Stavební bloky kryptografie

Kamil Malinka
malinka@fit.vutbr.cz
Fakulta informačních technologií

Módy blokových šifer

- Šifrování textu po blocích
  - 64, 80, 128, … bitové bloky

- Jak zašifrovat delší zprávy?
  - snaha zabránit výměně/promíchání bloků šifry
  - Příklad:
    - Zpráva obsahující bankovní příkaz, jeden blok obsahuje částku, jeho výměna změní význam celé zprávy
  - obrana?

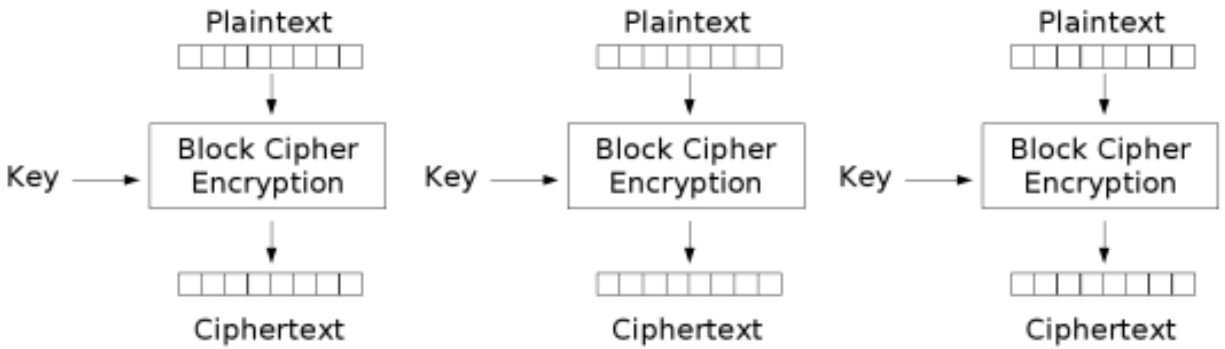
Logické svázání bloků

Konvence zápisu

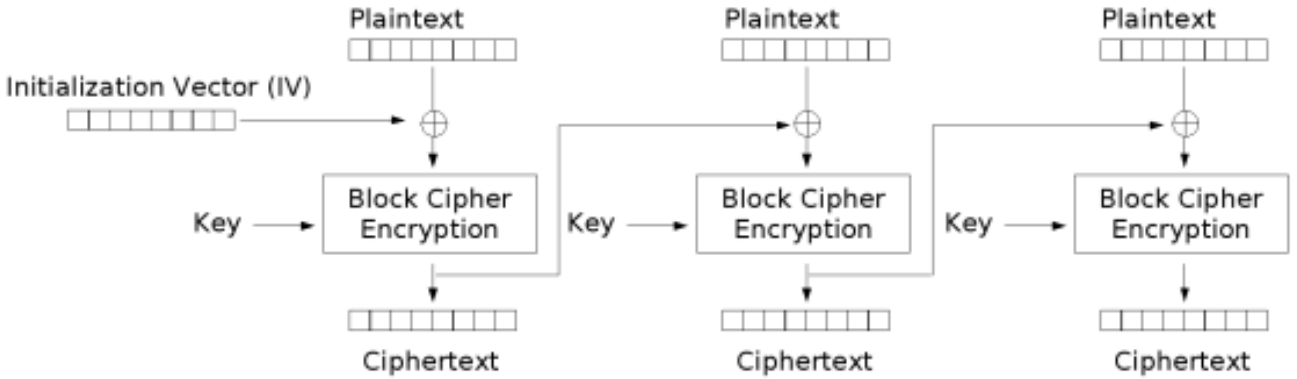
- $c_0,c_1,\ldots,c_j,\ldots$ - bloky šifrovaného textu
- $x_0,x_1,\ldots,x_j,\ldots$ - bloky otevřeného textu
- IV … inicializační vektor
- I_j … blok vstupující do šifry
- O_j … výstup šifry
- E_K … označení šifry s klíčem K
- $\oplus$ … operace xor

Základní módy

- ECB – electronic code book
  - nezávislé bloky, žádné řetězení
- CBC – cipher block chaining
  - $c_0=IV$, $c_j=E_K(c_{j-1} \oplus x_j)$
- OFB – output feedback mode
  - $I_1=IV$, $O_j=E_K(I_j)$, $c_j=x_j \oplus O_j$, $I_{i+1}=O_i$
- CFB – cipher feedback mode
  - $I_1=IV$, $O_j=E_K(I_j)$, $c_j=x_j \oplus O_j$, $I_{j+1}=c_j$



