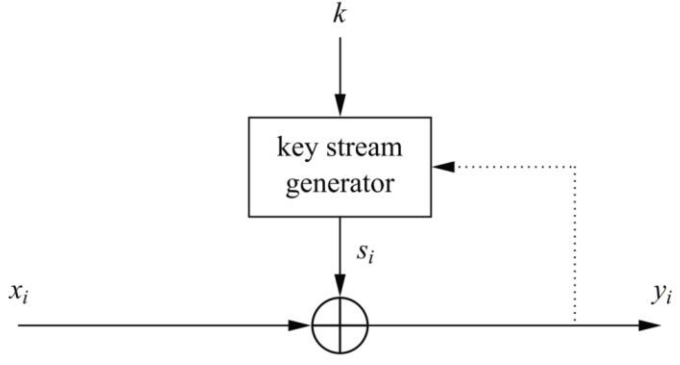
Kriptografija i kriptoanaliza

doc. dr. sc. Ante Đerek i prof. dr. sc. Marin Golub

Listopad 2023.

Algoritam kriptiranja toka podataka

- Protočna enkripcija / protočna šifra (*stream cipher).*
- Generira *tok ključa* koji se „zbraja" s jasnim tekstom operacijom XOR.
  - *Sinkrona protočna enkripcija* – tok ključa ovisi samo o ključu.
  - *Asinkrona protočna enkripcija* – tok ključa ovisi o ključu i prethodnim bitovima jasnog teksta.



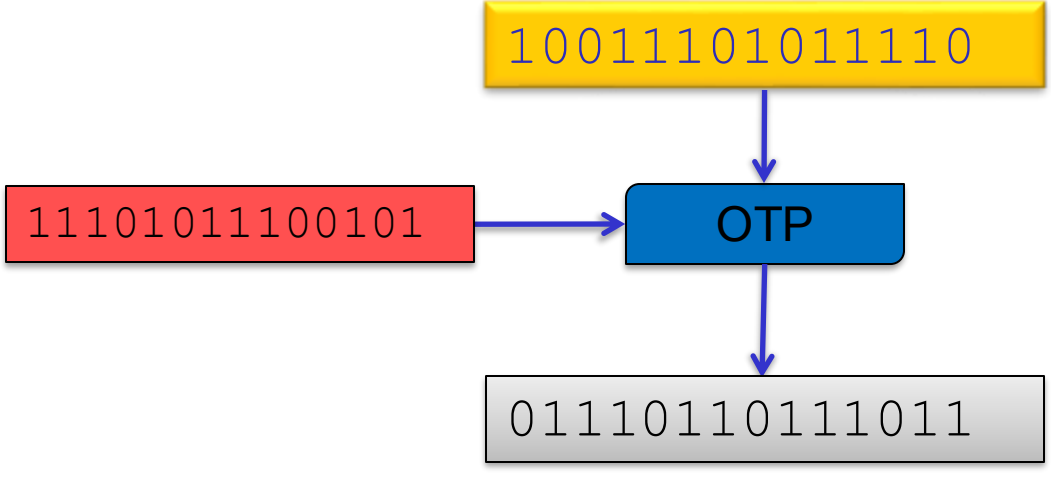
Slika je preuzeta iz udžbenika: ***Understanding Cryptography***, Christof Paar, Jan Pelzl; Springer-Verlag Berlin Heidelberg, 2009.

Primjeri protočne enkripcije

- RC4 (1987)
  - ključ veličine 40–2048 bitova
  - vrlo široko korišten, mnoštvo poznatih slabosti
- CSS (1996)
  - 40-bitni ključ
  - zaštita sadržaja na DVD-ovima
  - potpuno razbijen 1999. godine
- Salsa20/ChaCha (2005)
  - ključ 128 ili 256 bitova
  - podržan u TLS-u
  - alternativa AES-u zbog boljih performansi na uređajima gdje sklopovlje ne implementira AES

Ponavljanje: jednokratna bilježnica

- $M = K = C = \{0,1\}^n$
- $E(m,k) = m \oplus k$
- $D(c,k) = c \oplus k$



Ponavljanje: Savršena povjerljivost

- Claude Shannon, 1946
- Jednokratna bilježnica pruža savršenu povjerljivost:
  - Za svaku poruku $m \in \{0,1\}^n$ i šifrat $c \in \{0,1\}^n$ i vrijedi:

$$P_{k \leftarrow \{0,1\}^n}(E(m,k) = c) = \frac{1}{2^n}.$$

  - Alternativno, za svaku poruku $m \in \{0,1\}^n$ izlazi sljedeća dva vjerojatnostna algoritma imaju jednake razdiobe:

$$[k \xleftarrow{R} \{0,1\}^n; \text{output } E(m,k)]$$

$$[k \xleftarrow{R} \{0,1\}^n; m' \xleftarrow{R} \{0,1\}^n; \text{ output } E(m',k);]$$

Ponavljanje: jednokratna bilježnica – nedostaci

- Ključ
  - mora se generirati potpuno i uistinu slučajno!
  - mora biti jednako velik kao i poruka!
  - smije se koristiti najviše jednom!

$$c_1 = m_1 \oplus k$$
$$c_2 = m_2 \oplus k$$
$$c_1 \oplus c_2 = m_1 \oplus m_2$$

