνυμα.Ουσιαστικά ένα σύστημα κρυπτογραφίας δημόσιου κλειδιού απαρτίζεται από 3 αλγορίθμους.Έναν αλγόριθμο(G) ο οποίος αναλαμβάνει να δημιουργήσει το ζέυγος κλειδιών (pk, sk), έναν αλγόριθμο E(pk, sk) ο οποίος θα χρησιμοποιηθεί για την κρυπτογράφηση του μηνύματος m∈Mγια να μας παράγει ένα ciphertextc∈C και έναν αλγόριθμο D(sk, c) ο οποίος παίρνει το c∈Cκαι μας δίνει το m∈M ή ⊥

Τωρα θα ορίσουμε την ασφάλεια πανω σε ένα σχήμα κρυπτογράφησης δημοσίου κλειδιού.Πρώτα θα ορίσουμε την ασφάλεια ενάντια σε επιθέσεις eavesdroppingκαι μετά σε ενεργητικές επιθέσεις.Ξεκινάμε ορίζοντας 2 πειράματα(πείραμα 1 και πείραμα 0).Ο challengerθα δημιουργήσει μέσω του G ένα ζεύγος κλειδιών (pk, sk) και θα στείλει στη συνέχεια το δημόσιο κλειδί pkστον αντίπαλο(adversary) .Ο επιτιθέμενος θα βγάλει ως έξοδο 2 μηνύματα(m_0και m_1) τα οποία φυσικά έχουν το ίδιο μέγεθος.Θα λάβει την κρυπτογράφηση του m_0ή την κρυπτογράφηση του m_1.Στο πείραμα 0 θα λάβει την κρυπτογράφηση του m_0 ενώ στο πείραμα 1 θα λάβει την κρυπτογράφηση m_1(**Εικόνα 140**)



Εικόνα 139:Ασφάλεια κρυπτογράφησης δημοσίου κλειδιού(eavesdropping)

Ο επιτιθέμενος πρέπει να καταλάβει ποια από τις δύο κρυπτογραφήσεις έλαβε.Το σχήμα κρυπτογράφησης δημόσιου κλειδιού είναι σημασιολογικά ασφαλές έαν ο επιτιθέμενος δεν μπορεί να διακρίνει την κρυπτογράφηση m_0από την κρυπτογράφηση m_1δηλαδή:

$$\mathbf{Adv_{SS}[A, E]=| P[EXP(0)=1]-P[EXP(1)=1] |}$$

Πρωτού μεταβούμε στις ενεργητικές επιθέσεις, θα δούμε την σχέση μεταξύ του ορισμού της κρυπτογραφίας δημόσιου κλειδιού και του ορισμού της ασφάλειας από eavesdroppingεπιθέσεις.Οταν αναφερθήκαμε στην ασφάλεια ενάντια σε eavesdroppingεπιθέσεις στους συμμετρικούς αλγορίθμους, ξεχωρίσαμε τις περιπτώσεις οπου το κλειδί χρησιμοποιείται μία και μόνο φορά ή πολλές φορές.Δείξαμε ότι ο onetimepad(OTP) είναι ασφαλής μόνο έαν το κλειδί χρησιμοποιείται για να κρυπτογραφήσει ένα και μόνο μήνυμα ενώ είναι εντελώς ανασφαλής στην περίπτωση που το κλειδί χρησιμοποιηθεί για να κρυπτογραφήσει πολλαπλά μηνύματα.

Ορισμός της συμμετρικής κρυπτογράφησης όπως βλέπουμε είναι πολύ κοινός με τον ορισμό της onetimesecurityγια συμμετρικούς αλγορίθμους.Μας βγαίνει ως συμπέρασμα ότι δεν πρέπει να δώσουμε στον επιτιθέμενο το δικαίωμα την ικανότητα να ζητάει τις κρυπτογραφήσεις μηνυμάτων της επιλογής μιας και ένα το σχήμα είναι ασφαλές κάτω από τη χρήση ενός κλειδιού θα είναι ασφαλές κατω από τη χρήση πολλαπλών κλειδιών.Εαν του επιτρέπαμε να λαμβάνει της κρυπτογραφήσεις της επιλογής του τότε θα μπορεί να κάνει και αποκρυπτογράφηση μόνος του.

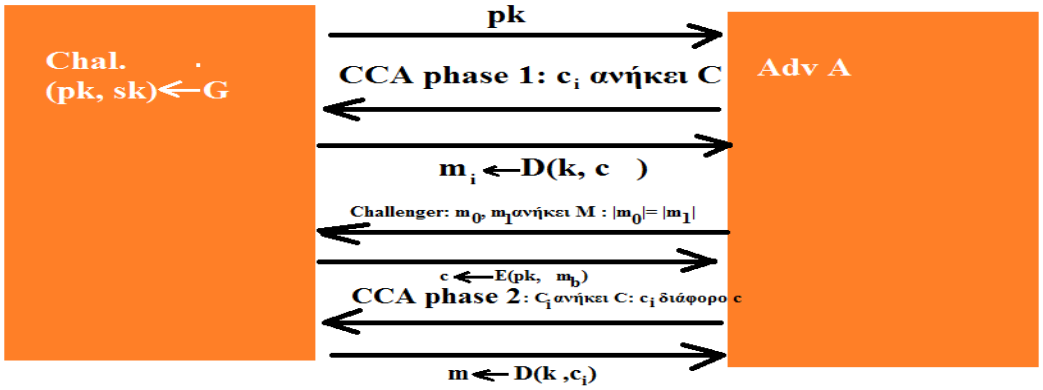
Τώρα μένει αν δούμε τι θα γίνει ένα κάποιος εφάρμοζε ενεργητική επίθεση.Ας δούμε ένα παράδειγμα που αφορά το email.Έστω ότι ο Bobθέλει να στείλει ένα email στην Caroline.Τυγχάνει η Carolineνα έχει έναν λογαριασμό(account) Gmailκαι ο τρόπος με τον οποίο λειτουργεί είναι ο εξής:το emailστέλνεται στον σέρβερ του

Gmailκρυπτογραφημένο.Στη συνέχεια ο Gmailσερβερ αποκρυπτογραφεί το μήνυμα .Αν ο παραλήπτης είναι η Carolinetότε προωθεί το emailσε αυτήν ενώ σε άλλη περίπτωσει προωθεί το emailστον επιτιθέμενο.Υποθέστε ότι ο Bobκρυπτογραφεί το μήνυμα με τέτοιο τρόπο ώστε να επιτρέπει τις ενεργητικές επιθέσεις από τον επιτιθέμενο χωρίς αυτός να μπορεί να γίνει αντιληπτός.

Πιο συγκεκριμένα υποθέστε ότι το μήνυμα κρυπτογραφήθηκε με την μέθοδο countermode ή κάτι παρόμοιο.Οταν ο επιτιθέμενος διακόψει το μήνυμα, μπορεί να αλλάξει τον αποστολέα και τώρα ο νέος παραλήπτης θα είναι ο επιτιθέμενος και όταν ο σέρβερ αποκρυπτογραφήσει το μήνυμα, θα το στείλει στον επιτιθέμενο που γράφει το μήνυμα δηλαδή στον επιτιθέμενο!Ετσι ο επιτιθέμενος μπορεί αν διαβάσει το emailπου προορίζονταν για την Caroline.

Οσκοπός μας είναι να δημιουργήσουμε σχήματα δημόσιας κρυπτογράφησης τα οποία είναι ασφαλή ακόμη και αν ο επιτιθέμενος μπορεί να χρησιμοποιήσει ενεργητικές επιθέσεις (tamperingattacks).Αυτού του είδους οι επιθέσεις μας αναγκάζουν να αλλάξουμε τον ορισμό της **ασφάλειαςενάντια σε επιθέσειςεπιλεγμένων ciphertext**(chosenciphertextsecurity)

Έστω ότι έχουμε ένα σχήμα κρυπτογράφησης (G, E, D).Ως συνήθως θα ορίσουμε δύο πειράματα.Το πείραμα 0 και το πείραμα 1. Ο challengerξεκινάει δημιουργώντας ένα ζεύγος μυστικού και δημόσιου κλειδιού(ps,sk) και δίνει το δημόσιο κλειδί στον επιτιθέμενο.Τότε ο επιτιθέμενος θα στείλει μερικά ciphertexts(**Εικόνα 141**)και θα απαιτήσει να του τα αποκρυπτογραφήσει ο challenger.



Εικόνα 140:Ασφάλεια σχήματος κρυπτογράφησης με δημόσιο κλειδί

Οταν τελειώσει αυτή η διαδικασία ο επιτιθέμενος υποβάλει δύο μηνύματα ίδιου μεγέθους(m_0και m_1) και θα λάβει ένα ciphertext το οποίο θα είναι η κρυπτογράφηση του m_0ή η κρυπτογράφηση του m_1.Για να περιορίσουμε λίγο το παιχνίδι θα επιτρέπουμε στον επιτιθέμενο να υποβάλλει κάθε ciphertextης επιλογής του εκτός από το ciphertextπου το έστειλε ο challenger.

Ορισμός

ΈνασχήμαΕείναιCCAασφαλές.Τότεθα πρέπει να εχουμε ότι:

$Addv_{CAA}$[A, E]=|P[EXP(0)=1] - P[EXP(1)=1]| να ειναι αμελητέο

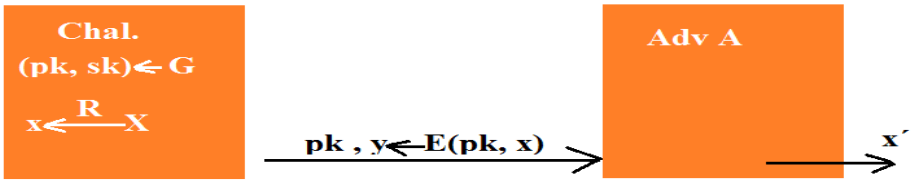
8.2Trapdoorfunctions[70]

Σε αυτήν την ενότητα θα κατασκευάσουμε συστήματα κρυπτογράφησης δημόσιου κλειδιού απο **trapdoorfunctions**(TDF).Αρχικά θα πρέπει να ορίσουμε την trapdoorfunction.

Ορισμός

Μια **trapdoorfunction**είναι ουσιαστικά μια συνάρτησηπου πηγαίνει απο το σέτ X στο σέτ Y και αποτελείται απο 3 αλγορίθμους (G, F, F^{-1}).Ο αλγόριθμος G μας δημιουργεί το ζεύγος (pk, sk), η συνάρτηση F(pk, ·) μας πηγαίνει απο το σέτ X αστο σέτ Y ,ενώ η συνάρτηση F^{-1}(sk , ·) μας πηγαίνει αποτο σετ Y στο σέτ X(αντιστρέφει τη συνάρτηση χρησιμοποιώντας το sk).

Η τριπλέτα (G, F, F^{-1}) είναι ασφαλής έαν η συνάρτηση F(pk, ·) είναι **μονόδρομη** συνάρτηση δηλαδή, δεν μπορεί να αντιστραφεί χωρίς το κλειδί sk.Θα χρησιμοποιήσουμε ως συνήθως ένα παιχνίδι για να το δείξουμε αυτο.Ο challengerθα δημιουργήσει ένα ζεύγος κλειδιών (pk, sk) μέσω της Gκαι μια τυχαία τιμή x ∈ X.Θα στείλει το κλειδί pkστον επιτιθέμενο και μετα θα υπολογίσει την τιμή της συνάρτησης Fβάζοντας την τιμή χπου επέλεξε πρίν λίγο και θα στείλει την τιμή υστον επιτιθέμενο(**Εικόνα 142**)



Adv [A, E]=P[x=x´]< απο αμελητέο

Εικόνα 141:Ασφάλεια στις trapdoorfunctions

Αυτό που επιτιθέμενος θα δεί είναι το δημόσιο κλειδί που του έστειλε ο challenger και την εικόνα της συνάρτησης που έχει υπολογίστει μέσω μιας τυχαίας τιμής x.Ο σκοπός του είναι να αντιστρέψει την συνάρτηση στο σημείο y για να μπορέσει να βρεί την τιμή x.Και θα λέμε οτι η trapdoorfunction**είναι ασφαλής** έαν η πιθανότητα του επιτιθεμενου να αντιστρέψει την συνάρτηση είναι μικρότερη απο αμελητέα.Ετσί χρησιμοποιώντας της trapdoorfunctionδεν ειναι πολύ δύσκολο να κατασκευάσουμε ένα σύστημα κρυπτογράφησης δημόσιου κλειδιού.

Έστωμιαtrapdoorfunction(G, F, F^{-1}) πουείναι ασφαλής, ένα σύστημα συμμετρικής αυθεντικής κρυπτογράφησης (E_s,E_s) που ορίζεται πάνω στο (K, M, C) και μια συνάρτηση κατακερματισμού H: X→K.Τώρα που διαθέτουμε αυτα τα 3 συστατικά μπορούμε να κατασκευάσουμε το σύστημα μας(Η γεννήτρια παραγωγής κλειδιών G που θα χρησιμοποιηθεί στο σύστημα κρυπτογράφησης δημοσίου κλειδιού είναι η ίδια με την G της trapdoorfunction)

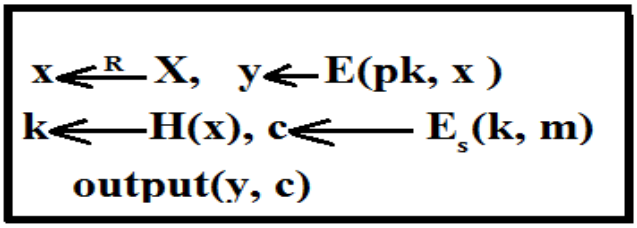
Θα τρέξουμε την συνάρτηση Gγια να πάρουμε ένα μυστικό και ένα δημόσιο κλειδί.Πώς όμως κρυπτογραφούμε και πώς αποκρυπτογραφούμε;Ο αλγόριθμος κρυπτογράφησης δέχεται ένα δημόσιο κλειδί και το μήνυμα δηλαδή E(pk, m).Αυτο

---

[70] Οι συναρτήσεις trapdoor, https://en.wikipedia.org/wiki/Trapdoor_function

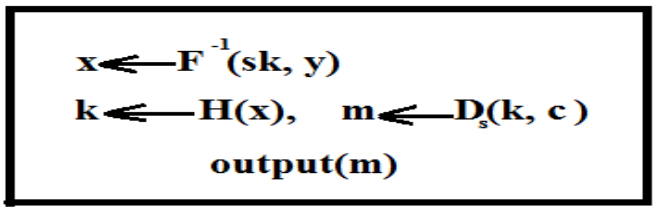
που θα κάνουμε είναι να δημιουργήσουμε μια τυχαία τιμή x∈Xκαι μετά θα εφαρμόσουμε μια trapdoorfunctionπάνω στην τιμή αυτή για να πάρουμε την τιμή y.

