



Wybrane zagadnienia kryptograficzne

wykład 3
Anna Grocholewska-Czuryło

Szyfry strumieniowe

- ❖ Czy pamiętacie szyfr OTP one-time pad?
 - ❖ $C = E_K(P) = P \oplus K$
 - ❖ klucz musi być ciągiem bitów równym długości wiadomości, wybranym losowo, innym dla każdej szyfrowanej wiadomości
- ❖ Rozwiązanie: użycie pseudo-losowego generatora liczb / ciągów (PRNG)
 - ❖ na wejście PRNG podajemy krótki, losowy, tajny klucz K (ziarno generatora) i generujemy wyglądający losowo ciąg bitów np. 128 bitów K zamieniamy w 10^6 pseudo-losowy ciąg

Przykłady szyfrów strumieniowych

- ❖ RC4
 - ❖ używany np. w WLAN, TLS, IPsec
- ❖ A5/1, A5/2
 - ❖ używany w GSM/GPRS
- ❖ SEAL
- ❖ AES używany w trybie Counter Mode
- ❖ ...

Cechy szyfrów strumieniowych

- ❖ zazwyczaj są bardzo szybkie (szybsze niż szyfry blokowe) - używane tam gdzie szybkość jest ważna: WiFi, DVD, przesyłanie w czasie rzeczywistym mowy,
- ❖ nie gwarantują bezwzględnego bezpieczeństwa,
 - ❖ są tak bezpieczne jak użyty generator PRNG
 - ❖ jeśli używane/implementowane są prawidłowo, mogą być tak bezpieczne jak szyfry blokowe
- ❖ PRNG jest, z definicji, nieprzewidywalny
 - ❖ mając n-bitów ciągu, nie jesteśmy przewidzieć n+1 bitu

Słabe punkty szyfrów strumieniowych

- ❖ nie zapewniają integralności
- ❖ mają cechę łączności i przemienności ($X \oplus Y \oplus Z = (X \oplus Z) \oplus Y$,
- ❖ $(P_1 \oplus \text{PRNG}(K)) \oplus P_2 = (P_1 \oplus P_2) \oplus \text{PRNG}(K)$
- ❖ Bardzo groźny jest atak ze znanym tekstem jawnym, jeśli klucz (strumień klucza się powtarza, jest okresowy)
 - ❖ z własności XOR: $X \oplus X = 0$,
 - ❖ $(P_1 \oplus \text{PRNG}(IV,K)) \oplus (P_2 \oplus \text{PRNG}(IV,K)) = P_1 \oplus P_2$
 - ❖ jeśli atakujący zna P_1 łatwo jest wyznaczyć P_2
 - ❖ nawet z $P_1 \oplus P_2$ można wyznaczyć tekst jawnego - przez redundancję języka - atak WPA2 Krack attack

