

Κρυπτογραφία

Ψευδοτυχασιότητα - Κρυπτοσυστήματα ροής

Άρης Παγουρτζής - Πέτρος Ποτίκας

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Περιεχόμενα

- 1 Εισαγωγή
- 2 Ψευδοτυχαίες γεννήτριες, συναρτήσεις, μεταθέσεις
- 3 Blum-Blum-Shub
- 4 Κρυπτοσυστήματα ροής (stream ciphers)
- 5 RC4
- 6 Linear Recurrence Keystream
- 7 Πρακτικά κρυπτοσυστήματα ροής με LFSRs
- 8 Σύγχρονα κρυπτοσυστήματα ροής

- ▶ Τυχαίοι αριθμοί αποτελούν σημαντικό στοιχείο της επιστήμης των υπολογιστών αλλά και της κρυπτογραφίας

- ▶ Τυχαίοι αριθμοί αποτελούν σημαντικό στοιχείο της επιστήμης των υπολογιστών αλλά και της κρυπτογραφίας
- ▶ Αλγόριθμοι και πρωτόκολλα που τους χρησιμοποιούν:
 - Κατανομή κλειδιών, σχήματα ταυτοποίησης χρηστών
 - Ακεραιότητα μηνύματος (MAC)
 - Παραγωγή κλειδιών συνεδρίας (session keys)
 - Παραγωγή ροής από bit για συμμετρική κρυπτογράφηση (**stream ciphers**)

Ψευδοτυχαίες γεννήτριες

- ▶ Επιτρέπουν ένα μικρό τυχαίο κλειδί (seed) να δώσει ένα μεγάλο “ψευδοτυχαίο”, αρκετά τυχαίο για έναν πολυωνυμικά φραγμένο αντίπαλο string.
- ▶ Το ψευδοτυχαίο κλειδί μπορεί να χρησιμοποιηθεί σαν κλειδί για το one-time pad (πράξη XOR).
- ▶ Παρεμφερής χρήση: σε κρυπτοσυστήματα ροής.
- ▶ Η ύπαρξη ψευδοτυχαίων γεννητριών σχετίζεται με την ύπαρξη μονόδρομων συναρτήσεων (one-way functions).
- ▶ RC4 (Rivest '87): μια σημαντική γεννήτρια / κρυπτοσύστημα ροής.

Ψευδοτυχαίες γεννήτριες

- ▶ Ιδέα: κάτι που “μοιάζει” με τυχαίο, αλλά δεν είναι πραγματικά
- ▶ Δε ξεχωρίζει ένα τυχαίο string από ένα που δημιουργείται από τη γεννήτρια ψευδοτυχειότητας
- ▶ Εφαρμογή ψευδοτυχειότητας και αλλού όπως π.χ. παίγνια, δειγματοληψία
- ▶ Την χρησιμοποιούμε είτε για την παραγωγή κλειδιών σε σχήματα συμμετρικής/ασύμμετρης κρυπτογράφησης είτε σε κρυπτογράφηση ροής

Ψευδοτυχαίες γεννήτριες

Ποιο είναι τυχαίο;

00101010100101010110

01010101010101010101

Ψευδοτυχαίες γεννήτριες

Ποιο είναι τυχαίο;

00101010100101010110

01010101010101010101

- ▶ Κατανομή πάνω σε strings:

$$D: \{0, 1\}^n \rightarrow [0, 1], \text{ ώστε } \sum_x D(x) = 1$$

Ψευδοτυχαίες γεννήτριες

Ποιο είναι τυχαίο;

00101010100101010110

01010101010101010101

- ▶ Κατανομή πάνω σε strings:

$$D: \{0, 1\}^n \rightarrow [0, 1], \text{ ώστε } \sum_x D(x) = 1$$

- ▶ Ορισμός ψευδοτυχειότητας μέσω στατιστικών τεστ: Μια κατανομή D πάνω σε n -bit strings είναι ψευδοτυχαία αν ικανοποιεί κάποια τεστ (NIST SP 800-22)

1. $\Pr_{x \leftarrow D}[\text{1ο bit του } x = 1] \simeq 1/2$
2. $\Pr_{x \leftarrow D}[\text{parity του } x = 1] \simeq 1/2$
3. $\Pr_{x \leftarrow D}[\#1 = \#0 \text{ in } x] \simeq 1/2$
4. ...

Ψευδοτυχαίες γεννήτριες

Ποιο είναι τυχαίο;

00101010100101010110

01010101010101010101

- ▶ Κατανομή πάνω σε strings:

$$D: \{0, 1\}^n \rightarrow [0, 1], \text{ ώστε } \sum_x D(x) = 1$$

- ▶ Ορισμός ψευδοτυχειότητας μέσω στατιστικών τεστ: Μια κατανομή D πάνω σε n -bit strings είναι ψευδοτυχαία αν ικανοποιεί κάποια τεστ (NIST SP 800-22)

1. $\Pr_{x \leftarrow D}[\text{1ο bit του } x = 1] \simeq 1/2$
2. $\Pr_{x \leftarrow D}[\text{parity του } x = 1] \simeq 1/2$
3. $\Pr_{x \leftarrow D}[\#1 = \#0 \text{ in } x] \simeq 1/2$
4. ...

- ▶ Όμως με αντίπαλο, δε γνωρίζουμε τα τεστ που έχει

Ψευδοτυχαίες γεννήτριες

Ποιο είναι τυχαίο;

00101010100101010110

01010101010101010101

- ▶ Κατανομή πάνω σε strings:

$D: \{0, 1\}^n \rightarrow [0, 1]$, ώστε $\sum_x D(x) = 1$

- ▶ Ορισμός ψευδοτυχειότητας μέσω στατιστικών τεστ: Μια κατανομή D πάνω σε n -bit strings είναι ψευδοτυχαία αν ικανοποιεί κάποια τεστ (NIST SP 800-22)

1. $\Pr_{x \leftarrow D}[\text{1ο bit του } x = 1] \simeq 1/2$
2. $\Pr_{x \leftarrow D}[\text{parity του } x = 1] \simeq 1/2$
3. $\Pr_{x \leftarrow D}[\#1 = \#0 \text{ in } x] \simeq 1/2$
4. ...