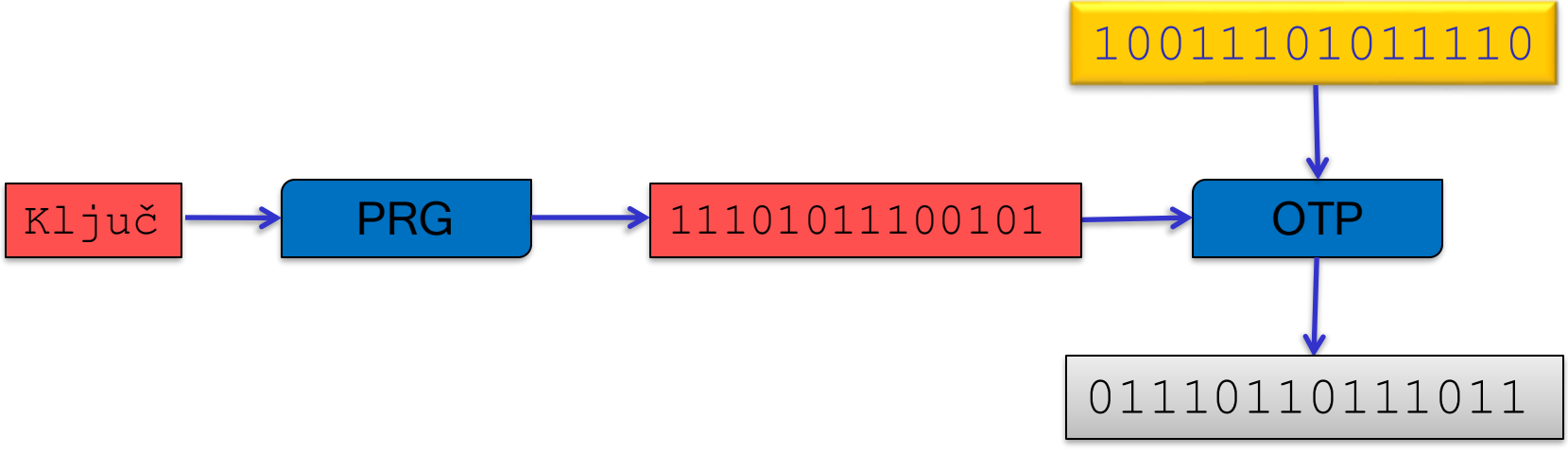
- Moguće je na predvidiv način izmijeniti poruku (engl. *malleable encryption)* !

$$c_1 = OTP(m_1, k) = m_1 \oplus k$$

$$c_2 = c_1 \oplus m_1 \oplus m_2 = m_1 \oplus k \oplus m_1 \oplus m_2 = m_2 \oplus k = OTP(m_2, k)$$

Algoritam kriptiranja toka podataka

- Ideja: umjesto slučajnog ključa koristimo *pseudoslučajni ključ*.
- Generator pseudoslučajnih brojeva na temelju ključa generira niz bitova koji se XOR-a s izvornim tekstom.



Generator pseudoslučajnih brojeva – definicija

- Generator pseudoslučajnih brojeva je efikasni deterministički algoritam $G: \{0,1\}^s \rightarrow \{0,1\}^n$ gdje je $n \gg s$.
- Ideja: G na deterministočan način od relativnog kratkog (npr. 128 bita) slučajnog *sjemena* (seed) generira vrlo dugačak (npr. 1GB) niz bitova koji *izgleda* slučajno.

Generator slučajnih brojeva – nomenklatura

- Generator slučajnih brojeva / *True random number generator* (TRNG)
  - Koristi prirodne slučajne procese ili ad-hoc informacije kako bi generirao *stvarno* slučajne brojeve.
- Generator pseudoslučajnih brojeva / *Pseudorandom number generators* (PRG)
  - Na temelju slučajnog *sjemena* deterministični računa vrijednosti koje *izgledaju slučajno*. Npr. rand() u programskom jeziku C.
- Kriptografski generator pseudoslučajnih brojeva / Cryptographically secure pseudorandom number generator
  - Generatori pseudoslučajnih brojeva s jakim sigurnosnim svojstvom nepredvidivosti.

Generator pseudoslučajnih brojeva – sigurnost

- Neformalno, generator pseudoslučajnih brojeva je *nepredvidiv* ako je napadaču (koji ne zna sjeme) jako teško predvidjeti izlaz generatora.
  - Čak i ako napadač vidi velik prefiks izlaza.

Protočna enkripcija

- Ako je $G: \{0,1\}^s \rightarrow \{0,1\}^n$ generator pseudoslučajnih brojeva onda možemo definirati protočnu enkripciju na sljedeći način:
  - $E(m,k) = m \oplus G(k)$
  - $D(c,k) = c \oplus G(k)$

Zadatak: protočna enkripcija s previdivim generatorom

- Neka je G generator pseudoslučajnih brojeva koji je predvidiv. Neka je $E(m,k) = m \oplus G(k)$.
- Pokaži da E nije sigurna simetrična enkripcija pod pretpostavkom da napadač može pogoditi prefiks poruke koja se kriptira.

"Obični" generatori pseudoslučajnih brojeva

- *Linear congruential generator:*
  - $S_0 = \text{seed}$
  - $S_i = (aS_{i-1} + b) \bmod m$
- Odlična statistička svojstva.
- Predvidiv i stoga neupotrebljiv u kriptografiji.