Electronic Codebook (ECB) mode encryption



Cipher Block Chaining (CBC) mode encryption

http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

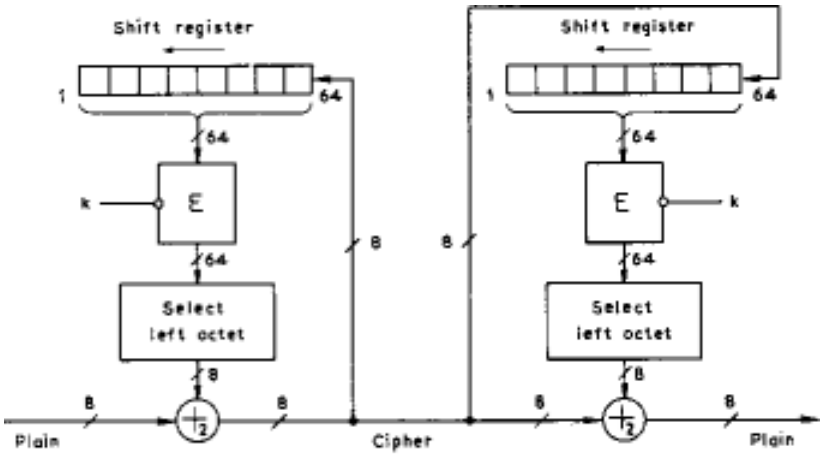


Fig. 4.7 8-bit cipher feedback

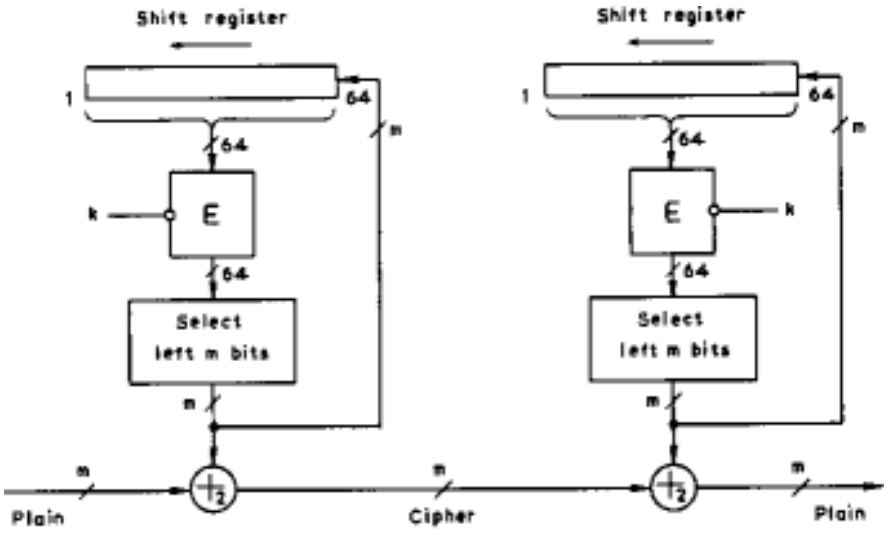


Fig. 4.12 m-bit output feedback

Který je CFB?

http://williamstallings.com/Extras/Security-Notes/lectures/blockA.html

Cipher-block chaining

- nejčastěji používaný mód
- změna v IV nebo prvním bloku => odlišný šifrovaný text
- 4 vlastnosti:
  - ***identické vstupy*** => identické výstupy (pokud je IV a klíč stejný), doporučeno něco změnit
  - ***vázací závislost***
    - c_j závisí na x_j a všech předcházejících blocích
    - změna pořadí bloků ovlivní dešifrování
    - správné dešifrování vyžaduje správné řazení bloků
  - ***propagace chyb***
    - chyba v 1 bitu ŠT ovlivní pouze dešifrování aktuálního a následujícího bloku
    - velikost chyby lze snadno odhadnout
    - 1 bitová chyba v c_j => změna 50% bitů v x_j
  - ***zotavení z chyb***
    - samosynchronizující
    - zotavení z chyby vs. zotavení ze ztráty

Output feedback (full - ISO10116, r-bit FIPS81)

- ISO – velikost výstupního bloku = velikost bloku šifry
- FIPS – použito r bitů bloku, kde r ≤ velikosti bloku šifry
- vlastnosti:
  - identické vstupy – totéž co CRC
  - propagace chyb – 1b chyba v OT => 1b chyba v ŠT
  - ztráta bitu => nemožnost provedení resynchronizace

Co se stane pokud nezměníme IV při stejném klíči?

Cipher feedback

- zpětná vazba přes šifrovaný text
- propagace chyb
  - změna 1b v bloku c_j => 1b změna v x_j a 50% změna v x_j+1
- samosynchronizující se jako CBC
- používá se pouze šifrování!

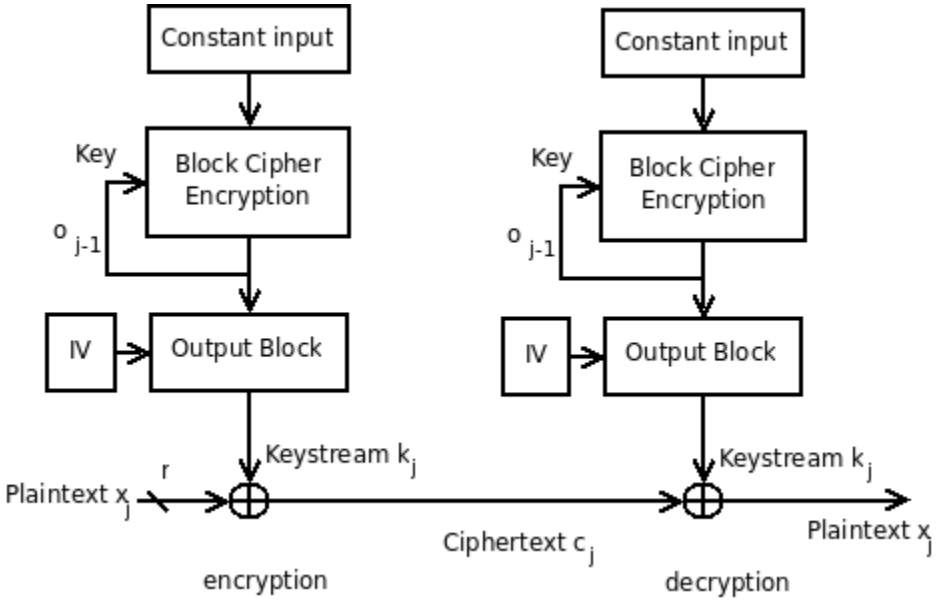
- Může útočník způsobit předvídatelné změny v bitech?