Szyfratory strumieniowe



$$C = C_1 || C_2 || \dots = E(M_1, K_1) || E(M_2, K_2) || \dots$$

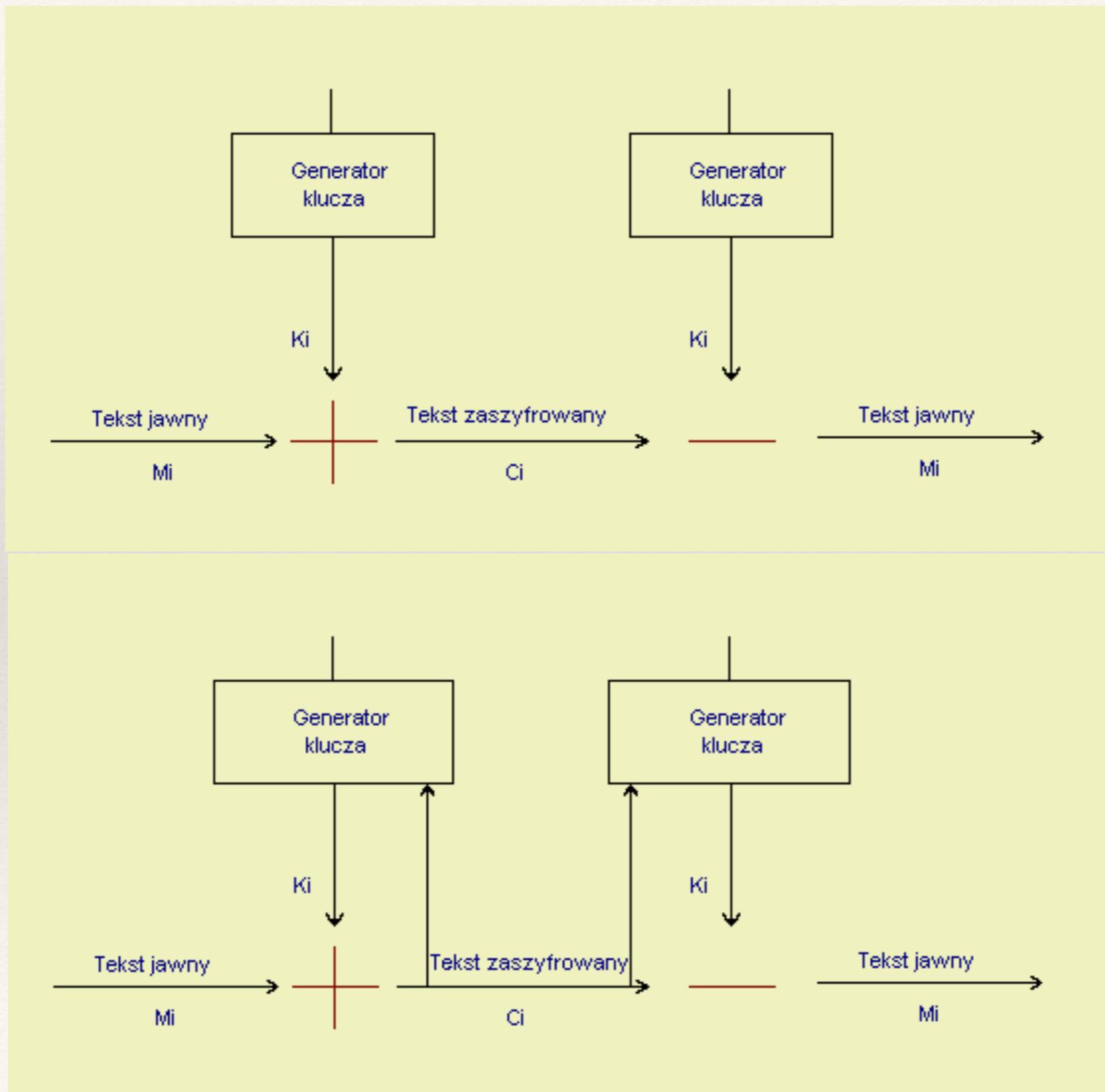
Bezpieczeństwo szyfratorów strumieniowych

- ❖ zależy od zastosowanego generatora strumienia klucza szyfrującego
- ❖ proces szyfrowania strumieniowego jest prostszy, niż w przypadku szyfrów blokowych - większa wydajność
- ❖ w addytywnych szyfrach strumieniowych tekst zaszyfrowany powstaje jako wynik operacji sumowania modulo 2
- ❖ redundancja związana z językiem naturalnym dla dobrze zaprojektowanych szyfrów strumieniowych jest bardzo mała, w przeciwieństwie do szyfrów blokowych

Rodzaje szyfrów strumieniowych

- **synchroniczne**,
- strumień klucza w szyfrach strumieniowych synchronicznych generowany jest niezależnie od tekstu jawnego czy też szyfrogramu
- **samosynchronizujące**
- każdy bit klucza jest zależny od pewnej stałej liczby n poprzednich bitów szyfrogramu