"Obični" generatori pseudoslučajnih brojeva

```
 * This is a linear congruential pseudorandom number generator, as
 * defined by D. H. Lehmer and described by Donald E. Knuth in
 * <cite>The Art of Computer Programming, Volume 2, Third edition:
 * Seminumerical Algorithms</cite>, section 3.2.1.
 *
 * @param  bits random bits
 * @return the next pseudorandom value from this random number
 *         generator's sequence
 * @since  1.1
 */
protected int next(int bits) {
    long oldseed, nextseed;
    AtomicLong seed = this.seed;
    do {
        oldseed = seed.get();
        nextseed = (oldseed * multiplier + addend) & mask;
    } while (!seed.compareAndSet(oldseed, nextseed));
    return (int)(nextseed >>> (48 - bits));
}
```

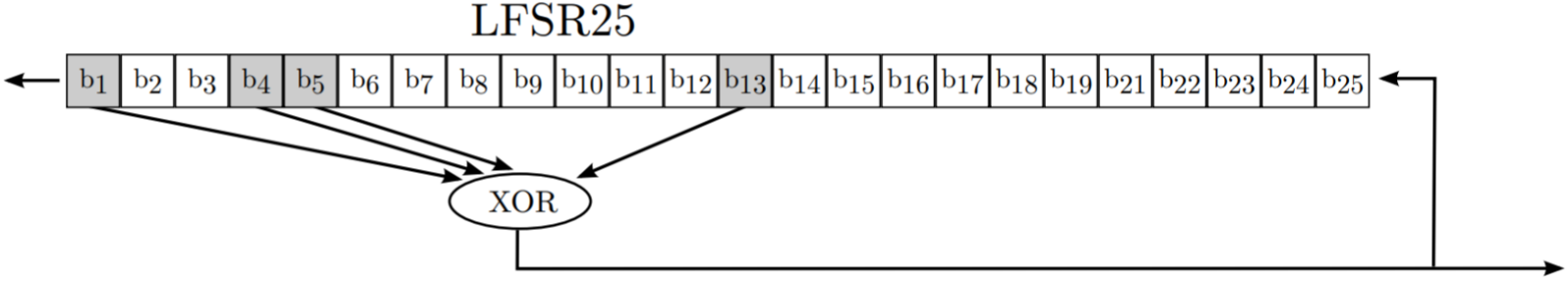
https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/util/Random.java

LFSR

- Linearni posmačni registar s povratnom vezom / *Linear Feedback Shift Register* (LFSR)
  - Stanje se sastoji od m bitova
  - U svakom ciklusu se izračuna XOR bitova na fiksnim (povratnim) pozicijama.
  - Rezultat b je izlaz ciklusa.
  - Stanje se posmiče i bit b se dodaje na kraj

Zadatak: Sigurnost LFSR-a

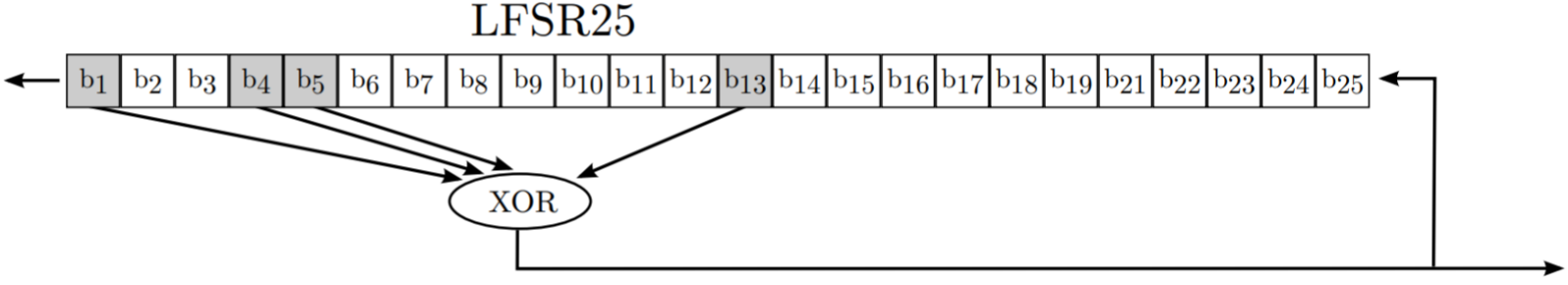
- Poznata je arhitektura 128-bitnog LFSR-a i poznato je prvih 1000 bitova njegovog izlaza. Predvidite ostale bitove.



Izvor: https://hsin.hr/pripreme2016/zadaci/prvi/zadaci.pdf

Zadatak: Sigurnost LFSR-a

- Poznata je arhitektura 128-bitnog LFSR-a i poznato je drugih 1000 bitova njegovog izlaza. Predvidite ostale bitove.