- ▶ Όμως με αντίπαλο, δε γνωρίζουμε τα τεστ που έχει
- ▶ Κρυπτογραφικά, η κατανομή D είναι ψευδοτυχαία, αν περνάει όλα τα αποδοτικά στατιστικά τεστ

Ορισμός

Μια **γεννήτρια ψευδοτυχειότητας (PRG)** είναι ένας αποδοτικός, ντετερμινιστικός αλγόριθμος που επεκτείνει ένα *μικρό, τυχαία επιλεγμένο σπόρο* σε μια μεγαλύτερη, ψευδοτυχαία έξοδο.

- ▶ Από λίγα πραγματικά τυχαία bits, παράγονται πολλά περισσότερα bits που “φαίνονται” τυχαία
- ▶ Παραγωγή πραγματικά τυχαίων bits είναι δύσκολη και χρονοβόρα
- ▶ Μέριμνα για τον σπόρο

Pseudorandom Generators (PRG)

Παρατήρηση: ασυμπτωτικά μιλάμε για $\text{Dist} = \{\text{Dist}_n\}$, όπου n η παράμετρος ασφαλείας.

Pseudorandom Generators (PRG)

Παρατήρηση: ασυμπτωτικά μιλάμε για $\text{Dist} = \{\text{Dist}_n\}$, όπου n η παράμετρος ασφαλείας.

Όπως η σημασιολογική (υπολογιστική) ασφάλεια είναι η υπολογιστική χαλάρωση της τέλει μυστικότητας έτσι και η ψευδοτυχειότητα είναι η υπολογιστική χαλάρωση της πραγματικής τυχειότητας.

Τυπικός ορισμός ψευδοτυχαίας κατανομής

Έστω συνάρτηση $G : \{0, 1\}^n \mapsto \{0, 1\}^l$.

Τυπικός ορισμός ψευδοτυχαίας κατανομής

Έστω συνάρτηση $G : \{0, 1\}^n \mapsto \{0, 1\}^l$.

Ορίζουμε Dist να είναι κατανομή σε l -bit strings που προκύπτει επιλέγοντας ομοιόμορφα τυχαία ένα $s \in \{0, 1\}^n$ δίνει έξοδο $G(s)$.

Τυπικός ορισμός ψευδοτυχαίας κατανομής

Έστω συνάρτηση $G : \{0, 1\}^n \mapsto \{0, 1\}^l$.

Ορίζουμε Dist να είναι κατανομή σε l -bit strings που προκύπτει επιλέγοντας ομοιόμορφα τυχαία ένα $s \in \{0, 1\}^n$ δίνει έξοδο $G(s)$.

Η G είναι ψευδοτυχαία αν η Dist είναι ψευδοτυχαία.

Πείραμα

- ▶ Θεωρούμε πως έχουμε έναν πολυωνυμικά φραγμένο αντίπαλο, ο οποίος λαμβάνει strings μήκους l .

Πείραμα

- ▶ Θεωρούμε πως έχουμε έναν πολυωνυμικά φραγμένο αντίπαλο, ο οποίος λαμβάνει strings μήκους l .
- ▶ Θέλουμε ο αντίπαλος να μην καταλαβαίνει αν παίρνουμε δείγμα από την κατανομή Dist ή αν παίρνουμε ομοιόμορφα τυχαία l -bit string

Πείραμα

- ▶ Θεωρούμε πως έχουμε έναν πολυωνυμικά φραγμένο αντίπαλο, ο οποίος λαμβάνει strings μήκους l .
- ▶ Θέλουμε ο αντίπαλος να μην καταλαβαίνει αν παίρνουμε δείγμα από την κατανομή Dist ή αν παίρνουμε ομοιόμορφα τυχαία l -bit string
- ▶ Θέλουμε ο αντίπαλος να μην καταλαβαίνει αν αυτά προήλθαν από την $G(s)$ (με ομοιόμορφα τυχαία επιλεγμένο s) ή αν αυτά προήλθαν ομοιόμορφα τυχαία από το $\{0, 1\}^l$ (δηλ. είναι πραγματικά τυχαία string μήκους l)

Pseudorandom Generators

Ορισμός

Έστω l πολυώνυμο, G ένας ντετερμινιστικός αλγόριθμος πολυωνυμικού χρόνου, τ.ώ. για κάθε n και είσοδο $s \in \{0, 1\}^n$ το αποτέλεσμα $G(s)$ είναι μήκους $l(n)$. Ο G είναι *ψευδοτυχαίος γεννήτορας (PRG)* αν:

1. Για κάθε n , $l(n) > n$
2. Για κάθε πιθανοτικό πολυωνυμικού χρόνου αλγόριθμο (PPT) D , υπάρχει μια αμελητέα¹ συνάρτηση $negl$, ώστε

$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq negl(n)$$

όπου η πρώτη πιθανότητα είναι από την ομοιόμορφη επιλογή του $s \in \{0, 1\}^n$ και την τυχειότητα του D , ενώ η δεύτερη από την ομοιόμορφη επιλογή του $r \in \{0, 1\}^{l(n)}$ και την τυχειότητα του D .

¹αμελητέα συνάρτηση f : για κάθε πολυώνυμο p , υπάρχει μια σταθερά N , τ.ώ. για κάθε $n > N$ ισχύει $f(n) < 1/p(n)$

Pseudorandom Generators

Παράδειγμα

Δίνεται ο $G(s) = s || \bigoplus_{i=1}^n s_i$

Είναι PRG;

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)
- ▶ Είναι τυχαία η κατανομή; Όχι τελείως!