Další módy

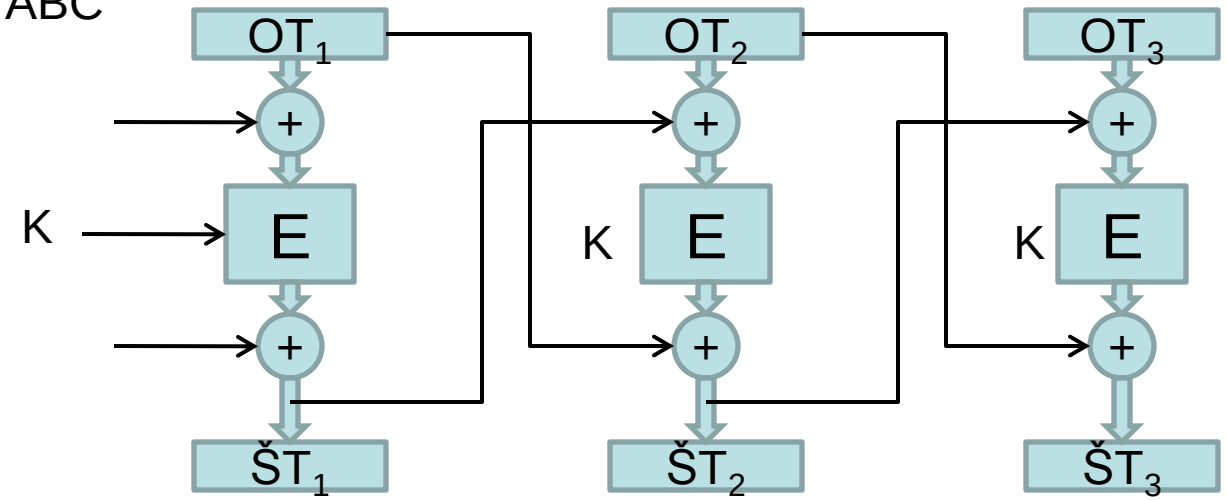
- celkový počet módů je mnohem větší…
- counter mode (CTR)
  - zjednodušené OFB, umožňuje paralelní zpracování dat, nebo zpracování bloků mimo pořadí
  - místo vazby na předchozí blok je použita vazba s hodnotou inkrementálního čítače
- key feedback (KFB)
  - výstup šifrování bloku x_i je použit jako klíč pro šifrování bloku x_{i+1}
- accumulated block chaining (ABC)
  - x_{i-1}, x_i, a c_i jsou šifrovány společně
  - náročnější na výpočet
  - $H_i = P_i \oplus h(H_{i-1})$
  - $C_i = E_k(H_i \oplus C_{i-1}) \oplus H_{i-1}$

http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/

KFB


Constant input
Key
Block Cipher Encryption
o j-1
IV
Output Block
Keystream k j
Plaintext x j
r
Ciphertext c j
encryption
decryption
Plaintext x j
...

ABC


OT1
OT2
OT3
K
E
ŠT1
ŠT2
ŠT3
...

Hashovací funkce

- základní hashovací funkce
  - mapují řetězce o libovolné délce na řetězce o konstantní délce
  - např. kontrolní součty, CRC…
- užití
  - kontrola integrity, jedinečnost
  - jednoduché a krátké vyjádření složitějšího…
  - např. otisk software ke stažení
- kryptografické hashovací funkce
  - je to jednosměrná funkce
  - je těžké najít dva vstupy, které se mapují na stejnou hodnotu
  - máme danou výstupní hodnotu y, je obtížné najít vzor *x* pro který platí $h(x) = y$
  - mějme vzor x_1, je těžké najít vzor x_2 takový, že $f(x_1) = f(x_2)$
    - tj. odolnost proti kolizím…