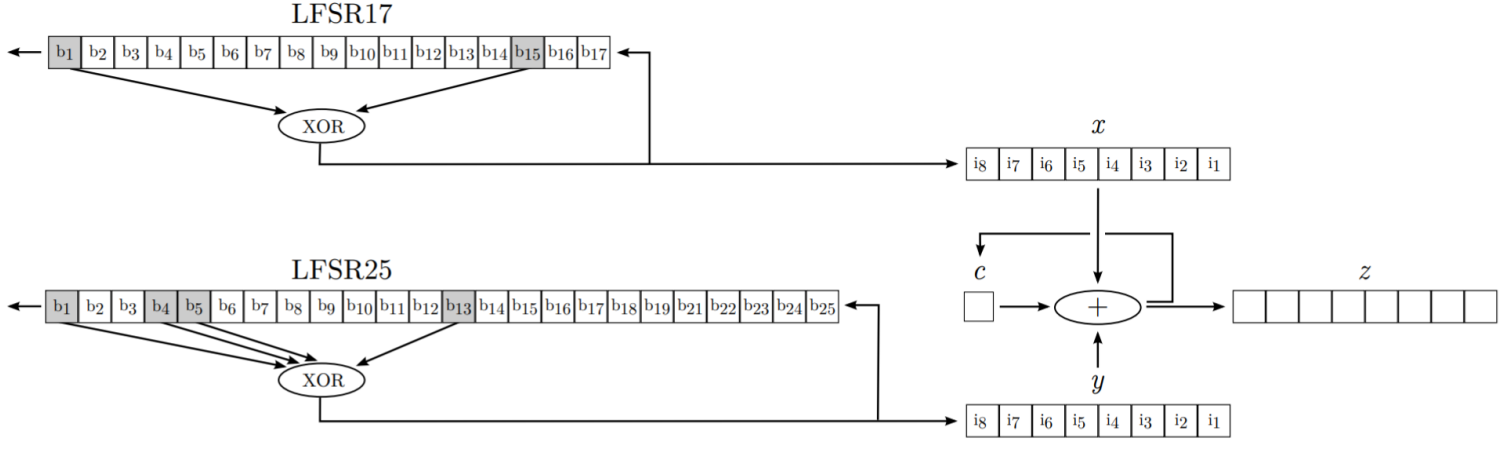
Izvor: https://hsin.hr/pripreme2016/zadaci/prvi/zadaci.pdf

LFSR

- Vrlo jednostavan i jeftin za sklopovsku implementaciju.
- Potpuno nesiguran za kriptografija.
- Baza za mnoge protočne enkripcije.

Primjer: CSS

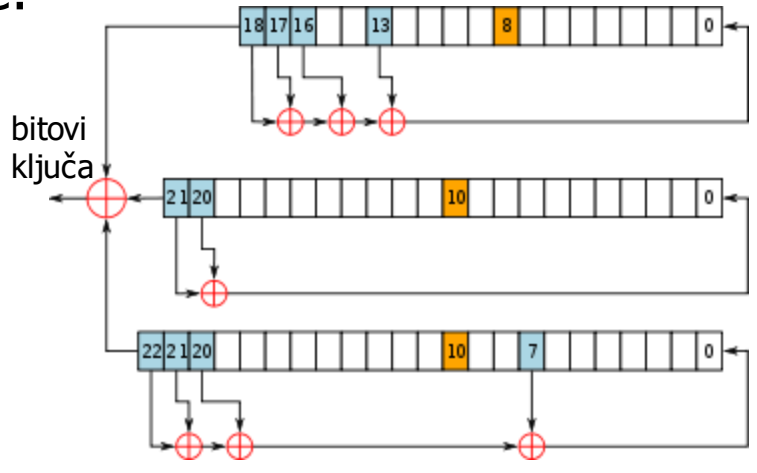
- *Content Scramble System*
- Mehanizam zaštite sadržaja na DVD-ovima
- 40-bitni ključevi
- Jednostavni napad poznatim jasnim tekstom u 2^{17} koraka.



Izvor: https://hsin.hr/pripreme2016/zadaci/prvi/zadaci.pdf

Primjer: A5/1

- U postupku generiranja enkripcijskog ključa koriste se 3 posmačna registra duljine 19, 22 i 23 bita.
- Postupak kriptiranja odvija se u 3 faze:
  1. učitavanje 64-bitnog ključa u 3 registra u 64 koraka
  2. postavljanje početnog stanja registara u 100 koraka
  3. stvaranje niza bitova ključa
- interaktivni simulator je dostupan na https://733amir.github.io/a51-cipher-simulator/



izvor: wikipedia.org

Primjer: A5/1

- Zabilježeni su brojni napadi.
- Eksperimentalno je utvrđeno da nakon postavljanja početnog stanja registara, koji se odvija u 100 koraka, stanje registara može poprimiti svega 15% svih mogućih stanja kojih ima 2^{64}
  - dakle, prostor pretraživanja se svodi na $2^{64} \times 0{,}15 \approx 2^{61{,}26}$
- 2006. g. kriptografi E. Barkan, E. Biham i N. Keller demonstrirali su napad na algoritam A5/1 koji omogućuje dekriptiranje razgovora u stvarnom vremenu.

eStream Contest

- 2004. – 2008.
- Profile 1: Stream ciphers for software applications with high throughput. Must support 128-bit key. Must support 64-bit IV and 128-bit IV.
- Profile 2: Stream ciphers for hardware applications with highly restricted resources. Must support 80-bit key. Must support 32-bit IV and 64-bit IV.

Profile 1 (software)	Profile 2 (hardware)
HC-128 [1]	Grain [2]
Rabbit [3]	MICKEY [4]
Salsa20/12 [5]	Trivium [6]
SOSEMANUK [7]	

Primjer: Salsa20/ChaCha

- Ulaz u generator je ključ i *nonce* vrijednost.
  - $E(m, k, r) = m \oplus G(k, r)$
  - Nonce vrijednost omogućuje ponovno korištenje ključa. Ako se dva puta koristi isti nonce s istim ključem onda sustav više nije siguran.
- Generator (nezavisno) generira 512-bitne blokove uz pomoć brojača.
  - $G(k, r) = C(k, r, 0)||C(k, r, 1)||C(k, r, 2)||$ ...
  - Moguća paralelizacija i brzo dekriptiranje na proizvoljnoj lokaciji unutar kriptiranog teksta.