Rodzaje szyfrów strumieniowych



Synchroniczne

Samosynchronizujące

Zalety i wady

- w szyfrach samosynchronizujących, każdy znak tekstu jawnego wpływa na cały szyfrogram, cechy statystyczne związane z tekstem jawnym są rozproszone w całym szyfrogramie (bardziej odporne na atak oparty na redundancji tekstu jawnego niż szyfry strumieniowe synchroniczne)
- wadą takiego trybu pracy jest propagacja błędów. Modyfikacja jednego bitu podczas przesyłania spowoduje niepoprawne wytworzenie n bitów klucza po stronie deszyfrującej, a co jest z tym związane n błędnych bitów w tekście odszyfrowanym

Generatory kluczy

- Generatory kluczy służą do generowania ciągów bitów, wykorzystywanych następnie jako klucze w strumieniowych algorytmach szyfrowania
- Mają one w większości przypadku wszystkie cechy statystyczne charakterystyczne dla ciągów losowych, lecz są powtarzalne, ze względu na to, iż są generowane przez algorytmy deterministyczne, korzystające jedynie z pewnych losowych danych wejściowych (ziarno losowe)

Generatory kluczy

Wielokrotne wywołanie tych algorytmów z takimi samymi danymi wejściowymi powoduje wygenerowanie takich samych wyjściowych ciągów. W przypadku kluczy kryptograficznych może to okazać się niebezpieczne, szczególnie, jeżeli kryptoanalytyk ma dostęp do generatora i zna klucz nadzędny !

Generatory kluczy

- Przykładowymi elementami, które można wykorzystać do generowania liczb losowych są:
 - szum fal radiowych,
 - szum termiczny rezystorów i diod,
 - licznik Geigera-Müllera,
 - czas przybywania cząsteczek promieniowania kosmicznego,
 - temperatura rdzenia procesora.
- Jednym z pierwszych algorytmów generowania ciągów pseudolosowych była zaproponowana przez Johna von Neumanna w roku 1946 metoda średka kwadratu. Zasada jej działania opiera się na podniesieniu do kwadratu poprzedniej liczby i wybraniu jej środkowej cyfry. Analiza tej metody wykazała jednak, iż istnieje duże prawdopodobieństwo przejścia do ciągu o krótkim okresie

Ogólna postać generatorów

- Niech X_0, X_1, \dots będzie ciągiem liczb naturalnych wytwarzanym w oparciu o równanie:
- $$X_{n+1} = aX_n^e + b \pmod{m}$$
- $n=0, 1, \dots,$
- a, b, e, m są odpowiednio dobranymi liczbami naturalnymi

Ogólna postać generatorów

- Dla $e=1$ oraz $b=0$ otrzymujemy generator liniowy, dla którego m jest najczęściej potęgą liczby pierwszej lub podwojoną potęgą liczby pierwszej, natomiast a równe jest pierwiastkowi pierwotnemu modulo m . Dzięki tak dobranym parametrom generator liniowy uzyskuje maksymalny okres.
- $$X_{n+1} = aX_n + b \pmod{m}$$
- Dla $e=1$ oraz liczby b różnej od zera mamy do czynienia z generatorem afiniczny. Przy właściwym doborze parametrów a i b możemy uzyskać okres generatora równy m .

Ogólna postać generatorów

- $$X_{n+1} = aX_n^e + b \pmod{m}$$
- Dla $e=2$ oraz $b=0$ otrzymujemy generator BBS. W tym przypadku $m=pq$, gdzie p,q są różnymi liczbami pierwszymi dającymi w wyniku dzielenia przez 4 resztę 3.
- Dla $b=0$, biorąc dowolną liczbę e , ale przy ograniczeniu, że $\text{NWD}(e,m-1)=1$ i przy założeniu, że m jest iloczynem dwóch liczb pierwszych postaci $4k+3$, otrzymujemy generator RSA.

Liniowe generatory kongruencyjne

Przyjmijmy następujące oznaczenia:

X_n – n -ty wyraz ciągu,

a – mnożnik (stała),

b – przyrost (stała),

m – moduł (stała),

X_0 – ziarno generatora.