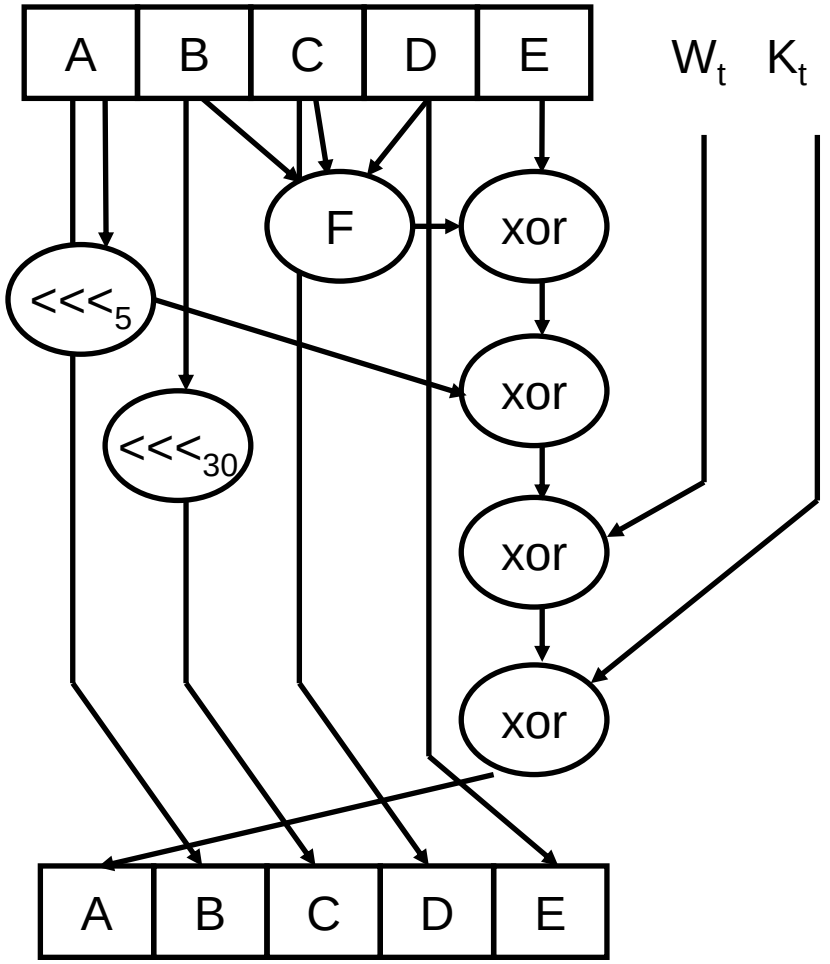
SHA-1

• založen na MD4, definován standardem (NIST)
• výstup 160 bitů

A,B,C,D,E – 32 bitové slova
F – měnící se nelineární funkce
<<< kruhový posun vlevo

80 opakování

SHA-256, 384, 512



SHA-1 příklad

- SHA1("The quick brown fox jumps over the lazy dog") = 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
- SHA1("The quick brown fox jumps over the lazy cog") = de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

Současnost:

- nejčastěji používané hashovací funkce procházejí krizí, kvůli vynikajícím kryptografickým výsledkům (Klíma)
- běžně používané implementace:
  - MD5 – nástupce MD4 – výstup dlouhý 128b., narozeninový paradox => efektivní bezpečnost 64 bitů…
  - rodina SHA

Message authentication code (MAC)

- autentizační kódy zpráv
- krátká zpráva, která umožňuje autentizovat obsah zprávy
- založeno na hashovací funkci, tajemství je přiřetězeno na konec zprávy
- alternativa: využití blokových šifer – heslo jako klíč nebo jeho kombinace se zprávou
- 2 funkce:
  - ověření autentičnosti zprávy
  - ověření integrity zprávy

MAC – PMAC/OMAC

- one-key MAC
- založen na blokových šifrách – např. AES
- šifra použita v CBC módu
  - pouze sekvenční zpracování

- P jako paralelní – Rogavay
  - klíč použit ke generování náhodné sekvence, která se xoruje s blokem zprávy
  - xor je komutativní => výpočet můžeme paralelizovat

MAC - HMAC

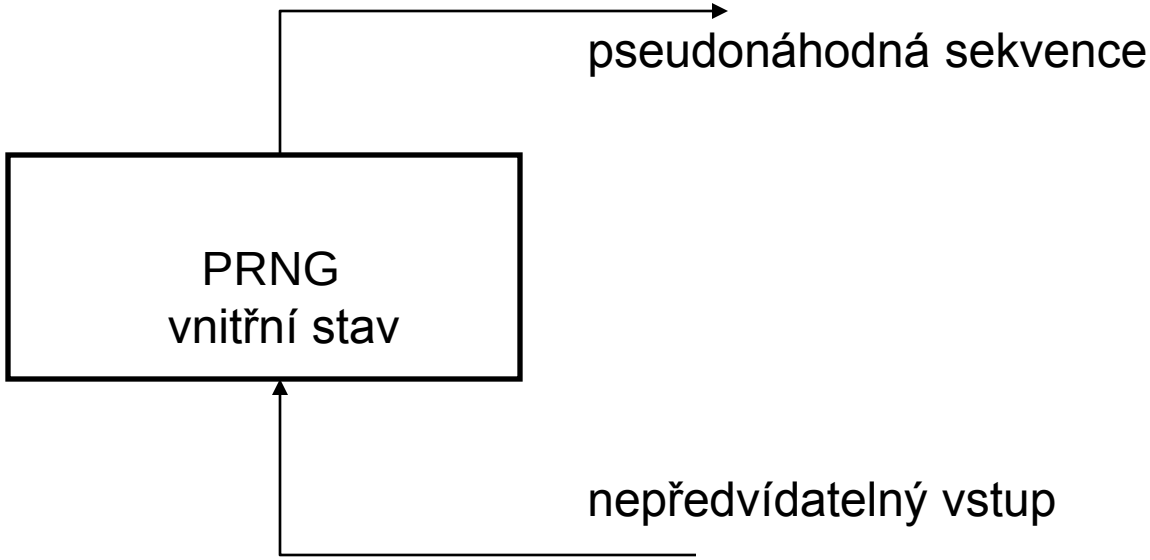
- $HMAC_K(m)=h(K \oplus opad||h((K \oplus ipad)||m))$

