Математична криптологія

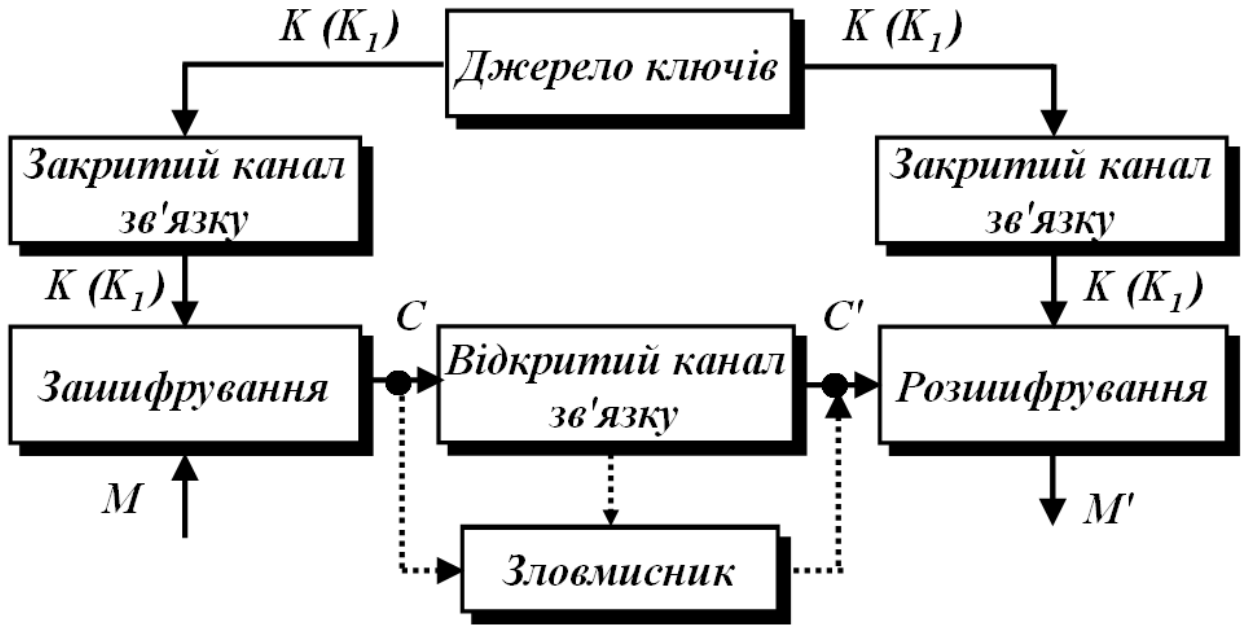
Шифр одноразового блокноту
Потокові шифри

Основи теорії секретного зв’язку (К. Шеннона)

Під *системою засекреченого зв'язку* будемо розуміти систему передачі даних у якій зміст інформації, що передається, ховається за допомогою криптографічних перетворень. При цьому сам факт передачі інформації не приховується. В основі кожної системи засекреченого зв'язку — використання алгоритмів шифрування як основного засобу збереження конфіденційності.

Шеннон (1949, Communication Theory and Secrecy Systems) розглянув питання теоретичної й практичної секретності. Для визначення теоретичної секретності Шеннон сформулював такі питання:

- Наскільки стійкою є система, якщо криптографічний аналітик супротивника не обмежений у часі та має всі необхідні засоби для аналізу криптограм?
- Чи має шифрований текст (криптограма) єдиний розв'язок?
- Який об'єм зашифрованого тексту необхідно перехопити криптографічному аналітику, щоб розв'язок став єдиним?



Основи теорії секретного зв’язку (К. Шеннона)

Для відповіді на ці питання Шеннон увів поняття досконалої секретності за допомогою такої умови: для усіх С апостеріорна ймовірність дорівнює апріорній ймовірності, тобто перехоплення зашифрованого повідомлення не дає криптографічному аналітику супротивника ніякої інформації. За теоремою Байєса:

$$P(M, C) = P(M)\,P(C|M) = P(C)\,P(M|C) \qquad (1)$$

де $P(M)$ — апріорна ймовірність повідомлення M; $P(C|M)$ — умовна ймовірність криптограми C за умови, що обрано повідомлення M, тобто сума ймовірностей усіх тих ключів, які приводять повідомлення M у криптограму C; $P(C)$ — ймовірність отримання криптограми C; $P(M|C)$ — апостеріорна ймовірність повідомлення M за умови, що перехоплено криптограму C.

Із виразу (1) випливає, що:

$$P(M|C) = \frac{P(M)P(C|M)}{P(C)}. \qquad (2)$$

Для досконалої секретності криптографічних системи значення $P(M|C)$ і $P(M)$ повинні бути рівні для усіх M і C, тобто $P(M|C) = \mathrm{P}(M)$. Якщо $P(C|M) = P(C)$, то з (2) випливає, що $P(M|C) = P(M)$ і система цілком таємна.

Необхідна й достатня умова для досконалої секретності полягає у тому, що $P(C|M) = P(C)$ для усіх M і C, тобто $P(C|M)$ не повинне залежати від M.

Іншими словами, повна ймовірність усіх ключів, що переводять повідомлення M_i у дане зашифроване повідомлення C, дорівнює повній ймовірність усіх ключів, що переводять повідомлення M_j у те саме зашифроване повідомлення C для усіх M_i, M_j і C.

Безумовна стійкість

Definition of perfect security
Let $\varepsilon = (E,D)$ be a Shannon cipher defined over (K,M,C). Consider a probabilistic experiment in which the random variable **k** is uniformly distributed over K. If for all $m_0, m_1 \in M$, and all $c \in C$, we have:

$$Pr[E(k,m_0) = c] = Pr[E(\mathbf{k},m_1) = c]$$

then we say that ε is a perfectly secure Shannon cipher.

Let $\varepsilon = (E,D)$ be a Shannon cipher defined over (K,M,C). Then the following statements are equivalent:

i) ε is perfectly secure;

ii) For every ciphertext $c \in C$ there exists Nc (possibly depending on c) such that for all messages $m \in M$, we have

$$|\{k \in K : E(k,m) = c\}| = Nc$$

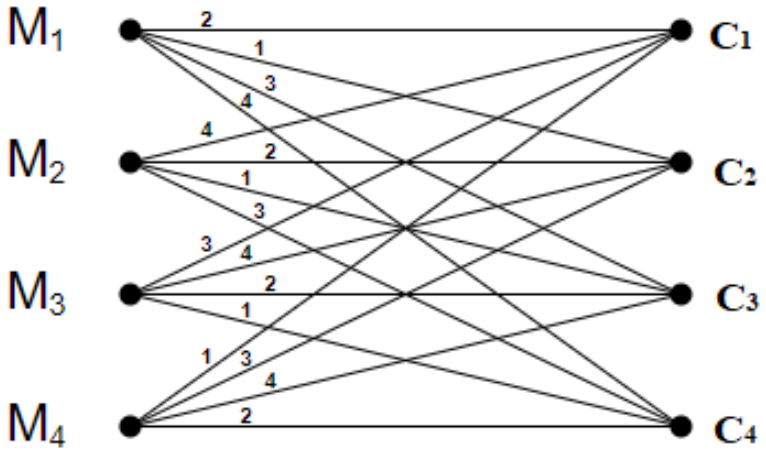
iii) If the random variable **k** is uniformly distributed over K, then each of the random variables $E(\mathbf{k},m)$ for $m \in M$ has the same distribution

Theorem 2.5 (Shannon's theorem). *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. If $\mathcal{E}$ is perfectly secure, then $|\mathcal{K}| \geq |\mathcal{M}|$.*

Безумовна стійкість. Приклад

Криптосистема може бути безумовно стійкою, коли кількість повідомлень точно дорівнює кількості ключів. Нехай M_i пронумеровані числами від *1* до *n*, так само як і C_i, і нехай використовуються *n* ключів. Тоді $E_i\ (M_j\ ) = C_s$, де $s = (i + j)\ mod\ n$. У цьому випадку справедливою є рівність:

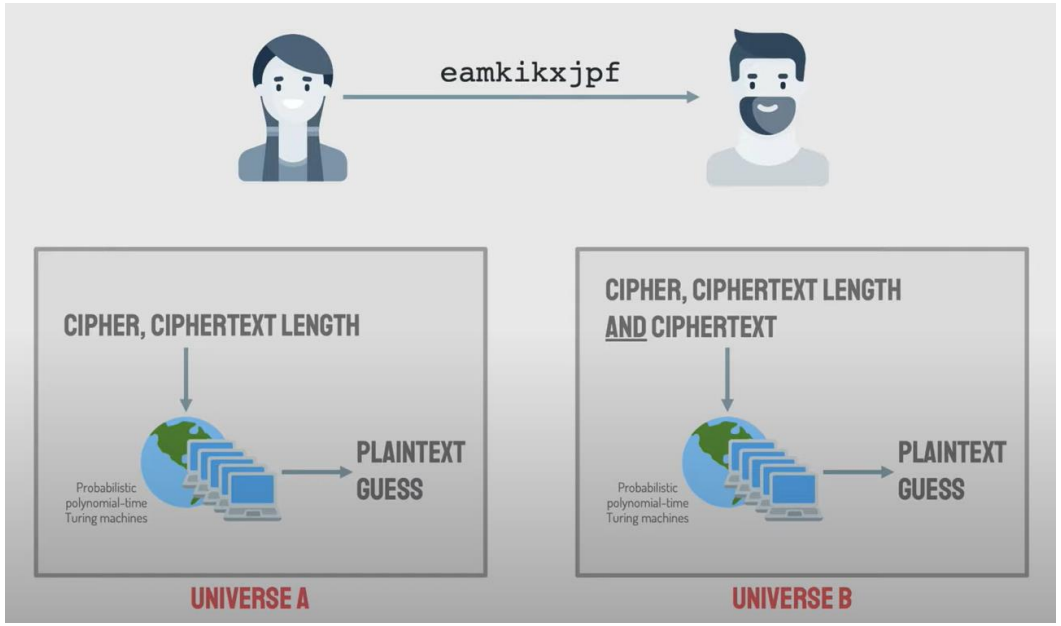
$p\ (M|C) = 1/n = p\ (C)$ і система є цілком секретною.

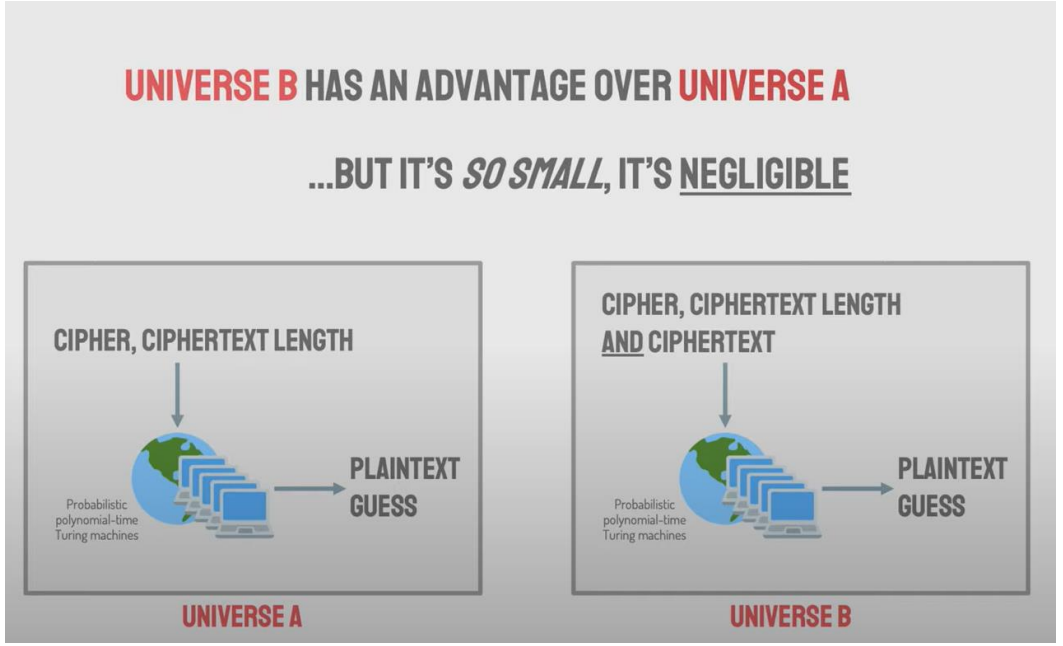


Семантична стійкість (semantic security)

Семантично стійкими є криптосистеми, в котрих з шифротексту можна витягнути лише незначну інформацію про відкритий текст. Семантична стійкість є аналогом концепції безумовної стійкості Шеннона. Безумовна стійкість означає, що шифротекст не розкриває жодної інформації про відкритий текст, тоді як семантична стійкість передбачає, що будь-яку розкриту інформацію неможливо витягнути виконуємим чином.

Поняття семантичної стійкості було вперше висунуто Голдвассером та Мікалі в 1982 році. Згодом Голдвассер та Мікалі продемонстрували, що семантична стійкість еквівалентна іншому визначенню безпеки, а саме нерозрізненості шифротексту під час атаки з підібраним відкритим текстом.





Гра підслуховування (Eavesdropping game)

LET'S PLAY A GAME

1. I pick and send you two "challenge messages", M1, M2

2. You flip a coin: heads you pick M1, tails you pick M2. You encrypt it and send me the "challenge ciphertext"

3. I guess which message you picked. If I'm right, I win

IMPLICATIONS OF THE "EAV" GAME

1. If I guess randomly, I win 50% of the time

2. If I can *distinguish* ciphertexts, I can win more than 50% of the time

3. If I can't distinguish ciphertexts, I can't do better than random guessing

$$\text{Adv} = \left|\Pr(\text{guessing correctly}) - \frac{1}{2}\right|$$

ADVANTAGE IS HOW FAR OFF 50% MY SUCCESS RATE IS

IS MY ADVANTAGE EVER ZERO IN THE "EAV" GAME?

No. I get the ciphertext, so I can try to brute-force decrypt it, and will succeed with some non-zero probability

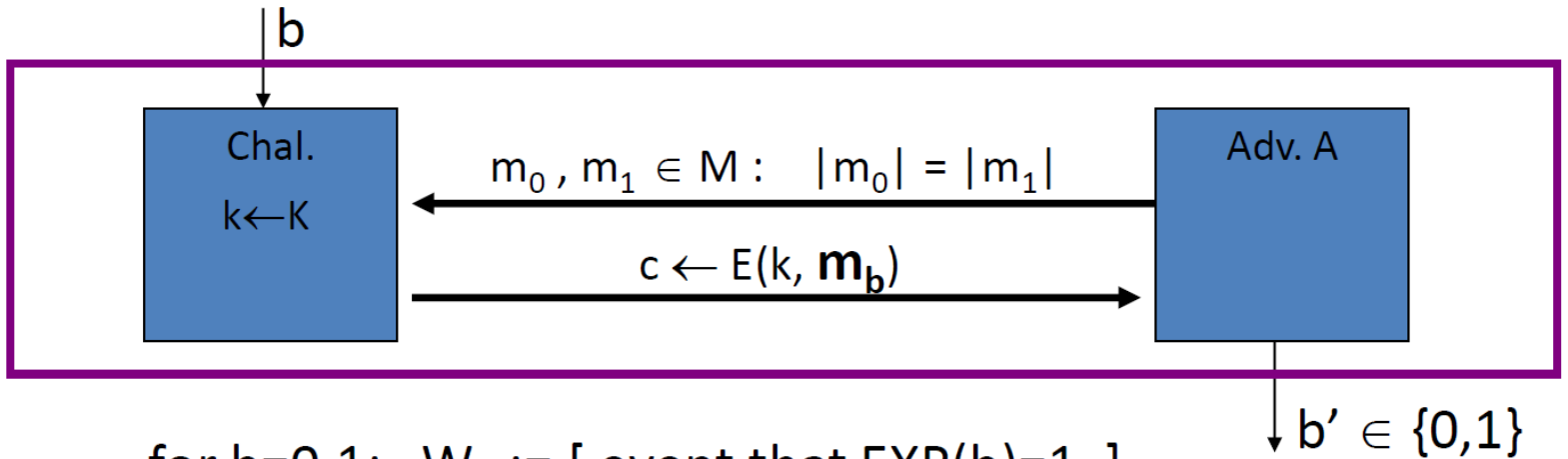
Незначна величина (negligible value)

- In practice: ε is a scalar and
  - ε non-neg: $\varepsilon \geq 1/2^{30}$ (likely to happen over 1GB of data)
  - ε negligible: $\varepsilon \leq 1/2^{80}$ (won't happen over life of key)

- In theory: ε is a function $\varepsilon: Z^{\geq 0} \longrightarrow R^{\geq 0}$ and
  - ε non-neg: $\exists d: \varepsilon(\lambda) \geq 1/\lambda^d$ inf. often ($\varepsilon \geq$ 1/poly, for many λ)
  - ε negligible: $\forall d, \lambda \geq \lambda_d: \varepsilon(\lambda) \leq 1/\lambda^d$ ($\varepsilon \leq$ 1/poly, for large λ)

Семантична стійкість

For b=0,1 define experiments EXP(0) and EXP(1) as:



for b=0,1: W_b := [event that EXP(b)=1]