Μετά θα δημιουργήσουμε ένα συμμετρικό κλειδί k κάνοντας hashτην τιμή x και τέλος κρυπτογραφούμε το plaintextμήνυμα m, με το κλειδί kκαι βγάζοντας ώς έξοδο την τιμή γκαι το ciphertextc(**Εικόνα 143**)



Εικόνα 142:Trapdoorfunction και κρυπτογράφηση

Ενώ για να αποκρυπτογραφήσουμε τότε πολύ απλά ο αλγόριθμος αποκρυπτογράφησης παίρνει το μυστικό κλειδί ώς είσοδο και το ciphertextτο οποίο περιέχει την τιμή γκαι το c.Το πρώτο βήμα που θα κάνουμε είναι βρούμε την τιμή χαπλά αντιστρέφοντας την trapdoorfunction(**Εικόνα 144**).Στη συνέχεια κάνουμε hashτην τιμή χγια να πάρουμε το κλειδί k.Μετα αποκρυπτογραφούμε το μήνυμα mχρησιμοποιώντας το κλειδί kκαι το ciphertextc.



Εικόνα 143: Trapdoor function και αποκρυπτογράφηση

Ουστιαστικά το ciphertextπεριέχει ώς σώμα του μηνύματος(body) την κρυπτογράφηση E(H(x), m) ενώς ώς επικεφαλίδα(header) την συνάρτηση F(pk, x).Πρώτα αποκρυπτογραφούμε την επικεφαλίδα και στη συνέχεια αποκρυπτογραφούμε το body.Επίσης δεν θα πρέπει **ποτέ** να εφαρμόζετε την trapdoorfunctionαπευθείας στο plaintextδιότι το σύστημα θα είναι σημασιολογικά ασφαλές και επιπλέον θα μπορούν να υπάρξουν επιθέσεις πάνω σε αυτό.

8.2.1TheRSAtrapdoorpermutation

Στηνπροηγούμενη ενότητα είδαμε το πώς μπορούμε να κατασκευάσουμε ένα σύστημα κρυπτογράφησης δημόσιου κλειδιού απο trapdoorfunctions.Σε αυτήν την ενότητα θα κατασκευάσουμε μια κλασσική trapdoorfunction η οποία ονομάζεται **RSA**.Πρώτα όμως θα πρέπει να περιγράψουμε τον τρόπο δημιουργίας των κλειδίων, την συναρτηση F, και την συνάρτηση F^{-1}.

Για να δημιουργήσουμε κλειδιά αυτο που κάνουμε είναι να δημιουργήσουμε δύοπρώτους αριθμούς ρκαιqπου έχουν μέγεθος περίπου 1024 bits.Θέτουμε ώς Ντο γινόμενο αυτών των δύο πρώτων, αρα **N=p·q**.Μετά θα πάρουμε δύο ακεραίους τους εκαι dτέτοιο ώστε **e·d=1 mod(φ(N))**.Βγάζουμε ώς έξοδο το δημόσιο κλειδί το οποίο είναι το **pk=(N,e)**ενώ το μυστικό κλειδί είναι το **sk=(N , d)**.Το εμερικές φορές

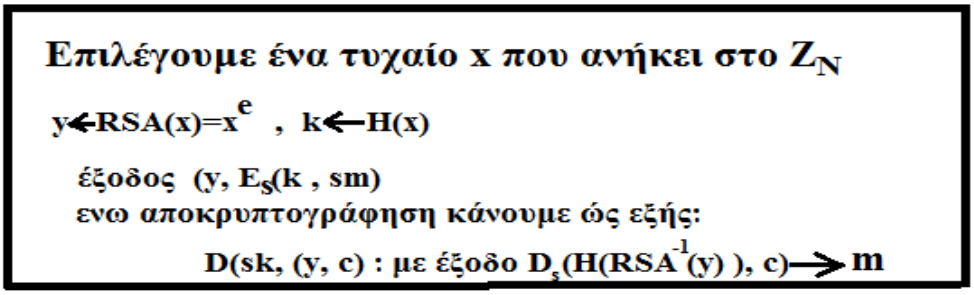
ονομάζεται και ώς **εκθέτης κρυπτογράφησης** και το dσαν **εκθέτης αποκρυπτογράφησης**.

Ο τροπος με τον οποίο ορίζεται η RSAfunctionείναι πολύ απλός.Έαν μας δίνουν μια έισοδο xτότε παίρνουμε το xκαι το υψώνουμε εις την e(στο Z_N)δηλαδή $\mathbf{RSA(x)=x^e}$ και ισχύει ότι **F(pk , x):**$\mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$.Για να αποκρυπτογραφήσουμε πρέπει να μας δώσουν την τιμή y.Μόλις μας δώσουν την τιμή αυτή τότε, μέσω της συνάρτησης F^{-1}έχουμε $\mathbf{F^{-1}(sk\ ,y)=y^d}$.Έπειτα παίρνουμε την τιμη y^dκαι υπολογίζουμε το $\mathbf{y^d=RSA(x)^d=x^{ed}=x^{k\varphi(N)+1}=(x^{\varphi(N)})^k \cdot x=x}$.Η συνάρτηση RSAείναι **onewaypermutation**εαν ισχύει για όλους τους αλγορίθμους A:

$$\mathbf{P[A(N, e, y)=y^{1/e}]<αποαμελητέο}$$

Τώρα που είδαμε πώς λειτουγεί η RSAfunctionείμαστε έτοιμοι να την να κατασκευασουμε το πρώτο ρεαλιστικο και πραγματικό σύστημα κρυπτογράφησης δημόσιου κλειδιού.Έχουμε όπως και πρίν ένα συμμετρικό σύστημα κρυπτογράφησης (E_s, D_s), μια $H:Z_N \rightarrow K$ όπου Κείναι ο κλειδόχωρος του (E_s, D_s).

Οτρόπος με τον οποίο λειτουργει το σύστημα είναι αρκετά απλός.Οαλγόριθμος Gδημιουργεί τις παραμέτρους pk= (N, e) και sk= (N, d) ενώ ο τρόπος με τον οποίο γίνεται η κρυπτογράφηση και η αποκρυπτογράφηση φαίνεται στην παρακάτω εικόνα (**Εικόνα 145**).

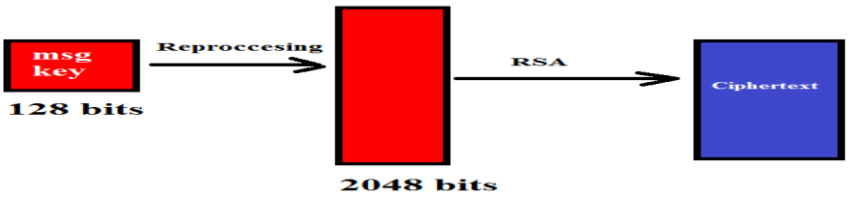


Εικόνα 144:Κρυπτογράφηση και αποκρυπτογράφηση με RSA

8.2.2 Public key cryptography standard number one (PKCS1)[71]

Στην πράξη τα πράγματα λειτουργούν εντελώς διαφορετικά.Δίνεται στην RSAένα συμμετρικό κλειδί για να το κρυπτογραφήσει και μετά θα εφαρμοσεί η RSAfunction.Εστω ότι δίνουμε στοRSAσύστημα ένα συμμετρικό κλειδί κρυπτογράφησης(πχ AESκλειδί 128bits).Τότε αρχικά παίρνουμε αυτα τα 128 bits και τα επεκτείνουμε για να φτάσουν σε μέγεθος τα 2048 bits και κατόπιν θα φαρμόσουμε την RSAfunction.Το ερώτημα είναι ποιά είναι η διαδικασία μετατροπής των 128 bitsσε 2048 bits;(**Εικόνα 146**).Πώς είμαστε σίγουροι ότι το σύστημα που προκύπτει είναι ασφαλές;

[71] Τοπρότυπο PKCS1, https://en.wikipedia.org/wiki/PKCS_1



Εικόνα 145:Απο 128 bitστα 2048 bit

Παίρνουμε το AESκλειδί των 128 bits και το χρησιμοποιούμε σαν το λιγότερο σημαντικό bitης τιμής που θέλουμε να δημιουργήσουμε.Στη συνέχεια βάζουμε 16 bitsτης τιμής 1 ή FF δίπλα στα 128 bits.Στη συνέχεια βάζουμε δίπλα στο FF ένα randompadμεγέθους 1900 bits το οποίο δέν περιέχει την τιμη FFπουθενά σε αυτο ενω τέλος βάζουμε τον αριθμό 02 σε μέγεθος 16 bits(**Εικόνα 147**).Ολο αυτό που δημιουργήσαμε είναι μια συμβολοσειρά μεγέθους 2048 bitsη οποία θα χρησιμοποιθεί στην RSAfunction.Η διαδικασία που χρησιμοποιήθηκε ονομάζεται **PKCS1 v1.5**

Εικόνα 146:Η χρήση του PKCS1 v1.5

Αυτός που θέλει να αποκρυπτογραφήσει το μήνυμα θα πρέπει να υπολογίστει την F^{-1} κατόπιν θα κοιτάξει στο πιο σημαντικό bitο οποίο είνα το 02 που αυτό θα τον ενημερώσει ότι πρόκειται για PKCS1.Στη συνέχεια θα αφαιρέσει το 02 και το randompad μέχρι το FFκαι ότι απομείνει θα είναι το αυθεντικό μήνυμα που περιέχει το συμμετρικό κλειδί των 128 bit.Δέν υπάρχει κάποιος μηχανισμός που να μας δείχνει ότι η διαδικασία PKCS1 v1.5 είναι αφαλής.Για να το δείξουμε αυτο θα παρουσιάσουμε μια **επίθεση** που στο PKCS1 που χρησιμοποιήθηκε διαμέσου του HTTPS.

Υποθέστε ότι ο επιτιθέμενος διακόπτει ένα ciphertexttύπου PKCS1 με σκοπό να το αποκρυπτογραφήσει.Ας δούμε τι μπορεί να κάνει ο επιτιθέμενος.Μπορεί να στείλει το ciphertextαπευθείας στον webserver, ο οποίος με την σειρά του θα το αποκρυπτογραφήσει χρησιμοποιώντας το μυστικό του κλειδί.Αυτό που θα κάνει ο serverμετα την αποκρυπτογράφηση είναι να «διερωτηθεί» εαυτό που έλαβε είναι τύπου PKCS1 ή κάτι άλλο, κοιτώντας το πιό σημαντικό bit(θα πρέπει να είναι το 02).Εαν όντως το σημαντικό bitείναι το ΄02΄ τότε θα συνεχίσει να το αποκρυπτογραφεί.Εαν όμως το πιό σημαντικό bitδεν είναι το ΄02΄ θα βγάλει ένα errorπου θα το στείλει στον επιτιθέμενο.

Ο επιτιθέμενος μπορεί αν υποβάλλει οποιοδήποτε ciphertext θέλει αυτός.Ο serverθα αντιστρέψει την RSAfunctionκαι θα «πει» στον επιτιθέμενο έαν η αντιστροφή ξεκινά με ΄02΄ ή οχι.Αυτό **είναι αρκετό** για να μπορέσει ο επιτιθέμενος να αποκρυπτογραφήσει οποιοδήποτε ciphertextθέλει αυτός.Ας δούμε τι μπορεί να κάνει ο επιτιθέμενος.

Έστω ότι έχει ένα ciphertextτο οποίο θέλει να αποκρυπτογραφήσει θα το στείλει στον serverγια να δεί εαν διαθέτει το ΄02΄.Στη συνέχεια θα επιλέξει μια τυχαία τιμή rπου ανήκει στο Z_N και θα προσπαθήσει να δημιουργήσει ένα νέο ciphertext(έστω c΄) για το οποίο θα ισχύει $\mathbf{c' \leftarrow r^e \cdot c = (r \cdot PKCS1(m))^e}$.Δηλαδή πολλαπλασιάζεται το plaintextμε μια

τιμή rτην οποία ελέγχει ο επιτιθέμενος.Θα δείξουμε τώρα το πώς μπορεί ένας επιτιθέμενος μπορει **να αποκρυπτογραφήσει** το ciphertext με ένα παράδειγμα.

Παράδειγμα

Έστω ότι ο επιτιθέμενος στέλνει ένα ciphertextστον webserverκαι αυτός με τη σειρά του θα χρησιμοποιήσει ένα μυστικό κλειδί για να το αποκρυπτογραφήσειαλλα υποθέστε ότι αντί να ελέγξει για το ΄02΄ θα ελέγξει για το έαν το πιο σημαντικό bitείναι 1.Για να γίνουν τα πράγματα πιο απλοϊκα υποθέστε επίσης οτι για την RSAισχύει ότι $\mathbf{N=2^n}$.Στέλοντας το ciphertextστον webservero επιτιθέμενος μαθαίνει το πιό σημαντικό bittου plaintextx.Αυτό που μπορεί να κάνει στη συνέχεια είναι :

$$\mathbf{2^e \cdot c = (2x)^e \sigma\tau o Z_N}$$

Έτσι μαθαίνουμε το πιο σημαντικό bittου 2x το οποίο είναι το δεύτερο πιο σημαντικό ψηφίο του x(ολίσθηση του χαριστερά).Αρα μάθαμε το δέυτερο bittου x.Στη συνέχεια κατα τον ίδιο τρόπο μαθαίνουμε το τρίτο πιο σημαντικό bittου x:

$$\mathbf{4^e \cdot c = (4x)^e \ στο \ Z_N}$$

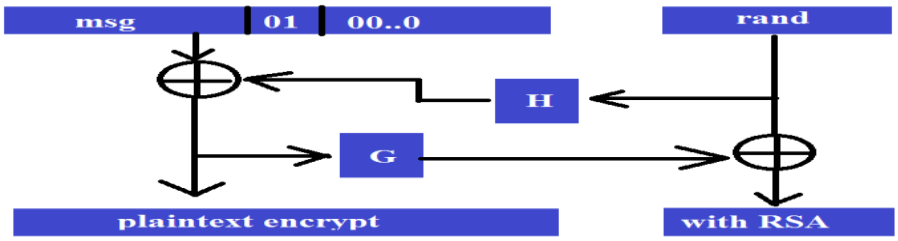
Η διαδικασια αυτή συνεχίζεται μέχρι να αποκρυπτογραφήσουμε **ολόκληρο** το ciphertext.

Πρέπει όμως να δούμε και το πως μπορούμε να αμυνθούμε απο αυτήν την επίθεση. Μετά την εφαρμογή της RSAκρυπτογράφησης και πάρουμε ένα plaintextτο οποίο δέν είναι κωδικοποιημένο με PKCS1, δηλαδή δέν ξεκινά με ΄02΄ , αυτό που θα κάνουμε είναι να επιλέξουμε μια τυχαία συμβολοσειρά r μεγέθους 46 bytesκαι θα προσποιηθούμε ότι το plaintexteίναι αυτή η τυχαία συμβολοσειρά r.Άρα δεν θα λέμε στον επιτιθέμενο έαν το plaintextμας ξεκινά με ΄02΄ αλλα θα προσποιηθούμε ότι το plaintexteίναι μια τυχαία συμβολοσειρά.

Εγείρεται το ερώτημα έαν μπορούμε να αλλάξουμε το PKCS1 πιο δραστικά με σκοπό να μας παρέχει ασφάλεια ενάντια σε chosen ciphertext επιθέσεις.Αυτό μας οδηγεί σε ένα διαφορετικό τρόπο με τον οποίο μπορούμε να κάνουμε κρυπτογράφηση χρησιμοποιώντας RSAο οποίος ονομάζεται **OptimalAssymetricEncryptionPadding**[72](OAEP).Το PKCS1 αναβαθμίστηκε σε PKCS1 2.0 και έτσι υποστηρίζει OAEP.Θα εξηγήσουμε το πώς λειτουργεί το OAEP.