Primjer: Salsa20/ChaCha

```c
void salsa20_block(uint32_t out[16], uint32_t const in[16])
{
    int i;
    uint32_t x[16];

    for (i = 0; i < 16; ++i)
        x[i] = in[i];
    // 10 loops × 2 rounds/loop = 20 rounds
    for (i = 0; i < ROUNDS; i += 2) {
        // Odd round
        QR(x[ 0], x[ 4], x[ 8], x[12]); // column 1
        QR(x[ 5], x[ 9], x[13], x[ 1]); // column 2
        QR(x[10], x[14], x[ 2], x[ 6]); // column 3
        QR(x[15], x[ 3], x[ 7], x[11]); // column 4
        // Even round
        QR(x[ 0], x[ 1], x[ 2], x[ 3]); // row 1
        QR(x[ 5], x[ 6], x[ 7], x[ 4]); // row 2
        QR(x[10], x[11], x[ 8], x[ 9]); // row 3
        QR(x[15], x[12], x[13], x[14]); // row 4
    }
    for (i = 0; i < 16; ++i)
        out[i] = x[i] + in[i];
}
```

Initial state of Salsa20

"expa"	Key	Key	Key
Key	"nd 3"	Nonce	Nonce
Pos.	Pos.	"2-by"	Key
Key	Key	Key	"te k"

```c
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d)(     \
    b ^= ROTL(a + d, 7),    \
    c ^= ROTL(b + a, 9),    \
    d ^= ROTL(c + b,13),    \
    a ^= ROTL(d + c,18))
```

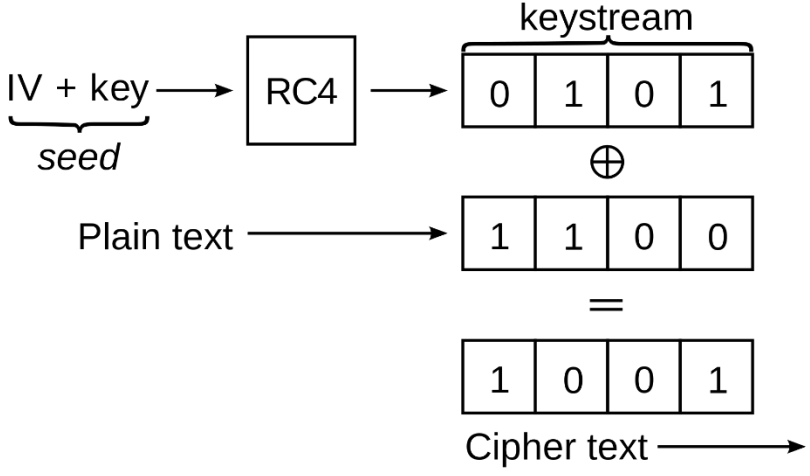
izvor: wikipedia.org

Napadi na protočnu enkripciju

- Svi napadi na OTP su i napadi na protočnu enkripciju!
  - Ako isti ključ koristimo dva puta, nema nikakve sigurnosti!
  - Enkripcija ne pruža nikakvo svojstvo integriteta!

Primjer: WEP

- Ključ (40-bitni ili 104-bitni) se spaja s 24-bitnim IV i koristi kao ključ za RC4 protočnu enkripciju.
- Ključevi su povezani, a RC4 je ranjiv na takozvani *related key* napad.
- Potpuno razbijen u 2001.



izvor: wikipedia.org

Generiranje entropije

- Ad-hoc slučajne informacije sa sistema
  - /dev/random na Linux sustavima koristi vremenske informacije s tipkovnice i miša, vremenske informacije o prekidima, vremenske informacije o operacijama diska, podatke o samom uređaju (npr. serijski brojevi).
  - https://github.com/torvalds/linux/blob/master/drivers/char/random.c
- Specijalizirani hardver
  - Koristi prirodne slučajne procese (termalni šum, fotoelektrični efekt, …)
- RDRAND (Intel)
  - Stvarna slučajnost i kriptografski generator pseudoslučajnih brojeva.
  - "The ES runs asynchronously on a self-timed circuit and uses thermal noise within the silicon to output a random stream of bits at the rate of 3 GHz."

Ako nema slučajnih brojeva nema ni sigurnosti!

- Sigurnost svih sustava ovisi u pretpostavci da je moguće na slučajan način generirati:
  - Ključeve
  - Inicijalizacijske vektore
  - Nonce vrijednosti u protokolima
  - …
- Ako napadač može predvidjeti ključeve, nikakva sigurnost nije garantirana!

Primjer: Netscape 1.1 (1995)

- Napad: Za sve moguće seed parametre
  - Generiraj ključ koristeći isti postupak
  - Provjeri je li moguće dešifrirati komunikaciju s dobivenim ključem

```
global variable seed;

RNG_CreateContext()
    (seconds, microseconds) = time of day; /* Time elapsed since 1970 */
    pid = process ID;  ppid = parent process ID;
    a = mklcpr(microseconds);
    b = mklcpr(pid + seconds + (ppid << 12));
     seed = MD5(a, b);

mklcpr(x) /* not cryptographically significant; shown for completeness */
    return ((0xDEECE66D * x + 0x2BBB62DC) >> 1);

MD5() /* a very good standard mixing function, source omitted */
```

Figure 2: *The Netscape 1.1 seeding process: pseudocode.*

```
RNG_GenerateRandomBytes()
    x = MD5(seed);
    seed = seed + 1;
    return x;

global variable challenge, secret_key;

create_key()
    RNG_CreateContext();
    tmp = RNG_GenerateRandomBytes();
    tmp = RNG_GenerateRandomBytes();
    challenge = RNG_GenerateRandomBytes();
    secret_key = RNG_GenerateRandomBytes();
```

Figure 3: *The Netscape v1.1 key-generation process: pseudocode.*

Izvor: Goldberg, Wagner, „Randomness and the Netscape Browser", 1996.

