Kryptografia i bezpieczeństwo danych
- SZYFRY STRUMIENIOWE

Sławomir Samolej
ssamolej.kia.prz.edu.pl
ssamolej@prz.edu.pl

1

Notatka 1

Definicja Szyfru Symetrycznego

- Szyfr jest zdefiniowany na trzech zbiorach $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
  - $\mathcal{K}$: zbiór wszystkich kluczy
  - $\mathcal{M}$: zbiór wszystkich wiadomości
  - $\mathcal{C}$: zbiór wszystkich szyfrogramów (zaszyfrowanych wiadomości)
- To jest para „efektywnych" algorytmów (***E***, ***D***),
  gdzie: $\boldsymbol{E}: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, $\boldsymbol{D}: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$
- Dla każdej wiadomości *m* należącej do $\mathcal{M}$ i każdego klucza *k* należącego do $\mathcal{K}$ zachodzi:

$$D(k, E(k, m)) = m$$

Algorytm ***E*** jest często losowy,
algorytm ***D*** jest zawsze deterministyczny.

2

Notatka 2

Szyfr z kluczem jednorazowym (One Time Pad) (Vernam 1917)

- Pierwszy przykład „bezpiecznego” szyfru.
- Przestrzeń wiadomości ($\mathcal{M}$) i szyfrów ($\mathcal{C}$) jest n-bitowym ciągiem.
- Klucz jest losowym cięgiem bitów o długości wiadomości.

3

Uwaga: przestrzeń wiadomości ma taki sam rozmiar co przestrzeń kluczy są to n-bitowe ciągi.

Definicja szyfru z kluczem jednorazowym

- C = E(k, m) = k ⊕ m
- D(k, m) = k ⊕ c

msg: 0 1 1 0 1 1 1

key: 1 0 1 1 0 1 0 ⊕

CT:

4

Własności operacji xor:

1) m xor k -> c, równocześnie c xor k -> m (mamy uniwersalny sposób szyfrowania i deszyfrowania z kluczem tajnym)
2) Jeśli weźmiemy dowolny ciąg losowy i niezależny od niego jednorodny ciąg losowy to po wykonaniu na nich xor otrzymamy jednorodny ciąg losowy.

Pytanie

- Mamy wiadomość (m) i jej zaszyfrowaną postać (c). Czy możemy obliczyć na ich podstawie klucz (k)?
  - Nie, nie można jej obliczyć
  - Tak, klucz wynosi: $k = m \oplus c$
  - Można obliczyć tylko połowę bitów klucza
  - Tak, klucz wynosi $k = m \oplus m$

5

Szyfr z kluczem jednorazowym

- Bardzo szybki!!!
  ale wymaga długiego klucza… (równego długości wiadomości)

- Czy jest dobrym szyfrem?
- Co oznacza, że szyfr jest dobry?
- Czy jest bezpieczny?
- Co oznacza, że szyfr jest bezpieczny?

6

Pierwsza definicja bezpiecznego szyfru

- Możliwości intruza: **Do dyspozycji mamy tylko zaszyfrowany teks** (na razie).
- Możliwe wymagania co do bezpieczeństwa:
  - 1. atakujący nie może odzyskać klucza
  - 2. atakujący nie może odszyfrować wiadomości
- Pomysł Shannona:
  - Wiadomość zaszyfrowana nie może naprowadzić nas na tekst wiadomości niezaszyfrowanej

7

(Shannon 1949)

- Podstawowa idea: Zaszyfrowany tekst nie może ujawnić informacji o tekście jawnym.
- Zostaje wprowadzone pojęcie **idealnego bezpieczeństwa**:
  - Dla każdej dowolnej pary wiadomości (o tej samej długości) m_0 i m_1 i klucza k (o rozkładzie równomiernym), jeśli je zaszyfruję, to patrząc na szyfr nie wiem, czy zaszyfrowana wiadomość była wiadomością m_0, czy m_1.
- Wnioski:
  - Nawet „najmocniejszy" intruz nie dowie się niczego o wiadomości posiadając jej zaszyfrowaną postać.
  - Jeśli dysponujemy **tylko zaszyfrowaną wiadomość** szyfr jest nie do złamania (wykonujemy atak tylko na zaszyfrowaną wiadomość)
  - Inne ataki są możliwe
  - **Szyfr z jednorazowym kluczem posiada idealne bezpieczeństwo.**
  - **Jeśli szyfr mam mieć właściwość idealnego bezpieczeństwa, to długość klucza musi być większa lub równa długości wiadomości.**
  - Szyfr z kluczem jednorazowym jest niepraktyczny (problem z dystrybucją klucza), ale wnioski z właściwości tego szyfru pozwalają tworzyć inne bezpieczne szyfry (ale nie posiadające idealnego bezpieczeństwa)