$Adv_{SS}[A,E] := \left| \Pr[W_0] - \Pr[W_1] \right| \in [0,1]$

Definition 2.2 (semantic security). *A cipher $\mathcal{E}$ is semantically secure if for all efficient adversaries $\mathcal{A}$, the value* $\mathrm{SSadv}[\mathcal{A}, \mathcal{E}]$ *is negligible.*

⇒ for all explicit $m_0, m_1 \in M$: $\{ E(k,m_0) \} \approx_p \{ E(k,m_1) \}$

Типи атак

IND-EAV SECURITY

A cipher is indistinguishable under eavesdropping (IND-EAV secure) if there exists no probabilistic polynomial-time Turing machine that can win the EAV game with a non-negligible advantage

WE HAVE A NEGLIGIBLE ADVANTAGE OF WINNING THE "EAV" GAME IF

$$\text{Adv} \leq \varepsilon(\lambda)$$

IND-CPA SECURITY

The chosen-plaintext attack (CPA) game runs exactly the same as the eavesdropping game except the guesser gets an additional "power:" the ability to make encryption queries under the same key used to create the challenge ciphertext

IND-CCA2 SECURITY

The *adaptive* chosen-ciphertext attack (CCA2) game runs exactly the same as the CCA1 game except the guesser gets an additional "power:" the ability to make decryption queries after the challenge ciphertext is received*

IND-CCA1 SECURITY

The chosen-ciphertext attack (CCA1) game runs exactly the same as the CPA game except the guesser gets an additional "power:" the ability to make decryption queries under the same key as the challenge ciphertext, until the challenge ciphertext is received

Приклад – Шифр Віженера

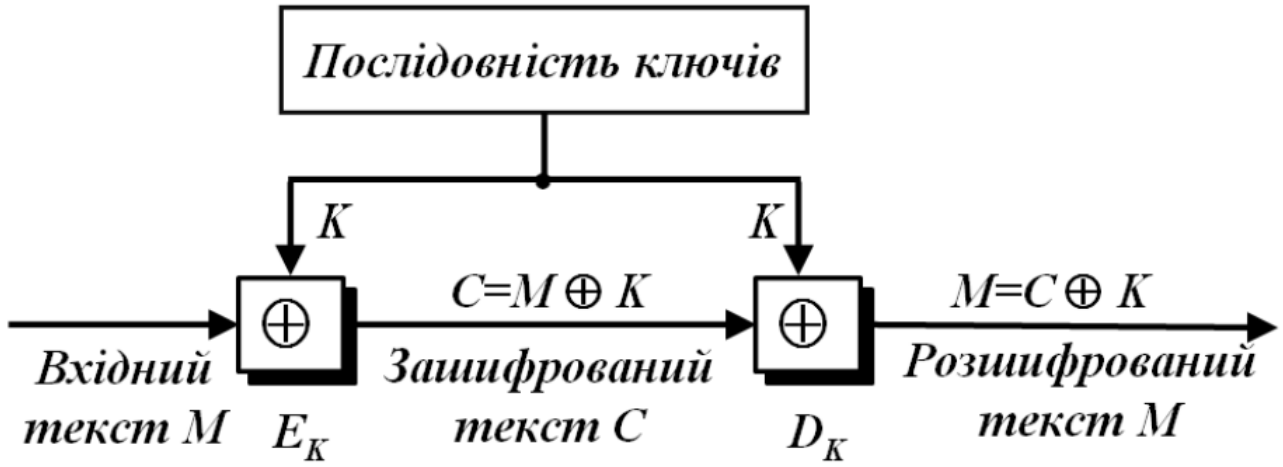
Довести що шифр Віженера не є семантично стійким

Шифр одноразового блокноту

Одна із цілей криптографії — ідеальна секретність. Майже всі застосовувані на практиці шифри характеризуються як умовно надійні, оскільки вони можуть бути розкриті за наявності необмежених обчислювальних можливостей. Абсолютно надійні шифри не можна зруйнувати навіть за використання необмежених обчислювальних можливостей. Існує єдиний такий шифр, що винайдений 1917 р. американцями Г. Вернамом і Д. Моборном - шифр одноразового блокноту:

$$c_i = m_i \oplus k_i;\ M{=}C = \{0,1\}^n;\ K{=}\{0,1\}^n$$

$$m_i = c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i\ , i = 1,2,3, \dots, n;$$



$$\begin{array}{l} 00000100000001000100000010001 \\ \oplus\ 00110111010110001001100011101 0 \\ \hline 00110011010111001101100010101 1 \end{array}$$

Шифр одноразового блокноту

„**XOR** is an involution so the function's inverse is the function itself"

EXAMPLE

Originally one time pad algorithm used **XOR** operation so first we consider the binary representation

→ we find the **ASCII** value for every letter in the text

→ then we convert the decimal value into binary

output is **0** or **1**
with **50%** probability

x	**y**	x **XOR** y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

So we want to shift every letter in the plaintext which means addition
Addition is the same as bitwise **XOR** (if there are no carry bits)

For example: let's use **XOR** to add **16** and **10**

$$\mathbf{00010000 = 0x2^0 + 0x2^1 + 0x2^2 + 0x2^3 + 1x2^4 + 0x2^5 + 0x2^6 + 0x2^7 = 16}$$

$$\mathbf{00001010 = 0x2^0 + 1x2^1 + 0x2^2 + 1x2^3 + 0x2^4 + 0x2^5 + 0x2^6 + 0x2^7 = 10}$$

00010000 | 00001010 = 00011010 = 26

Шифр одноразового блокноту
One Time Pad (OTP)

EXAMPLE

Plaintext: **HELLO**

Key: **11001000001011101111011011000000**

1.) let's convert the **ASCII** values of the letters into **binary**

72 = 01001000

69 = 01000101

76 = 01001100

79 = 01001111

72 69 76 76 79

H E L L O = 0100100001000101010011000100111 1

```
  01001000010001010100110001001111
XOR 11001000001011101111011011000000
-------------------------------------
  10000000011010111011101010001111
```

Надійність шифру одноразового блокноту

Дослідження К. Шеннона (Information Theoretic Security, 1949) показали, що ідеальна секретність може бути досягнута за виконання таких правил: $\forall\, \boldsymbol{m_0, m_1} \in \boldsymbol{M}, \forall \boldsymbol{c} \in \boldsymbol{C}: \boldsymbol{len(m_0) = len(m_1)},\ \boldsymbol{P[E(k, m_0) = c] = P[E(k, m_1) = c]}$ де $k \in K$.

- ключ для шифрування вибирається випадково;
- довжина ключа повинна дорівнювати довжині тексту, що шифрують ($|K| \geq |M|$);
- ключ повинен використовуватися тільки один раз.

Lemma: OTP has perfect secrecy.

Proof:

$$\forall m, c: \quad \Pr_{K}[E(K,m)=c] = \frac{\#\text{keys } K \in \mathcal{K} \text{ s.t. } E(K,m)=c}{|\mathcal{K}|}$$

So: if $\forall m, c: \#\{K \in \mathcal{K} : E(K,m)=c\} = \text{const.}$

$\Rightarrow$ cipher has perfect secrecy

For OTP: $\forall m, c$: if $E(K,m)=c$

$\Rightarrow K \oplus m = c \Rightarrow K = m \oplus c$

$\Rightarrow \boxed{\#\{K \in \mathcal{K} : E(K,m)=c\} = 1}$

$\Rightarrow$ OTP has perfect secrecy ∎

Вразливість Two Time Pad

Two Time Pad is Insecure

Two Time Pad:

$c_1 = m_1 \oplus k$

$c_2 = m_2 \oplus k$

Enough redundancy in ASCII (and english) that $m_1 \oplus m_2$ is enough to know m_1 and m_2

Eavesdropper gets c_1 and c_2.
What is the problem?