- RFC2104
- h je iterovaná hashovací funkce (možnost definovat počet iterací hashe pro každý blok)
- K je tajný klíč doplněný potřebným počtem nul
- opad = outer padding – 0x5c5c…5c
- ipad = inner padding – 0x3636…36

} dvě hexadecimální konstanty o délce jednoho bloku

- hashovací funkce jsou rychlejší <1 cycle/bit, block ciphers <4 cycles/bit

Pseudorandom number generators (PRNG)



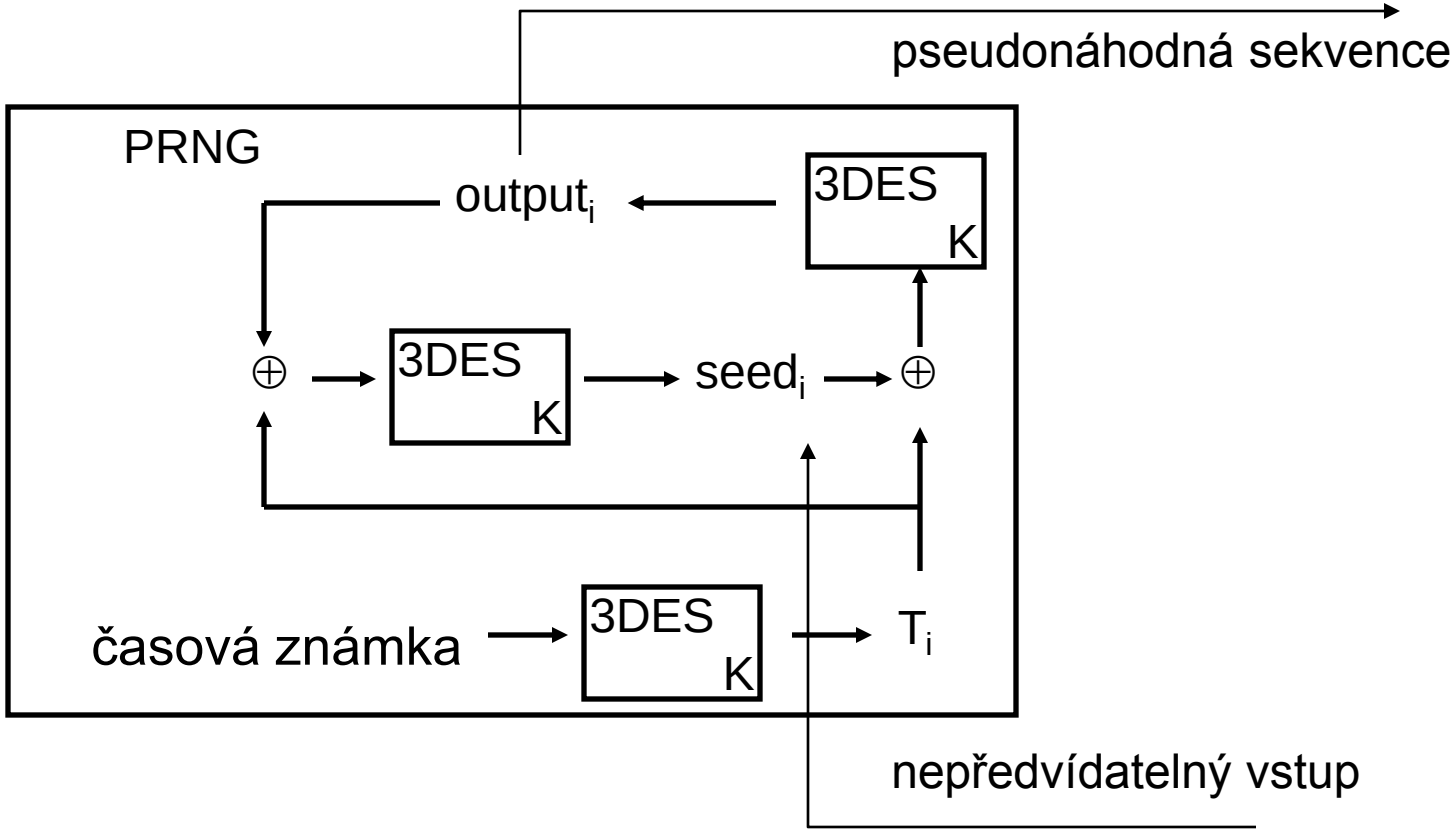
Pseudorandom number generators (PRNG)

- data vygenerovaná pomocí funkce => nejsou úplně náhodná
- PRNG – dostatečné pro běžné počítačové aplikace a různé simulace
- kryptografie je náročnější => definice kryptograficky bezpečných PRNG
- bezpečnost založena na 3 předpokladech:
  1. nepředvídatelnost vstupu
  2. bezpečnost vnitřní funkce PRNG
  3. důvěrnost vnitřního stavu

Útoky na PRNG

- kvalita PRNG je odvozena od obtížnosti odlišit výstup od náhodné sekvence
- útočník má dva řetězce – PRNG a náhodný
- přímý kryptoanalytický útok
  - analýza vnitřní funkce, rozlišení řetězců přímým útokem na kryptografickou bezpečnost
- útok založený na vstupu
  - útočník kontroluje vstupní data
- rozšíření kompromitace stavu
  - využití znalosti vnitřního stavu pro identifikaci výstupu
  - backtracking, trvalá kompromitace, meet-in-the-middle, iterativní hádání