Zasada działania generatora:

$$X_n = (aX_{n-1} + b) \text{ mod } m.$$

Liniowe generatory kongruencyjne

- duża szybkość działania.
- ciąg generowany przez ten generator będzie miał maksymalny okres równy m .
- długość okresu uzależniony jest od doboru stałych a , b oraz m .
- generatorы są efektywne, a generowane ciągi cechują dobre właściwości statystyczne
- cała rodzina generatorów kongruencyjnych (liniowe, kwadratowe, sześciennne, obcięte liniowe generatorы) została złamana.

Nazwa	m	a	c
Numerical Recipes	2^{32}	1664525	1013904223
Borland C/C++	2^{32}	22695477	1
GNU Compiler Collection	2^{32}	69069	5
ANSI C	2^{32}	1103515245	12345
Borland Delphi, Virtual Pascal	2^{32}	134775813	1
Microsoft Visual/Quick C/C++	2^{32}	214013	2531011
ANSIC	2^{31}	1103515245	12345
MINSTD	$2^{31}-1$	16807	0
RANDU	2^{31}	65539	0
SIMSCRIPT	$2^{31}-1$	630360016	0
BCSLIB	2^{35}	30517578125	7261067085
BCPL	2^{32}	2147001325	715136305
URN12	2^{31}	452807053	0
APPLE	2^{35}	1220703125	0
Super-Duper	2^{32}	69069	0

Funkcja Eulera

$$\varphi: \mathbb{N} \rightarrow \mathbb{N}$$

$$\varphi(n) = \text{card} \{ b \leq n \mid \text{NWD}(b, n) = 1\}$$

$$\varphi(1)=1$$

$$\varphi(2)=1$$

$$1. \quad p \text{ jest pierwsza } a \geq 1 \Rightarrow \varphi(p^a) = p^{a-1}(p-1)$$

$$\varphi(3)=2$$

$$\bullet \quad p \text{ jest pierwsza, } a=1 \Rightarrow \varphi(p) = p-1$$

$$\varphi(4)=2$$

$$2. \quad p, q \text{ względnie pierwsze} \Rightarrow \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$$

$$\varphi(5)=4$$

$$\varphi(6)=3$$

$$\varphi(7)=6$$

$$\varphi(8)=4$$

$$\text{NWD}(k, n) = 1 \text{ i } kx \bmod n = 1 \Rightarrow x = k^{\varphi(n)-1} \bmod n$$

Generator BBS

- Algorytm ten oparty jest na obliczaniu reszt kwadratowych modulo n
- Bierzemy dwie duże liczby pierwsze p i q , które są kongruentne z 3 modulo 4. Iloczyn tych liczb n nazywamy liczbą Bluma. Jeśli n jest liczbą Bluma, to każda reszta kwadratowa (mod n) ma dokładnie 4 pierwiastki kwadratowe. W kolejnym kroku wybieramy inną losową liczbę całkowitą x , która jest względnie pierwsza z n . Obliczamy wartość początkową dla generatora:
 - $x_0 = x^2 \text{ mod } n.$

Generator BBS

Elementem o numerze i w generowanym ciągu pseudolosowym jest najmniej znaczący bit x_i , gdzie x_i otrzymujemy z następującego wzoru:

$$x_i = x_{i-1}^2 \bmod n.$$

Jeżeli chcemy obliczyć bit i ciągu pseudolosowego bez wyznaczania wszystkich iteracji wcześniejszych, możemy skorzystać z poniższego wzoru:

$$x_i = x_0 (2^i) \bmod ((p-1)(q-1)) \bmod n.$$

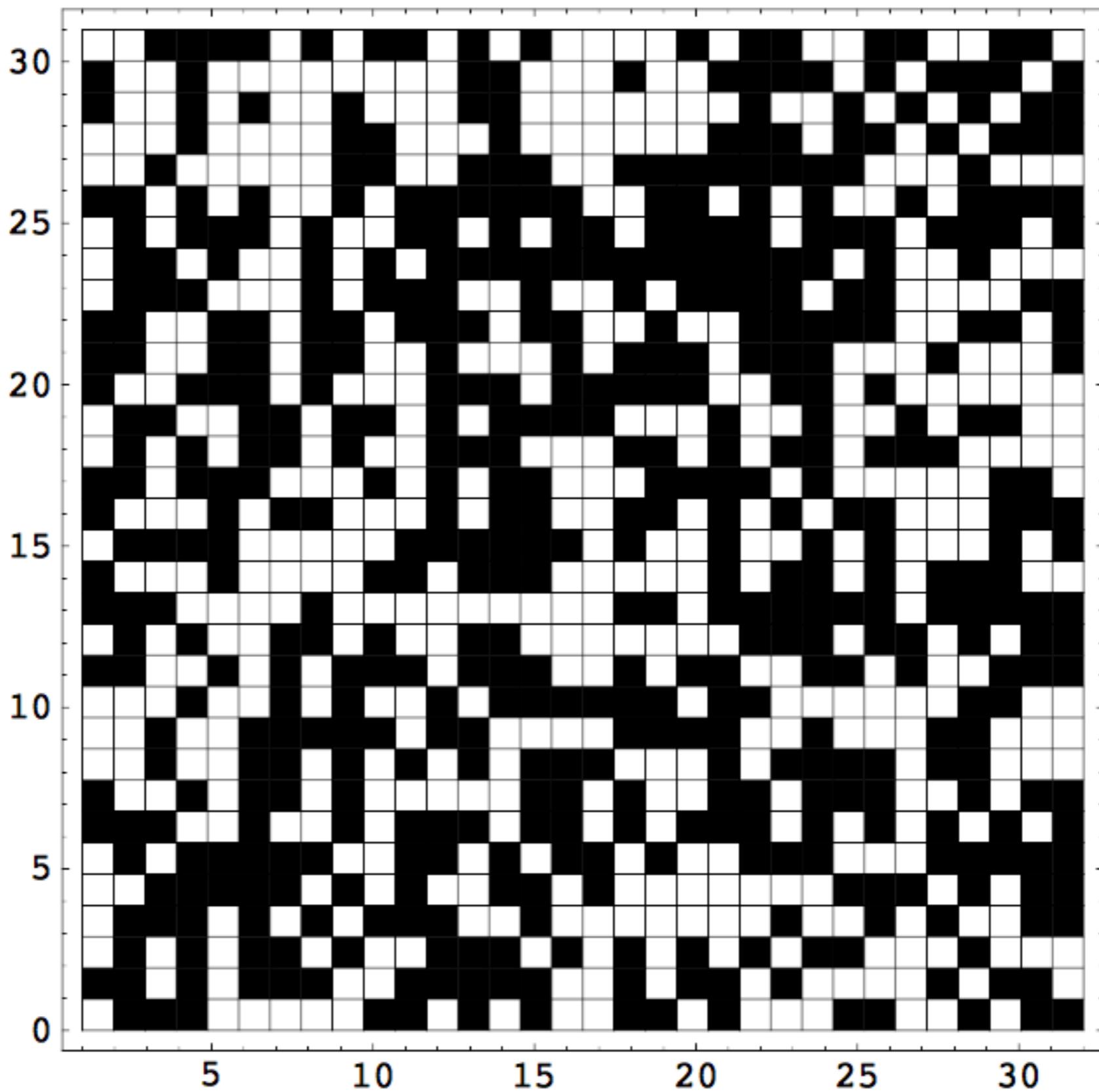
1. wybieramy dwie liczby pierwsze Bluma $p=7$, $q=11$, muszą spełniać warunek $p,q - 3 = 0 \bmod 4$
obliczamy liczbę Bluma $p^*q = m = 77$,

2. wybieramy losowe ziarno $s=53$,
 $1 <= s <= m-1$ i $\text{NWD}(m,s) = 1$ (są względnie pierwsze)

$$x_0 = s^2 \bmod m = 1$$

3. Ostatnie bity kolejnych liczb tworzą ciąg pseudolosowy

i	0	1	2	3	...
x_i	37	60	58	53	...
u_i	1	0	0	1	...