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)
- ▶ Είναι τυχαία η κατανομή; Όχι τελείως!
- ▶ αν $l(n) = n + 1$, τότε στην ομοιόμορφη κατανομή στο $\{0, 1\}^{n+1}$ κάθε συμβολοσειρά έχει ακριβώς $1/2^{n+1}$ πιθανότητα να επιλεγεί

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)
- ▶ Είναι τυχαία η κατανομή; Όχι τελείως!
- ▶ αν $l(n) = n + 1$, τότε στην ομοιόμορφη κατανομή στο $\{0, 1\}^{n+1}$ κάθε συμβολοσειρά έχει ακριβώς $1/2^{n+1}$ πιθανότητα να επιλεγεί
- ▶ αν $|dom(G)| = 2^n$, $|range(G)| = 2^{n+1}$, τότε η πιθανότητα μια συμβολοσειρά μήκους $n + 1$ να εμφανιστεί στην έξοδο της G είναι τουλάχιστον $1/2^n$ για τις μισές το πολύ συμβολοσειρές και 0 για τις υπόλοιπες

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)
- ▶ Είναι τυχαία η κατανομή; Όχι τελείως!
- ▶ αν $l(n) = n + 1$, τότε στην ομοιόμορφη κατανομή στο $\{0, 1\}^{n+1}$ κάθε συμβολοσειρά έχει ακριβώς $1/2^{n+1}$ πιθανότητα να επιλεγεί
- ▶ αν $|dom(G)| = 2^n$, $|range(G)| = 2^{n+1}$, τότε η πιθανότητα μια συμβολοσειρά μήκους $n + 1$ να εμφανιστεί στην έξοδο της G είναι τουλάχιστον $1/2^n$ για τις μισές το πολύ συμβολοσειρές και 0 για τις υπόλοιπες
- ▶ Αν ο διαχωριστής είναι εκθετικού χρόνου, τότε με εξαντλητική αναζήτηση μπορεί να ξεχωρίσει την κατανομή D από την ομοιόμορφη

Pseudorandom Generators

Παρατηρήσεις:

- ▶ Ο αλγόριθμος είναι ντετερμινιστικός και αποδοτικός (πολυωνυμικός)
- ▶ Είναι τυχαία η κατανομή; Όχι τελείως!
- ▶ αν $l(n) = n + 1$, τότε στην ομοιόμορφη κατανομή στο $\{0, 1\}^{n+1}$ κάθε συμβολοσειρά έχει ακριβώς $1/2^{n+1}$ πιθανότητα να επιλεγεί
- ▶ αν $|dom(G)| = 2^n$, $|range(G)| = 2^{n+1}$, τότε η πιθανότητα μια συμβολοσειρά μήκους $n + 1$ να εμφανιστεί στην έξοδο της G είναι τουλάχιστον $1/2^n$ για τις μισές το πολύ συμβολοσειρές και 0 για τις υπόλοιπες
- ▶ Αν ο διαχωριστής είναι εκθετικού χρόνου, τότε με εξαντλητική αναζήτηση μπορεί να ξεχωρίσει την κατανομή D από την ομοιόμορφη
- ▶ Ο σπόρος πρέπει να μείνει μυστικός και αρκετά μεγάλος, ώστε να μη γίνεται επίθεση με εξαντλητική αναζήτηση

- ▶ Υπάρχουν γεννήτριες ψευδοτυχειότητας;

- ▶ Υπάρχουν γεννήτριες ψευδοτυχειότητας; Άγνωστο, χωρίς κάποια υπόθεση.

- ▶ Υπάρχουν γεννήτριες ψευδοτυχαιότητας; Άγνωστο, χωρίς κάποια υπόθεση.
- ▶ Μπορούν να κατασκευαστούν με την υπόθεση ότι υπάρχουν μονόδρομες συναρτήσεις (one-way functions).

- ▶ Υπάρχουν γεννήτριες ψευδοτυχειότητας; Άγνωστο, χωρίς κάποια υπόθεση.
- ▶ Μπορούν να κατασκευαστούν με την υπόθεση ότι υπάρχουν μονόδρομες συναρτήσεις (one-way functions).
- ▶ Υπάρχουν, με την υπόθεση ότι το πρόβλημα της παραγοντοποίησης μεγάλων αριθμών είναι δύσκολο.

- ▶ Υπάρχουν γεννήτριες ψευδοτυχειότητας; Άγνωστο, χωρίς κάποια υπόθεση.
- ▶ Μπορούν να κατασκευαστούν με την υπόθεση ότι υπάρχουν μονόδρομες συναρτήσεις (one-way functions).
- ▶ Υπάρχουν, με την υπόθεση ότι το πρόβλημα της παραγοντοποίησης μεγάλων αριθμών είναι δύσκολο.
- ▶ Υποψήφιος: stream ciphers, block ciphers (OFB, CFB, CTR mode)

Κρυπτοσυστήματα Ροής

- ▶ Συνήθως θέλουμε ροή από ψευδοτυχαία bit
- ▶ Δύο αλγόριθμοι: *Init* που παίρνει είσοδο ένα seed και ίσως ένα IV και επιστρέφει μια state st_0 και *GetBit* που παίρνει μια state st_i και βγάζει ένα y και ένα νέο state st_{i+1} .
- ▶ Ο δεύτερος αλγόριθμος τρέχει όσο χρειαζόμαστε.
++++
- ▶ Ισχύει: G γεννήτρια ψευδοτυχαιότητας αν G μη προβλέψιμη

Ορισμός

(Προβλέψιμη) Υπάρχει πολυωνυμικός αλγόριθμος A τέτοιος ώστε:

$$\Pr[A(G(K)_{1..i}) = G(K)_{i+1}] > \frac{1}{2} + \epsilon$$

για μη αμελητέο ϵ

Κρυπτοσυστήματα Ροής

- ▶ Συνήθως θέλουμε ροή από ψευδοτυχαία bit
- ▶ Δύο αλγόριθμοι: *Init* που παίρνει είσοδο ένα seed και ίσως ένα IV και επιστρέφει μια state st_0 και *GetBit* που παίρνει μια state st_i και βγάζει ένα y και ένα νέο state st_{i+1} .
- ▶ Ο δεύτερος αλγόριθμος τρέχει όσο χρειαζόμαστε.
++++
- ▶ Ισχύει: G γεννήτρια ψευδοτυχειότητας αν G μη προβλέψιμη

Ορισμός

(Προβλέψιμη) Υπάρχει πολυωνυμικός αλγόριθμος A τέτοιος ώστε:

$$\Pr[A(G(K)_{1..i}) = G(K)_{i+1}] > \frac{1}{2} + \epsilon$$

για μη αμελητέο ϵ

Επιπλέον, θα πρέπει να έχουμε και προς τα πίσω μη προβλεψιμότητα: οι τιμές που έχουν εμφανιστεί δεν αποκαλύπτουν το σπόρο.

