Α 44 – ΚΡΥΠΤΟΓΡΑΦΙΑ
ΣΗΜΕΙΩΣΕΙΣ #2

ΘΕΟΔΟΥΛΟΣ ΓΑΡΕΦΑΛΑΚΗΣ

1. Συμμετρικοι κωδικες

Οι συμμετρικοί κώδικες (symmetric ciphers) μπορουν αφηρημένα να περιγραφούν με τους παρακάτω μετασχηματισμους:

$$c = e_k(m) \quad \kappa\alpha\iota \quad m = d_k(c),$$

όπου

- m είναι το καθαρό μήνυμα,
- e είναι η συνάρτηση κρυπτογράφησης,
- d είναι η συνάρτηση αποκρυπτογράφησης,
- k είναι το κλείδι,
- c είναι το κρυπτογράφημα.

Το βασικό που πρέπει να παρατηρήσουμε είναι ότι τόσο η κρυπτογράφηση όσο και η αποκρυπτογράφηση χρησιμοποιούν το ίδιο κλειδί. Επίσεις, το μόνο κρυφό είναι το κλειδί k (και φυσικά το μύνημα m). Δηλαδή οι συναρτήσεις e και d θεωρούνται (και είναι) γνωστές.

Γενικά οι συμμετρικοί κώδικες χωρίζονται σε κώδικες ροής (stream ciphers) και τμηματικούς κώδικες (block ciphers).

2. Κωδικες ροης

Το χαρακτηριστικό τους είναι ότι οι χαρακτήρες του μυνήματος κρυπτογραφούνται ένας προς έναν. Το πιο χαρακτηριστικό ίσως παράδειγμα είναι το one-time pad. Για περισσότερες πληροφορίες μπορείτε να διαβάσετε το κεφάλειο 5.1.2 του βιβλίου του Smart ή το σχετικό κεφάλαιο του βιβλίου των Menezes, van Oorschot, Vanstone.

3. Τμηματικοι κωδικες

Είναι ο βασικός τύπος συμμετρικών κωδίκων που χρησιμοποιούνται σήμερα. Στηρίζονται στη γενική κατασκευή του H. Feistel. Το μήνυμα κόβεται σε τμήματα n χαρακτήρων τα οποία κρυπτογραφούνται χωριστά. Από εδώ και στο εξής υποθέτουμε ότι το μήνυμα είναι γραμμένο στο δυαδικό σύστημα. Δηλαδή κάθε τμήμα είναι n διαδοχικά δυαδικά ψηφία (bits).

Ένας Feistel cipher δουλεύει σε γύρους. Κάθε γύρος έχει σαν είσοδο ένα n-bit μπλοκ και ένα κλείδι γύρου. Το κλειδί γύρου προκύπτει από το βασικό κλειδί με ένα αλγόριθμο σχεδιασμού κειδιών. Το μπλοκ εισόδου χωρίζεται σε δύο ίσα μέρη: το αριστερό (L) και το δεξιό (R). Στο i-στο γύρο, η είσοδος είναι: το κλειδί k_i (που προέκυψε από τον αλγόριθμο σχεδιασμού κλειδιών), και τα L_{i-1} και R_{i-1} που προέκυψαν από τον προηγούμενο γύρο. Η έξοδος του γύρου είναι:

$$L_i = R_{i-1} \quad \textit{και} \quad R_i = L_{i-1} \oplus F(k_i, R_{i-1}). \tag{1}$$

Εδώ το σύμβολο $\oplus$ είναι πρόσθεση modulo 2 κατά συντεταγμένες (τα L_i και R_i μπορεί κανείς να τα δεί σαν διανύσματα πάνω στο $\mathbb{F}_2$). Επίσεις η F είναι μια απεικόνιση

$$F : K \times \{0,1\}^{n/2} \longrightarrow \{0,1\}^{n/2}.$$

Με τον τρόπο αυτό δημιουργούμε δύο ακολουθίες L_i και R_i. Οι αρχικές τιμές της ακολουθίας L_0 και R_0 ορίζονται από το καθαρό μήνυμα. Αν ο κώδικας μας δουλεύει σε r γύρους, τότε το τελικό κρυπτογράφημα είναι L_r, R_r.

Πως αποκρυπτογραφούμε; Θυμηθείτε ότι ο νόμιμος παραλήπτης ξέρει το κλειδί k και ο αλγόριθμος σχεδιασμού κελιδιών είναι γνωστός σε όλους. Άρα μπορεί να υπολογίσει τα κλειδιά γύρου k_i, $i = 1, \ldots, r$. Από τις Εξισώσεις (1) βλέπουμε ότι

$$R_{i-1} = L_i \quad \textit{και} \quad L_{i-1} = R_i \ominus F(k_i, L_i).$$

Αρχίζοντας λοιπόν από τα L_r και R_r και χρησιμοποιώντας τα κλειδιά γύρου με αντίστροφη σειρά υπολογίζουμε κατά σειρά τα ζεύγη $(L_r, R_r), (L_{r-1}, R_{r-1}), \ldots, (L_0, R_0)$ και αποκαλύπτουμε το μήνυμα.

Μερικοί από τους πιο σημαντικούς τμηματικούς κώδικες είναι οι: DES, MARS, RC5, RC6, Twofish, Serpent, Rijndael.