Primjer: Debian OpenSSL (2005)

On May 13th, 2008 the Debian project announced that Luciano Bello found an interesting vulnerability in the OpenSSL package they were distributing. The bug in question was caused by the removal of the following line of code from *md_rand.c*

```
MD_Update(&m,buf,j);
[ .. ]
MD_Update(&m,buf,j); /* purify complains */
```

These lines were removed because they caused the Valgrind and Purify tools to produce warnings about the use of uninitialized data in any code that was linked to OpenSSL. You can see one such report to the OpenSSL team here. Removing this code has the side effect of crippling the seeding process for the OpenSSL PRNG. Instead of mixing in random data for the initial seed, the only "random" value that was used was the current process ID. On the Linux platform, the default maximum process ID is 32,768, resulting in a very small number of seed values being used for all PRNG operations.

Izvor:
schneier.com

Druge konstrukcije PRG-ova

- Bazirane na sustavima kriptiranja bloka.
  - Npr. CRT_DBRG
- Bazirane na kriptografskim hash funkcijama
  - HASH_DBRG, HMAC_DBRG
- Specijalizirane konstrukcije.

Formalne definicije sigurnosti

- Cilj: pokazati kako se formalno može definirati sigurnost i kako se može dokazati sigurnost kriptografske konstrukcije na temelju sigurnosti pojedinih primitiva.

PRG

- Želimo na neki način definirati što znači da napadač ne može razlikovati generator pseudoslučajnih brojeva od generatora stvarno slučajnih brojeva.
- Neka je $G: K \rightarrow \{0,1\}^n$ PRG, želimo da sljedeće dvije razdiobe budu „nerazlučive" (*indistinguishable).*

$$[k \xleftarrow{R} K; \text{output } G(k)]$$

$$[r \xleftarrow{R} \{0,1\}^n; \text{output } r;]$$

Statistički test

- *Statistički test* je bilo koji algoritam koji prima niz od *n* bitova i pokušava odrediti je li taj niz slučajan.
- $A: \{0,1\}^n \rightarrow \{0,1\}$
- Primjer:
  - $A(x) = 1$ ako je razlika broja jedinica i broja nula manja od $\sqrt{n}$, a $A(x) = 0$ inače.

Prednost

- Neka je $G: K \rightarrow \{0,1\}^n$ PRG
- Neka je $A: \{0,1\}^n \rightarrow \{0,1\}$ statistički test
- Prednost (advantage) statističkog testa A definiramo kao:

$$\mathrm{Adv}_{PRG}(A,G) = |P_{k \leftarrow K}(A(G(k)) = 1) - P_{x \leftarrow \{0,1\}^n}(A(x) = 1)|$$

- Prednost blizu nuli: test ne razlikuje G od slučajnih brojeva.
- Prednost blizu jedinici: test razlikuje G od slučajnih brojeva.

Zadatak: Prednost

- Neka je $G: K \rightarrow \{0,1\}^n$ PRG koji ima svojstvo da je XOR prvih 10 bitova jednak 1 s vjerojatnošću ¾.
- Postoji li statistički test koji ima veliku prednost?

Definicija sigurnosti

- Neka je $G: K \rightarrow \{0,1\}^n$ PRG. Kažemo da je G *siguran* ako ako svaki efikasni statistički test ima zanemarivo malu prednost.

Što znače efikasno i zanemarivo?

- Praktični pristup: fiksne vrijednosti.
- G je (t, ε)-siguran PRG ako svaki algoritam koji radi t koraka ima prednost strogo manju od ε.
- Konkretne poželjne vrijednosti ovise o situaciji, npr: $t = 2^{80}, \varepsilon = 2^{-80}$.

Što znače efikasno i zanemarivo?

- Teorijski pristup: funkcija sigurnosnog parametra.
- G_i je siguran PRG ako je svaki vjerojatnostni algoritam A koji radi u polinomom vremenu njegova prednost ε_i zanemariva funkcija.
  - Funkcija $f: \mathbb{N} \to \mathbb{R}$ je *zanemariva* ako za svaki broj $c \in \mathbb{R}^+$ postoji $n_0 \in N$ takav da za sve $n > n_0$ vrijedi $|f(n)| < \frac{1}{n^c}$.
  - Intuicija: zanemariva funkcija raste sporije nego inverz svakog polinoma.