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση
- ▶ Τυχαία συνάρτηση: $Func_n =$ όλες οι συναρτήσεις από το $\{0, 1\}^n$ στο $\{0, 1\}^n$

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση
- ▶ Τυχαία συνάρτηση: $Func_n =$ όλες οι συναρτήσεις από το $\{0, 1\}^n$ στο $\{0, 1\}^n$
- ▶ Πόσες;

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση
- ▶ Τυχαία συνάρτηση: $Func_n =$ όλες οι συναρτήσεις από το $\{0, 1\}^n$ στο $\{0, 1\}^n$
- ▶ Πόσες; Μπορούμε να αναπαραστήσουμε μια συνάρτηση στο $Func_n$ με $n2^n$ bits

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση
- ▶ Τυχαία συνάρτηση: $Func_n =$ όλες οι συναρτήσεις από το $\{0, 1\}^n$ στο $\{0, 1\}^n$
- ▶ Πόσες; Μπορούμε να αναπαραστήσουμε μια συνάρτηση στο $Func_n$ με $n2^n$ bits
- ▶ Άρα, $|Func_n| = 2^{n2^n}$
- ▶ Τυχαία συνάρτηση: διάλεξε ομοιόμορφα μια $f \in Func_n$

Pseudorandom Functions - PRF

- ▶ Συνάρτηση που φαίνεται ίδια με μια τυχαία συνάρτηση
- ▶ Τυχαία συνάρτηση: $Func_n =$ όλες οι συναρτήσεις από το $\{0, 1\}^n$ στο $\{0, 1\}^n$
- ▶ Πόσες; Μπορούμε να αναπαραστήσουμε μια συνάρτηση στο $Func_n$ με $n2^n$ bits
- ▶ Άρα, $|Func_n| = 2^{n2^n}$
- ▶ Τυχαία συνάρτηση: διάλεξε ομοιόμορφα μια $f \in Func_n$
- ▶ Ισοδύναμα: σε κάθε θέση του πίνακα τιμών διάλεξε ομοιόμορφα ένα string από το $\{0, 1\}^n$

Pseudorandom Functions - PRF

- ▶ Δεν έχει νόημα να μιλάμε για σταθερή συνάρτηση, αλλά θέλουμε κάποια κατανομή

Pseudorandom Functions - PRF

- ▶ Δεν έχει νόημα να μιλάμε για σταθερή συνάρτηση, αλλά θέλουμε κάποια κατανομή
- ▶ Αν έχουμε μια $F: \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^*$, τότε αν κρατήσουμε σταθερή την πρώτη παράμετρο έχουμε συναρτήσεις $F_k(x) = F(k, x)$, όπου k κλειδί (επιλέγεται ομοιόμορφα)

Pseudorandom Functions - PRF

- ▶ Δεν έχει νόημα να μιλάμε για σταθερή συνάρτηση, αλλά θέλουμε κάποια κατανομή
- ▶ Αν έχουμε μια $F: \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^*$, τότε αν κρατήσουμε σταθερή την πρώτη παράμετρο έχουμε συναρτήσεις $F_k(x) = F(k, x)$, όπου k κλειδί (επιλέγεται ομοιόμορφα)
- ▶ Επιλέγοντας το κλειδί $k \leftarrow \{0, 1\}^n$ επιλέγεται μια $F_k: \{0, 1\}^n \rightarrow \{0, 1\}^n$

Pseudorandom Functions - PRF

- ▶ Δεν έχει νόημα να μιλάμε για σταθερή συνάρτηση, αλλά θέλουμε κάποια κατανομή
- ▶ Αν έχουμε μια $F: \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^*$, τότε αν κρατήσουμε σταθερή την πρώτη παράμετρο έχουμε συναρτήσεις $F_k(x) = F(k, x)$, όπου k κλειδί (επιλέγεται ομοιόμορφα)
- ▶ Επιλέγοντας το κλειδί $k \leftarrow \{0, 1\}^n$ επιλέγεται μια $F_k: \{0, 1\}^n \rightarrow \{0, 1\}^n$
- ▶ Άρα η F με κλειδί, ορίζει μια κατανομή στις συναρτήσεις της $Func_n$

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή
- ▶ Έχουμε ένα μαντείο O που είτε είναι ίσο με F_k (για ομοιόμορφο k) ή με f (για ομοιόμορφη f)

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή
- ▶ Έχουμε ένα μαντείο O που είτε είναι ίσο με F_k (για ομοιόμορφο k) ή με f (για ομοιόμορφη f)
- ▶ Μπορούμε να ρωτήσουμε για όποιο x θέλουμε, αλλά ίδια απάντηση για το ίδιο x .

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή
- ▶ Έχουμε ένα μαντείο O που είτε είναι ίσο με F_k (για ομοιόμορφο k) ή με f (για ομοιόμορφη f)
- ▶ Μπορούμε να ρωτήσουμε για όποιο x θέλουμε, αλλά ίδια απάντηση για το ίδιο x .
- ▶ Μόνο πολυωνυμικά πολλές ερωτήσεις γίνονται στο μαντείο.

Τυπικός ορισμός PRF

- ▶ Η αναπαράσταση με $n2^n$ bits είναι αδύνατο να ελεγχθεί από έναν πολυωνυμικό διαχωριστή
- ▶ Έχουμε ένα μαντείο O που είτε είναι ίσο με F_k (για ομοιόμορφο k) ή με f (για ομοιόμορφη f)
- ▶ Μπορούμε να ρωτήσουμε για όποιο x θέλουμε, αλλά ίδια απάντηση για το ίδιο x .
- ▶ Μόνο πολυωνυμικά πολλές ερωτήσεις γίνονται στο μαντείο. Οι ερωτήσεις προσαρμόζονται.

Ψευδοτυχαία συνάρτηση - Ορισμός

Ορισμός

Έστω συνάρτηση $F : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}^n$ αποδοτικά υπολογίσιμη. Η F είναι *ψευδοτυχαία συνάρτηση* αν για κάθε πολυωνυμικού χρόνου διαχωριστή D υπάρχει αμελητέα συνάρτηση $negl$ ώστε:

$$|Pr_{k \leftarrow \{0,1\}^n}[D^{F_k}() = 1] - Pr_{f \leftarrow Func_n}[D^f() = 1]| \leq negl(n)$$

Ψευδοτυχαία συνάρτηση - Ορισμός

Ορισμός

Έστω συνάρτηση $F : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}^n$ αποδοτικά υπολογίσιμη. Η F είναι *ψευδοτυχαία συνάρτηση* αν για κάθε πολυωνυμικού χρόνου διαχωριστή D υπάρχει αμελητέα συνάρτηση $negl$ ώστε:

$$|Pr_{k \leftarrow \{0,1\}^n}[D^{F_k^{()}}(1^n) = 1] - Pr_{f \leftarrow Func_n}[D^{f^{()}}(1^n) = 1]| \leq negl(n)$$

Σημείωση: Αν δοθεί το κλειδί, παύει να είναι PRF.