Generator RSA

Algorytm działa na następującej zasadzie:

p, q – liczby pierwsze,

N – iloczyn liczb p i q ,

e – liczba całkowita względnie pierwsza z $(p-1)(q-1)$,

x_0 – losowa wartość początkowa, mniejsza niż N ,

$$x_{i+1} = x_i^e \bmod N.$$

Najmniej znaczący bit liczby x_i tworzy ciąg pseudolosowy.

Generator uważa się za bezpieczny, jeżeli liczba N jest dostatecznie duża.

Algorytm potęgowania

WEJŚCIE: $x, a, n \in \mathbb{N}$.

WYJŚCIE: $y = x^a \bmod n$.

METODA:

1. Przedstaw liczbę a w postaci dwójkowej:
 $a := (a_r, a_{r-1}, \dots, a_0)_2, a_r = 1;$
2. $y := 1;$
3. **for** $i := r$ **downto** 0 **do**
 $y := y \cdot y \bmod n;$
if $a_i = 1$ **then** $y := y \cdot x \bmod n;$
4. **return** (y);

Ciągi statystycznie losowe - testy

- ❖ Testy statystyczne FIPS 140-2
- ❖ **Test pojedynczych bitów** dla ciągów o długości 20 000 bitów:
 - ❖ $9725 < n(1) < 10275$
- ❖ **Test serii:** seria w danym ciągu to maksymalny podciąg następujących po sobie zer lub jedynek

Długość serii	Przedział
1	2315-2685
2	1114-1386
3	527-723
4	240-384
5	103-209
6 i więcej	103-209

Testy cd.

- ❖ Test długiej serii - serię zer albo jedynek nazywamy długą, jeśli ma długość 26 lub więcej. Test zakończy się sukcesem jeśli w próbce o długości 20 000 bitów nie ma takiej serii.
- ❖ Test pokerowy:
 - ❖ ciąg o długości 20 000 bitów należy podzielić na 5000 segmentów.
 - ❖ należy policzyć i zapamiętać liczbę wystąpień każdej z możliwych 16 - 4 bitowych wartości

Test pokerowy cd.

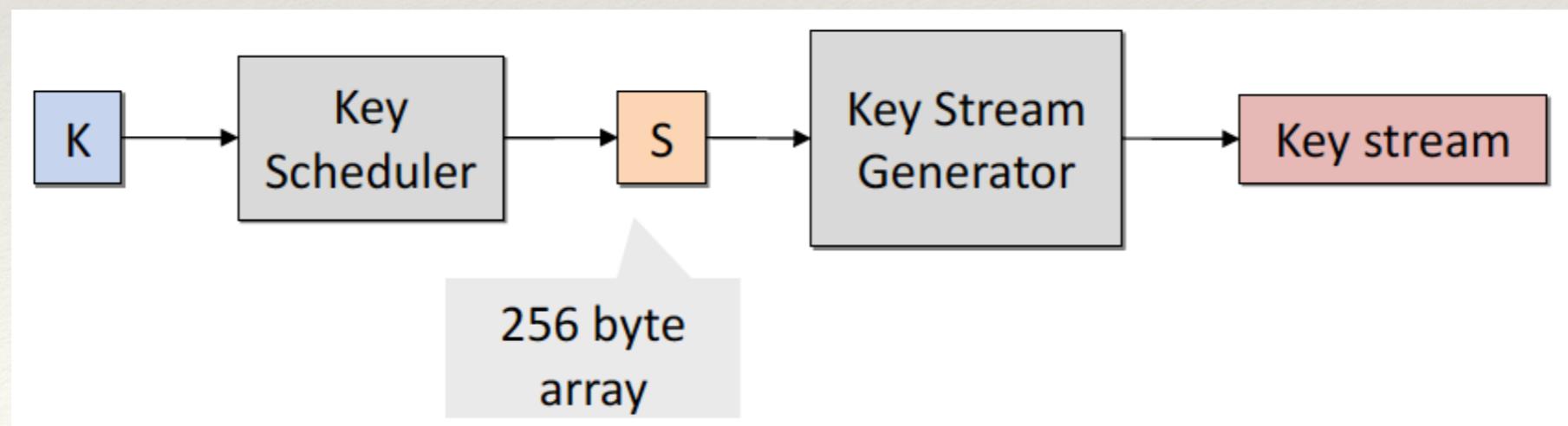
- ❖ Niech $s(i)$, gdzie $0 \leq i \leq 15$ oznacza liczbę wystąpień segmentów o wartości dziesiątnej i .
- ❖ Test zakończy się sukcesem, jeśli $2,16 < X < 46,17$,
- ❖ gdzie X :