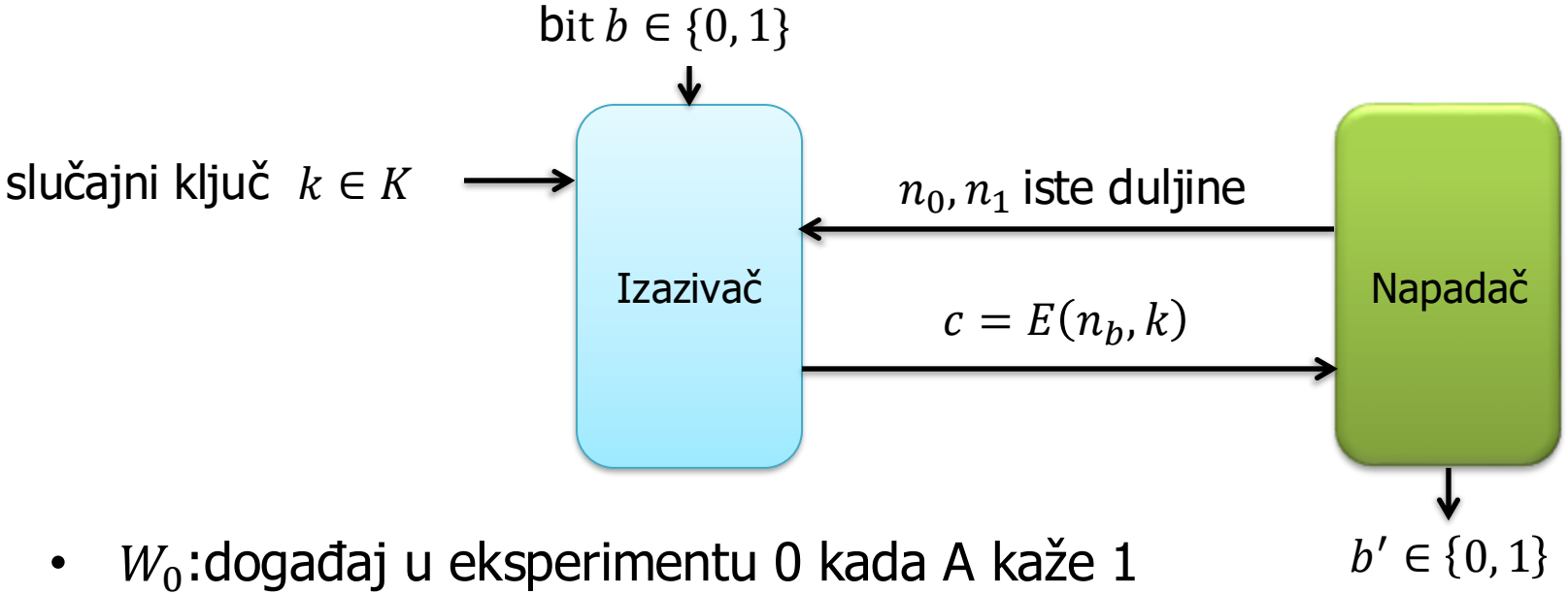
Zadatak: Sigurnost i nepredvidivost

- Neka je $G: K \rightarrow \{0,1\}^n$ siguran PRG, pokažite da je G nepredvidiv.
- Postoji li statistički test koji ima veliku prednost?
- Vrijedi i obrat: svaki nepredvidiv PRG je siguran (Yao, 1982).

Semantička sigurnost

- Želimo definirati sigurnost simetrične enkripcije koja je općenitija od savršene povjereljivosti.
  - Ali dokle god se ključ koristi samo jednom, dakle ne uzima u obzir napade poznatim jasnim tekstom, odabranim jasnim tekstom, itd.