Ψευδοτυχαία συνάρτηση - Ορισμός

Ορισμός

Έστω συνάρτηση $F : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}^n$ αποδοτικά υπολογίσιμη. Η F είναι *ψευδοτυχαία συνάρτηση* αν για κάθε πολυωνυμικού χρόνου διαχωριστή D υπάρχει αμελητέα συνάρτηση $negl$ ώστε:

$$|Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] - Pr_{f \leftarrow Func_n}[D^{f(\cdot)}(1^n) = 1]| \leq negl(n)$$

Σημείωση: Αν δοθεί το κλειδί, παύει να είναι PRF.

Παράδειγμα

$F(k, x) = k \oplus x$. Είναι ψευδοτυχαία συνάρτηση;

Ψευδοτυχαία μετάθεση (Pseudorandom permutation)

Υπάρχει και η έννοια της ψευδοτυχαίας μετάθεσης, δηλ. συνάρτηση που είναι 1-1 και επί (άρα έχει και αντίστροφη)

Ψευδοτυχαία μετάθεση (Pseudorandom permutation)

Υπάρχει και η έννοια της ψευδοτυχαίας μετάθεσης, δηλ. συνάρτηση που είναι 1-1 και επί (άρα έχει και αντίστροφη)

Ο υπολογισμός της αντίστροφης πρέπει να γίνεται αποδοτικά. Όμως έχουμε oracle και για την αντίστροφη, οπότε ο ορισμός της ασφάλειας πρέπει να αλλάξει (strong pseudorandom permutation)

PRF vs PRG

Από PRF σε PRG:

PRF vs PRG

Από PRF σε PRG:

Από μια ψευδοτυχαία συνάρτηση μπορούμε να πάρουμε μια ψευδοτυχαία γεννήτρια: $G(k) = F_k(0) || F_k(1) || \dots$

PRF vs PRG

Από PRF σε PRG:

Από μια ψευδοτυχαία συνάρτηση μπορούμε να πάρουμε μια ψευδοτυχαία γεννήτρια: $G(k) = F_k(0) || F_k(1) || \dots$

Αντίστροφα, από μια PRG μπορούμε να πάρουμε μια PRF:

PRF vs PRG

Από PRF σε PRG:

Από μια ψευδοτυχαία συνάρτηση μπορούμε να πάρουμε μια ψευδοτυχαία γεννήτρια: $G(k) = F_k(0) || F_k(1) || \dots$

Αντίστροφα, από μια PRG μπορούμε να πάρουμε μια PRF:

Έστω PRG G με παράγοντα επέκτασης $n2^{t(n)}$, τότε ορίζεται μια συνάρτηση $f: \{0, 1\}^n \times \{0, 1\}^{t(n)} \mapsto \{0, 1\}^n$

PRF vs PRG

Από PRF σε PRG:

Από μια ψευδοτυχαία συνάρτηση μπορούμε να πάρουμε μια ψευδοτυχαία γεννήτρια: $G(k) = F_k(0) || F_k(1) || \dots$

Αντίστροφα, από μια PRG μπορούμε να πάρουμε μια PRF:

Έστω PRG G με παράγοντα επέκτασης $n2^{t(n)}$, τότε ορίζεται μια συνάρτηση $f: \{0, 1\}^n \times \{0, 1\}^{t(n)} \mapsto \{0, 1\}^n$

Για να υπολογίσουμε το $F_k(i)$, υπολογίζουμε το $G(k)$ και ερμηνεύουμε το αποτέλεσμα σαν look-up table με $2^{t(n)}$ γραμμές, όπου κάθε γραμμή έχει ένα n -bit string

PRF vs PRG

Από PRF σε PRG:

Από μια ψευδοτυχαία συνάρτηση μπορούμε να πάρουμε μια ψευδοτυχαία γεννήτρια: $G(k) = F_k(0) || F_k(1) || \dots$

Αντίστροφα, από μια PRG μπορούμε να πάρουμε μια PRF:

Έστω PRG G με παράγοντα επέκτασης $n2^{t(n)}$, τότε ορίζεται μια συνάρτηση $f: \{0, 1\}^n \times \{0, 1\}^{t(n)} \mapsto \{0, 1\}^n$

Για να υπολογίσουμε το $F_k(i)$, υπολογίζουμε το $G(k)$ και ερμηνεύουμε το αποτέλεσμα σαν look-up table με $2^{t(n)}$ γραμμές, όπου κάθε γραμμή έχει ένα n -bit string

Βελτίωση: Δέντρο $G(k) = G_0(k) || G_1(k)$ και $F_k(011) = G_0(G_1(G_1(k)))$

Σχήμα?

Δημιουργία πραγματικής τυχαιότητας

- ▶ υλικό, φυσικά φαινόμενα π.χ. θερμικός ή ηλεκτρικός θόρυβος
- ▶ λογισμικό π.χ. πάτημα πλήκτρων πληκτρολογίου, κίνηση του ποντικιού

Γεννήτριες τυχαίων αριθμών γενικού σκοπού είναι μη κατάλληλες για την κρυπτογραφία π.χ. `rand()` της C.

Intel, random.org ...

‘Αποδεδειγμένα’ ασφαλείς γεννήτριες ψευδοτυχαίων

- ▶ RSA-based (Micali-Schnorr), BBS.
- ▶ Βασίζονται σε (γενικά παραδεκτές) αριθμοθεωρητικές μονόδρομες συναρτήσεις: ύψωση σε δύναμη modulo n , τετραγωνισμός modulo n .
- ▶ Λειτουργία: διαδοχικές εφαρμογές της συνάρτησης, έξοδος κάθε φορά το λιγότερο σημαντικό bit του αριθμού (ή κάποια από τα λιγότερο σημαντικά bit).
- ▶ Είναι ασφαλείς κάτω από την υπόθεση δυσκολίας αντιστροφής της αντίστοιχης συνάρτησης.
- ▶ Απαιτούν μεγαλύτερη υπολογιστική προσπάθεια.