Παίρνουμε το μήνυμα που θέλουμε να κρυπτογραφήσουμε πχ AESκλειδί των 128bitκαι το πρώτο που κάνουμε είναι να εφαρμόσουμε ένα μικρό padσε αυτό πχ 0100...000.Το πόσα μηδενικά θα βάλουμε βασίζεται στο εκάστοτε standardπου χρησιμοποιούμε.Εδώ θα έχουμε 128 μηδενικά.Μετα θα επιλέξουμε μια τυχαία συμβολοσειρά(**Εικόνα 148**) μεγέθους 2047 bits.Πρίν εφαρμόσουμε την RSAfunction, παίρνουμε αυτήν την τυχαία συμβολοσειρά και την βάζουμε μέσα σε μια hashfunction(έστω H).

[72] Το σχήμα επικάλυψης OAEP, https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding



Εικόνα 147:Η λειτουργία του OAEP

Κατόπιν κάνουμε πράξη XORμεταξύ της εξόδου της hashfunctionκαι του μηνύματος μας που έχει δίπλα το pad.Το αποτέλεσμα της διαδικασίας αυτό το κρατάμε.Μετα βάζουμε σε μια άλλη hashfunction(έστω η G) το αποτέλεσμα που κρατήσαμε και εφαρμόζουμε πάλι XORμε την τυχαία μας συμβολοσειρά.

Τελικά έχουμε δύο τιμές.Την τιμή που κρατήσαμε απο την πράξη XORμεταξύ της hashfunction και το μηνύματος με το pad μαζι **και** το αποτέλεσμα της hashfunctionG που έγινε XORμεταξύ της τυχαίας μας συμβολοσειράς.Αυτα τα δύο αποτελέσματα έχουν μαζί μέγεθος 2047bitsκαι αυτά θα τοποθετηθούν στην RSAfunction.

8.3 ElGamalpublickeysystem

8.3.1 Η λειτουργία του συστήματος ElGamal

Σεαυτήν την ενότητα θα δούμε το πώς μπορούμε να κατασκευάσουμε συστήματα κρυπτογραφίας δημόσιου κλειδιού απο το πρωτόκολλο Diffie-Hellaman.Θα προσπαθήσουμε να τροποποιήσουμε το πρωτόκολλο αυτό κατάλληλα για να κατασκευάσουμε το σύστημα μας.Θα θεωρήσουμε ένα **G={1,g^2, g^3,…,g^{n-1}}**ενώ το Geίναι ένας **κυκλικός δακτύλιος** ώστε **G=$(Z_p)^*$**βαθμού n.Θα θεωρήσουμε ότι θέλουν να επικοινωνήσουν η Aliceκαι ο Bob.Η Aliceθα επιλέξει μια τυχαία τιμή aπου ανήκει{1,...,n) ενώ ο Bobθα επιλέξει μια τιμή bπου ανήκει στο {1,...,n}.Στη συνέχεια η Aliceθα υπολογίσει το **A=g^a**ενώ ο Bobτο **B=g^b**_.

Οσο αφορά την Aliceθα θα θεωρήσουμε ότι το δημόσιο κλειδί της είναι το Aενώ το μυστικό της κλειδί το a.Σε κάποια χρονική στιγμή ο Bobθέλει να κρυπτογραφήσει ένα μήνυμα που θα το στείλει στην Aliceμε το δημόσιο κλειδί της.Πώς όμως θα το κάνει αυτό;Αρχικά θα υπολογίσει το **B=g^b**και στη συνέχεια θα υπολογίσει το **g^{ab}=A^b**.Απο το **g^{ab}**θα παράγει ένα συμμετρικό κλειδί kκαι τέλος θα κρυπτογραφήσει το μήνυμα mμε το κλειδί αυτό.

Απο την πλευρά της η Alice, θα υπολογίσει το **g^{ab}=B^a**μιας και διαθέτει το Bαπο τον Bobκαι το aαπο την ίδια.Στη συνέχεια απο το **g^{ab}** θα υπολογίσει το συμμετρικό κλειδί k με το οποίο θα αποκρυπτογραφήσει το μήνυμα.Τοσύστημα που μόλις περιγράψαμε ονομάζεται **ElGamal**προς τιμήν του TaherElGamal[73] που ήταν ο δημιουργός του το 1985.

<u>Πιο αναλυτικά</u>

Ας δούμε το σύστημα ElGamalμε περισσότερη λεπτομέρεια.Έστω ένας κυκλικός δακτύλιος Gβαθμού n,ένα συμμετρικό κρυπτοσύστημα (E_s, D_s) που ορίζεται πάνω στο

[73]Ο κρυπτογράφος TaherElGamal, https://en.wikipedia.org/wiki/Taher_Elgamal

(K, M , C) και μια συνάρτηση κατακερματισμού η οποία παίρνει 2 στοιχεία του δακτυλίου G και τα πηγαίνει στο κλειδόχωρο K.Για να δημιουργήσουμε το σύστημα μας θα χρειαστούμε να περιγράψουμε 3 αλγορίθμους.Τον αλγόριθμο παραγωγής δημόσιων και μυστικών κλειδιών, τον αλγόριθμο κρυπτογράφησης και τον αλγόριθμο αποκρυπτογράφησης.

Ο αλγόριθμος παραγωγής κλειδιών λειτουργεί ώς εξής:θα επιλέξουμε ένα τυχαίο g που ανήκει στο G και ένα τυχαίο aπου ανήκει στο Z_N.Το μυστικό κλειδί θα είναι το**ps=a**ενώ το δημόσιο κλειδί θα είναι το **pk=(g, h=g^a)**.Θα δούμε το πώς μπορούμε να κρυπτογραφήσουμε(**Εικόνα 149**) με αυτόν τον τρόπο.

E(pk=(g, h), m):

$b \xleftarrow{R} Z_n$, $u \leftarrow g^b$, $v \leftarrow h^b$

$k \leftarrow H(u, v)$, $c \leftarrow E_s(k, m)$

output (u, c)

Εικόνα 148:Κρυπτογράφηση ElGamal

Οταν ο Bob θέλει να κρυπτογραφήσει το μήνυμα θα επιλέξει ένα $\mathbf{b} \xleftarrow{R} \mathbf{Z_N}$, θα υπολογίσει το $\mathbf{u} \leftarrow \mathbf{g^b}$ και το $\mathbf{h} \leftarrow \mathbf{h^b}=\mathbf{g^{ab}}$.Στη συνέχεια θα υπολογίσει το συμμετρικό κλειδί $\mathbf{k} \leftarrow \mathbf{H(u, v)}$ και τέλος θα κρυπτογραφήσει το μήνυμα με το k, δηλαδή $\mathbf{c} \leftarrow \mathbf{E_s(k, m)}$ και θα βγάλει ώς έξοδο το ciphertext (u, c).Η Alice για να αποκρυπτογραφήσει το μήνυμα θα χρησιμοποιήσει το μυστικό της κλειδί a, την συνεισφορά του Bob στο Diffie-Hellman πρωτόκολλο(δηλαδή το u) και την συμμετρική κρυπτογράφηση του μηνύματος που ο Bob έστειλε(**Εικόνα 150**)

D(sk=a, (u,c))

$v \leftarrow u^a = (g^b)^a = g^{ab}$

$k \leftarrow H(u, v)$, $m \leftarrow D_s(k ,m)$

output m

Εικόνα 149:Αποκρυπτογράφηση ElGamal

Θα υπολογίσει το $\mathbf{v} \leftarrow \mathbf{u^a}=(\boldsymbol{g^b})^{\boldsymbol{\alpha}}$ και μετά θα παράγει το μυστικό κλειδί κλειδι $\mathbf{k} \leftarrow \mathbf{H(u, v)}$αφού διαθέτει και το υαλλα και αφού πριν λίγο υπολόγισε το v.Τέλος θα αποκρυπτογραφήσει το μήνυμα $\mathbf{m} \leftarrow \mathbf{D_s(k, c)}$

8.3.2Ασφάλεια στο σύστημα ElGamal

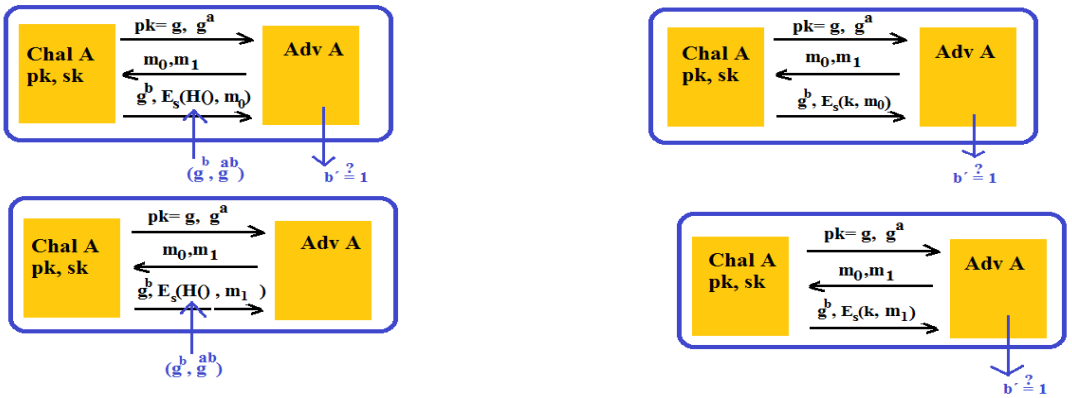
Οταν είχαμε αναφερθεί στην ασφάλεια του πρωτοκόλου Diffie-Hellmanδείξαμε ότι εαν εχουμε ένα G βαθμού ντότε έαν γνωρίζουμε τα g, g^a,g^bτότε είναι πολύ δύσκολο να υπολογίσουμε το g^{ab}.Αυτό ονομάζεται **computationalDiffieHellmanassumption**(CDH).Θα προχωρήσουμε δημιουργώντας δυσκολότερη υπόθεση που ονομάζεται **hashDiffieHellmanassumption**(HDH).Η συνάρτηση κατακερματισμού Ηπαίρνει ζευγάρια τιμών απο το Gκαι τα τροφοδοτεί στον κλειδόχωρο K.Τότε έαν μας δωθούν (g, g^a, g^b, H(g^b, g^{ab})) τότε η κατανομή αυτή είναι περίπου ίδια με την κατανομή (g, g^a, g^b, R)όπου a, b$\leftarrow Z_N$, R$\leftarrow$K.

Αυτό που μας δείχνει αυτό το συμπέρασμα είναι ότι το συμμετρικό κλειδί το οποίο παράχθηκε κατα την διάρκεια της ElGamalκρυπτογράφησης είναι περίπου ίδιο απο

ένα πραγματικά τυχαίο κλειδί το οποίο παράχθηκε ανεξάρτητα απο τις παραμέτρους του συστήματος μας.Η υπόθεση hashDiffieHellmanείναι πίο δυνατή απο την υπόθεση computationalDiffieHellman.Θα δείξουμε ότι το σύστημα ElGamalείναι σημασιολογικά ασφαλές με την χρήση του HDH.Αυτό θα γίνει με τη χρήση δύο πειραμάτων.

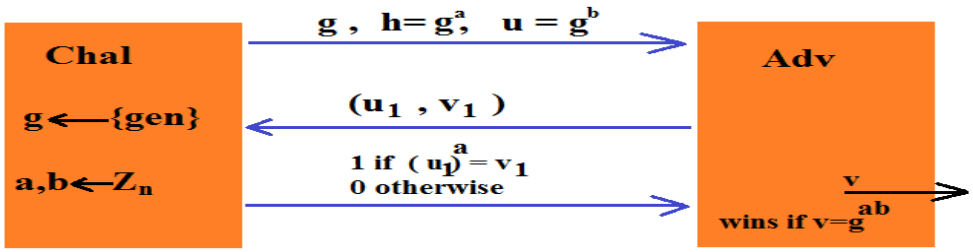
Στο πείραμα 1 δίνεται στον επιτιθέμενο η κρυπτογράφηση του m_1ενώ στο πείραμα 0 δίνεται η κρυπτογράφηση m_0.Ochallengerξεκινάει στέλντοντας το δημόσιο κλειδί **pk=(g, g^a)**στον επιτιθέμενο(**Εικόνα 151**).Ο επιτιθέμενος στη συνέχεια επιλέγει δύο μηνύματα m_0και m_1ίδιο μεγέθους και τα στέλνει.Στο πείραμα 1 λαμβάνει την κρυπτογράφηση του m_1ενώ στο πείραμα 0 λαμβάνει την κρυπτογράφηση του m_0.

Λόγω της υπόθεσηςHDHγνωρίζουμε ότι ακόμη και αν ο επιτιθέμενος δεί τις τιμές g, g^a, g^bτότε το hashτων (g, g^{ab}) δέν διαφέρει απο κάτι πραγματικά τυχαίο.Το ίδιο ακριβώς συμβαίνει και στο πείραμα 1.Ο επιτιθέμενος δεν μπορεί να διακρίνει την έξοδο της συνάρτησης κατακερματισμού απο ένα πραγματικά τυχαίο κλειδί.



Εικόνα 150:ElGamalσημασιολογικά ασφαλές με χρήση HDH

Το θέμα είναι ότι δεν αρκεί μόνο αυτό.Αυτό που πραγματικά θέλουμε είναι να έχουμε ασφάλεια ενάντια σε επιθέσεις chosenciphertextκαι το ερώτημα είναι έαν το σύστημα ElGamalείναι ασφαλές ενάντια σε τέτοιου είδους επιθέσεις.Για να αποδείξουμε την ασφάλεια ενάντια σε chosenciphertextεπιθέσεις θα πρέπει να κάνουμε ακόμη πιο ισχυρή υπόθεση που ονομάζεται **interactiveDiffie-Hellmanassumption**(IDH) στο G.Θα θεώρησουμε ένα πείραμα(**Εικόνα 152**).Ο challengerθα γεννήσει ένα τυχαίο g και θα δημιουργήσει τα a, bπου ανήκουν στο Z_N.Θα στείλει στον επιτιθέμενο τα **g, h=g^a, u=g^b**.Ο επιτιθέμενος θελει να υποογίσει το **v=g^{ab}**.



Εικόνα 151:Ασφάλεια στο IDH

Χρειαστήκαμε αυτήν την υπόθεση IDHγια να του δώσουμε τη δυνατότητα να μπορεί να υποβάλλει ερωτήματα.Θα υποβάλλει δύο στοιχεία του συνόλου Gτα u_1και v_1.Ochallengerθα του επιστρέψει 1 έαν το $\mathbf{(u_1)^a=v_1}$και 0 σε κάθε άλλη περίπτωση.