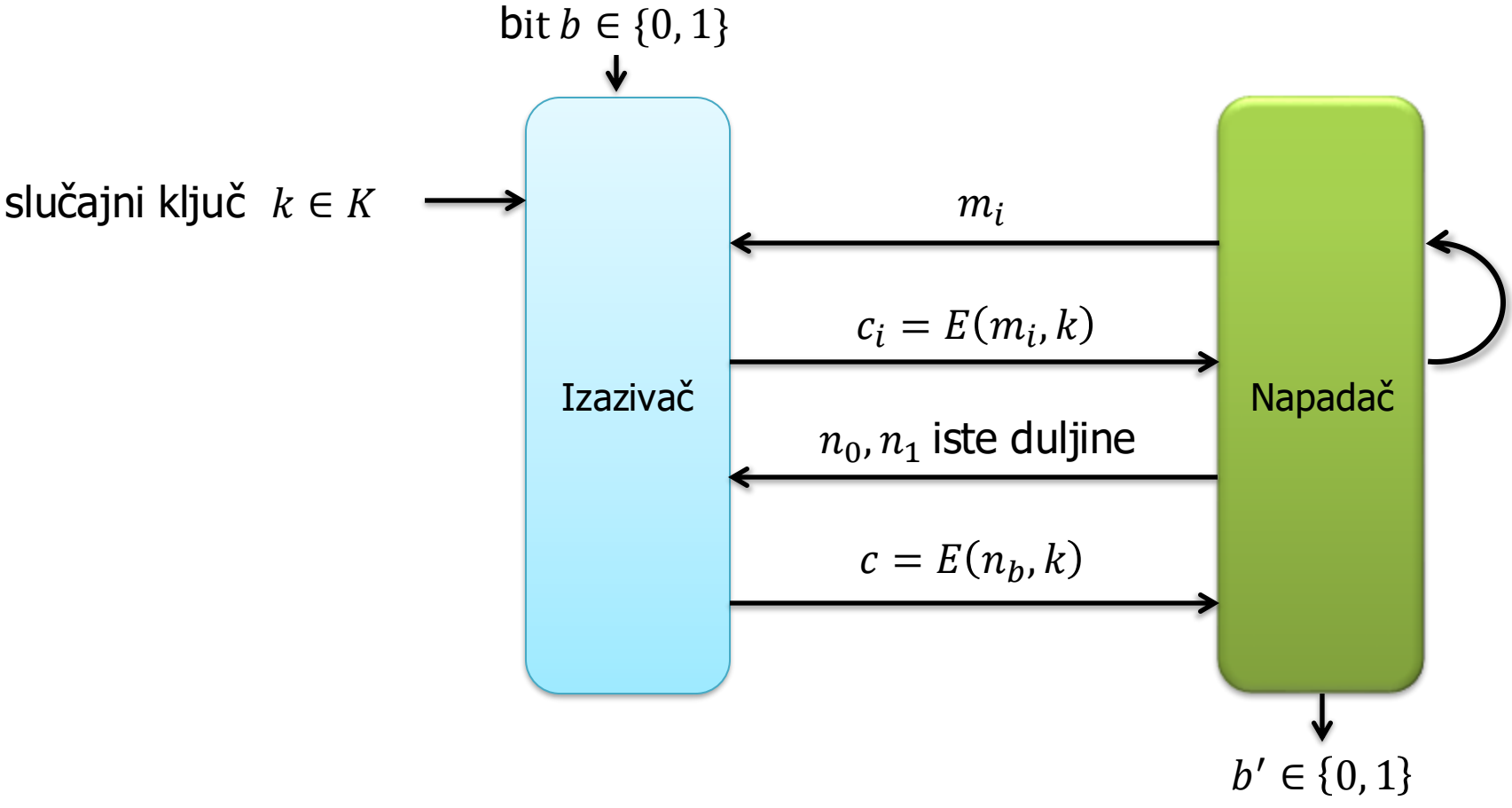
Semantička sigurnost



- W_0:događaj u eksperimentu 0 kada A kaže 1
- W_1:događaj u eksperimentu 1 kada A kaže 1

$$\mathrm{Adv}_{SS}(A) = |P(W_0) - P(W_1)|$$

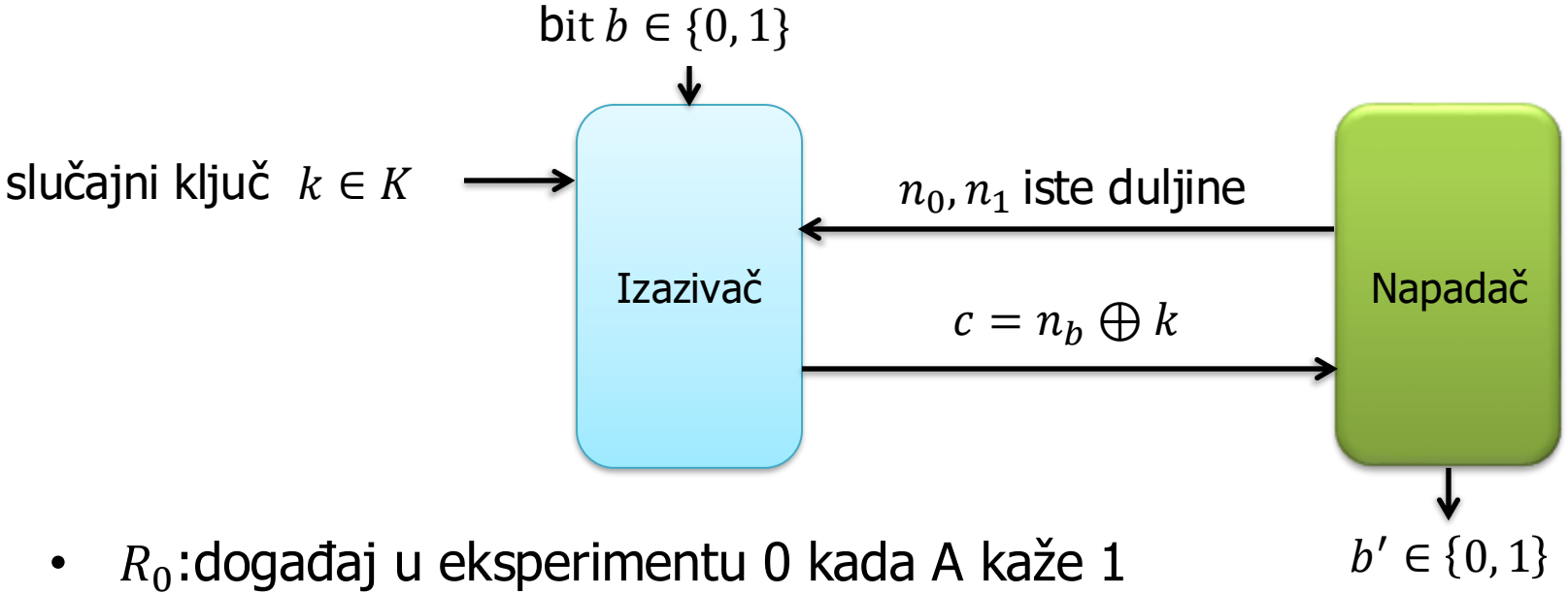
Digresija: Jače definicije sigurnosti



Semantička sigurnost

- Simetrična enkripcija je semantički sigurna ako je za svakog efikasnog napadača A njegova prednost $\mathrm{Adv}_{SS}(A)$ zanemarivo mala.

Semantička sigurnost jednokratne bilježnice



- R_0:događaj u eksperimentu 0 kada A kaže 1
- R_1:događaj u eksperimentu 1 kada A kaže 1

$$\mathrm{Adv}_{SS}(A) = |P(R_0) - P(R_1)| = 0$$

Sigurnost protočne enkripcije

- Teorem: Ako je G siguran PRG onda je $E(m,k) = m \oplus G(k)$ semantički sigurna enkripcija.
- Dokaz ćemo pokazati ali ga nije potrebno znati.