8

Uwaga: Istnieje tw., że jeśli chcemy zachować idealne bezpieczeństwo, to długość klucza musi być równa lub większa od długości wiadomości…

Szyfr strumieniowy: przekształcenie pomysłu na szyfr z kluczem jednorazowym na coś praktycznego

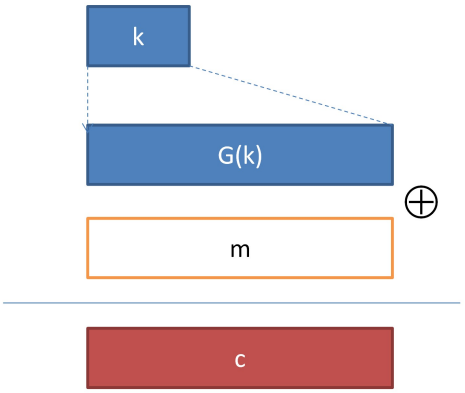
- Pomysł: należy zastąpić „losowy” klucz kluczem generowanym przez algorytm „pseudolosowy”.
- Do generowania kluczy służą generatory liczb pseudolosowych (**PRG – pseudo-random generator**): algorytm, który na podstawie pewnego ciągu zero-jedynkowego (**ziarna**) generuje znacznie dłuższy ciąg mający charakter pseudolosowy.
- Ciąg musi być efektywnie „wyliczalny” i generowany przez deterministyczny algorytm.
- Wygenerowany ciąg musi „wyglądać” jak losowy.

9

Szyfr strumieniowy zasada działania

$C = E(k,m) = m \oplus G(k)$

$D(k,c) = c \oplus G(k)$



10

Czy szyfr strumieniowy może posiadać właściwość perfekcyjnego bezpieczeństwa?

- Tak, jeśli PRG jest naprawdę „bezpieczny”
- Nie, nie ma szyfrów z perfekcyjnym bezpieczeństwem
- Tak, każdy szyfr ma perfekcyjne bezpieczeństwo
- Nie, ponieważ klucz jest krótszy od wiadomości

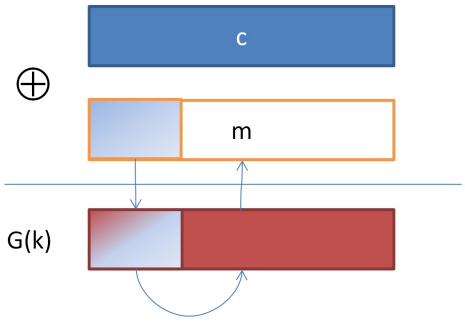
11

Inna definicja bezpieczeństwa…

- Szyfry strumieniowe nie mają perfekcyjnego bezpieczeństwa
- Wnioski;
  - Musi się pojawić inna definicja bezpieczeństwa
  - Bezpieczeństwo będzie zależeć od PRG

PRG musi być nieprzewidywalny

- Nieodpowiedni algorytm PRG pozwala na określenie kolejnej sekwencji (pseudolosowych) bitów na podstawie pewnej początkowej sekwencji



Formalnie: PRG jest nieprzewidywalny, jeśli następny element ciągu pseudolosowego jest możliwy do obliczenia na podstawie wcześniej obliczonych wartości z bardzo małym prawdopodobieństwem (np. $1/2^{30}$)

13

Jeśli znamy część wiadomości (np. wiadomo, że jest to zaszyfrowany email do znanej nam osoby, wtedy zawsze zaczyna się od „Do:Slawka”) i algorytm generowania liczb pseudolosowych jest przewidywalny (to znaczy, że na podstawie pierwszych i-bitów ciągu losowego można obliczyć pozostałe n-bitów), to wystarczy wykonać xor z pierwszej części szyfrogramu i znanej nam części wiadomości. Rezultatem będzie pierwsza część ciągu losowego wygenerowanego przez przewidywalny algorytm. Na jej podstawie można obliczyć resztę tego ciągu i wiedząc, że jest kluczem odszyfrować resztę wiadomości.
Problem występuje nawet wtedy, gdy bylibyśmy w stanie przewidzieć nawet 1 bit po i-bitach wcześniej wygenerowanych.