Blum-Blum-Shub (1986)

Αλγόριθμος

- ▶ Πάρε δύο μεγάλους πρώτους p, q , με $p \equiv q \equiv 3 \pmod{4}$, και θέσε $n = pq$.
- ▶ Επίλεξε τυχαία ένα s_0 σχετικά πρώτο με το n .
- ▶ Πάρε

$$s_0 = s_{-1}^2 \pmod{n}$$

Για $1 \leq i \leq \infty$

$$s_i = s_{i-1}^2 \pmod{n}, z_i = s_i \pmod{2}$$

Παρατήρηση: σχετικά αργό, αλλά ασφαλές με την υπόθεση ότι ο έλεγχος τετραγωνικών υπολοίπων \pmod{n} είναι δύσκολος αν δεν είναι γνωστή η παραγοντοποίηση του n .

Παράδειγμα BBS

Έστω $n = 192649 = 383 * 503$ και $s_0 = 101355^2 \pmod n = 20749$.

Τα πρώτα 5 bits που παράγονται από τον BBS είναι

11001

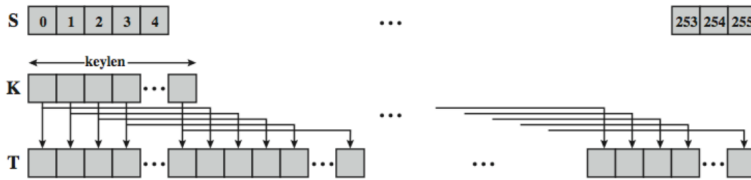
και προκύπτουν:

i	s_i	z_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1

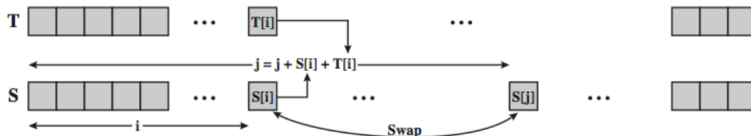
Η γεννήτρια ψευδοτυχαίων RC4

- ▶ Rivest (1987)
- ▶ Ιδιωτικό της εταιρίας RSA Data Security, Inc (κλειστό)
- ▶ Διέρρευσε το 1994
- ▶ Χρήση σε πολύ διαδεδομένα πρωτόκολλα: WEP/WPA, SSL/TLS

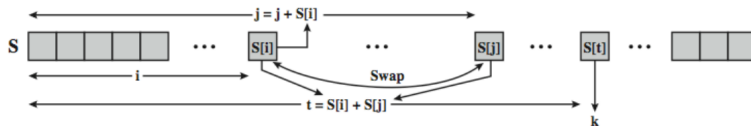
RC4 σχηματικά



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation

Η γεννήτρια ψευδοτυχαίων RC4

- ▶ Συστατικά: 2 arrays of bytes:
 - ▶ Μετάθεση $P[0..255]$. Αρχικοποίηση:
for all $i \in \{0..255\}$ **do** : $P[i] = i$
 - ▶ Κλειδί $K[0..keylen - 1]$, $keylen \leq 256$ – συνήθως $keylen \in [5..8]$.
Επιλέγεται από χρήστη.
- ▶ Δημιουργία σειράς κλειδιών (key-scheduling algorithm – KSA). Η αρχική (ταυτοτική) μετάθεση P μετατρέπεται μέσω μιας σειράς ανταλλαγών (swap) σε μια (φαινομενικά τυχαία) μετάθεση.
Το “ανακάτεμα” επηρεάζεται από το αρχικό κλειδί K .
- ▶ Παραγωγή ψευδοτυχαίων bytes (pseudorandom generation algorithm – PRGA)
Επαναληπτικός βρόχος. Σε κάθε επανάληψη επιλέγεται κάποιο byte της P ως κλειδί εξόδου με τρόπο που καθορίζεται από τα τρέχοντα περιεχόμενα της P . Οι επαναλήψεις συνεχίζονται για όσο χρειάζεται (δηλ. μέχρι να τελειώσει το stream). Σε κάθε επανάληψη γίνεται και ένα νέο swap.

Η γεννήτρια ψευδοτυχαίων RC4

Περιγραφή KSA, PRGA

- ▶ Δημιουργία σειράς κλειδιών (KSA)

$j = 0$

for $i = 0$ **to** 255 **do** :

$j = (j + P[i] + K[i \bmod \text{keylen}]) \bmod 256$

swap($P[i]$, $P[j]$)

- ▶ Παραγωγή ψευδοτυχαίων bytes (PRGA)

$i = 0; j = 0$

while next key needed :

$i = (i + 1) \bmod 256 ; j = (j + P[i]) \bmod 256$

swap($P[i]$, $P[j]$)

$K_o = P[(P[i] + P[j]) \bmod 256]$

output K_o

Κάθε κλειδί εξόδου K_o χρησιμοποιείται για την κρυπτογράφηση ενός byte αρχικού κειμένου.

Η γεννήτρια ψευδοτυχαίων RC4

Παρατηρήσεις

- ▶ Με ίδιο αρχικό κλειδί K προκύπτει η ίδια σειρά κλειδιών εξόδου.
- ▶ Απλή και γρήγορη στην υλοποίηση με software (σε αντίθεση με άλλα stream cipher, π.χ. αυτά που βασίζονται σε LFSRs).
- ▶ Η ασφάλεια της γεννήτριας RC4 έχει αμφισβητηθεί έντονα. Κάποιοι τρόποι χρήσης ιδιαίτερα ανασφαλείς (π.χ. WEP) – επίθεση Fluhrer, Mantin, Shamir (2001).