$$x = \frac{16}{5000} \cdot \sum_{i=0}^{15} s(i)^2 - 5000$$



RC4

- ❖ zaprojektowany przez Rona Rivesta w 1987
- ❖ prosty, szybki, szeroko wykorzystywany
- ❖ SSL/TLS , WLAN
- ❖ Struktura szyfratora: 2 algorytmy: wyznaczania klucza dla generatora, algorytm generowania ciągu losowego



RC4 Key Scheduler

- ❖ RC4, używany jest często wraz z wektorem IV (losowym lub jako wartość licznika), dołączanym do klucza
- ❖ RC4 jest niewystarczająco losowy! jego pierwszy bajt wygenerowanego ciągu zależy tylko od 3 komórek tablicy S, RSA sugeruje żeby pominać pierwszych 256 bitów wygenerowanego ciągu

Fluhrer-Martin-Shamir attack

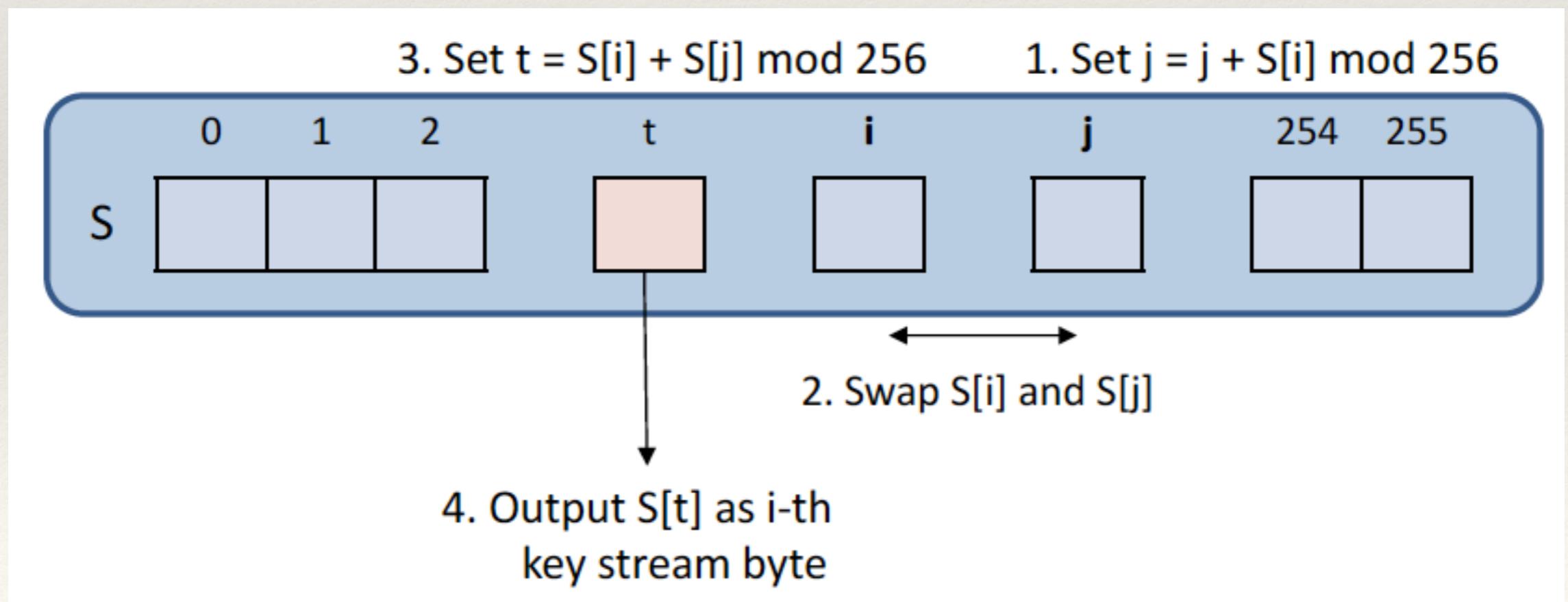
```
Divide key K into L bytes  
for i = 0 to 255 do  
    S[i] := i  
    j := 0  
    for i = 0 to 255 do  
        j := (j+S[i]+K[i mod L]) mod 256  
        swap(S[i],S[j])
```