Słabe generatory liczb pseudolosowych (nie do użycia w kryptografii)

- LCG (linear congruential generator) = Liniowy Generator Kongruentny
  r[i] = a*r[i-1] + b mod p (**ziarno=r[0]**)
  wyjście: bity liczby r[i]
  i++

- glibc random():

  $r[i] \leftarrow (r[i-3] + r[i-31]) \ \% \ 2^{32}$

  output r[i] >> 1

- Mają niezłe właściwości statystyczne, ale można ten ciąg przewidzieć.
- Nigdy nie należy używać do zastosowań kryptograficznych!

14

Uwaga: nigdy nie należy używać standardowej funkcji random() w zastosowaniach kryptograficznych. System szyfrowania Keberos w wersji 4 używał tej funkcji i został łatwo złamany.

Atak 1 na szyfr z kluczem jednorazowym

- Zastosowanie dwa razy tego samego klucza nie jest bezpieczne!!!

$C_1 \leftarrow m_1 \oplus PRG(k)$

$C_2 \leftarrow m_2 \oplus PRG(k)$

- Atakujący wykonuje:

$C_1 \oplus C_2 \rightarrow$

- Jest wystarczająca liczba powtarzających się znaków, żeby:

$m_1 \oplus m_2 \rightarrow m_1, m_2$

- **WNIOSEK: w szyfrze z kluczem jednorazowym i w szyfrach strumieniowych nie wolno dwa razy zastosować tego samego klucza!!!**

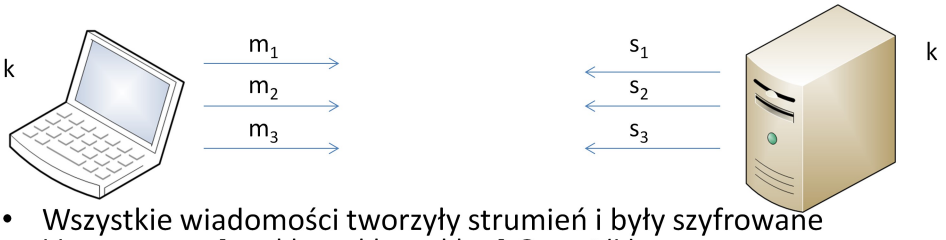
15

Rzeczywiste przykłady (1)

- Projekt Venona (Związek Radziecki, 1941-1946).
  - szyfrowanie z kluczem jednorazowym
  - klucz był uzyskiwany przez notowanie rzutem kostką
  - ponieważ było to pracochłonne, to używano tego samego klucza do kodowania wielu wiadomości
  - Rząd Stanów Zjednoczonych odczytał około 3000 zaszyfrowanych wiadomości

Rzeczywiste przykłady (2)

- MS-PPTP (Point to Point Tuneling Protocol Windows NT)



- Wszystkie wiadomości tworzyły strumień i były szyfrowane kluczem: $[m_1 \,||\, m_2 \,||\, m_3 \,||\, \ldots] \oplus PRG(k)$
- Niestety odpowiedzi na wiadomości były szyfrowane tym samym kluczem: $[s_1 \,||\, s_2 \,||\, s_3 \,||\, \ldots] \oplus PRG(k)$
- Taki system można zaatakować tak jak na slajdzie nr 15.
- **Wniosek: Przy szyfrowaniu komunikacji Klient-Serwer i Serwer-Klient potrzeba stosować różne klucze**
- **Klucz do komunikacji jest wtedy parą kluczy i obie strony muszą znać oba klucze.**

17

Rzeczywiste przykłady (3)

802.11b WEP:

- Ramka jest szyfrowana szyfrem strumieniowym z uzgodnionym kluczem k
- Klucz każdej ramki uzyskuje się łącząc go z 24 bitowym ciągiem IV
- Można sobie wyobrazić, że ten ciąg rozpoczyna się od 0 i co ramkę jest zwiększany o jeden
- Pomysł gwarantował zmianę klucza co ramkę. Odbiorca znał klucz i dostawał wraz z ramką wartość IV
- Problemem okazała się długość IV: jest tylko 2^{24} IV, w przybliżeniu 16 milionów i potem następuje reset IV.
- **Problem 1:** W dostatecznie długim strumieniu ramek są więc dwie zaszyfrowane tak samo
- **Problem 2:** Po resecie **niektórych** kart bezprzewodowej komunikacja startowała od nowa z IV == 0

18

Rzeczywiste przykłady (4)
– należy unikać „powiązanych ze sobą kluczy”

802.11b WEP:

- Problem 3: kolejne klucze uzyskuje się przez połączenie stałego klucza i wartości IV zwiększonej o 1.
  - Klucze są do siebie podobne
- Problem 4: zastosowany algorytm do generowania liczb pseudolosowych RC4 był źle stosowany (udany atak opublikowany 2001 roku)

19

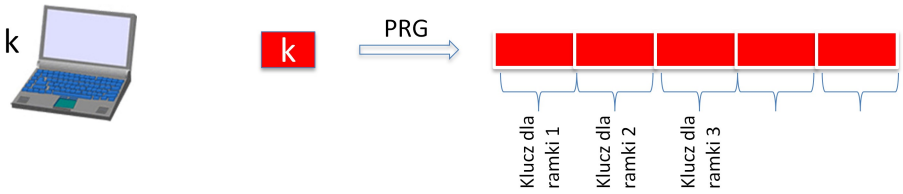
Pr. 3: PRG zastosowany w WEP nie generował bezpiecznych ciągów losowych, kiedy dane wejściowe niewiele się od siebie różniły.
Pr. 4: Znając właściwości szyfrowania strumienia danych w WEP (bliskie klucze) i algorytm RC4 jako PRG opublikowano atak pokazujący, że po podsłuchaniu 10^6 ramek można odkryć klucz. Późniejsze prace pokazały, że wystarczy tylko 40 000 ramek do przeprowadzenie takiego ataku (czas ataku: minuty!!!).

Jak można by było sprawę poprawić? (1)

1. Pakiety można by było potraktować jako strumień danych zaszyfrowany kluczem generowanym za pomocą PRG:
   $[m_1 \,||\, m_2 \,||\, m_3 \,||\, \ldots] \oplus PRG(k)$

20

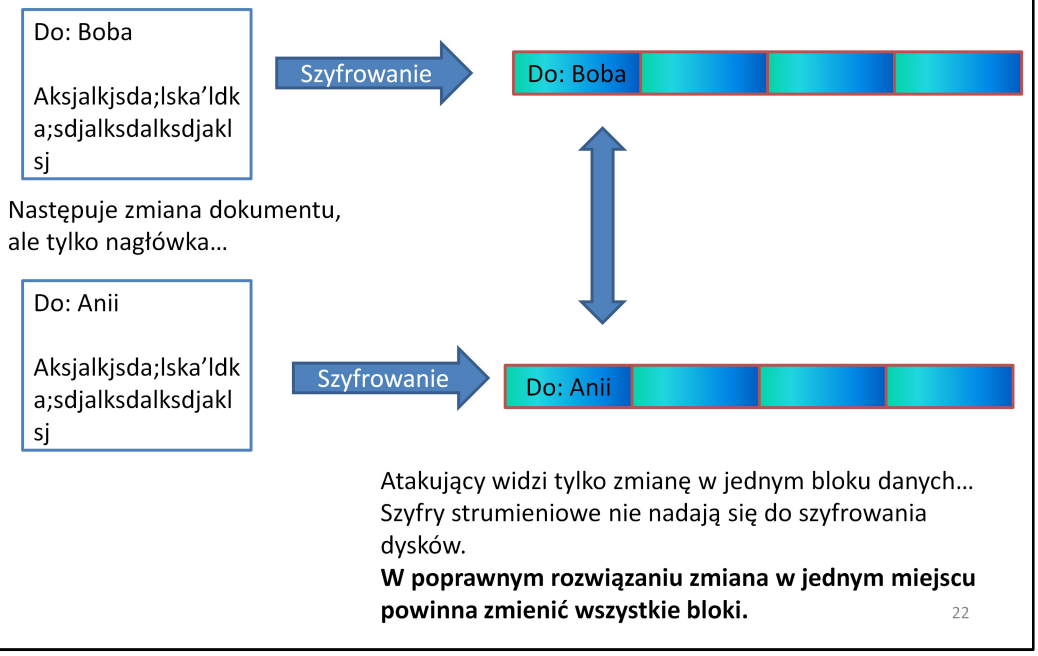
Jak można by było sprawę poprawić? (2)
(jeśli nie chcemy danych potraktować jako strumień)