Η γεννήτρια ψευδοτυχαίων RC4

Παρατηρήσεις

- ▶ Με ίδιο αρχικό κλειδί K προκύπτει η ίδια σειρά κλειδιών εξόδου.
- ▶ Απλή και γρήγορη στην υλοποίηση με software (σε αντίθεση με άλλα stream cipher, π.χ. αυτά που βασίζονται σε LFSRs).
- ▶ Η ασφάλεια της γεννήτριας RC4 έχει αμφισβητηθεί έντονα. Κάποιοι τρόποι χρήσης ιδιαίτερα ανασφαλείς (π.χ. WEP) – επίθεση Fluhrer, Mantin, Shamir (2001).
Απόδειξη στον πίνακα

Η γεννήτρια ψευδοτυχαίων RC4

Παρατηρήσεις

- ▶ Με ίδιο αρχικό κλειδί K προκύπτει η ίδια σειρά κλειδιών εξόδου.
- ▶ Απλή και γρήγορη στην υλοποίηση με software (σε αντίθεση με άλλα stream cipher, π.χ. αυτά που βασίζονται σε LFSRs).
- ▶ Η ασφάλεια της γεννήτριας RC4 έχει αμφισβητηθεί έντονα. Κάποιοι τρόποι χρήσης ιδιαίτερα ανασφαλείς (π.χ. WEP) – επίθεση Fluhrer, Mantin, Shamir (2001).
Απόδειξη στον πίνακα
- ▶ Ουσιαστικό πρόβλημα η παραλλαγή του RC4 με χρήση IV, όπου μπορεί να αποκαλυφθεί το πραγματικό κλειδί (WEP)

Η γεννήτρια ψευδοτυχαίων RC4

Παρατηρήσεις

- ▶ Με ίδιο αρχικό κλειδί K προκύπτει η ίδια σειρά κλειδιών εξόδου.
- ▶ Απλή και γρήγορη στην υλοποίηση με software (σε αντίθεση με άλλα stream cipher, π.χ. αυτά που βασίζονται σε LFSRs).
- ▶ Η ασφάλεια της γεννήτριας RC4 έχει αμφισβητηθεί έντονα. Κάποιοι τρόποι χρήσης ιδιαίτερα ανασφαλείς (π.χ. WEP) – επίθεση Fluhrer, Mantin, Shamir (2001).
Απόδειξη στον πίνακα
- ▶ Ουσιαστικό πρόβλημα η παραλλαγή του RC4 με χρήση IV, όπου μπορεί να αποκαλυφθεί το πραγματικό κλειδί (WEP)
- ▶ Άμυνα: απόρριψη αρχικού τμήματος κλειδοροής (RC4-drop[n]),
ενδεικτικά: $n = 768$ bytes, συστήνεται ακόμη και $n = 3072$.

Η γεννήτρια ψευδοτυχαίων RC4

Παρατηρήσεις

- ▶ Με ίδιο αρχικό κλειδί K προκύπτει η ίδια σειρά κλειδιών εξόδου.
- ▶ Απλή και γρήγορη στην υλοποίηση με software (σε αντίθεση με άλλα stream cipher, π.χ. αυτά που βασίζονται σε LFSRs).
- ▶ Η ασφάλεια της γεννήτριας RC4 έχει αμφισβητηθεί έντονα. Κάποιοι τρόποι χρήσης ιδιαίτερα ανασφαλείς (π.χ. WEP) – επίθεση Fluhrer, Mantin, Shamir (2001).
Απόδειξη στον πίνακα
- ▶ Ουσιαστικό πρόβλημα η παραλλαγή του RC4 με χρήση IV, όπου μπορεί να αποκαλυφθεί το πραγματικό κλειδί (WEP)
- ▶ Άμυνα: απόρριψη αρχικού τμήματος κλειδοροής (RC4-drop[n]),
ενδεικτικά: $n = 768$ bytes, συστήνεται ακόμη και $n = 3072$.
- ▶ **Μη ασφαλές!**

Κρυπτοσυστήματα ροής (stream ciphers)

Παραγωγή ακολουθίας κλειδιών με βάση κάποιο αρχικό κλειδί, και (πιθανά) το plaintext.

Ορισμός

- ▶ Plaintext: x_0, x_1, \dots, x_{n-1}
- ▶ Ciphertext: y_0, y_1, \dots, y_{n-1}
- ▶ Αρχικό κλειδί: k
- ▶ Βοηθητικές συναρτήσεις: $f_i, 0 \leq i < m$
- ▶ Key stream: $z_i = f_{i \bmod m}(k, x_0, \dots, x_{i-1}, z_0, \dots, z_{i-1})$
- ▶ Κρυπτογράφηση: $y_i = enc_{z_i}(x_i)$
- ▶ Αποκρυπτογράφηση: $x_i = dec_{z_i}(y_i)$

Π.χ. για δυαδικές ακολουθίες:

$$enc_z(x) = x \oplus z = x + z \bmod 2$$

$$dec_z(y) = y \oplus z = y + z \bmod 2$$

Κρυπτοσυστήματα ροής - Τρόποι λειτουργίας

Διακρίνονται σε **synchronous** (το κλειδί δεν εξαρτάται από το plaintext), και **asynchronous** (λέγονται και **self-synchronizing**).

Επίσης σε **periodic** ($\forall i : z_{i+d} = z_i$, όπου d η περίοδος) και **aperiodic**.

Παράδειγμα: το Vigenère είναι synchronous και periodic.

Κρυπτοσυστήματα ροής: Linear Recurrence Keystream

Αρχικό διάνυσμα κλειδιών: $(z_0, z_1, \dots, z_{m-1})$.

Τα υπόλοιπα κλειδιά υπολογίζονται ως εξής:

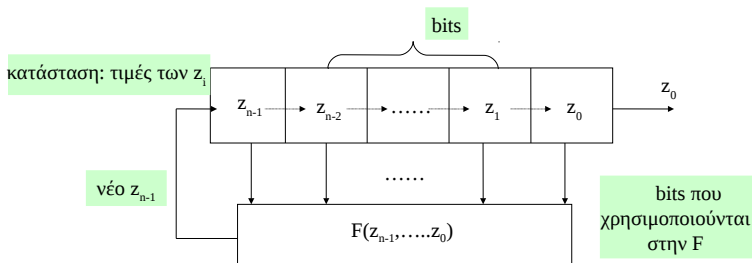
$$z_{i+m} = \sum_{j=0}^{m-1} c_j \cdot z_{i+j} \pmod{2}, \quad \forall j, c_j \in \{0, 1\}$$

Εάν το πολυώνυμο $c_0 + c_1x + c_2x^2 + \dots + c_{m-1}x^{m-1} + x^m$ είναι **primitive**, τότε το κρυπτοσύστημα έχει περίοδο $d = 2^m - 1$.