Θεώρημα ασφάλειας

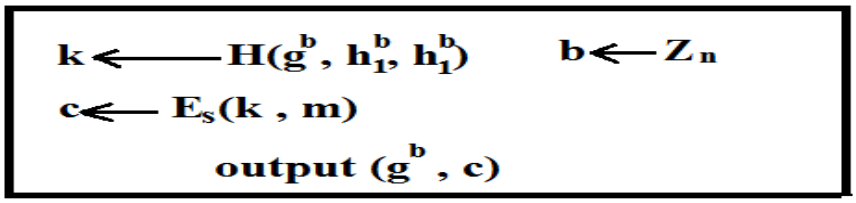
Έαν ισχυει το IDHγια ένα G και το (E_s, D_s) μας παρεχει αυθεντική κρυπτογράφηση καιη $H:G^2 \rightarrow$Κείναι έναrandomoracle, τότε το ElGamalείναι CCAασφαλές.

8.3.3 Παραλλαγές του ElGamal και περισσότερη ασφάλεια

Ένα ερώτημα που εγείρεται είναι έαν μπορούμε να αποδείξουμε την ασφάλεια ενάντια σε επιθέσεις chosenciphertextμε χρήση του CDH;Υπάρχουν δύο τρόποι να το κάνουμε αυτό.Ο **πρώτος** τρόπος είναι να χρησιμοποιήσουμε ένα Gόπου θα ισχύει ότι CDH=IDH ενώ ο**δεύτερος** τρόπος είναι να αλλάξουμε κάπως το σύστημα ElGamal.Ο πρώτος τρόπος είναι ύκολος και μπορεί να αποδειχθεί με χρήση προχωρημένης άλγεβρας πχ θα μπορούσαμε να χρησιμοποιήσουμεgroups ελλειπτικών καμπυλών.

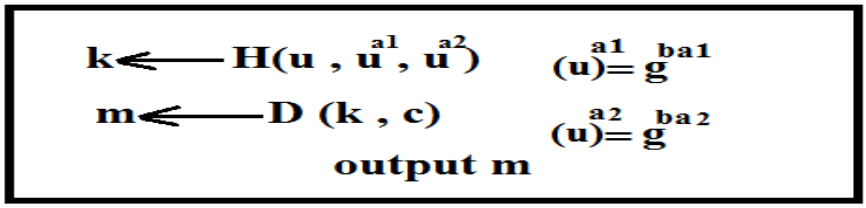
Όσο αφορά τον δεύτερο τρόπο μπορούμε να κατασκευάσουμε μια εκλεπτυσμένη κατασκευή που ονομάζεται **twinElGamal** που είναι μια παραλλαγή του αρχικού ElGamal.Θα επιλέξουμε ένα τυχαίο gκαι αυτή τη φορά δύο τιμές a_1 ,$a_2 \leftarrow Z_N$.Το μυστικό κλειδί θα είναι τώρα το $\mathbf{sk=(a_1, a_2)}$ενώ το δημόσιο κλειδί θα είναι το $\mathbf{pk=(g, h_1=(g)^{a_1}, h_2=(g)^{a_2})}$

Οτρόπος κρυπτογράφησης είναι παρόμοιος με τον ElGamal.Αυτή την φορά θα κάνουμε hashτις τιμές (g^b, h_1^b, h_2^b) για να δημιουργήσουμε το κλειδί k(**Εικόνα 153**).Στη συνέχεια θα κρυπτογραφήσουμε το μήνυμα μας mμε το κλειδί kκαι θα λάβουμε το ciphertextc.Θα βγάλουμε ώς έξοδο το ciphertextκαι το g^b.



Εικόνα 152:Κρυπτογράφηση με twinElGamal

Για να αποκρυπτογραφήσουμε πρέπει να βρούμε το κλειδί k, άρα θα κάνουμε hashτις τιμές $\mathbf{(u, (u)^{a_1}, (u)^{a_2})}$αφού $(u)^{a_1}=(g^b)^{a_1}$και $(u)^{a_2}=(g^b)^{a_2}$.Στη συνέχεια θα αποκρυπτογραφήσουμε το μήνυμα χρησιμοποιώντας το κλειδί kκαι το ciphertext(**Εικόνα 154)**



Εικόνα 153:Αποκρυπτογράφηση με twinElGamal

Θεώρημα ασφάλειας

Έστω ότι εφαρμόζεται το CDHσε ένα G και το σχήμα (E_s,D_s) που παρέχει αυθεντική κρυπτογράφηση και μια συνάρτηση κατακερματισμού $H:G^3 \rightarrow K$ είναι ένα randomoracletότε ο twinElGamalείναι ασφαλής ενάντια σε επιθέσεις CCA.

8.4 Ερωτήσειςκεφαλαίου

Ερώτηση1

Recall that with symmetric ciphers it is possible to encrypt a 32-bit message and obtain a 32-bit ciphertext (e.g. with the one time pad or with a nonce-based system). Can the same be done with a public-key system?

a) No, public-key systems with short ciphertexts can never be secure.
b)Yes, when encrypting a short plaintext the output of the public-key encryption algorithm can be truncated to the length of the plaintext
c) It is possible and depends on the specifics of the system.
d) Yes, the RSA-OAEP system can produce 32-bit ciphertexts.

Απάντηση

The correct answer is a: An attacker can use the public key to build a dictionary of all 2^{32} ciphertexts of length 32 bits along with their decryption and use the dictionary to decrypt any captured ciphertext.

Ερώτηση 2

Let (Gen,E,D) be a semantically secure public-key encryption system. Can algorithm E be deterministic?

a) Yes, some public-key encryption schemes are deterministic.
b) Yes, RSA encryption is deterministic.
c) No, but chosen-ciphertext secure encryption can be deterministic.
d) No, semantically secure public-key encryption must be randomized.

Απάντηση

The correct answer is d: That's correct since otherwise an attacker can easily break semantic security.

Ερώτηση 3

Let (Gen,E,D) be a chosen ciphertext secure public-key encryption system with message space $\{0,1\}^{128}$. Which of the following is also chosen ciphertext secure?

a)(Gen, E′,D′) where E′(pk, m)=[c←E(kp, m) output (c,c)]and D′(sk,(c_1,c_2))=

$$\begin{cases} D(k,c_1)\ if\ c_1 = c_2 \\ \perp\ otherwise \end{cases}$$

b)(Gen,E′,D′) where E′(pk, m)=(E(pk,m), 0^{128}) and D′(sk,(c_1,c_2))=D(sk,c_1)

c)(Gen,E′,D′) where E′(pk, m)=(E(pk,m),E(pk,0^{128}) and D′(sk,(c_1,c_2))=D(sk,c_1)

d) (Gen,E′,D′) where E′(pk, m)=E(pk,m$\oplus 1^{128}$) and D′(sk,c)=D(sk,c) $\oplus 1^{128}$

Απάντηση

The anser a is correct because it is a ciphertext secure.

The answer b is not corretct:
This construction is not chosen-ciphertext secure. An attacker can output two messages $m_0=0^{128}$ and $m_1=1^{128}$ and be given back a challenge ciphertext (c_1,c_2). The attacker would then ask for the decryption of $(c_1,1^{128})$ and be given in response m0 or m1 thereby letting the attacker win the game. Note that the decryption query is valid since it is different from the challenger ciphertext (c_1,c_2).
The answer c is not correct:
This construction is not chosen-ciphertext secure. An attacker can output two messages $m_0=0^{128}$ and $m_1=1^{128}$ and be given back a challenge ciphertext (c_1,c_2). The attacker would then ask for the decryption of $(c_1,E(pk,1^{128})$ and be given in response m0 or m1 thereby letting the attacker win the game. Note that the decryption query is valid since it is different from the challenger ciphertext (c_1,c_2)
The answer d is correct because it it a ciphertext secure

Ερώτηση 4
Recall that an RSA public key consists of an RSA modulus N and an exponent e. One might be tempted to use the same RSA modulus in different public keys. For example, Alice might use (N,3) as her public key while Bob may use (N,5) as his public key. Alice's secret key is $d_a=3^{-1}\mod\varphi(N)$ and Bob's secret key is $d_b=5^{-1}\mod\varphi(N)$.

In this question and the next we will show that it is insecure for Alice and Bob to use the same modulus N. In particular, we show that either user can use their secret key to factor N. Alice can use the factorization to compute φ(N) and then compute Bob's secret key. As a first step, show that Alice can use her public key (N,3) and private key da to construct an integer multiple of φ(N). Which of the following is an integer multiple of φ(N)?

a)d_a+1
b)d_a-1
c)$3d_a-1$
d)$5d_a-1$

Απάντηση
Since $da=3^{-1}\mod\varphi(N)$ we know that $3d_a=1\mod\varphi(N)$ and therefore $3d_a-1$ is divisibly by φ(N).So the **correct answer is c**.

Ερώτηση 5
Now that Alice has a multiple of φ(N) let's see how she can factor N=pq. Let x be the given muliple of φ(N). Then for any g in $\mathbb{Z}_N^*$ we have $g^x=1$ in $\mathbb{Z}_N$.
Alice chooses a random g in $\mathbb{Z}*N$ and computes the sequence

$$\mathbf{g^x, g^{x/2}, g^{x/4}, g^{x/8}\ldots \text{ in } \mathbb{Z}N}$$

and stops as soon as she reaches the first element $y=g^{x/2^i}$ such that $y\neq1$ (if she gets stuck because the exponent becomes odd, she picks a new random g and tries again).
It can be shown that with probability 1/2 this y satisfies:

$$\begin{cases} y=1\ mod\ p \\ y=-1\ mod q \end{cases} \text{ or } \begin{cases} y=-1 mod\ p \\ y=1\ mod\ q \end{cases}$$

How can Alice use this y to factor N?

a) Compute gcd(N, y^2-1)
b) Compute gcd(N−1, y)
c) Compute gcd(N, y−1)
d) Compute gcd(N, y^2)

Απάντηση
The correct answer is c: We know that y−1 is divisible by p or q, but not divisible by the other. Therefore, gcd(N, y−1) will output a non-trivial factor of N.

Ερώτηση 6
In standard RSA the modulus N is a product of two distinct primes. Suppose we choose the modulus so that it is a product of three distinct primes, namely N=pqr. Given an exponent e relatively prime to φ(N) we can derive the secret key as $d=e^{-1} \bmod \varphi(N)$. The public key (N,e) and secret key (N,d) work as before. What is φ(N) when N is a product of three distinct primes?

a) φ(N)=(p−1)(q−1)(r−1)
b) φ(N)=pqr−1
c) φ(N)=(p−1)(q−1)
d) φ(N)=(p−1)(q−1)(r+1)

Απάντηση
The correct answer is a: When is a product of distinct primes then $|\mathbb{Z}_N^*|$ satisfies $|\mathbb{Z}_N^*|=|\mathbb{Z}p|\cdot|\mathbb{Z}_q^*|\cdot|\mathbb{Z}_r^*|=(p-1)(q-1)(r-1)$.

Ερώτηση 7
n administrator comes up with the following key management scheme: he generates an RSA modulus N and an element s in $\mathbb{Z}_N^*$. He then gives user number i the secret key $s_i=s^{r_i}$ in ℤN where r_i is the i'th prime (i.e. 2 is the first prime, 3 is the second, and so on).

Now, the administrator encrypts a file that is accssible to users i,j and t with the key $k=s^m$ where$m=r_ir_jr_t$ in $\mathbb{Z}_N$. It is easy to see that each of the three users can compute k. For example, user i computes k as $k=s_i^f$ where $f=r_ir_j$. The administrator hopes that other than users I ,j and t, no other user can compute k and access the file.

Unfortunately, this system is terribly insecure. Any two colluding users can combine their secret keys to recover the master secret s and then access all files on the system. Let's see how. Suppose users 1 and 2 collude. Because r1 and r2 are distinct primes there are integers a and b such that $ar_1+br_2=1$. Now, users 1 and 2 can compute s from the secret keys s1 and s2 as follows:

a)$s=s_1^a+s_2^b$
b)$s=s_1^a\cdot s_2^b$
c) $s=s_1^a/s_2^b$
d) $s=s_1^b+s_2^a$

Απάντηση
The correct answer is b: $s=s_1^a\cdot s_2^b=s^{r_1a}\cdot s^{r_2b}=s$

Ερώτηση 8

Let G be a finite cyclic group of order n and consider the following variant of ElGamal encryption in G:

- Gen: choose a random generator g in G and a random x in $\mathbb{Z}_n$. Output pk=(g,h=g^x) and sk=(g,x).
- E(pk,m∈G): choose a random r in $\mathbb{Z}$n and output (g^r, m·h^r).
- D(sk,(c_0,c_1)): output c_1/c_0^x.

This variant, called plain ElGamal, can be shown to be semantically secure under an appropriate assumption about G. It is however not chosen-ciphertext secure because it is easy to compute on ciphertexts. That is, let (c_0,c_1) be the output of E(pk,m_0) and let (c_2,c_3) be the output of E(pk,m_1). Then just given these two ciphertexts it is easy to construct the encryption of $m_0 \cdot m_1$ as follows:

a) ($c_0 - c_2$, $c_1 - c_3$) is an encryption of of $m_0 \cdot m_1$.
b) ($c_0 c_3$, $c_1 c_2$) is an encryption of of $m_0 \cdot m_1$.
c) ($c_0 c_2$, $c_1 c_3$) is an encryption of of $m_0 \cdot m_1$.
d) (c_0/c_3, c_1/c_2) is an encryption of of $m_0 \cdot m_1$.

Απάντηση

The correct answer is c:Indeed, ($c_0 c_2$, $c_1 c_3$)=($g r_0+r_1$, $m_0 m_1 h^{r0+r1}$), which is a valid encryption of $m_0 m_1$.

Ερώτηση 9

Let G be a finite cyclic group of order n and let pk=(g,h=g_a) and sk=(g,a) be an ElGamal public/secret key pair in G as described in Segment 12.1. Suppose we want to distribute the secret key to two parties so that both parties are needed to decrypt.

Moreover, during decryption the secret key is never re-constructed in a single location. A simple way to do so it to choose random numbers a_1,a_2 in $\mathbb{Z}$n such that $a_1+a_2=a$. One party is given a_1 and the other party is given a_2. Now, to decrypt an ElGamal ciphertext (u,c) we send u to both parties. What do the two parties return and how do we use these values to decrypt?

a) Party 1 returns $u_1 \leftarrow u(a_1^2)$, party 2 returns $u_2 \leftarrow u(a_2^2)$ and the results are combined by computing $v \leftarrow u_1 \cdot u_2$.
b) Party 1 returns $u_1 \leftarrow u^{a_1}$, party 2 returns $u_2 \leftarrow u^{a_2}$ and the results are combined by computing $v \leftarrow u_1 \cdot u_2$.
c) Party 1 returns $u_1 \leftarrow u^b$, party 2 returns u2$\leftarrow u^e$where b=1/a_1 and e=1/$a_2$and the results are combined by computing $v \leftarrow u_1+u_2$.
d) party 1 returns $u_1 \leftarrow u^{a_1}$, party 2 returns $u_2 \leftarrow u^{a_2}$ and the results are combined by computing $v \leftarrow u_1/u_2$.

Απάντηση

The correct answer is b: Indeed, $v=u_1 \cdot u_2=g^{a1+a2}=g^a$ as needed for decryption. Note that the secret key was never re-constructed for this distributed decryption to work.