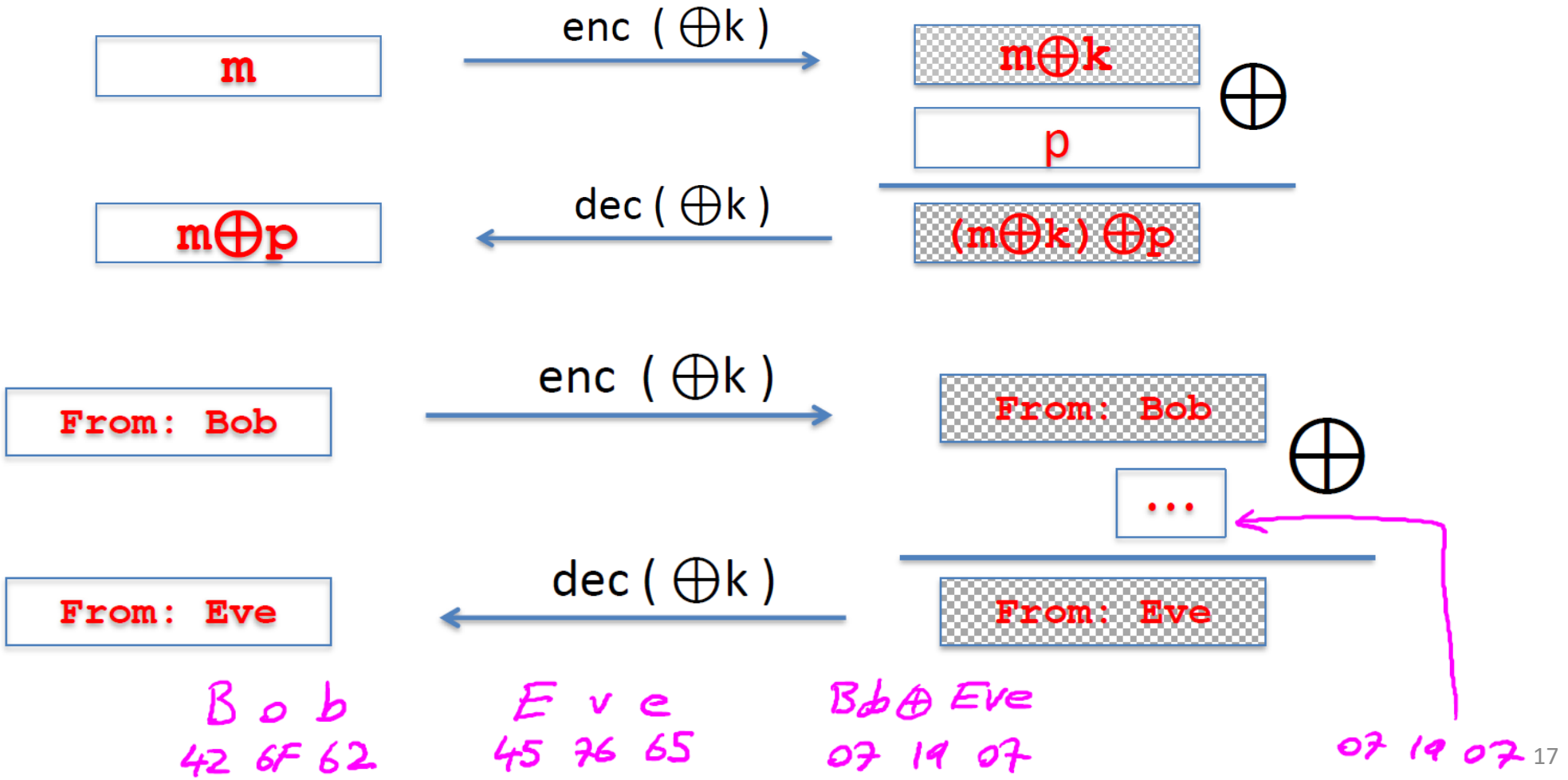
$c_1 \oplus c_2 = m_1 \oplus m_2$

Real World Examples

- Project Venona (~1942-1945)
  - Russians used same OTP twice → break by American and British cryptographers

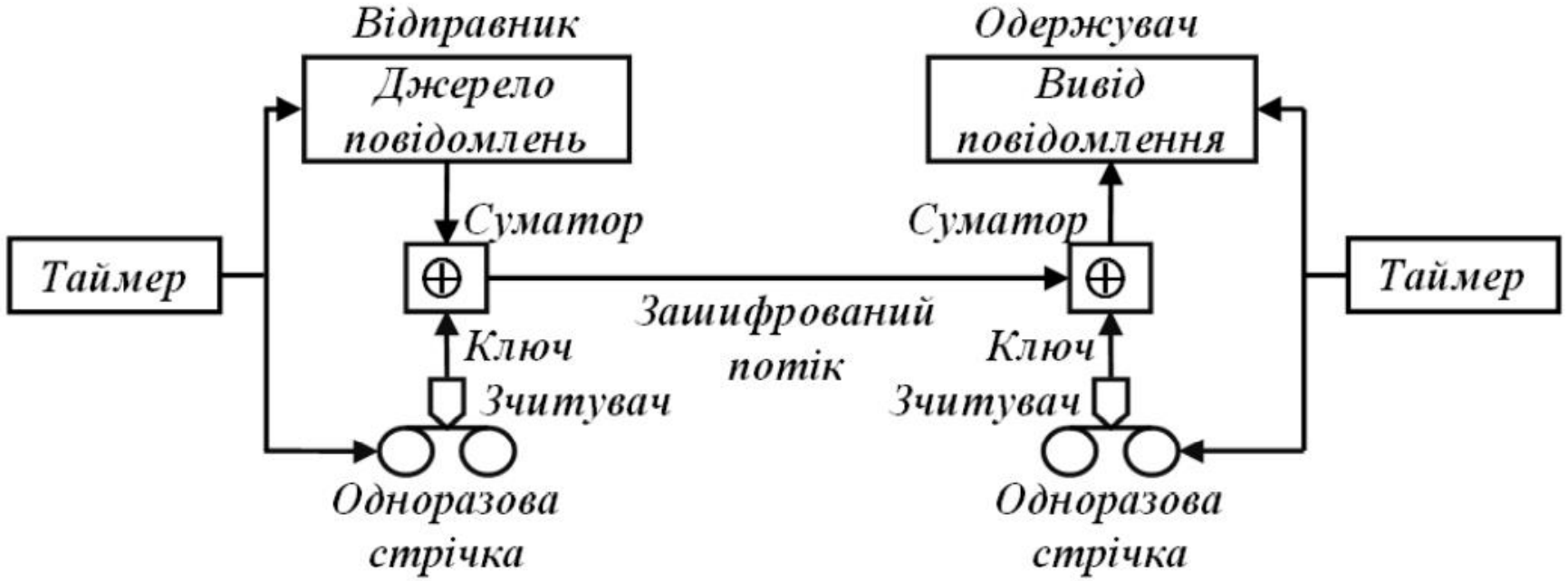
Вразливості шифру одноразового блокноту

Модифікація шифротексту зловмисником не може бути виявлена та має передбачуваний вплив (OTP is malleable)



Система Вернама

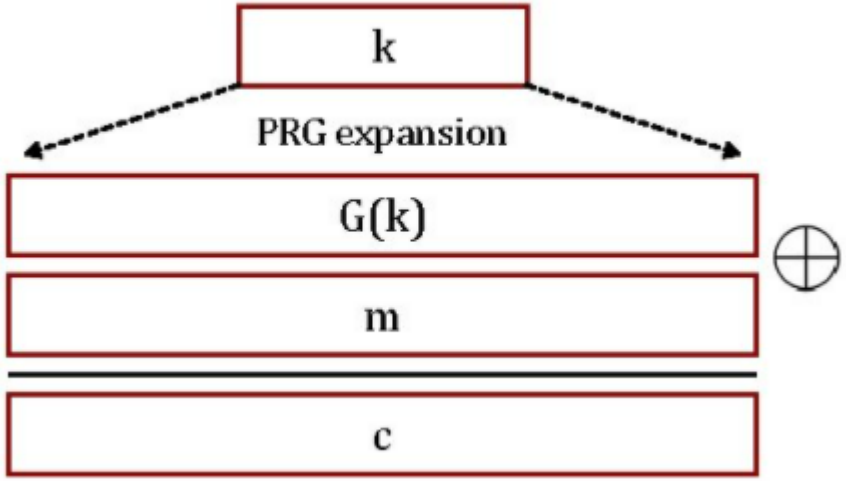
У реальних системах спочатку готують дві однакові стрічки з випадковими цифрами ключа. Одна залишається у відправника, а інша — передається “неперехоплюючим” способом, наприклад, кур'єром з охороною, законному одержувачу. Коли відправник хоче передати повідомлення, він спочатку перетворює його у двійкову форму й поміщає в пристрій, який до кожної цифри повідомлення додає за модулем 2 цифри, визначені в ключовій стрічці. На при ймаючій стороні зашифроване повідомлення записується та пропускається через машину, схожу на пристрій, який використовується для зашифрування. Ключова стрічка повинна просуватися абсолютно синхронно із своїм дублікатом, що використовується для зашифрування.