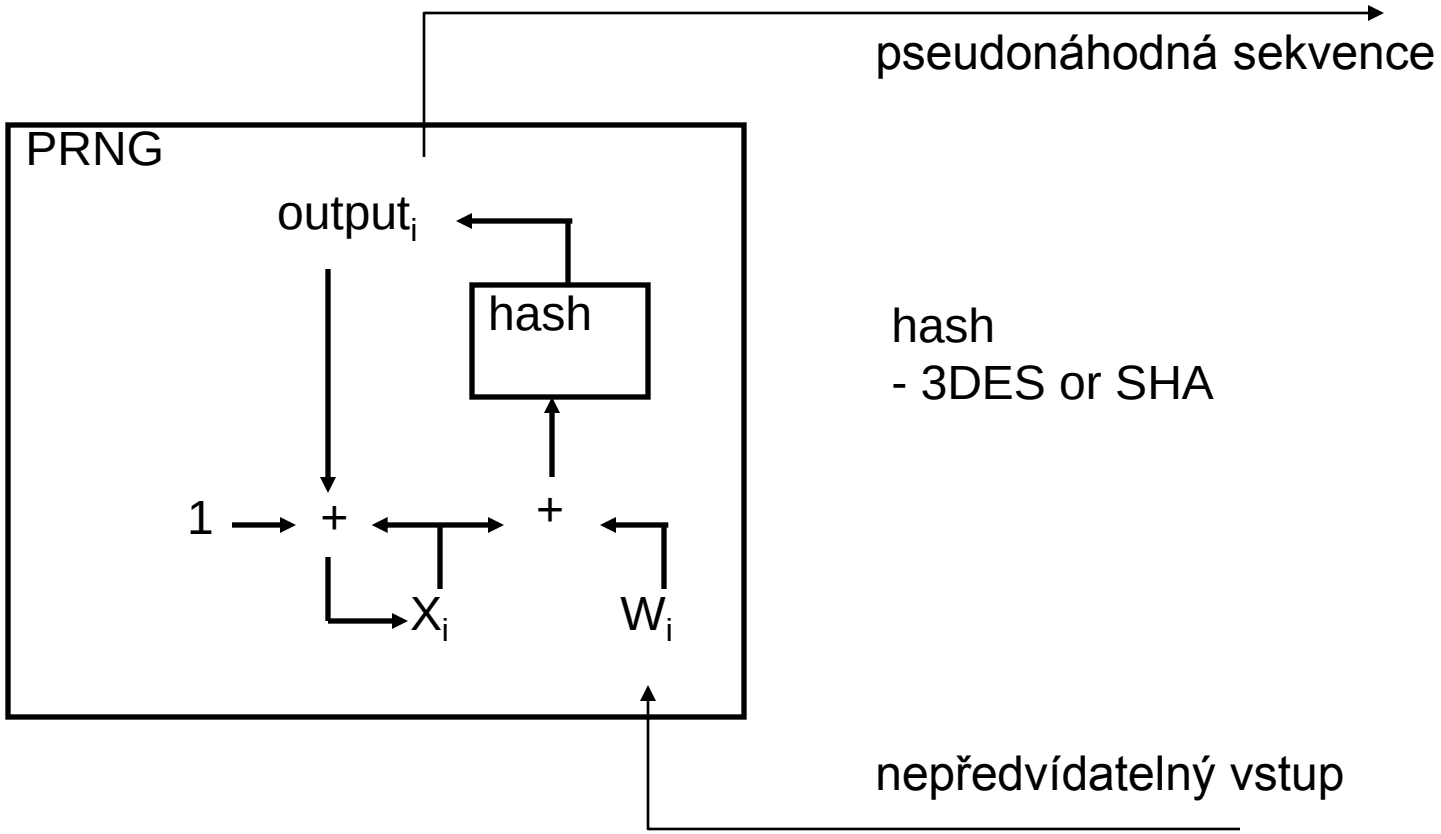
ANSI X9.17 PRNG



ANSI X9.17 PRNG

- state compromise extension attack – with the K compromised, the security of PRNG never recovers
- backtracking works as well as moving forward
- meet-in-the-middle – finding seed i and i+8, guessing $T_{i+1}…T_{i+4}$, and backwards T_{i+5},T_{i+8} correct $seed_{i+4}$ will be in both lists => bits of entropy are halved