Ερώτηση 10

Suppose Alice and Bob live in a country with 50 states. Alice is currently in state $a \in \{1,\ldots,50\}$ and Bob is currently in state $b \in \{1,\ldots,50\}$. They can communicate with one another and Alice wants to test if she is currently in the same state as Bob. If they are in the same state, Alice should learn that fact and otherwise she should learn nothing else about Bob's location. Bob should learn nothing about Alice's location.

They agree on the following scheme:

- They fix a group G of prime order p and generator g of G
- Alice chooses random x and y in $\mathbb{Z}p$ and sends to Bob $(A_0,A_1,A_2)=(g^x, g^y, g^{xy+a})$
- Bob choose random r and s in $\mathbb{Z}_p$ and sends back to Alice $(B_1,B_2)=(A_1{}^r g^s, (A_2/g^b)^r A_0{}^s)$

What should Alice do now to test if they are in the same state (i.e. to test if a=b)?

Note that Bob learns nothing from this protocol because he simply recieved a plain ElGamal encryption of ga under the public key gx. One can show that if $a \neq b$ then Alice learns nothing else from this protocol because she recieves the encryption of a random value.

a) Alice tests if a=b by checking if $B_2 B_1{}^x=1$
b) Alice tests if a=b by checking if $B_1{}^x B_2=1$.
c) Alice tests if a=b by checking if $B_2/B_1{}^x=1$.
d) Alice tests if a=b by checking if $B_2{}^x B_1=1$.

Απάντηση

The correct answer is c: The pair (B_1,B_2) from Bob satisfies $B_1=g^{yr+s}$ and $B_2=(g^x)^{yr+s} g^{r(a-b).}$ Therefore, it is a plain ElGamal encryption of the plaintext $g^{r(a-b)}$ under the public key (g,g^x). This plaintext happens to be 1 when a=b. The term $B_2/B_1{}^x$computes the ElGamal plaintext and compares it to 1.

Note that when $a \neq b$ the $r(a-b)$ term ensures that Alice learns nothing about b other than the fact that $a \neq b$. Indeed, when $a \neq b$ then $r(a-b)$ is a uniform non-zero element of $\mathbb{Z}p$.

Ερώτηση 11

[OPTIONAL: EXTRA CREDIT] What is the bound on d for Wiener's attack when N is a product of three equal size distinct primes?

a) $d<N^{1/6}/c$ for some constant c.
b) $d<N^{1/5}/c$ for some constant c.
c) $d<N^{2/3}/c$ for some constant c.
d) $d<N^{1/4}/c$ for some constant c.

Απάντηση

The correct answer is a: The only change to the analysis is that $N-\varphi(N)$ is now on the order of $N^{2/3}$. Everything else stays the same. Plugging in this bound gives the answer. Note that the bound is weaker in this case compared to when N is a product of two primes making the attack less effective.

Προγραμματιστικήάσκηση 6

Your goal in this project is to break RSA when the public modulus N is generated incorrectly. This should serve as yet another reminder not to implement crypto primitives yourself.

Normally, the primes that comprise an RSA modulus are generated independently of one another. But suppose a developer decides to generate the first prime p by choosing a random number R and scanning for a prime close by. The second prime q is generated by scanning for some other random prime also close to R. We show that the resulting RSA modulus N=pq can be easily factored.

Suppose you are given a composite N and are told that N is a product of two relatively close primes p and q, namely p and q satisfy:

$$|p-q|<2N^{1/4}$$

Your goal is to factor N.

Let A be the arithmetic average of the two primes, that is A=p+q/2. Since p and q are odd, we know that p+q is even and therefore A is an integer

o factor N you first observe that under condition (*) the quantity $\sqrt{N}$is very close to A. In particular

$$A-\sqrt{N}<1$$

as shown below. But since A is an integer, rounding $\sqrt{N}$up to the closest integer reveals the value of A. In code, A=ceil(sqrt(N)) where "ceil" is the ceiling function. Visually, the numbers p,q,$\sqrt{N}$and A are ordered as follows(Εικόνα):

Since A is the exact mid-point between p and q there is an integer x such that p=A−x and q=A+x. But then:

$$N=pq=(A-x)(A+x)=A^2-x^2 \text{ and therefore } x=sqrt(A^2-X^2)$$

Now, given x and A you can find the factors p and q of N since $p=A-x$ and $q=A+x$. In the following challenges, you will factor the given moduli using the method outlined above. To solve this assignment it is best to use an environment that supports multi-precision arithmetic and square roots. In Python you could use the gmpy2 module. In C you can use GMP.

Factoring challenge #1: The following modulus N is a products of two primes p and q where $|p-q|<2N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Factoring challenge #2: The following modulus N is a products of two primes p and q where $|p-q|<2^{11}N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Hint: in this case $A-\sqrt{N}<2^{20}$so try scanning for A from$\sqrt{N}$upwards, until you succeed in factoring N.

Factoring challenge #3: (extra credit) The following modulus N is a products of two primes p and q where $|3p-2q|<N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Hint: use the calculation below to show that $\sqrt{6N}$ is close to 3p+2q/2 and then adapt the method above to factor N.

Challenge #4

The challenge ciphertext provided below is the result of encrypting a short secret ASCII plaintext using the RSA modulus given in the first factorization challenge. The encryption exponent used is e=65537. The ASCII plaintext was encoded using PKCS v1.5 before the RSA function was applied, as described in Lecture 11.4.

Use the factorization you obtained for this RSA modulus to decrypt this challenge ciphertext and enter the resulting English plaintext in the box below. Recall that the factorization of N enables you to compute φ(N) from which you can obtain the RSA decryption exponent.

Challenge ciphertext (as a decimal integer):

```
22096451867410381776306561134883418017410069787892831071731839143676135600120
53800428232965047350942434394621975151225646583996794288946076454204058156474
89880137348641204523252293201764879166664029975091887299716905260832220677716
00019329260870009579993724077458967773697817571267229951148662959627934791540
```

After you use the decryption exponent to decrypt the challenge ciphertext you will obtain a PKCS1 encoded plaintext. To undo the encoding it is best to write the decrypted value in hex.

You will observe that the number starts with a '0x02' followed by many random non-zero digits. Look for the '0x00' separator and the digits following this separator are the ASCII letters of the plaintext. (note: the separator used here is '0x00', not '0xFF' as stated in the lecture)

Απάντηση

Τον κώδικα της προγραμματιστικής άσκησης θα τον βρείτε εδώ.

8.5 Αναλυτική βαθμολογία

You submitted this homework on **Fri 5 Jun 2015 11:39 AM EEST**. You got a score of **11.00** out of **11.00**.

Question 1

Recall that with symmetric ciphers it is possible to encrypt a 32-bit message and obtain a 32-bit ciphertext (e.g. with the one time pad or with a nonce-based system). Can the same be done with a public-key system?

Your Answer		Score	Explanation
No, public-key systems with short ciphertexts can never be secure.	✔	1.00	An attacker can use the public key to build a dictionary of all 2^{32} ciphertexts of length 32 bits along with their decryption and use the dictionary to decrypt any captured ciphertext.
Yes, when encrypting a short plaintext the output of the public-key encryption algorithm can be truncated to the length of the plaintext.			
It is possible and depends on the specifics of the system.			
Yes, the RSA-OAEP system can produce 32-bit ciphertexts.			
Total		1.00 / 1.00	

Εικόνα 154:Ερώτηση 1-Week 6

Question 2

Let (Gen, E, D) be a semantically secure public-key encryption system. Can algorithm E be deterministic?

Your Answer		Score	Explanation
Yes, some public-key encryption schemes are deterministic.			
Yes, RSA encryption is deterministic.			
No, but chosen-ciphertext secure encryption can be deterministic.			
No, semantically secure public-key encryption must be randomized.	✔	1.00	That's correct since otherwise an attacker can easily break semantic security.
Total		1.00 / 1.00	

Εικόνα 155:Ερώτηση 2-Week 6

Question 3

Let (Gen, E, D) be a chosen ciphertext secure public-key encryption system with message space $\{0,1\}^{128}$. Which of the following is also chosen ciphertext secure?

Your Answer		Score	Explanation
(Gen, E', D') where $E'(\mathrm{pk}, m) = \left[c \leftarrow E(\mathrm{pk}, m),\ \text{output } (c, c)\right]$ and $D'\left(\mathrm{sk}, (c_1, c_2)\right) = \begin{cases} D(\mathrm{sk}, c_1) & \text{if } c_1 = c_2 \\ \perp & \text{otherwise} \end{cases}$.	✔	0.25	This construction is chosen-ciphertext secure. An attack on (Gen, E', D) gives an attack on (Gen, E, D).
(Gen, E', D') where $E'(\mathrm{pk}, m) = \left(E(\mathrm{pk}, m),\ 0^{128}\right)$ and $D'\left(\mathrm{sk}, (c_1, c_2)\right) = D(\mathrm{sk}, c_1)$.	✔	0.25	This construction is not chosen-ciphertext secure. An attacker can output two messages $m_0 = 0^{128}$ and $m_1 = 1^{128}$ and be given back a challenge ciphertext (c_1, c_2). The attacker would then ask for the decryption of $(c_1, 1^{128})$ and be given in response m_0 or m_1 thereby letting the attacker win the game. Note that the decryption query is valid since it is different from the challenger ciphertext (c_1, c_2).
(Gen, E', D') where $E'(\mathrm{pk}, m) = \left(E(\mathrm{pk}, m),\ E(\mathrm{pk}, 0^{128})\right)$ and $D'\left(\mathrm{sk}, (c_1, c_2)\right) = D(\mathrm{sk}, c_1)$.	✔	0.25	This construction is not chosen-ciphertext secure. An attacker can output two messages $m_0 = 0^{128}$ and $m_1 = 1^{128}$ and be given back a challenge ciphertext (c_1, c_2). The attacker would then ask for the decryption of $(c_1, E(pk, 1^{128})$ and be given in response m_0 or m_1 thereby letting the attacker win the game. Note that the decryption query is valid since it is different from the challenger ciphertext (c_1, c_2).
(Gen, E', D') where $E'(\mathrm{pk}, m) = E(\mathrm{pk}, m \oplus 1^{128})$ and $D'(\mathrm{sk}, c) = D(\mathrm{sk}, c) \oplus 1^{128}$	✔	0.25	This construction is chosen-ciphertext secure. An attack on (Gen, E', D) gives an attack on (Gen, E, D).
Total		1.00 /	

Εικόνα 156:Ερώτηση 3-Week 6

Question 4

Recall that an RSA public key consists of an RSA modulus N and an exponent e. One might be tempted to use the same RSA modulus in different public keys. For example, Alice might use $(N, 3)$ as her public key while Bob may use $(N, 5)$ as his public key. Alice's secret key is $d_a = 3^{-1} \bmod \varphi(N)$ and Bob's secret key is $d_b = 5^{-1} \bmod \varphi(N)$.

In this question and the next we will show that it is insecure for Alice and Bob to use the same modulus N. In particular, we show that either user can use their secret key to factor N. Alice can use the factorization to compute $\varphi(N)$ and then compute Bob's secret key.

As a first step, show that Alice can use her public key $(N, 3)$ and private key d_a to construct an integer multiple of $\varphi(N)$. Which of the following is an integer multiple of $\varphi(N)$?

Your Answer		Score	Explanation
$d_a + 1$			
$d_a - 1$			
$3d_a - 1$	✔	1.00	Since $d_a = 3^{-1} \bmod \varphi(N)$ we know that $3d_a = 1 \bmod \varphi(N)$ and therefore $3d_a - 1$ is divisibly by $\varphi(N)$.
$5d_a - 1$			
Total		1.00 / 1.00	

Εικόνα 157:Ερώτηση 4-Week 6

Question 5

Now that Alice has a multiple of $\varphi(N)$ let's see how she can factor $N = pq$. Let x be the given muliple of $\varphi(N)$. Then for any g in $\mathbb{Z}_N^*$ we have $g^x = 1$ in $\mathbb{Z}_N$. Alice chooses a random g in $\mathbb{Z}_N^*$ and computes the sequence

$g^x, g^{x/2}, g^{x/4}, g^{x/8}$ ... in $\mathbb{Z}_N$

and stops as soon as she reaches the first element $y = g^{x/2^l}$ such that $y \neq 1$ (if she gets stuck because the exponent becomes odd, she picks a new random g and tries again). It can be shown that with probability 1/2 this y satisfies

$$\begin{cases} y = 1 \bmod p, \text{ and} \\ y = -1 \bmod q \end{cases} \quad \text{or} \quad \begin{cases} y = -1 \bmod p, \text{ and} \\ y = 1 \bmod q \end{cases}$$

How can Alice use this y to factor N?

Your Answer		Score	Explanation
compute $gcd(N, y^2 - 1)$			
compute $gcd(N - 1, y)$			
compute $gcd(N, y - 1)$	✔	1.00	We know that $y - 1$ is divisible by p or q, but not divisible by the other. Therefore, $gcd(N, y - 1)$ will output a non-trivial factor of N.
compute $gcd(N, y^2)$			
Total		1.00 / 1.00	

Εικόνα 158:Ερώτηση 5-Week 6

Question 6

In standard RSA the modulus N is a product of two distinct primes. Suppose we choose the modulus so that it is a product of three distinct primes, namely $N = pqr$. Given an exponent e relatively prime to $\varphi(N)$ we can derive the secret key as $d = e^{-1} \bmod \varphi(N)$. The public key (N, e) and secret key (N, d) work as before. What is $\varphi(N)$ when N is a product of three distinct primes?

Your Answer		Score	Explanation
$\varphi(N) = (p-1)(q-1)(r-1)$	✔	1.00	When is a product of distinct primes then $\lvert\mathbb{Z}_N^*\rvert$ satisfies $\lvert\mathbb{Z}_N^*\rvert = \lvert\mathbb{Z}_p^*\rvert \cdot \lvert\mathbb{Z}_q^*\rvert \cdot \lvert\mathbb{Z}_r^*\rvert = (p-1)(q-1)(r-1)$.
$\varphi(N) = pqr - 1$			
$\varphi(N) = (p-1)(q-1)$			
$\varphi(N) = (p-1)(q-1)(r+1)$			
Total		1.00 / 1.00	

Εικόνα 159:Ερώτηση 6-Week 6

Question 7

An administrator comes up with the following key management scheme: he generates an RSA modulus N and an element s in $\mathbb{Z}_N^*$. He then gives user number i the secret key $s_i = s^{r_i}$ in $\mathbb{Z}_N$ where r_i is the i'th prime (i.e. 2 is the first prime, 3 is the second, and so on).

Now, the administrator encrypts a file that is accssible to users i, j and t with the key $k = s^{r_i r_j r_t}$ in $\mathbb{Z}_N$. It is easy to see that each of the three users can compute k. For example, user i computes k as $k = (s_i)^{r_j r_t}$. The administrator hopes that other than users i, j and t, no other user can compute k and access the file.