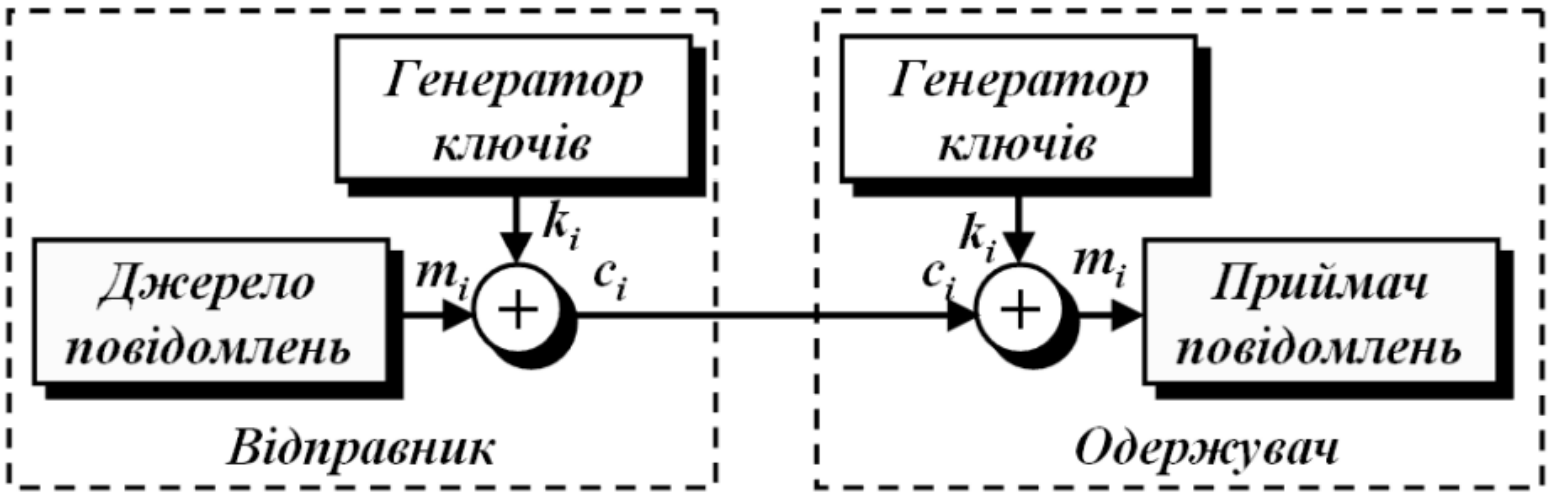


Потокові шифри

PRG можуть використовуватися як генератори ключів у потокових шифрах. Метою використання PRG є отримання нескінченного ключового слова, за використання відносно малої довжини самого ключа. PRG створює послідовність бітів, схожу на випадкову.

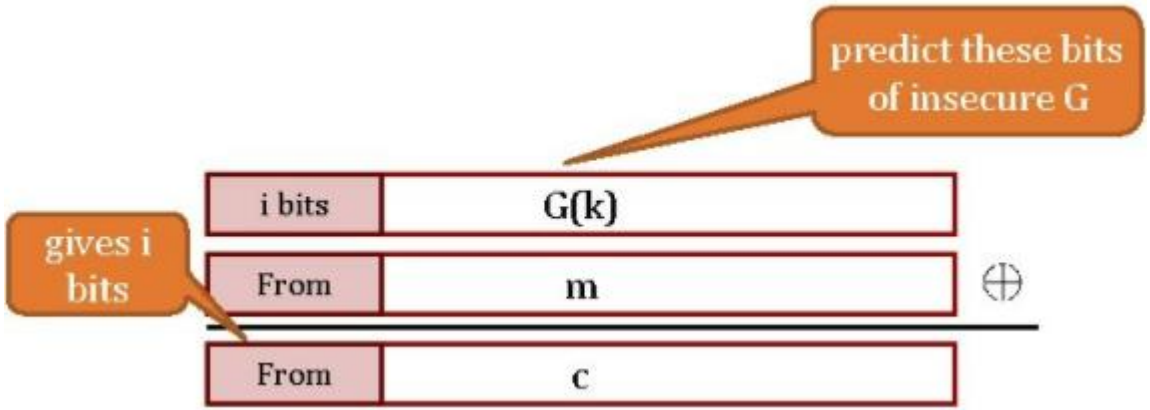
$\boldsymbol{G}: \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{s}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{n}}, \boldsymbol{n} \gg \boldsymbol{s}$;
$\boldsymbol{C} = \boldsymbol{E}(\boldsymbol{k}, \boldsymbol{m}) := \boldsymbol{m} \oplus \boldsymbol{G}(\boldsymbol{k})$; $\boldsymbol{D}(\boldsymbol{k}, \boldsymbol{c}) := \boldsymbol{c} \oplus \boldsymbol{G}(\boldsymbol{k})$.





PRG повинен бути непередбачуваним

Припустимо PRG передбачуваний: $\exists\, \boldsymbol{i}, \boldsymbol{G}(\mathbf{k})|_{1,\dots,i} \xrightarrow{Alg} \boldsymbol{G}(\boldsymbol{k})|_{i+1,\dots,n}$



PRG передбачуваний якщо $\exists\, \boldsymbol{Alg}, \boldsymbol{P}[\boldsymbol{Alg}(\boldsymbol{G}(\boldsymbol{k})|_{1,\dots,i}) = \boldsymbol{G}(\boldsymbol{k})|_{i+1}] > \frac{1}{2} + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \geq \frac{1}{2^{30}}$

Приклад вразливого PRG – Middle-Square Method

MIDDLE-SQUARE METHOD

It was invented by
John von Neumann in **1949**

The input of the algorithm is a **seed**: because computers are programmed to execute well-defined operations, it's impossible to generate random numbers

→ but we can define algorithms to mimic randomness
~ these are the pseudo-random numbers

→ the initial position (this is the **seed**) determines the sequence itself

The seed should be a truly random number: measurement of noise or current time in milliseconds

THE SEED IS THE INPUT OF A SIMPLE CALCULATION

ALGORITHM:
1.) multiply the seed by itself
2.) get the middle of the result
3.) the result is the seed in the next iteration

The randomness of the sequence depends on the randomness of the seed exclusively

Приклад вразливого PRG – Middle-Square Method

MIDDLE-SQUARE METHOD

ALGORITHM:

1.) multiply the seed by itself
2.) get the middle of the result
3.) the result is the seed in the next iteration

seed: **152**

152 x 152 = 2**310**4

310

seed: **310**

310 x 310= 9**610**0

310610

seed: **610**

610 x 610= 37**210**0

310610210

Приклад вразливого PRG – Middle-Square Method

It is a **pseudo-random number sequence**: so first problem is that if we know the initial seed, we can reproduce the sequence

→ if the algorithm reaches a seed it previously used then the sequence keeps repeating itself

→ this is called the **period**: the length before a pseudo-random sequence repeats

→ the period depends on the initial **seed** exclusively

2 digits seed: algorithm uses at most **100** digits before reusing the seed
3 digits seed: algorithm uses at most **1000** digits before reusing the seed

.
.

N digits seed: algorithm uses **10** digits before reusing the seed

Приклад вразливого PRG – Linear Congruential Gen

$$X_{n+1} = (a X_n + c) \bmod m$$

→ as usual we have to define a **seed** which is the X_0

→ the values of the parameters **a, c** and **m** determine the **period**

glibc random():

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

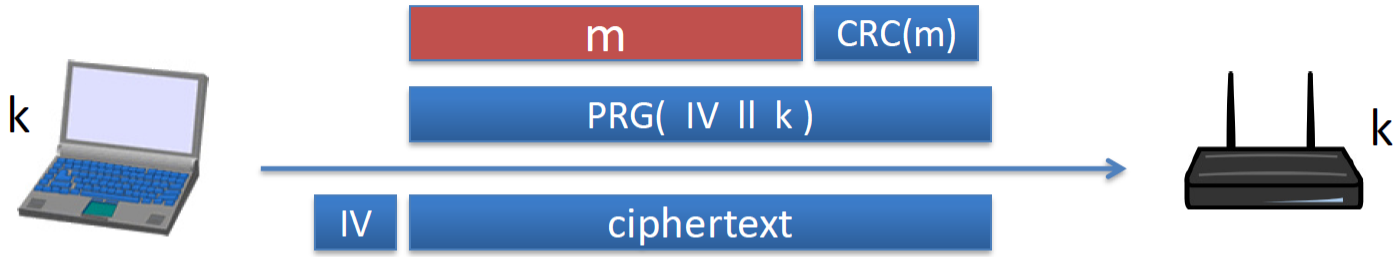
output $r[i] >> 1$

Ніколи не використовуйте random( ) для криптографії

WEP 802.11b

Wired Equivalent Privacy (WEP) - найстаріший стандарт захисту бездротового трафіку, заснований на алгоритмі потокового шифрування RC4 (з використанням загального секретного ключа). Існують варіанти з довжиною ключа 64, 128 і 256 бітів.