1. Klucze nie są ze sobą powiązane
2. Jeśli PRG spełnia wymagania bezpieczeństwa, to szyfrowanie jest bezpieczne

21

Szyfrowanie dysków tym samym kluczem



Atakujący widzi tylko zmianę w jednym bloku danych…
Szyfry strumieniowe nie nadają się do szyfrowania dysków.
W poprawnym rozwiązaniu zmiana w jednym miejscu powinna zmienić wszystkie bloki.

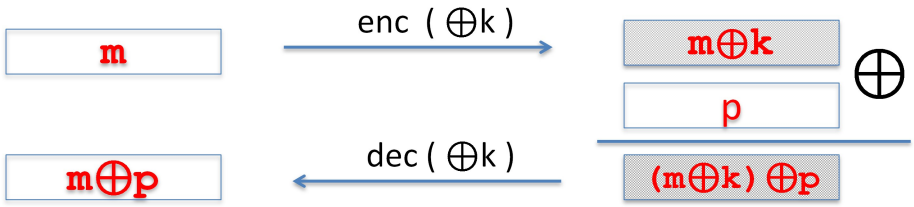
22

Atakujący wykonuje zrzut z dysku po pierwszym i po drugim zaszyfrowaniu. Zauważa, że zmianie ulega tylko jeden z bloków danych. Pomimo że atakujący nie wie, co się zmieniło, to wie, gdzie się zmieniło i toteż już jest ujawnienie informacji!

Podsumowanie ataków na szyfry, w których wielokrotnie użyto tego samego klucza:

- **W szyfrowaniu strumieniowym klucz może być użyty tylko raz!!!**
  - Transmisja danych w sieci: należy negocjować klucze dla każdej sesji (np. TLS)
  - Szyfrowanie dysków: najczęściej nie stosuje się do tego szyfru strumieniowego

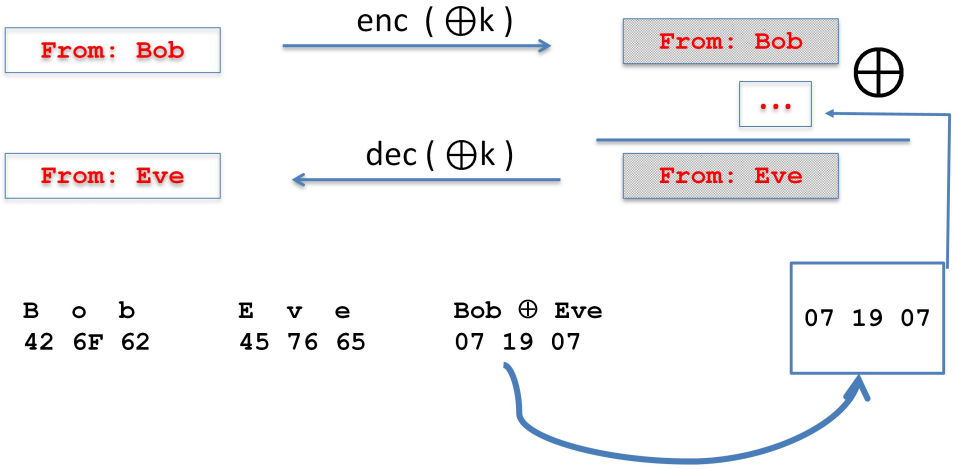
Atak 2: brak integralności (1)



- Modyfikacja wiadomości zaszyfrowanej jest **niewykryta** i ma **przewidywalny wpływ na treść wiadomości**.

24

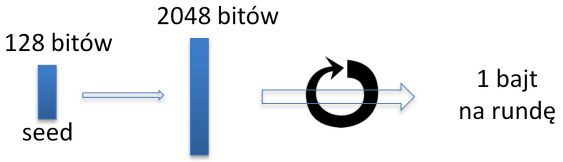
Atak 2: brak integralności (2)



Modyfikacja wiadomości zaszyfrowanej jest **niewykryta** i ma **przewidywalny wpływ na treść wiadomości**.