Unfortunately, this system is terribly insecure. Any two colluding users can combine their secret keys to recover the master secret s and then access all files on the system. Let's see how. Suppose users 1 and 2 collude. Because r_1 and r_2 are distinct primes there are integers a and b such that $ar_1 + br_2 = 1$. Now, users 1 and 2 can compute s from the secret keys s_1 and s_2 as follows:

Your Answer		Score	Explanation
$s = s_1^a + s_2^b$ in $\mathbb{Z}_N$.			
$s = s_1^a \cdot s_2^b$ in $\mathbb{Z}_N$.	✔	1.00	$s = s_1^a \cdot s_2^b = s^{r_1 a} \cdot s^{r_2 b} = s^{r_1 a + r_2 b} = s$ in $\mathbb{Z}_N$.
$s = s_1^b / s_2^a$ in $\mathbb{Z}_N$.			
$s = s_1^b + s_2^a$ in $\mathbb{Z}_N$.			
Total		1.00 / 1.00	

Εικόνα 160:Ερώτηση 7-Week 6

Question 8

Let G be a finite cyclic group of order n and consider the following variant of ElGamal encryption in G:

- Gen: choose a random generator g in G and a random x in $\mathbb{Z}_n$. Output $\text{pk} = (g, h = g^x)$ and $\text{sk} = (g, x)$.
- $E(\text{pk}, m \in G)$: choose a random r in $\mathbb{Z}_n$ and output $(g^r, m \cdot h^r)$.
- $D(\text{sk}, (c_0, c_1))$: output c_1 / c_0^x.

This variant, called plain ElGamal, can be shown to be semantically secure under an appropriate assumption about G. It is however not chosen-ciphertext secure because it is easy to compute on ciphertexts. That is, let (c_0, c_1) be the output of $E(\text{pk}, m_0)$ and let (c_2, c_3) be the output of $E(\text{pk}, m_1)$. Then just given these two ciphertexts it is easy to construct the encryption of $m_0 \cdot m_1$ as follows:

Your Answer		Score	Explanation
$(c_0 - c_2, c_1 - c_3)$ is an encryption of of $m_0 \cdot m_1$.			
$(c_0 c_3, c_1 c_2)$ is an encryption of of $m_0 \cdot m_1$.			
$(c_0 c_2, c_1 c_3)$ is an encryption of of $m_0 \cdot m_1$.	✔	1.00	Indeed, $(c_0 c_2, c_1 c_3) = (g^{r_0 + r_1}, m_0 m_1 h^{r_0 + r_1})$, which is a valid encryption of $m_0 m_1$.
$(c_0 / c_3, c_1 / c_2)$ is an encryption of of $m_0 \cdot m_1$.			
Total		1.00 / 1.00	

Εικόνα 161:Ερώτηση 8-Week 6

Question 9

Let G be a finite cyclic group of order n and let $\text{pk} = (g, h = g^a)$ and $\text{sk} = (g, a)$ be an ElGamal public/secret key pair in G as described in Segment 12.1. Suppose we want to distribute the secret key to two parties so that both parties are needed to decrypt. Moreover, during decryption the secret key is never re-constructed in a single location. A simple way to do so it to choose random numbers a_1, a_2 in $\mathbb{Z}_n$ such that $a_1 + a_2 = a$. One party is given a_1 and the other party is given a_2. Now, to decrypt an ElGamal ciphertext (u, c) we send u to both parties. What do the two parties return and how do we use these values to decrypt?

Your Answer	Score	Explanation
party 1 returns $u_1 \leftarrow u^{(a_1^2)}$, party 2 returns $u_2 \leftarrow u^{(a_2^2)}$ and the results are combined by computing $v \leftarrow u_1 \cdot u_2$.		
party 1 returns $u_1 \leftarrow u^{a_1}$, party 2 returns $u_2 \leftarrow u^{a_2}$ and the results are combined by computing $v \leftarrow u_1 \cdot u_2$.	✓ 1.00	Indeed, $v = u_1 \cdot u_2 = g^{a_1 + a_2} = g^a$ as needed for decryption. Note that the secret key was never re-constructed for this distributed decryption to work.
party 1 returns $u_1 \leftarrow u^{1/a_1}$, party 2 returns $u_2 \leftarrow u^{1/a_2}$ and the results are combined by computing $v \leftarrow u_1 + u_2$.		
party 1 returns $u_1 \leftarrow u^{a_1}$, party 2 returns $u_2 \leftarrow u^{a_2}$ and the results are combined by computing $v \leftarrow u_1 / u_2$.		
Total	1.00 / 1.00	

Εικόνα 162:Ερώτηση 9-Week 6

Question 10

Suppose Alice and Bob live in a country with 50 states. Alice is currently in state $a \in \{1, \ldots, 50\}$ and Bob is currently in state $b \in \{1, \ldots, 50\}$. They can communicate with one another and Alice wants to test if she is currently in the same state as Bob. If they are in the same state, Alice should learn that fact and otherwise she should learn nothing else about Bob's location. Bob should learn nothing about Alice's location.

They agree on the following scheme:

- They fix a group G of prime order p and generator g of G
- Alice chooses random x and y in $\mathbb{Z}_p$ and sends to Bob $(A_0, A_1, A_2) = \left(g^x,\ g^y,\ g^{xy+a}\right)$
- Bob choose random r and s in $\mathbb{Z}_p$ and sends back to Alice $(B_1, B_2) = \left(A_1^r g^s,\ (A_2/g^b)^r A_0^s\right)$

What should Alice do now to test if they are in the same state (i.e. to test if $a = b$) ?

Note that Bob learns nothing from this protocol because he simply recieved a plain ElGamal encryption of g^a under the public key g^x. One can show that if $a \neq b$ then Alice learns nothing else from this protocol because she recieves the encryption of a random value.

Your Answer	Score	Explanation
Alice tests if $a = b$ by checking if $B_2 B_1^x = 1$.		
Alice tests if $a = b$ by checking if $B_1^x B_2 = 1$.		
Alice tests if $a = b$ by checking if $B_2 / B_1^x = 1$.	✓ 1.00	The pair (B_1, B_2) from Bob satisfies $B_1 = g^{yr+s}$ and $B_2 = (g^x)^{yr+s} g^{r(a-b)}$. Therefore, it is a plain ElGamal encryption of the plaintext $g^{r(a-b)}$ under the public key (g, g^x). This plaintext happens to be 1 when $a = b$. The term B_2 / B_1^x computes the ElGamal plaintext and compares it to 1. Note that when $a \neq b$ the $r(a-b)$ term ensures that Alice learns nothing about b other than the fact that $a \neq b$. Indeed, when $a \neq b$ then $r(a-b)$ is a uniform non-zero element of $\mathbb{Z}_p$.
Alice tests if $a = b$ by checking if $B_2^x B_1 = 1$.		

Εικόνα 163:Ερώτηση 10-Week 6

Question 11

[OPTIONAL: EXTRA CREDIT] What is the bound on d for Wiener's attack when N is a product of **three** equal size distinct primes?

Your Answer	Score	Explanation
◉ $d < N^{1/6}/c$ for some constant c.	✓ 1.00	The only change to the analysis is that $N - \varphi(N)$ is now on the order of $N^{2/3}$. Everything else stays the same. Plugging in this bound gives the answer. Note that the bound is weaker in this case compared to when N is a product of two primes making the attack less effective.
○ $d < N^{1/5}/c$ for some constant c.		
○ $d < N^{2/3}/c$ for some constant c.		
○ $d < N^{1/4}/c$ for some constant c.		
Total	1.00 / 1.00	

Εικόνα 164:Ερώτηση 11-Week 6

You submitted this homework on **Fri 5 Jun 2015 11:42 AM EEST**. You got a score of **4.00** out of **4.00**.

Question 1

Your goal in this project is to break RSA when the public modulus N is generated incorrectly. This should serve as yet another reminder not to implement crypto primitives yourself.

Normally, the primes that comprise an RSA modulus are generated independently of one another. But suppose a developer decides to generate the first prime p by choosing a random number R and scanning for a prime close by. The second prime q is generated by scanning for some other random prime also close to R. We show that the resulting RSA modulus $N = pq$ can be easily factored.

Suppose you are given a composite N and are told that N is a product of two relatively close primes p and q, namely p and q satisfy

$|p - q| < 2N^{1/4}$ (*)

Your goal is to factor N.

Let A be the arithmetic average of the two primes, that is $A = \frac{p+q}{2}$. Since p and q are odd, we know that $p + q$ is even and therefore A is an integer.

To factor N you first observe that under condition (*) the quantity $\sqrt{N}$ is very close to A. In particular

$A - \sqrt{N} < 1$

as shown below. But since A is an integer, rounding $\sqrt{N}$ up to the closest integer reveals the value of A. In code, $A = \text{ceil}(\text{sqrt}(N))$ where "ceil" is the ceiling function. Visually, the numbers $p, q, \sqrt{N}$ and A are ordered as follows:

Εικόνα 165:Προγραμματιστική άσκηση 6-a

Since A is the exact mid-point between p and q there is an integer x such that $p = A - x$ and $q = A + x$. But then

$N = pq = (A - x)(A + x) = A^2 - x^2$ and therefore $x = \sqrt{A^2 - N}$

Now, given x and A you can find the factors p and q of N since $p = A - x$ and $q = A + x$.

In the following challenges, you will factor the given moduli using the method outlined above. To solve this assignment it is best to use an environment that supports multi-precision arithmetic and square roots. In Python you could use the gmpy2 module. In C you can use GMP.

Factoring challenge #1: The following modulus N is a products of two primes p and q where $|p - q| < 2N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

```
N = 179769313486231590772930519078902473361797697894230657273430081l5 \
    773267580550562068698537949211298295958550138753716401571013985l6 \
    478337786069255834975410851965916151280575759407526350074759352l8 \
    710823649949407718956170543611494748650467110151815639406805275l \
    007158456087857766374304008634074285527854909258l
```

Factoring challenge #2: The following modulus N is a products of two primes p and q where $|p - q| < 2^{11}N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

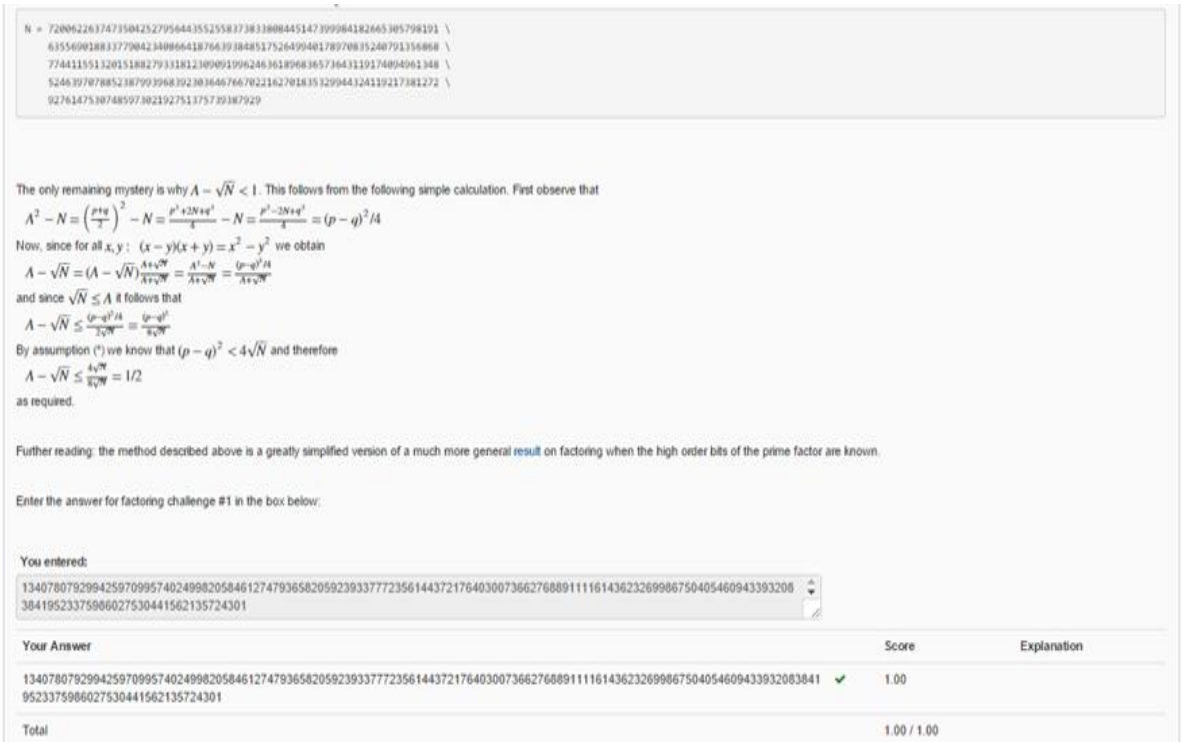
Hint: in this case $A - \sqrt{N} < 2^{20}$ so try scanning for A from $\sqrt{N}$ upwards, until you succeed in factoring N.

```
N = 648455842808071669662824265346772278726343720706976263060439070378l \
    973086180811164627140152760614175691955873218402545206554249067198l \
    242884484183935328197298853131051173864896596258282150258499026445l \
    100885281673303711142296421027840289307657458645233683357077834689l \
    158386460882396402368662522117900857878l7
```

Factoring challenge #3: (extra credit) The following modulus N is a products of two primes p and q where $|3p - 2q| < N^{1/4}$. Find the smaller of the two factors and enter it as a decimal integer.

Hint: use the calculation below to show that $\sqrt{6N}$ is close to $\frac{3p+2q}{2}$ and then adapt the method above to factor N.

Εικόνα 166:Προγραμματιστική άσκηση 6-b



```
N = 720062263747350425279564435525583738338084451473999841826653057981911 \
    635546901883377704423408664187643938485175264990401789708352407913568681 \
    774411551320151882793318123090919962463618968365736431191740949613481 \
    524639707885238799396832303646766702216270183532994432411921738127211 \
    927614753074859730219275137573938792l
```

The only remaining mystery is why $A - \sqrt{N} < 1$. This follows from the following simple calculation. First observe that

$A^2 - N = \left(\frac{p+q}{2}\right)^2 - N = \frac{p^2+2N+q^2}{4} - N = \frac{p^2-2N+q^2}{4} = (p-q)^2/4$

Now, since for all x, y : $(x - y)(x + y) = x^2 - y^2$ we obtain

$A - \sqrt{N} = (A - \sqrt{N})\frac{A+\sqrt{N}}{A+\sqrt{N}} = \frac{A^2-N}{A+\sqrt{N}} = \frac{(p-q)^2/4}{A+\sqrt{N}}$

and since $\sqrt{N} \le A$ it follows that

$A - \sqrt{N} \le \frac{(p-q)^2/4}{2\sqrt{N}} = \frac{(p-q)^2}{8\sqrt{N}}$

By assumption (*) we know that $(p - q)^2 < 4\sqrt{N}$ and therefore

$A - \sqrt{N} \le \frac{4\sqrt{N}}{8\sqrt{N}} = 1/2$

as required.

Further reading: the method described above is a greatly simplified version of a much more general result on factoring when the high order bits of the prime factor are known.