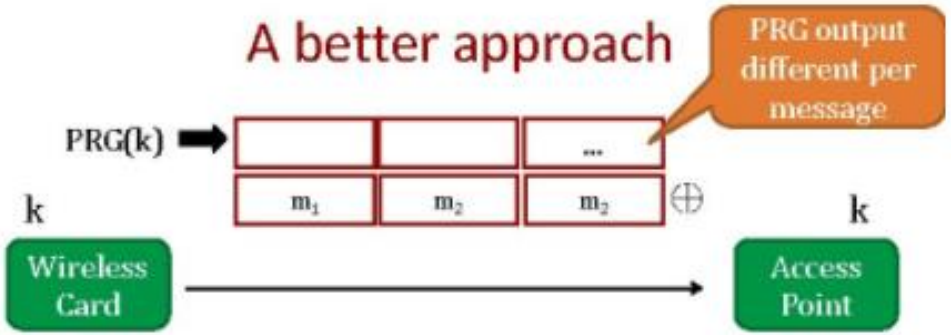
802.11b WEP:



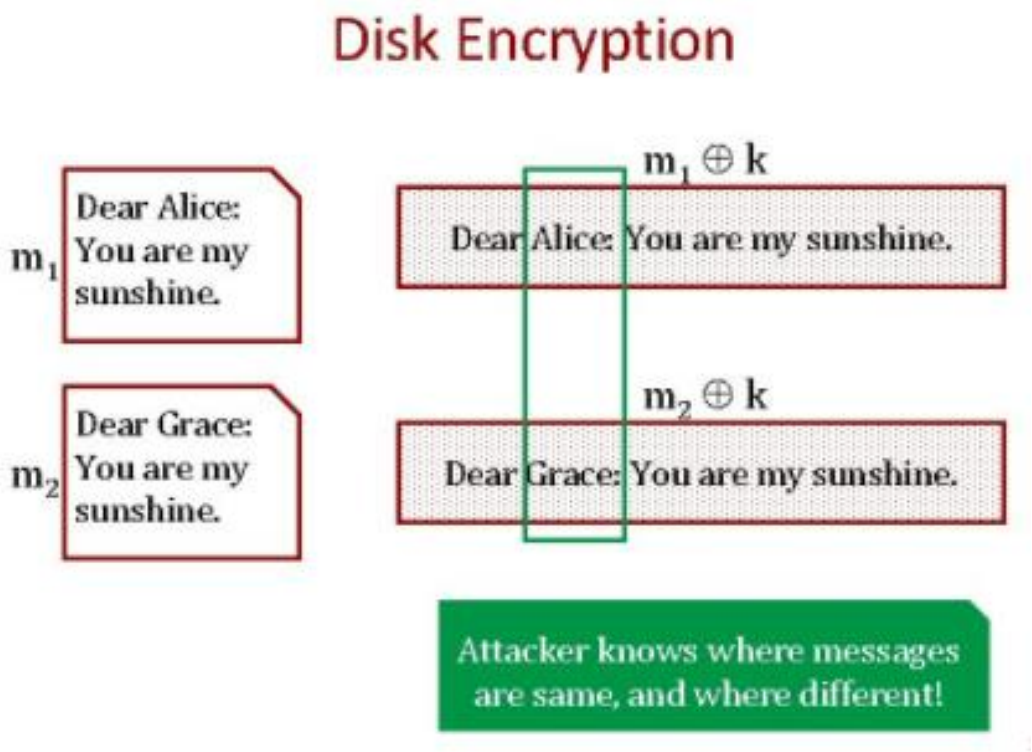
Length of IV: 24 bits

- Repeated IV after $2^{24} \approx 16M$ frames
- On some 802.11 cards: IV resets to 0 after power cycle

WPA (англ. Wi-Fi Protected Access) — один з протоколів безпеки для захисту бездротових мереж. Створений для заміни застарілого протоколу WEP. Заснований на TKIP (англ. Temporary Key Integrity Protocol — протокол тимчасової цілісності ключів), який ефективно бореться з проблемою, що лежить в основі вразливості WEP, — повторного використання ключів шифрування.



Шифрування диску



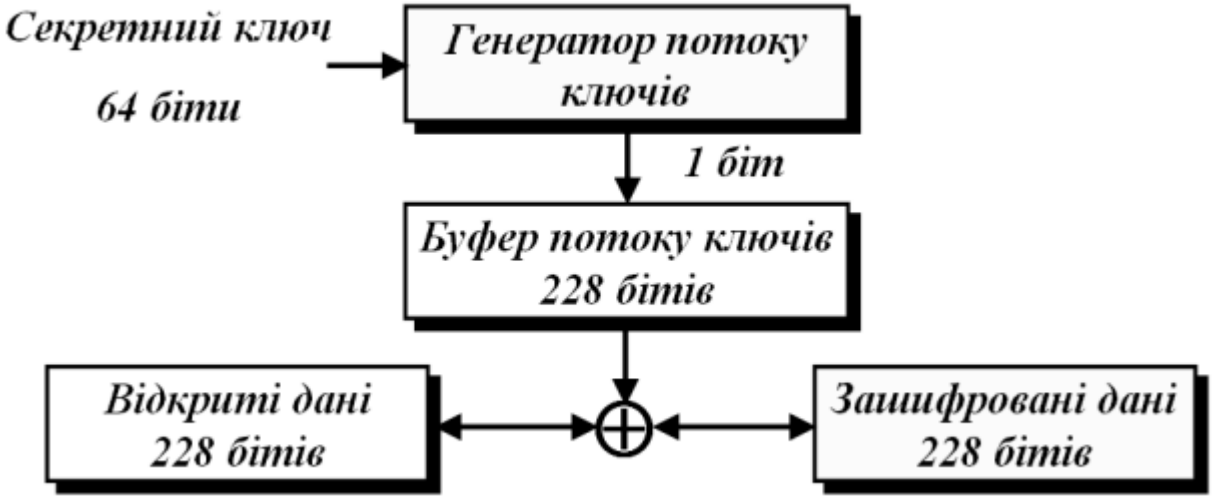
Класифікація потокових шифрів.

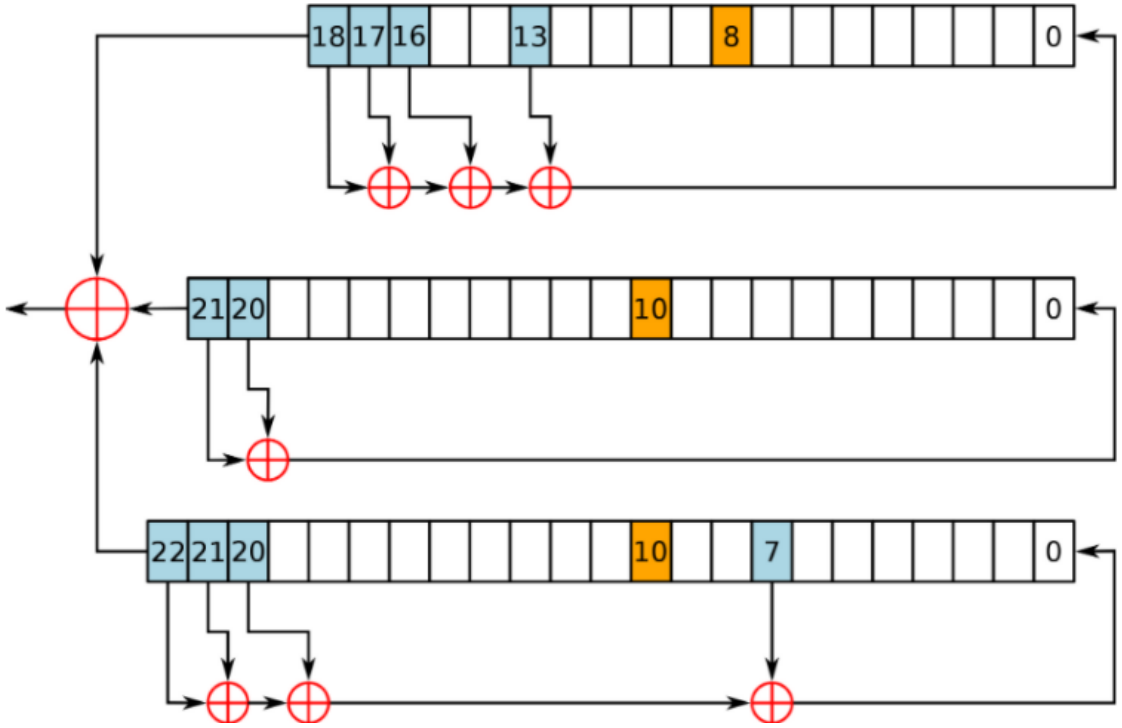
Синхронні потокові шифри (СПШ) - це шифри, в яких потік ключів генерується незалежно від відкритого й зашифрованого повідомлення. Під час зашифрування генератор потоку ключів видає біти потоку ключів, які ідентичні бітам потоку ключів під час розшифрування. Втрата знака зашифрованого повідомлення призведе до порушення синхронізації між цими двома генераторами й неможливості розшифрування частини повідомлення. Очевидно, що в цій ситуації відправник та одержувач повинні повторно синхронізуватися для продовження роботи.