25

Realne szyfry strumieniowe (1)
(programowy) RC4 (1987)

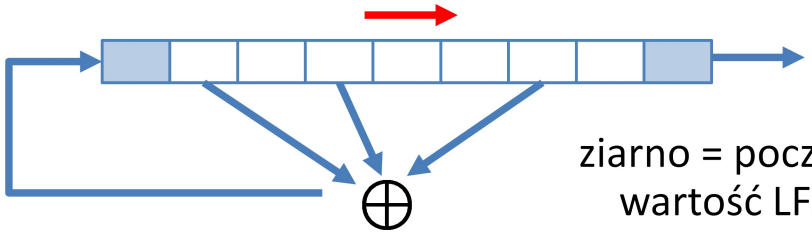


- 128 bitowe ziarno
- Ziarno jest rozszerzane do długości 2048 bitów będące wewnętrznym stanem generatora
- Następnie w pętli generowane są pojedyncze bajty klucza
- Stosowany w HTTPs (przez Google) i WEP (ale nie w bezpieczny sposób)
- W nowych projektach nie jest zalecany
- Słabości:
  - Stronnicza wartość drugiego bajta danych: Pr[2nd byte = 0] = 2/256
  - Okazuje się, że wartość pierwszego i trzeciego też jest stronnicza
  - Zaleca się, żeby w implementacjach ominąć pierwsze 256 bajtów i stosować wygenerowaną liczbę wyjścia od 257
  - Prawdopodobieństwo uzyskania pary bajtów (0,0) wynosi $1/256^2 + 1/256^3$ (pojawia się po wygenerowaniu kilku GB danych)
  - Jeśli klucze są do siebie podobne, to RC4 generuje podobne ciągi pseudolosowe więc można przeprowadzić atak

26

Realne szyfry strumieniowe (2)
(sprzętowy) CSS (ang. Content Scrambling System)
(złamany)

Rejestr przesuwający z liniowym sprzężeniem zwrotnym
(LSFR – Linear Feedback Shift Register)



ziarno = początkowa wartość LFSR

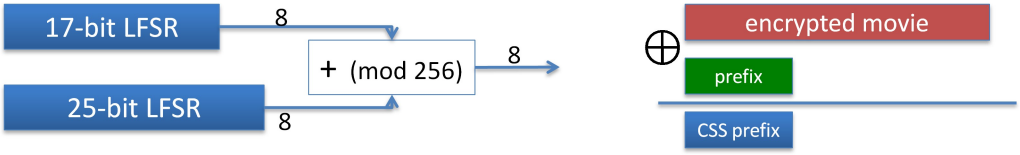
Szyfrowanie DVD (CSS):	2 LFSR	Wszystkie złamane
Szyfrowanie GSM (A5/1, 2):	3 LFSRs	
Bluetooth (E0):	4 LFSR	

27

- LSFR to rejestr bitowy

Realne szyfry strumieniowe (3), Atak na CSS

- CSS: ziarno = 5 bajtów = 40 bitów



• Znamy prefix kodowania MPEG (np.20 bitów)

• Dla każdego z 17-bitowego LFSR generujemy 20-bitowe wyjście z generatora szyfru

• Znamy prefix, wygenerowaliśmy wyjście dla jednego z 17-bitowych LSFR, możemy wyliczyć 20-bitowe wyjście 25-bitowego LFSR i znaleźć początkową wartość 25-bitowego LSFR

• Znając wartości początkowe obu rejestrów, możemy wygenerować resztę klucza i odszyfrować film DVD

Współczesne szyfry strumieniowe: eStream (2008)

PRG: $\{0,1\}^s \times R \longrightarrow \{0,1\}^n$, gdzie $n >> s$

ziarno

Nonce (wartość jednorazowa)

Nonce: niepowtarzalna wartość przy zastosowaniu danego klucza.

$$E(k, m ; r) = m \oplus PRG(k ; r)$$

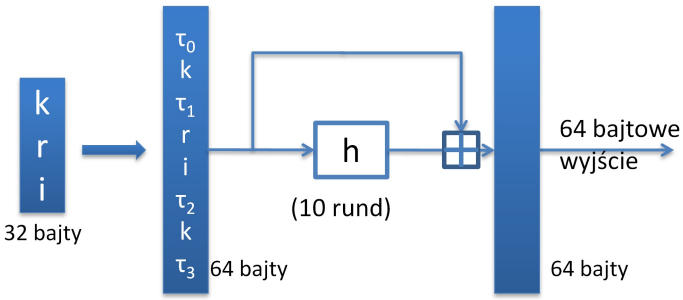
- para (k,r) nie jest więcej razy stosowana niż raz
- klucz może być stosowany wielokrotnie, ponieważ para (k, r) jest unikalna.