Enter the answer for factoring challenge #1 in the box below:

You entered:

```
134078079299425970995740249982058461274793658205923933777235614437217640300736627688911116143623269986750405460943393208
384195233759860275304415621357243O1
```

Your Answer	Score	Explanation
134078079299425970995740249982058461274793658205923933777235614437217640300736627688911116143623269986750405460943393208384195233759860275304415621357243O1 ✔	1.00	
Total	1.00 / 1.00	

Εικόνα 167:Προγραμματιστική άσκηση 6-c

Πίνακας 6:Βαθμολογίες-Week 6

	1η προσπάθεια	2η προσπάθεια	3η προσπάθεια	4η προσπάθεια
Ερώτησεις	9.00/11.00	11.00/11.00	-	-
Άσκηση	4.00/4.00	-	-	-

8.8 Πιστοποιητικό παρακολούθησης(certificate)

JULY 06, 2015

Online Course Statement of Accomplishment

WITH DISTINCTION

GEORGIOS OGLOU

HAS SUCCESSFULLY COMPLETED A FREE ONLINE OFFERING OF THE FOLLOWING COURSE PROVIDED BY STANFORD UNIVERSITY THROUGH COURSERA INC.

Cryptography I

This course covers the theory and practice of cryptographic systems. Topics included symmetric encryption, data integrity, public-key encryption, and key exchange. The course emphasized the correct use of these primitive.

Dan Boneh

DAN BONEH
PROFESSOR OF COMPUTER SCIENCE,
STANFORD UNIVERSITY

ΠαράρτημαΑ:Ακρωνύμια - Συντομογραφίες

ΟΛΑ τα ακρωνύμια και οι συντομογραφίες που χρησιμοποιήθηκαν στην αναφορά, να ορίζονται παρακάτω.

0-9	**3DES**	Triple Data Encryption Standard
	5GL	5th Generation Language
A	**ACE**	Access Control Entry
	AES	Advanced Encryption Standard
B	**BIOS**	Basic Input/Output System
	BCD	Binary Coded Decimal
C	**CAPICOM**	Cryptographic API for COM
	CERT	Computer Emergency Response Team
D	**DCOM**	Distributed Component Object Model
	DEP	Data Execution Prevention
E	**ECM**	Enterprise Configuration Manager
	EFS	Encrypting File System
F	**FEK**	File Encryption Key
	FIPS	Federal Information Processing Standard
G	**GINA**	Graphical Identification and Authentication
	GPMC	Group Policy Management Console
H	**HKLM**	HKEY_Local_Machine
	HTTPS	HTTP Over SSL
I	**ICF**	Internet Connection Firewall
	IDS	Intrusion Detection System
J	**JCL**	Job Control Language
	JKS	Java Key Store
K	**KDC**	KerberosKeyDistributionCenter
	KEK	Key Encrypting Key
L	**L2TP**	Layer 2 Tunneling Protocol
	LSA	Local Security Authority
M	**MBSA**	Microsoft Baseline Security Analyzer
	MIIS	Microsoft Identify Integration Server

N	**NAP**	Network Access Protection
	NSA	National Security Agency
O	**OMB**	Office of Management and Budget
	OVAL	Open Vulnerability and Assessment Language
P	**PIN**	Personal Identification Number
	PKI	Public Key Infrastructure
Q	**QoS**	Quality of Service
	QRA	Quantitative Risk Analysis
R	**RA**	Remote Assistance
	RID	User Relative ID
S	**SACL**	System Access Control List
	SAM	Security Accounts Manager
T	**TGS**	Ticket Granting Service
	TLS	Transport Layer Security
U	**UDP**	User Datagram Protocol
	UPS	Uninterruptible Power Supply
V	**VBS**	Visual Basic Script
	VPN	Virtual Private Network
W	**WebDAV**	Web Distributed Authoring and Versioning
	WPA	Wi-Fi Protected Access
X	**XCCDF**	Extendible Configuration Checklist Description Format
	XNOS	Experimental Network Operating System
Y	**Y2K**	Bug Year 2000
	YB	YottaByte 10^{24}Bytes
		]
Z	**ZAC**	Zero Administration Client
	ZBR	Zero Bug Release

Παράρτημα Β: Προγραμματιστικές ασκήσεις

Προγραμματιστική άσκηση 1(assignment1.py)

```
"""Quickly written one-time pad decryption based on 10 cyphertext with same
key.

"""
import itertools
import string

CTS = [ct.decode('hex') for ct
in"""315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb
778cdf2d3aff021dfff5b403b510d0d0455468aeb98622b137dae857553ccd8883a7bc37520e0
6e515d22c954eba5025b8cc57ee59418ce7dc6bc41556bdb36bbca3e8774301fbcaa3b83b2208
09560987815f65286764703de0f3d524400a19b159610b11ef3e234c02ecbbfbafa3ed18510ab
d11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef4081ffde5ad46a5069d
9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919cf3a9f1cb74151f6d551f4480
c82b2cb24cc5b028aa76eb7b4ab24171ab3cdadb8356f32510ba9a7b2bba9b8005d43a304b571
4cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e51d04e0e7b207b70b9b8261112
bacb6c866a232dfe257527dc29398f5f3251a0d47e503c66e935de81230b59b7afb5f41afa8d6
61cb32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47
a8ad3306ef5021eafe1ac01a81197847a5c68a1b78769a37bc8f4575432c198ccb4ef63590256
e305cd3a9544ee4160ead45aef520489e7da7d835402bca670bda8eb775200b8dabbba246b130
f040d8ec6447e2c767f3d30ed81ea2e4c1404e1315a1010e7229be6636aaa3f561ba9adb4b6eb
ec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d9607cee021dafe1e
001b21ade877a5e68bea88d61b93ac5ee0d562e8e9582f5ef375f0a4ae20ed86e935de81230b5
9b73fb4302cd95d770c65b40aaa065f2a5e33a5a0bb5dcaba43722130f042f8ec85b7c2070325
10bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8de28
7be84d07e7e9a30ee714979c7e1123a8bd9822a33ecaf512472e8e8f8db3f9635c1949e640c62
1854eba0d79eccf52ff111284b4cc61d11902aebc66f2b2e436434eacc0aba938220b084800c2
ca4e693522643573b2c4ce35050b0cf774201f0fe52ac9f26d71b6cf61a711cc229f77ace7aa8
8a2f19983122b11be87a59c355d25f8e432510bfbacfbb9befd54415da243e1695ecabd58c519
cd4bd90f1fa6ea5ba47b01c909ba7696cf606ef40c04afe1ac0aa8148dd066592ded9f8774b52
9c7ea125d298e8883f5e9305f4b44f915cb2bd05af51373fd9b4af511039fa2d96f83414aaaf2
61bda2e97b170fb5cce2a53e675c154c0d9681596934777e2275b381ce2e40582afe67650b13e
72287ff2270abcf73bb028932836fbdecfecee0a3b894473c1bbeb6b4913a536ce4f9b13f1eff
f71ea313c8661dd9a4ce315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde
54ce56801d943ba708b8a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f89ee04132ecb
9298f5fd2d5e4b45e40ecc3b9d59e9417df7c95bba410e9aa2ca24c5474da2f276baa3ac32591
8b2daada43d6712150441c2e04f6565517f317da9d3271946f9bbb2aeadec111841a81abc300e
caa01bd8069d5cc91005e9fe4aad6e04d513e96d99de2569bc5e50eeeca709b50a8a987f4264e
db6896fb537d0a716132ddc938fb0f836480e06ed0fcd6e9759f40462f9cf57f4564186a2c177
8f1543efa270bda5e933421cbe88a4a52222190f471e9bd15f652b653b7071aec59a2705081ff
e72651d08f822c9ed6d76e48b63ab15d0208573a7eef02766d06ece998b7a2fb1d464fed2ced7
641ddaa3cc31c9941cf110abbf409ed39598005b3399ccfafb61d0315fca0a314be138a9f3250
3bedac8067f03adbf3575c3b8edc9ba7f537530541ab0f9f3cd04ff50d66f1d559ba520e89a2c
b2a83""".split()]

TCT="32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba
6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b57bddef121336cb85ccb8f3315f4b5
2e301d16e9f52f904".decode('hex')
KEY=list("_______________________________________________________________________
_____________________")
defstrxor(a, b):     # xor two strings of different lengths
iflen(a) >len(b):
return"".join([chr(ord(x) ^ ord(y)) for (x, y) inzip(a[:len(b)], b)])
else:
return"".join([chr(ord(x) ^ ord(y)) for (x, y) inzip(a, b[:len(a)])])

defmain():
for CT1, CT2 in itertools.permutations(CTS, 2):
        xor = strxor(CT1, CT2)
for i, c inenumerate(xor):
if i >= len(KEY):
continue# we don't care
if KEY[i] != '_':
continue
```

```
if c in string.ascii_uppercase:
for ct in CTS:
if i <len(ct) and ct != CT2 and strxor(CT1, ct)[i] in string.ascii_uppercase:
# there is a space in CT1 and c lower in CT2
                        KEY[i] = strxor(CT1[i], ' ')
continue
if i <len(ct) and ct != CT1 and strxor(CT2, ct)[i] in string.ascii_uppercase:
# there is a space in CT2 and c lower in CT1
                        KEY[i] = strxor(CT2[i], ' ')
continue
if c in string.ascii_lowercase:
for ct in CTS:
if i <len(ct) and ct != CT2 and strxor(CT1, ct)[i] in string.ascii_lowercase:
# there is a space in CT1 and c lower in CT2
                        KEY[i] = strxor(CT1[i], ' ')
continue
if i <len(ct) and ct != CT1 and strxor(CT2, ct)[i] in string.ascii_lowercase:
# there is a space in CT2 and c lower in CT1
                        KEY[i] = strxor(CT2[i], ' ')
continue
    print KEY
print"decrypted text: ", list(strxor("".join(KEY), TCT))
if __name__ == '__main__':
main()
```

Προγραμματιστική ασκηση 2(asssignment2.py)

```
# -*- coding: utf-8 -*-
from Crypto.Cipher import AES
# Question 1 (CBC)
key = "140b41b22a29beb4061bda66b6747e14".decode('hex')
ciphertext =
"4ca00ff4c898d61e1edbf1800618fb2828a226d160dad07883d04e008a7897ee2e4b7465d529
0d0c0e6c6822236e1daafb\
94ffe0c5da05d9476be028ad7c1d81".decode('hex')
size = AES.block_size
#iv = Random.new().read(AES.block_size)
iv = ciphertext[:size]
cipher = AES.new(key, AES.MODE_CBC, iv)
msg = cipher.decrypt(ciphertext[16:])
print msg
# Question 2 (CBC)
key = "140b41b22a29beb4061bda66b6747e14".decode('hex')
ciphertext =
"5b68629feb8606f9a6667670b75b38a5b4832d0f26e1ab7da33249de7d4afc48\
e713ac646ace36e872ad5fb8a512428a6e21364b0c374df45503473c5242a253".decode('hex
')
iv = ciphertext[:AES.block_size]
cipher = AES.new(key,AES.MODE_CBC, iv)
msg = cipher.decrypt(ciphertext[16:])
print msg
defpycrypto_decrypt(key, iv, data):
    crypt = AES.new(key, AES.MODE_CTR, counter=lambda: iv)
return crypt.decrypt(data)
# Question 3 (CTR)
key = "36f18357be4dbd77f050515c73fcf9f2".decode('hex')
cyphertext =
"69dda8455c7dd4254bf353b773304eec0ec7702330098ce7f7520d1cbbb20fc3\
88d1b0adb5054dbd7370849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f5f51eeca32eab
edd9afa9329"
iv1 = cyphertext[:32].decode('hex')
iv2 = hex(int(cyphertext[:32], 16) + 1)[2:][:-1].decode('hex')  # iv+1
iv3 = hex(int(cyphertext[:32], 16) + 2)[2:][:-1].decode('hex')  # iv+2
iv4 = hex(int(cyphertext[:32], 16) + 3)[2:][:-1].decode('hex')  # iv+3
ct1 = cyphertext[32:64].decode('hex')
ct2 = cyphertext[64:96].decode('hex')
```

```
ct3 = cyphertext[96:128].decode('hex')
ct4 = cyphertext[128:].decode('hex')
print pycrypto_decrypt(key, iv1, ct1) + pycrypto_decrypt(key, iv2, ct2) +
pycrypto_decrypt(key, iv3, ct3) + pycrypto_decrypt(key, iv4, ct4)
# Question 4 (CTR)
key = "36f18357be4dbd77f050515c73fcf9f2".decode('hex')
ctr_cyphertext2 =
"770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa\
0e311bde9d4e01726d3184c34451"
iv1 = ctr_cyphertext2[:32].decode('hex')
iv2 = hex(int(ctr_cyphertext2[:32], 16) + 1)[2:][:-1].decode('hex')  # iv+1
iv3 = hex(int(ctr_cyphertext2[:32], 16) + 2)[2:][:-1].decode('hex')  # iv+2
iv4 = hex(int(ctr_cyphertext2[:32], 16) + 3)[2:][:-1].decode('hex')  # iv+3
ct1 = ctr_cyphertext2[32:64].decode('hex')
ct2 = ctr_cyphertext2[64:96].decode('hex')
ct3 = ctr_cyphertext2[96:128].decode('hex')
ct4 = ctr_cyphertext2[128:].decode('hex')
print pycrypto_decrypt(key, iv1, ct1) + pycrypto_decrypt(key, iv2, ct2) +
pycrypto_decrypt(key, iv3, ct3) + pycrypto_decrypt(key, iv4, ct4)
```

Προγραμματιστικήάσκηση 3(asssignment3.py)

```
import hashlib

defget_h_from_file(filename):
    blocks = []
    prev_h = ''
    with open(filename, 'rb') as f:
        data = None
while data != '':
            data = f.read(1024)
if data != '':
                blocks.insert(0, data)

for block in blocks:
        m = block + prev_h
        h = hashlib.sha256(m)
        prev_h = h.digest()
        hex_h = h.hexdigest()

return hex_h

print('sample: ' + get_h_from_file('assignment3-sample.mp4'))
```