Π.χ. $c_0 = c_1 = 1, c_2 = c_3 = 0$ ορίζουν το πολυώνυμο $x^4 + x + 1$, και με δεδομένο αρχικό κλειδί z_0, \dots, z_3 έχουμε $z_{4+i} = z_i + z_{i+1} \pmod{2}$.

Το κρυπτοσύστημα αυτό έχει περίοδο 15.

Υλοποίηση με **Linear Feedback Shift Register (LFSR)**.



Σχήμα: FSR

Καταχωρητές Ολίσθησης Γραμμικής Ανάδρασης - LFSRs

- ▶ Δημιουργούν περιοδικές ακολουθίες, με περίοδο το πολύ $2^L - 1$, όπου L το πλήθος των ψηφίων.
- ▶ Αν το αντίστοιχο πολυώνυμο είναι primitive έχουμε **maximum-length LFSR**. Πολλά γνωστά primitive πολυώνυμα.

Καταχωρητές Ολίσθησης Γραμμικής Ανάδρασης - LFSRs

- ▶ Δημιουργούν περιοδικές ακολουθίες, με περίοδο το πολύ $2^L - 1$, όπου L το πλήθος των ψηφίων.
- ▶ Αν το αντίστοιχο πολυώνυμο είναι primitive έχουμε **maximum-length LFSR**. Πολλά γνωστά primitive πολυώνυμα.
- ▶ Σημαντικό μέγεθος για ακολουθίες: **γραμμική πολυπλοκότητα (linear complexity)**. Είναι το ελάχιστο μέγεθος LFSR που παράγει την ίδια ακολουθία.
- ▶ Αλγόριθμος Berlekamp-Massey: υπολογίζει τη γραμμική πολυπλοκότητα και τον αντίστοιχο LFSR.

Καταχωρητές Ολίσθησης Γραμμικής Ανάδρασης - LFSRs

- ▶ Δημιουργούν περιοδικές ακολουθίες, με περίοδο το πολύ $2^L - 1$, όπου L το πλήθος των ψηφίων.
- ▶ Αν το αντίστοιχο πολυώνυμο είναι primitive έχουμε **maximum-length LFSR**. Πολλά γνωστά primitive πολυώνυμα.
- ▶ Σημαντικό μέγεθος για ακολουθίες: **γραμμική πολυπλοκότητα (linear complexity)**. Είναι το ελάχιστο μέγεθος LFSR που παράγει την ίδια ακολουθία.
- ▶ Αλγόριθμος Berlekamp-Massey: υπολογίζει τη γραμμική πολυπλοκότητα και τον αντίστοιχο LFSR.
- ▶ Αύξηση γραμμικής πολυπλοκότητας: χρήση περισσότερων LFSRs, συνδυασμός εξόδων με μη γραμμικό τρόπο.

Π.χ. Geffe generator συνδυάζει 3 maximum-length LFSRs με μήκος L_1, L_2, L_3 και εξόδους x_1, x_2, x_3 :

$$f(x_1, x_2, x_3) = x_1x_2 \oplus (1 \oplus x_2)x_3$$

έχει περίοδο $(2^{L_1} - 1) \cdot (2^{L_2} - 1) \cdot (2^{L_3} - 1)$ και γραμμική πολυπλοκότητα $L = L_1L_2 + L_2L_3 + L_3$

Κρυπτοσυστήματα ροής με LFSRs

- ▶ LFSR: εύκολη υλοποίηση σε hardware, καλές στατιστικές ιδιότητες, αλλά μη ασφαλή γιατί τα bits εξόδου έχουν γραμμική σχέση
- ▶ Λύσεις
 - ▶ μη γραμμική ανάδραση
 - ▶ μη γραμμικός συνδυασμός των registers δίνει την έξοδο
 - ▶ συνδυασμός των εξόδων περισσότερων LFSRs, αλλά χωρίς εξάρτηση της τελικής εξόδου από κάποια από τις επιμέρους

Κρυπτοσυστήματα ροής με LFSRs

- ▶ LFSR: εύκολη υλοποίηση σε hardware, καλές στατιστικές ιδιότητες, αλλά μη ασφαλή γιατί τα bits εξόδου έχουν γραμμική σχέση
- ▶ Λύσεις
 - ▶ μη γραμμική ανάδραση
 - ▶ μη γραμμικός συνδυασμός των registers δίνει την έξοδο
 - ▶ συνδυασμός των εξόδων περισσότερων LFSRs, αλλά χωρίς εξάρτηση της τελικής εξόδου από κάποια από τις επιμέρους
- ▶ Χρήση σε:
 1. DVD (CSS): 2 LFSRs, (ανάκτηση σπόρου σε 2^{17})
 2. GSM (A5/1): 3 LFSRs ($2^{39.91}$, με προεργασία 2^{38}), (A5/2): 4 LFSRs
 3. Bluetooth (E0): 4 LFSRs (ανάκτηση σπόρου σε 2^{38})

- ▶ eStream project: 2004-2008
- ▶ Κατηγορίες:
 - ▶ Μήκος κλειδιού 128 bits και ένα IV (initialization vector) μήκους 64 και/ή 128 bits (SW)
 - ▶ Μήκος κλειδιού 80 bits και ένα IV (initialization vector) μήκους 32 και/ή 64 bits (HW)
- ▶ Ξεχωριστές προτάσεις για SW και για HW
- ▶ Αξιολόγηση:
 - ▶ Ασφάλεια
 - ▶ Δωρεάν αδειοδότηση
 - ▶ Επιδόσεις και φάσμα εφαρμογών
- ▶ Η επιτροπή απλά μάζεψε τις συμμετοχές, η αξιολόγηση έγινε από την κοινότητα

eStream

- ▶ Κριτήρια ασφάλειας
 - ▶ οποιαδήποτε επίθεση ανάκτησης κλειδιού πρέπει να είναι τόσο δύσκολη όσο η εξαντλητική αναζήτηση
 - ▶ Απλότητα σχεδίασης
- ▶ Κριτήρια υλοποίησης
 - ▶ SW και HW αποδοτικότητα
 - ▶ Εκτέλεση και μνήμη
 - ▶ Επίδοση
 - ▶ Ευελιξία χρήσης

SW	HW
HC-128	Grain v1
Rabbit	MICKEY 2.0
Salsa20	Trivium
Sosemanuk	

Λεπτομέρειες στο www.ecrypt.eu.org/stream