DSA PRNG - NIST

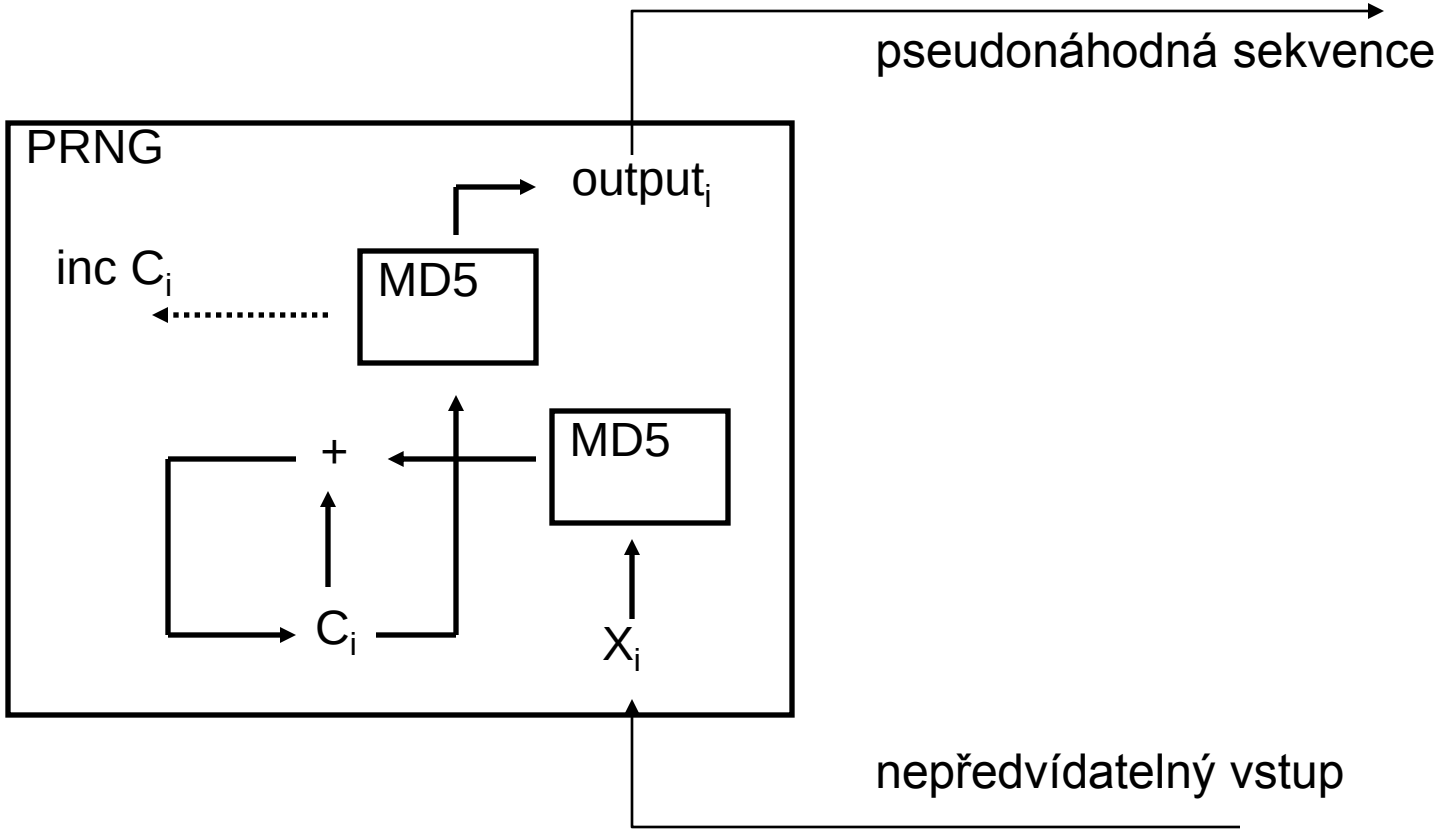


DSA PRNG – NIST

- Input-based attacks – we can repeat outputs
  $W_i = W_{i-1} - output_{i-1} - 1$
- Filling in the gaps
  $output_i = X_{i+2} - X_{i-2} - output_{i+1}$