29

eStream: Salsa 20 (sprzęt lub program)

Salsa20: $\{0,1\}^{128 \text{ lub } 256} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{n}$ (max $n = 2^{73}$ bitów)

Salsa20(k ; r) := H(k , (r, 0)) ll H(k , (r, 1)) ll …



h: funkcja odwracalna. Zaprojektowana, żeby była szybko wykonywana na procesorach x86 (SSE2)

30

Bezpieczeństwo?

- Nieznane: nie istnieje **udowodnione** bezpieczeństwo generatora liczb losowych
- W rzeczywistości: nie ma znanych lepszych ataków niż pełne przeszukiwanie kluczy

31

Wydajność (biblioteka Crypto++ 5.6.0)

AMD Opteron, 2.2 GHz (Linux)

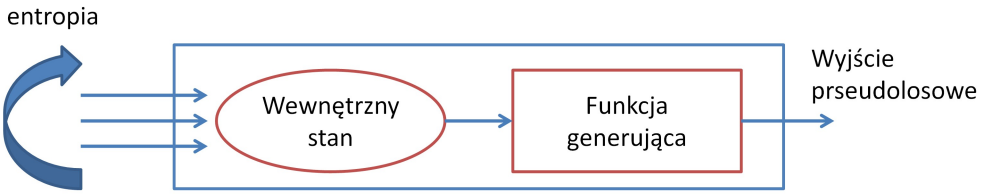
	PRG	Speed (MB/sec)
	RC4	126
eStream	Salsa20/12	643
eStream	Sosemanuk	727

32

Kiedy PRG jest bezpieczny?

- Wtedy, gdy żaden efektywny tzw. test statystyczny nie jest w stanie odróżnić rezultatu PRG od losowego ciągu
- Ale: nie znaleziono matematycznego dowodu że dany PRG jest bezpieczny

Generowanie Przypadkowości



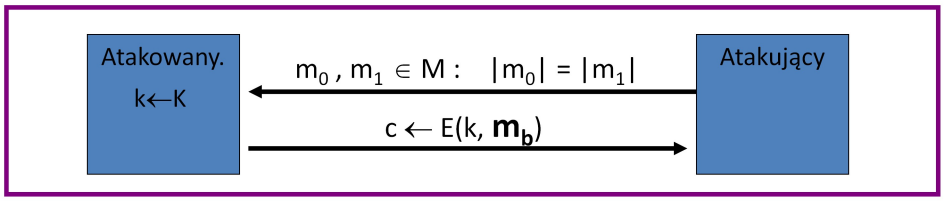
W praktyce: (np. /dev/random)

- Entropia jest ciągle dodawana do stanu
- Źródła entropii:
  - Sprzętowe: Intel **RdRand** inst. (Ivy Bridge). 3Gb/sec.
  - Czasowe: przerwania od sprzętu (mysz, klawiatura)

NIST SP 800-90: Generatory są zatwierdzane przez NIST

34

Bezpieczeństwo Semantyczne



- Atakujący może wybrać dwie wiadomości o identycznej długości m_0 i m_1 i wysłać je do zaszyfrowania. Otrzymuje szyfrogram jednej z nich. Schemat szyfrowania jest **semantycznie bezpieczny** jeśli atakujący jest w stanie z pomijalnie małym prawdopodobieństwem stwierdzić czy otrzymany szyfrogram jest zaszyfrowaną wiadomością m_0 czy m_1.
- Szyfr strumieniowy jest semantycznie bezpieczny, jeśli udowodni się, że generator liczb pseudolosowych jest bezpieczny…
- Bezpieczeństwo semantyczne nie gwarantuje odporności na atak na integralność wiadomości.

35