Για παράδειγμα στο DES το n (μέγεθος του μπλοκ) είναι 64. Το μέγεθος του βασικού κλειδιού ειναι 56 bits. Δουλεύει σε 16 γύρους (δηλαδή $r = 16$). Κάθε κλειδί γύρου έχει μέγεθος 48 bits. Η απεικόνιση F περιγράφεται στο βιβλίο του Smart καθώς και στο βιβλίο του Stinson.

4. Θεση λειτουργιας

Μέχρι τώρα είδαμε πως λειτουργει ένας τμηματικός κώδικας. Κατασκευάσαμε ένα αλγόριθμο (κουτί) E και ένα αλγόριθμο D που μπορούν να κρυπτογραφήσουν ένα μήνυμα μεγέθους n χαρακτήρων. Τυπικά το n είναι σχετικά μικρός αριθμός: 64 για το DES, 128 για πιο σύγχρονους κώδικες. Πώς μπορούμε να κρυπτογραφήσουμε ένα μήνυμα οποιουδήποτε μήκους; Είναι προφανές ότι θα κόψουμε το μήνυμα σε τμήματα μήκους n. Αν το αρχικό μήκος του μηνύματος δεν είναι πολλαπλάσιο του n, απλα προσθέτουμε τους απαραίτητους χαρακτήρες στο τέλος του μηνυματος (κενά για παράδειγμα). Έχουμε τώρα να κρυπτογραφήσουμε τα μπλοκ $m_1, m_2, \ldots, m_l$. Το πως ακριβώς χρησιμοποιούμε τον E για να κρυπτογραφήσουμε τα m_i, $i = 1, \ldots, l$ ονομάζεται θέση λειτουργίας του κώδικα (mode of operation).

4.1. **ECB mode.** Είναι το απλούστερο mode που έχει και τις περισσότερες αδυναμίες. Η ακολουθία των κρυπτογραμμάτων δίνεται ως εξής:

$$c_i = E_k(m_i), \quad i = 1, \ldots, l.$$

Φανερά η αποκρυπτογράφηση γίνεται ως εξής:

$$m_i = D_k(m_i), \quad i = 1, \ldots, l.$$

Τα προβλήματα προκύπτουν από την ιδιότητα: Αν $m_i = m_j$ για κάποια $i \neq j$ τότε $c_i = c_j$. Αυτό δημιουργεί ιδιαίτερα προβλήματα αν ο επιτηθέμενος δεν παρακολουθεί απλά παθητικά, αλλά παρεμβαίνει στην επικοινωνία.

4.2. **CBC mode.** Όπως και πριν έχουμε να κρυπτογραφήσουμε τα $m_1, m_2, \ldots, m_l$. Αυτή τη φορά χρησιμοποιουμε ένα αρχικό διάνυσμα IV που δεν είναι μυστικό, και κάνουμε τα εξής:

$$c_1 = E_k(m_1 \oplus IV), \quad c_i = E_k(m_i \oplus c_{i-1}), \quad i > 1.$$

Η αποκρυπτογράφηση είναι και πάλι δυνατή:

$$m_1 = D_k(c_1) \ominus IV, \quad m_i = D_k(c_i) \ominus c_{i-1}, \quad i > 1.$$

Αυτό που έχουμε επιτύχει είναι ότι δίνουμε context (συμφραζόμενα;) στην κρυπτογράφηση. Δηλαδή το κρυπτογράφημα ενός μπλοκ είναι διαφορετικό ανάλογα με το δείκτη του, τη θέση του δηλαδή στην ακολουθία των 'καθαρών' μπλοκ. Μεταβολή στο m_i δημιουργεί μεταβολή στο c_i (αυτό ισχύει και στο ECB mode) αλλά και στο c_{i+1} (αυτό **δεν** ισχύει στο ECB mode).

5. Κωδικες πιστοποιησης ακεραιοτητας

Είναι η πρώτη εφαρμογή που βλέπουμε, όπου στοχος μας δεν είναι η διατήρηση της μυστικότητας, αλλά η πιστοποίηση της ακεραιότητας (integrity) του μηνύματος. Θέλουμε να στείλουμε το μήνυμα M και σκοπός μας δεν είναι να το κρατήσουμε μυστικό, αλλά ο παραλήπτης να είναι σε θέση να καταλάβει αν έχει γίνει κάποια αλλαγή στο μήνυμα.

Η στρατηγική μας είναι να βρούμε μια απεικόνιση f που παραμετροποιείται από ένα κλειδί k, και να υπολογίσουμε την τιμή $f_k(M)$. Στέλνουμε το ζεύγος $M, f_k(M)$.

Για να κατασκευάσουμε την f θα χρησιμοποιήσουμε ένα block cipher σε CBC mode. Θα υπολογίσουμε δηλαδή το $e_k(M)$ και θα στείλουμε το $(M, e_k(M))$.

Μια παρατήρηση: η τιμή $e_k(M)$ χρησιμοποιούνται μόνο για να πιστοποιήσουμε την ακεραιότητα του M. Άρα αντί της τιμής $e_k(M)$ (που έχει μέγεθος όσο και το μήνυμα) θα μπορούσαμε να είχαμε στείλει ένα μέρος της τιμής μόνο, ας πούμε τα μισά πρώτα bits.