- not good as a general purpose PRNG

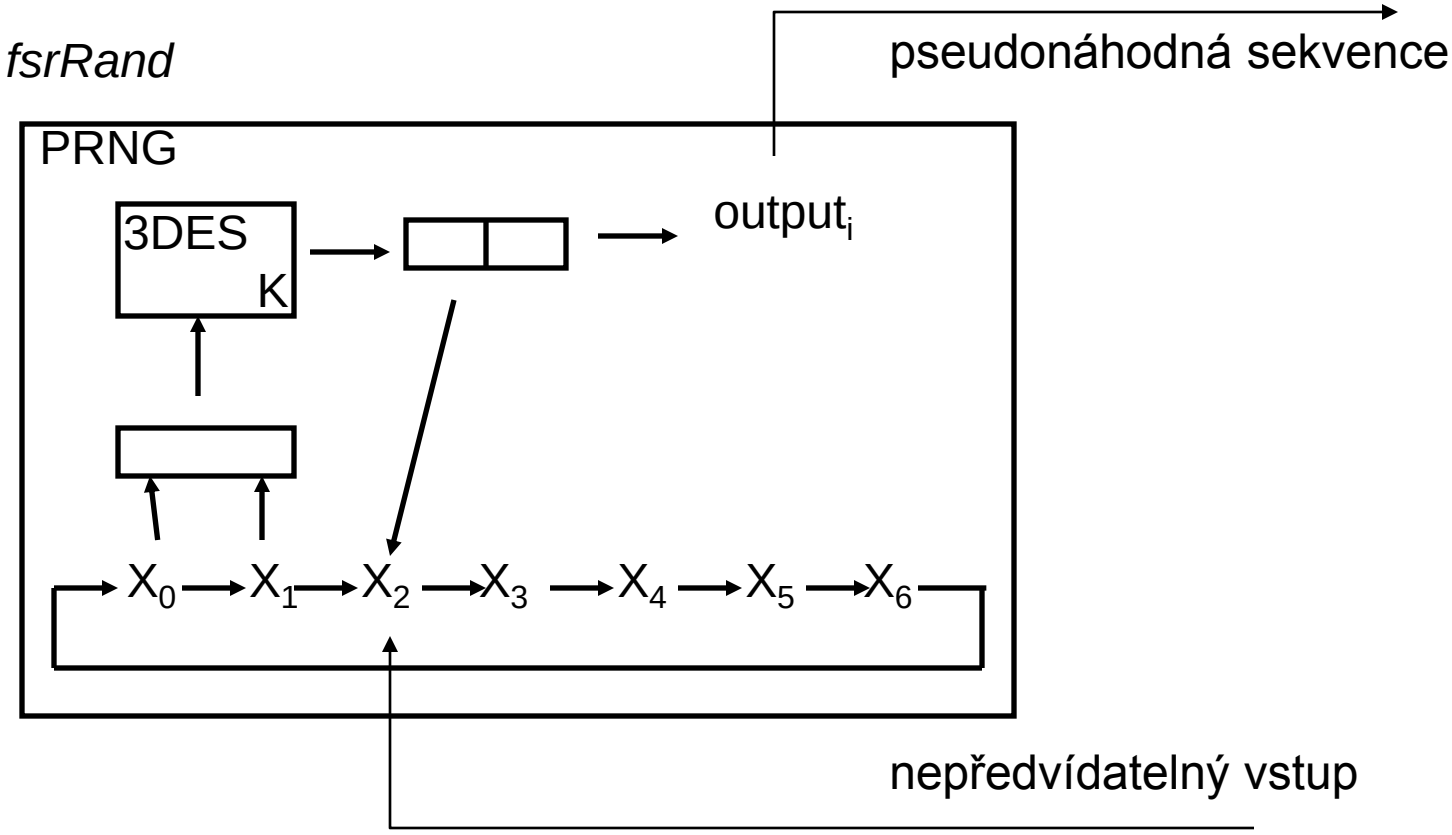
RSAREF PRNG


pseudonáhodná sekvence
PRNG
output$_i$
inc C_i
MD5
MD5
+
C_i
X_i
nepředvídatelný vstup

RSAREF PRNG

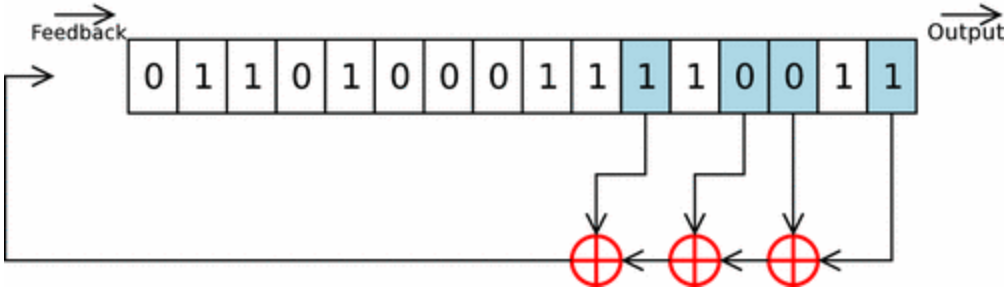
- chosen input attack – to shorten cycle
- chosen-input timing attack – to uncover secret state
- iterative guessing, backtracking
- inputs affect state in an order-independent way

Cryptlib's PRNG



Proudové šifry

- symetrické šifry pracující nad jednotlivými bity
- též jako generátory pseudonáhodných čísel
- LFSR (linear feedback shift register) – základní proudová šifra

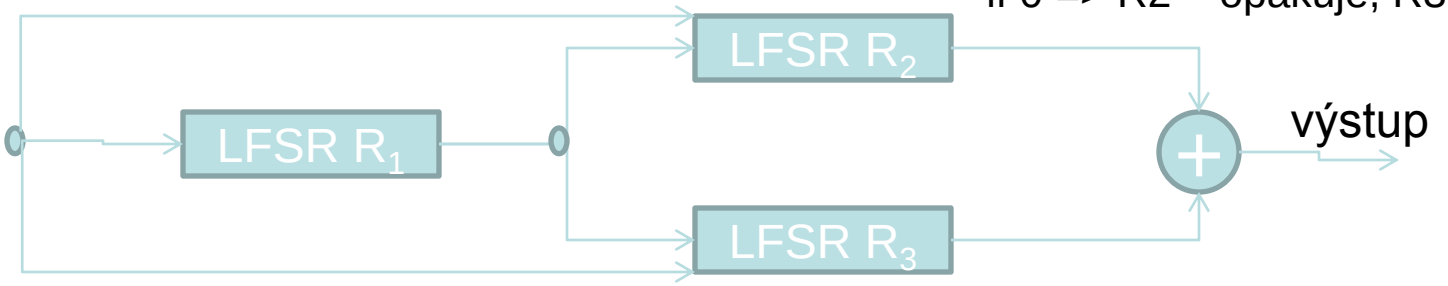


- A5 (šifrování GSM) je založeno na LFSR (19,22,23 bitů)

Generátory řízené hodinami

- stop and go generator
  - 2 LSFR – je li výstup prvního 1, tak se spouští druhý, jinak pouze opakuje svůj předchozí výstup
- alternating step generator

if 1 => R2 – pracuje, R3 – opakuje
if 0 => R2 – opakuje, R3 – pracuje



- shrinking generator