Προγραμματιστική άσκηση 4(asssignmen4.py)

```
importurllib2
import sys

defquery(q):
        target = 'http://crypto-class.appspot.com/po?er=' + urllib2.quote(q)
# Create query URL
        req = urllib2.Request(target) # Send HTTP request to server
try:
            f = urllib2.urlopen(req)    # Wait for response
except urllib2.HTTPError, e:
if e.code == 404:
returnTrue # good padding
returnFalse # bad padding
ct =
'f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb
515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4'
def int2hex(i):
returnhex(i)[2:] iflen(hex(i)[2:]) == 2 else '0' + hex(i)[2:]
```

```
defexor_pad(i):
assert(i > 0)
assert(i<=16)
return'00' * (16 -i) + int2hex(i) * i
defexor_g(g,pos):
assert(pos>=0)
assert(pos<16)
return'00' * (15-pos) + int2hex(g) + '00' * pos

def rellenar_zero(s):
return'0' * (32 - len(s)) + s

defstrxor(a, b):     # xor two strings of different lengths
if len(a) >len(b):
return"".join([chr(ord(x) ^ ord(y)) for (x, y) inzip(a[:len(b)], b)])
else:
return"".join([chr(ord(x) ^ ord(y)) for (x, y) inzip(a, b[:len(a)])])
def hexexor(s1,s2):
return strxor(s1.decode("hex"),s2.decode("hex")).encode("hex")
blocks = ()
while ct:
    blocks = blocks + (ct[:32],)
    ct = ct[32:]
message = ()
probable_range = [ord(' '),] +  range(ord('a'),ord('z')) +
range(ord('A'),ord('Z')) + range(17)
for b inrange(1,len(blocks)):
    iv = blocks[b-1]
    block = blocks[b]
    msg = ""
for pos inrange(1,17):
        pad = exor_pad(pos)
        lastmsg = rellenar_zero(msg.encode("hex"))
        getletter = 0
for g in probable_range:
            gpad = exor_g(g,pos-1)
print"Try " + int2hex(g)
if query("".join(blocks[:(b-1)]) +
hexexor(lastmsg,hexexor(iv,hexexor(gpad,pad))) + block):
                getletter = 1
break
 if getletter:
            msg = int2hex(g).decode("hex") + msg
        else:
            msg = '?' + msg
print"message='" + msg + "'"
    message = message + (msg,)
print "Message Block " + str(b) + ":" + message[b-1]



iv = blocks[2]
block = blocks[3]
msg = '09'.decode("hex") * 9
for pos inrange(10,17):
        pad = exor_pad(pos)
        lastmsg = rellenar_zero(msg.encode("hex"))
        getletter = 0
for g in probable_range:
            gpad = exor_g(g,pos-1)
print"Try " + int2hex(g)
if query("".join(blocks[:(b-1)]) +
hexexor(lastmsg,hexexor(iv,hexexor(gpad,pad))) + block):
                getletter = 1
break
if getletter:
            msg = int2hex(g).decode("hex") + msg
```

```
else:
            msg = '?' + msg
print"message='" + msg + "'"
        message = message + (msg,)
print"Message Block " + str(b) + ":" + message[b-1]
```

Προγραμματιστικήάσκηση 5(assignment5.py)

```
#!/usr/bin/python
import sys
import math
import gmpy2
from gmpy2 import mpz
from gmpy2 import invert
from gmpy2 import powmod
from gmpy2 import divm

defcompute_x0s(p,h,g,B):
return ((i, powmod(g, B*i, p)) for i inrange(B))

defdiscrete_log(p, h, g, maxExp=40):
""" Computes x such that h = g^x mod p
   """
   B = mpz(2**(int(maxExp/2)))
   g = mpz(g)
   h = mpz(h)
   p = mpz(p)
print("Computing x1s...")
   x1s = { divm(h, powmod(g,i,p), p) : i for i inrange(B) }
print("Checking for equality...")

for x0, exprR in compute_x0s(p,h,g,B):
       x1 = x1s.get(exprR)
if x1 is notNone:
print("Found values!")
print("x0 ={}".format(x0))
print("x1 = {}".format(x1))
return mpz(x0)*B+mpz(x1)
 raiseValueError("No suitable x0, x1 found!")

defself_test():
p =
13407807929942597099574024998205846127479365820592393377723561443721764030073
54697680187429816690342769003185818648605085375388281194656994643364900608417
1
      g =
11717829880366207009516117596335367088558084999998952205599979459063929499736
58374667057217647146031292859482967542827946656652711521274846758989460196556
8

    h =
32394751040504504435652643787280657886490975209524495278347924529719819761432
92558073856937958553180532878928001494706097394108577585732452307673444020333

print("Running tiny test")
    xTiny = 3
    x = discrete_log(97, 20, 57, 6)
print("x== {}".format(x))
assert(xTiny == x)
print("Tiny test passed!")
print("")
print("Running short test")
    xShort = 23232
    x = discrete_log(1938281, 190942, 1737373, 16)
print("x == {}".format(x))
assert(xShort == x)
print("Short test passed!")
```

```
print("")
print("Running long test")
    x = discrete_log(p, h, g, 40)
assert(h == powmod(g,x,p))
print("x == {}".format(x))
print("Long test passed”)
print("")

if __name__ == "__main__":
    self_test()
```

Προγραμματιστική ασκηση 6(factor.py-challenges.py)

factor.py

```
#!/usr/bin/python
import gmpy2
from gmpy2 import mpz
from gmpy2 import powmod
from gmpy2 import isqrt
from gmpy2 import isqrt_rem
from gmpy2 import div
from gmpy2 import invert
from math import ceil
from binascii import unhexlify
import challenges

defceil_sqrt(x):
    s,t = isqrt_rem(x)
return s + (1 if t else 0)
defcheck_factors(p,q,N):
return p*q == N
deffactor_with_average(A, N):
    x = isqrt(A**2 - N)
return (A - x, A + x)
defcheck_ch3(i,A,N):
    p,q = (div(A + i - 1,3), div(A - i,2))
if check_factors(p,q,N):
return p,q
    p,q = (div(A - i,3), div(A + i - 1,2))
if check_factors(p,q,N):
return p,q
returnNone

defch3_factor(N):
""" Valid when |3p - 2q| < N^(1/4)
"""
    A = ceil_sqrt(6*N)

    # let M = (3p+2q)/2
    # M is not an integer since 3p + 2q is odd
    # So there is some integer A = M + 0.5 and some integer i such that
    # 3p = M + i - 0.5 = A + i - 1
    # and
    # 2q = M - i + 0.5 = A - i
    # N = pq = (A-i)(A+i-1)/6 = (A^2 - i^2 - A + i)/6
    # So 6N = A^2 - i^2 - A + i
    # i^2 - i = A^2 - A - 6N
    # Solve using the quadratic equation!

    a = mpz(1)
    b = mpz(-1)
    c = -(A**2 - A - 6*N)
    det = b**2 - 4*a*c
    roots = (div(-b + isqrt(b**2 - 4*a*c), 2*a),
         div(-b - isqrt(b**2 - 4*a*c), 2*a))
```

```
for i in roots:
if i >= 0:
            f = check_ch3(i,A,N)
if f:
return f
# We should have found the root
assert(False)
deffactor(N):
    N = mpz(N)
    # Valid when |p-q| < 2N^(1/4)
    A = ceil_sqrt(N)
    p,q = factor_with_average(A, N)
if check_factors(p,q,N):
return (p,q)
# Valid when |p-q| < 2^11 * N^(1/4)
for i inrange(2**20):
        A += 1
        p,q = factor_with_average(A, N)
if check_factors(p,q,N):
return (p,q)
return ch3_factor(N)
defdecrypt_RSA(ciphertext, pk):
    N, e = pk
    p,q = factor(N)
    phiN = N - p - q + 1
    d = invert(e, phiN)
return powmod(ciphertext, d, N)
def self_test():
    Ns = challenges.Ns
for num,N inenumerate(Ns):
        p,q = factor(N)
if check_factors(p,q,N):
print("N[{}]: Found p = \n{}\n".format(num, min(p,q)))
else:
print("ERROR: Incorrectly factored N[{}]!".format(num))
# Find the plaintext
    pt = decrypt_RSA(challenges.ciphertext_1, (challenges.N_1,
challenges.e_1))
    ptHex = hex(pt)
    pos = ptHex.find("00")
print("Plaintext:")
print(ptHex)
print("Message:")
print(unhexlify(ptHex[pos+2:]))
if __name__ == "__main__":
    self_test()
```

challenges.py

```
N_1 =
17976931348623159077293051907890247336179769789423065727343008115773267580550
56206869853794492129829595855013875371640157101398586478337786069255834975410
85196591615128057575940752635007475935288710823649949940771895617054361149474
86504671101510156394068052754007158456087857766374304008634074285527854909258
1

N_2 =
64845584280807166966282426534677227872634372070697626306043907037879730861808
11164627140152760614175691955873218402545206554249067198924288448418393532819
72988531310511738648965962582821502504990264452100885281673303711142296421027
84028930765745864523368335707783468971583864608823964023686625221179008578787
7

N_3 =
72006226374735042527956443552558373833808445147399984182665305798191635569018
83377904234086641876639384851752649940178970835240791356868774411551320151882
```

```
7933181230909199624636189683657364311917409496134852463970788523879939683923036467667022162701835329944324119217381272927614753074859730219275137573938792
9

ciphertext_1 =
2209645186741038177630656113488341801741006978789283107173183914367613560012053800428232965047350942434394621975151225646583996794288946076454204058156474898801373486412045232522932017648791666640299750918872997169052608322206777160001932926087000957999372407745896777369781757126722995114866295962793479154 0

e_1 = 65537

Ns = (N_1, N_2, N_3)
```

Παράρτημα Γ: Βιβλιογραφία

Ελληνική Βιβλιογραφία

Κρυπτογραφικοί αλγόριθμοι ροής,Wikipedia,
https://el.wikipedia.org/wiki/%CE%9A%CF%81%CF%85%CF%80%CF%84%CE%BF%CE%B3%CF%81%CE%B1%CF%86%CE%B9%CE%BA%CE%BF%CE%AF_%CE%91%CE%BB%CE%B3%CF%8C%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CE%B9_%CE%A1%CE%BF%CE%AE%CF%82
Στεφανής Παναγιώτης,Προγραμματισμός και οπτικοποίηση αλγορίθμων εκτέλεσης πράξεων αριθμητικής υπολοίπων,πανεπιστήμιο Μακεδονίας,2007-2008,
https://dspace.lib.uom.gr/bitstream/2159/13379/2/StefanisMsc2008.pdf
Άρης παγουρτζής-Στάθης Ζάχος,Στοιχεία θεωρίας αριθμών και εφαρμογές στην Κρυπτογραφία-Κρυπτογραφία και πολυπλοκότητα,Corelab,ολες οι διαλέξεις,
http://www.corelab.ntua.gr/courses/crypto/
Στάθης Ζάχος-Άρης παγουρτζής,Στοιχεία θεωρίας αριθμών και εφαρμογές στην Κρυπτογραφία,Εθνικό Μετσόβειο Πολυτεχνείο,Σημειώσεις διαλέξεων,12 Νοεμβρίου 2012,
http://www.corelab.ntua.gr/courses/crypto/notes2012/crypto-12-Nov-2012.pdf
Γιώργος Ρίζος,Μαθηματικά σύμβολα,mathematica.gr,
http://www.mathematica.gr/forum/viewtopic.php?f=6&t=728
ainigmatica000,Εισαγωγή στην Κρυπτογραφία(Μέρος πρώτο),
http://kryptografies.blogspot.gr/2014/01/blog-post.html
ainigmatica000,Εισαγωγή στην Κρυπτογραφία(Μέρος δέυτερο),
http://kryptografies.blogspot.gr/2015/04/blog-post_81.html
Κωνσταντίνου Ελισάβετ,Εφαρμοσμένη κρυπτογραφία Ι,
http://www.icsd.aegean.gr/website_files/metaptyxiako/365678859.pdf
Κωνσταντίνου Ελισάβετ,Εφαρμοσμένη κρυπτογραφία Ι(συνέχεια),
http://www.icsd.aegean.gr/website_files/metaptyxiako/997371597.pdf
Ο μαθηματικός όλων των μαθητών,2008-2009,
http://www.mathimatikos.com/EXERCISES/Lykeio/C/typologio/Pithanotites.pdf
Πολυωνυμικοί δακτύλιοι,Αριστοτέλειο πανεπιστήμιο Θεσσαλονίκης,
http://users.auth.gr/epsom/algdomes/polyring/main.htm
http://users.auth.gr/epsom/algdomes/rings/ideals.htm

Ξένηβιβλιογραφία

High school Mathematics Extensions/Discrete Probability,Wikepedia,

https://en.wikibooks.org/wiki/High_School_Mathematics_Extensions/Discrete_Probability
Dan Boneh,Cryptography I,Coursera,Stanford
https://www.coursera.org/course/crypto
Cryptography,Wikipedia,
https://en.wikipedia.org/wiki/Cryptography
Vigenere cipher,Wikipedia
https://en.wikipedia.org/wiki/Vigenère_cipher
Enigma machine,Wikipedia
https://en.wikipedia.org/wiki/Enigma_machine
Alan turing,Wikipedia
https://en.wikipedia.org/wiki/Alan_Turing
National Security Agency,Wikipedia
https://en.wikipedia.org/wiki/National_Security_Agency
Data Encryption Standard,Wikipedia,
https://www.coursera.org/course/crypto
https://el.wikipedia.org/wiki/Data_Encryption_Standard
National Institute of Standards and Technology,Wikipedia,
https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technolo
One time Pad,Wikipedia,
http://en.wikipedia.org/wiki/One-time_pad
Pseudorandom generator,Wikipedia,
http://en.wikipedia.org/wiki/Pseudorandom_generator
Random variable,Wikipedia,
https://en.wikipedia.org/wiki/Random_variable
Fiestel cipher diagram en,Wikipedia
https://en.wikipedia.org/wiki/Feistel_cipher#/media/File:Feistel_cipher_diagram_en.svg
Fiestel cipher,Wikipedia,
https://en.wikipedia.org/wiki/Feistel_cipher
Linear cryptanalysis,Wikipedia,
https://en.wikipedia.org/wiki/Linear_cryptanalysis
Block cipher mode of operation,Wikipedia,
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Fiestel network,securityblog.redhat.com
https://www.google.gr/search?q=feistel+network&rlz=1C1AVNE_enGR629GR629&es_sm=93&biw=1366&bih=623&source=lnms&tbm=isch&sa=X&ei=mxyYVdn7JYarsAGepIWwDg&ved=0CAYQ_AUoAQ#imgrc=vt4rwH_Pdg2d3M%3A
Margaret Rouse,Whati is one time pad,
http://searchsecurity.techtarget.com/definition/one-time-pad
Saffi Goldwasher-Mihir Bellare,Lectures on Cryptography, July 2008,
http://cseweb.ucsd.edu/~mihir/papers/gb.pdf
Ronald L.Rivest,Cryptography and security,
http://people.csail.mit.edu/rivest/crypto-security.html
Chris tensen,Double Des,
http://www.nku.edu/~christensen/3DES.pdf
Phant0m,25 January 2013,XOR-hex Strings,
http://stackoverflow.com/questions/14526231/python-xor-hex-strings
Python,Built in Functions,
https://docs.python.org/2/library/functions.html

Python,iterools,
https://docs.python.org/2/library/itertools.html
Pycrypto,API Docs,
https://www.dlitz.net/software/pycrypto/api/current/
gmpy2,Multiple-precision integers,
https://gmpy2.readthedocs.org/en/latest/mpz.html#mpz-methods
Victor Shoup,A computational Introduction to Number Theory and Algebra,2008,
http://shoup.net/ntb/ntb-v2.pdf
The decision Diffie-Hellman problem, D.Boneh ANTS-3, 1998
http://link.springer.com/chapter/10.1007%2FBFb0054851#page-1
Universal hash proofs and paradigms for chosen ciphertext secure public key encryption , R.Cramer and V.Shoup,Eurocrypt 2002
http://www.shoup.net/papers/
Chosen ciphertext security from Identity Based Encryption D.Boneh, R.Carnetti,S.Halevi, J.Katz, SICOMP 2007
https://people.csail.mit.edu/shaih/pubs.html
The twin Diffie Hellman problem and applications, D.Cash, E.Kilitz, V.Shoup, Eurocrypt 2008
http://link.springer.com/chapter/10.1007%2F978-3-540-78967-3_8#page-1
Efficent chosen ciphertet security via extractable has protocols, H.Wee ,Crypto 2010
http://link.springer.com/chapter/10.1007%2F978-3-642-14623-7_17#page-1