Самосинхронізуючі потокові шифри (асинхронні потокові шифри (АПШ)) — це шифри, в яких потік ключів створюється функцією ключа й фіксованою кількістю знаків зашифрованого повідомлення. Отже, внутрішній стан генератора потоку ключів є функцією попередніх n бітів зашифрованого повідомлення. Тому генератор потоку ключів, який розшифровує, прийнявши n бітів, автоматично синхронізується із шифрувальним генератором.

Потоковий шифр A5

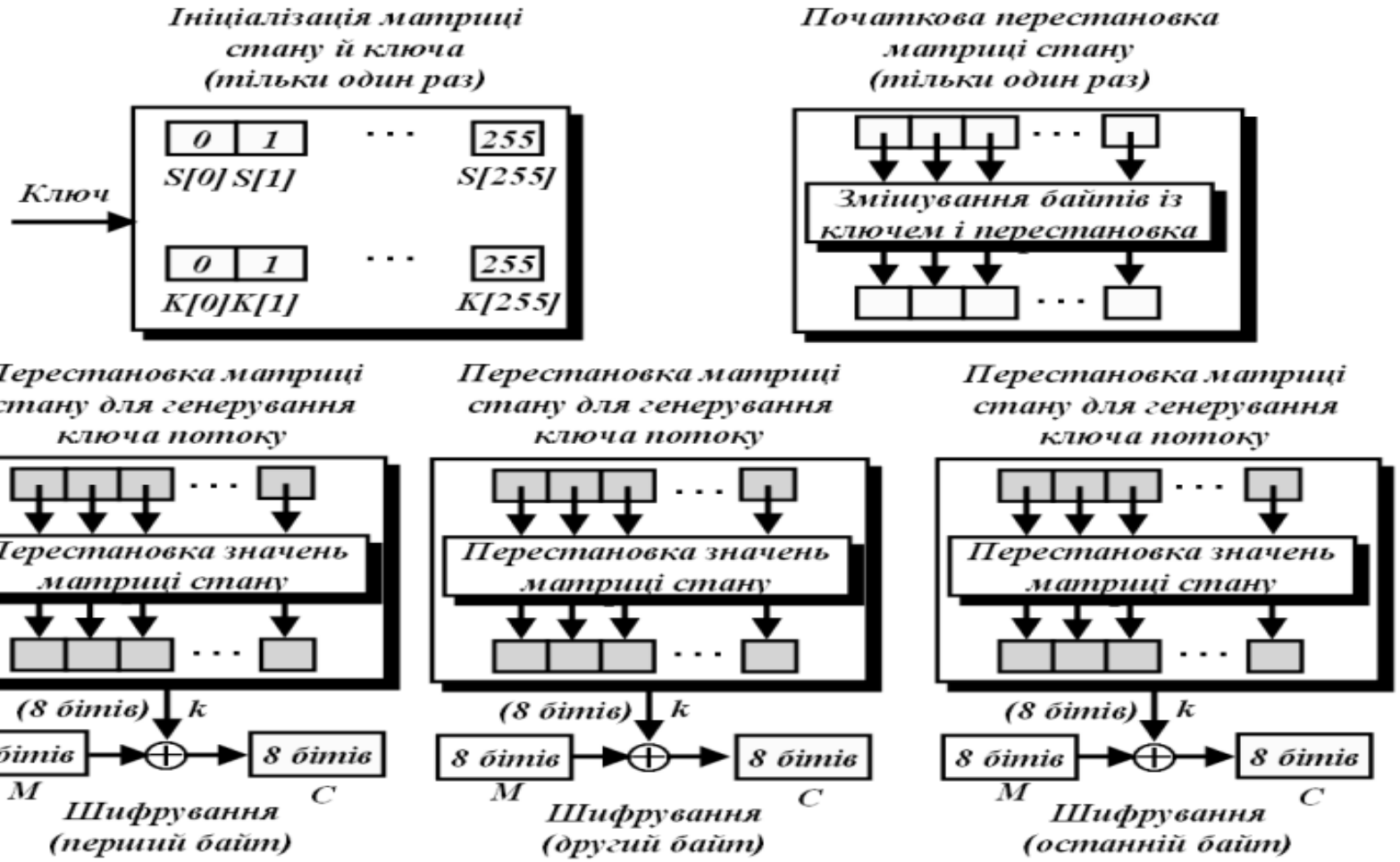
A5 — це потоковий алгоритм шифрування, що використовується для забезпечення конфіденційності даних, які передаються між телефоном і базовою станцією в європейській системі мобільного цифрового зв'язку GSM (Group Special Mobile). Цей шифр забезпечував достатній рівень захищеності потоку, що забезпечувало конфіденційність розмови. Спочатку експорт стандарту з Європи не передбачався, але незабаром у цьому з'явилася необхідність. Саме тому, A5 перейменували в A5/1 і стали поширювати в Європі та США. Для решти країн алгоритм модифікували, значно знизивши криптографічну стійкість шифру. A5/2 був спеціально розроблений як експортний варіант для країн, що не входили до Євросоюзу.

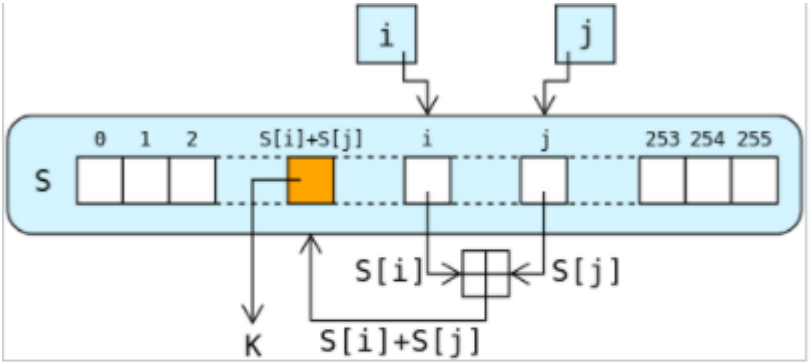




Потоковий шифр RC4

Алгоритм RC4 розробив Рональд Лінн Рівест 1987 р. спеціально як генератор потоку ключової інформації з ключем змінної довжини. RC4 базується на понятті матриці станів. У кожний момент матриця станів (256 байтів) активізується, з неї випадково вибирається один байт, який буде ключем для шифрування. Ідея може бути показана у вигляді масиву байтів: S[0], S[1], S[2], …, S[254], S[255]. Було доведено, що сучасні атаки на RC4 дозволяють зламати його протягом декількох днів або навіть годин. Тому в лютому 2015 року Internet Engineering Task Force (IETF) запропонувала в документі RFC 7465 припинити застосування RC4 в протоколі та реалізаціях TLS