Key can be any length
up to 2048 bits, i.e. at
most 256 byte

Generate initial
permutation
from key K

RC4 Generator Strumienia bajtów

- ❖ Key scheduler wypełnia tablicę bajtów S
- ❖ i-ty bajt strumienia jest generowany w pętli, gdzie na początku $i=j=0$, iterowanej po i:



Generator strumienia klucza

- ❖ W każdej rundzie pętli wyznaczany jest bajt strumienia klucza

```
i = j := 0
loop
    i := (i+1) mod 256
    j := (j+S[i]) mod 256
    swap(S[i], S[j])
    output S[(S[i]+S[j]) mod 256]
end loop
```

Bezpieczeństwo RC4

- ❖ 2001 - Fluhrerm Shamir, Mantin - opisali słabość w algorytmie Key scheduling
 - ❖ złamanie protokołu WEP - 2001
- ❖ 2013 - AlFardan - atak ze znanym tekstem zaszyfrowanym (odzyskanie tekstu jawnego) na TLS (jeśli używa RC4)
- ❖ 2015 - Vanhoef - atak ze znanym tekstem zaszyfrowanym (odzyskanie tekstu jawnego) na używany RC4 w WPA i TLS (HTTPs)
- ❖ 2015 - Van der Merwe - atak na odzyskanie hasła w TLS używającym RC4
- ❖ RC4 już nie powinien być używany!

Podsumowując

- ❖ Pseudo Random Number Generators (**PRNGs**) używane są w kryptografii do:
 - ❖ generowania klucza symetrycznego
 - ❖ generowania klucza asymetrycznego lub parametrów w algorytmach generowania takiego klucza
- ❖ PRNGs używają **generatorów pseudolosowych bitów** - które generują po jednym losowym bicie
- ❖ Niektóre standardy używają pseudolosowych funkcji - Pseudo Random Function (PRF) zamiast PRNG

Kryptograficznie bezpieczny PRNG

- ❖ PRNG przechodzi pozytywnie **test następnego bitu** jeśli nie istnieje wielomianowy algorytm, który pozwalałby na podstawie n-bitów wejściowych obliczyć kolejny bit $n+1$ z prawdopodobieństwem większym niż 0,5.
- ❖ Algorytm Berlekampa-Masseya - wyznaczający wielomian, na podstawie n-bitów, generujący ten ciąg bitów
- ❖ w budowie kryptograficznie bezpiecznych PRNG używa się:
 - ❖ funkcji skrótu z kluczem
 - ❖ szyfrów blokowych
 - ❖ algorytmów z teorii liczb

do poczytania..

- ❖ Randomness Recommendations for Security: RFC 1750