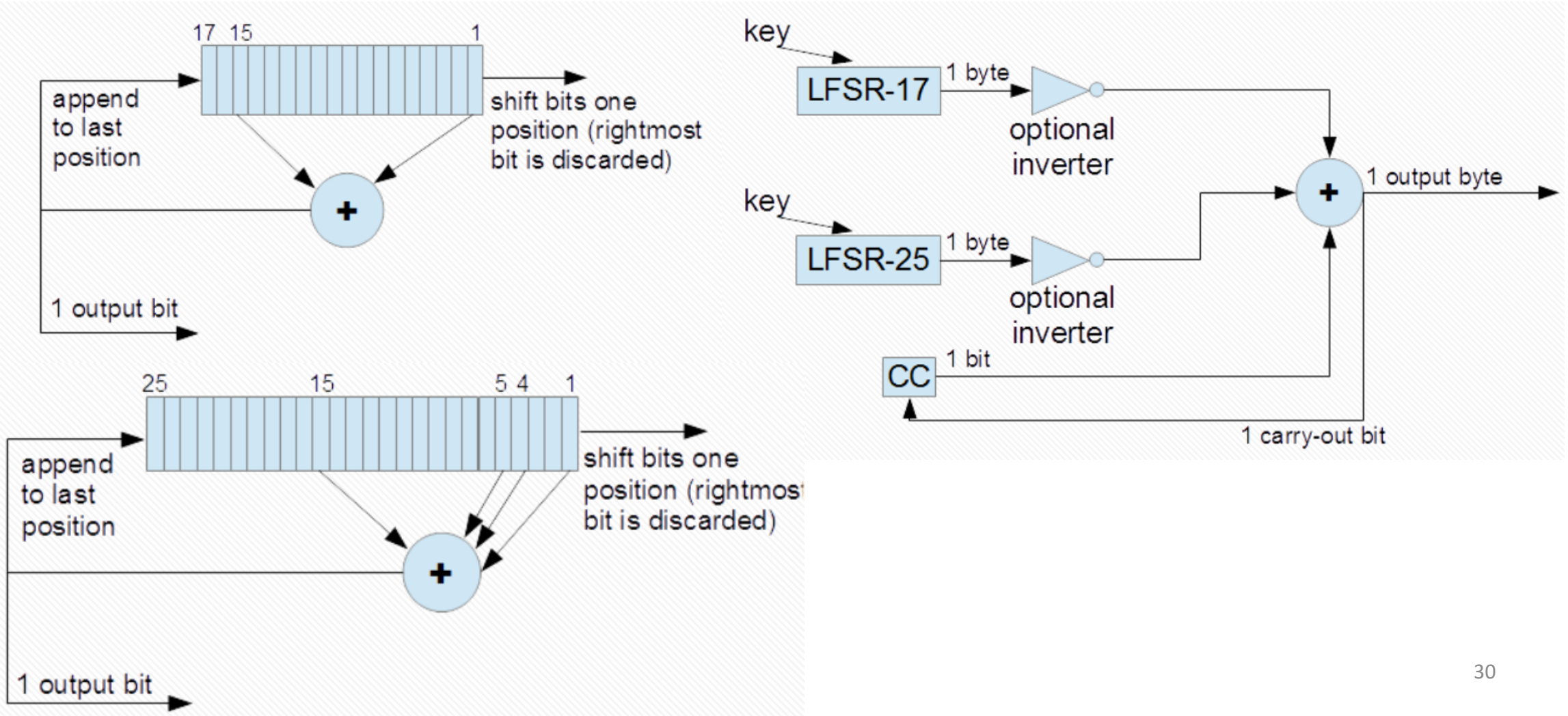


The lookup stage of RC4. The output byte is selected by looking up the values of `S[i]` and `S[j]`, adding them together modulo 256, and then using the sum as an index into `S`; `S(S[i] + S[j])` is used as a byte of the key stream, K.

CSS (Content Scramble System)

CSS потоковий шифр, який був розроблений (1996) для захисту відео даних на DVD дисках. Вперше був розкритий у 1999р.

LFSR - Linear Feedback Shift Registers (hardware implementation is very cheap):



Сучасні потокові шифри: eStream

$$\text{PRG:} \quad \underbrace{\{0,1\}^s}_{\text{seed}} \times R \longrightarrow \{0,1\}^n$$

nonce

Nonce: a non-repeating value for a given key.

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

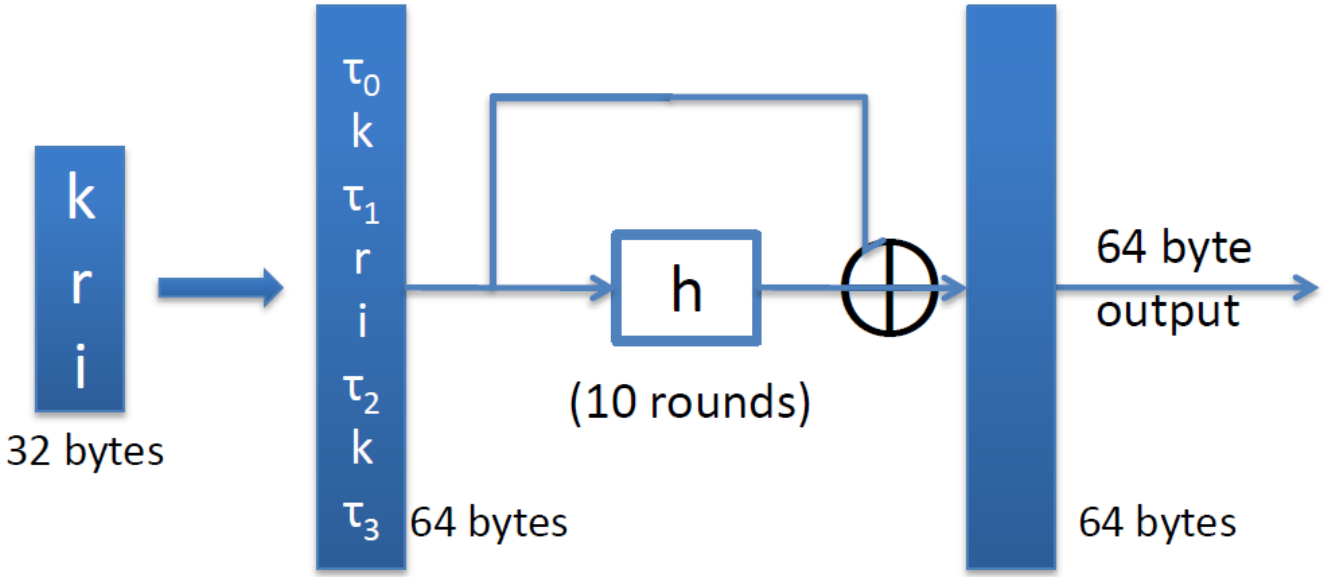
The pair (k,r) is never used more than once.

Сучасні потокові шифри: Salsa 20

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

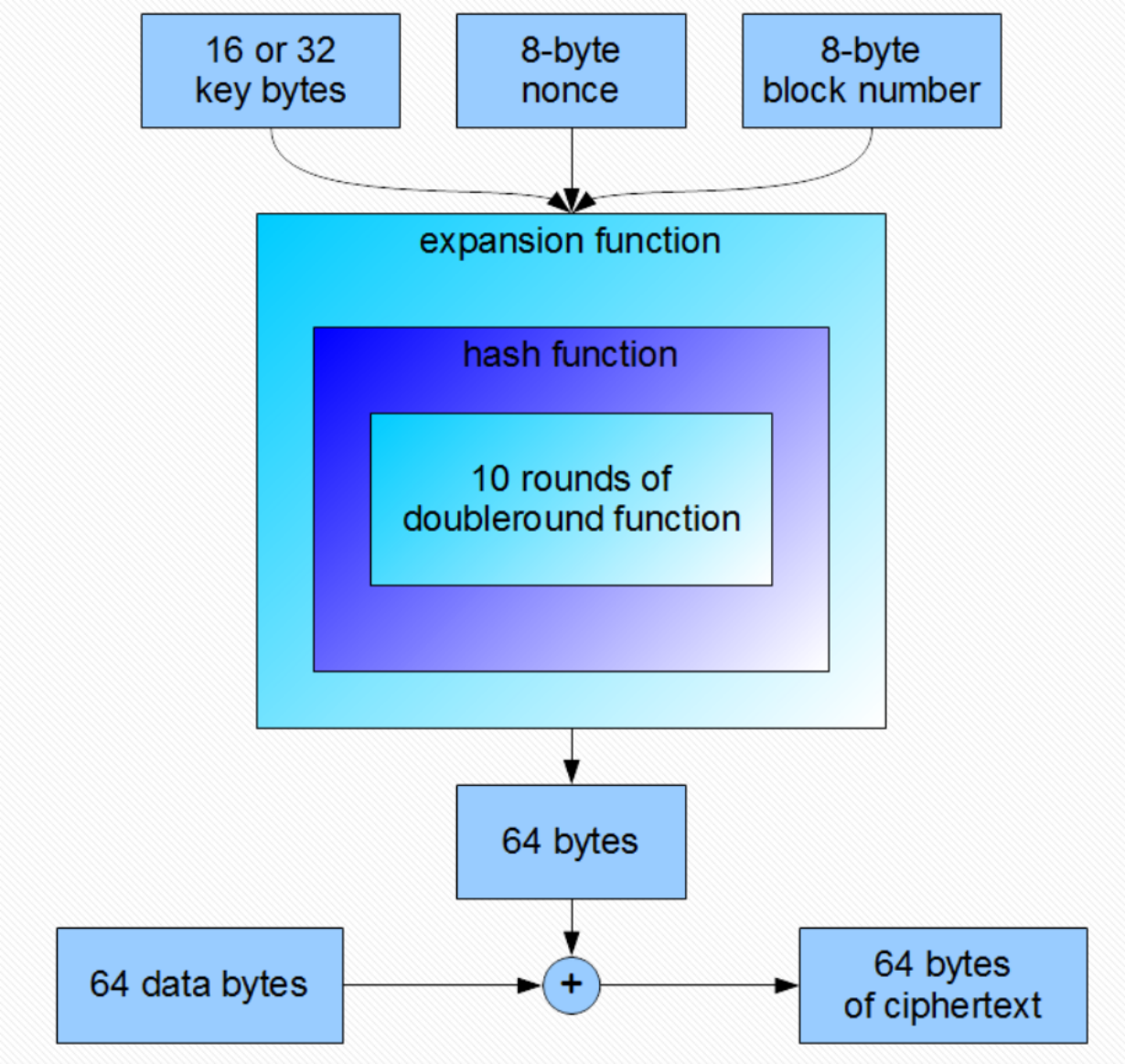
nonce

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



h: invertible function. designed to be fast on x86 (SSE2)

Сучасні потокові шифри: Salsa 20



Performance

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109