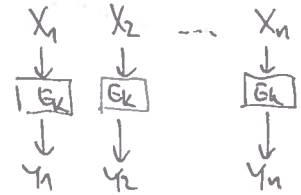
POUŽITÍ BLOKOVÝCH ŠIFER aneb šifrovací módy

- padding – musí být reversibilní!
  - 80 00 ... 00 (P-1)
  - P P P ... P (P)
  - 00 00 ... 00 P (P-1)

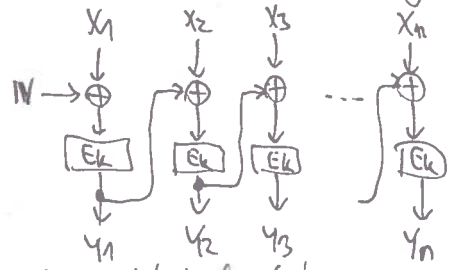
(?) chceme kontrolovat, že padding má správný formát? (nebo náhodné bajty)

- ECB (Electronic Code Book)



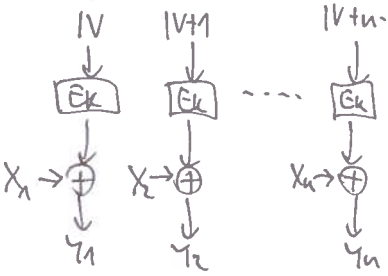
  - fatálně rozbitý, nepoužívat!
  - odhaluje rovnost bloků
  - nemá žádnou IV
  - změna bitu v Y_i změní celý X_i, ostatní X_j nedotčeny
  - vynechání/prohození bloků nevšimnes nedokáže

- CBC (Cipher Block Chaining)



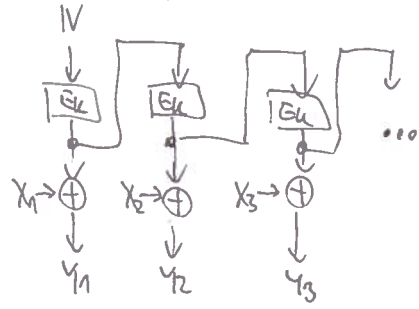
  - rozumnější dešifrování
  - IV má být náhodný (jinak fail)
  - má důkaz bezpečnosti proti CPA
  - změna bitu v Y_i změní celý X_i a bit v X_{i+1}
  - vynechání/prohození bloků ovlivní tyto bloky a 1 následující
  - pokud zpráva není moc dlouhá? Jinak se zopakují bloky ciphertextu → zjistím rovnost některých bloků plaintextu

- CTR (Counter)



  - proudová šifra → netřeba padding
  - nesmíme zopakovat IV!
  - bit flip v Y_i ⇒ bit flip v X_i
  - lze paralelizovat & má random access

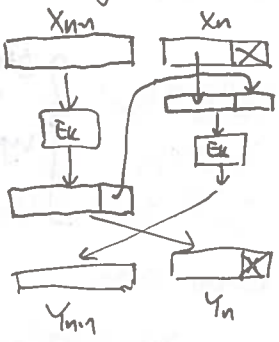
- OFB (Output FeedBack)



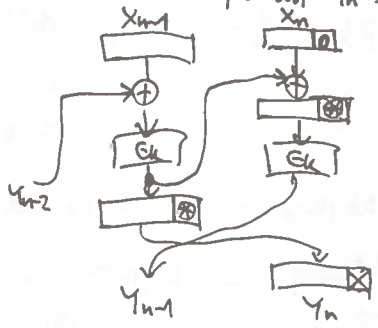
  - také proudová šifra
  - pozor na krátké cykly
  - keystream jsou vlastně 0* šifrované CBC

- Ciphertext stealing – jak se vyhnout paddingu

u ECB:



u CBC: stačí dopadovat nulami a prohodit Y_n s Y_{n-1}



→ část dat šifrujeme 2x

→ propagace chyb se chová trochu jinak u posl. 2 bloků

Další zajímavé blokové šifry z finále AES:

- Serpent: bloky 128b, klíč 128–256b, 32-rundová SPN + lineární transf.
  - velmi konzervativní, nevyužil kvůli pomalosti
- Twofish: bloky 128b, klíč do 256b, 16-rundová feistelovská síť
  - pomalá inicializace (key schedule), S-boxy vypočítá z klíče

Padding oracle attacks

- ukážeme pro CBC s paddingem typu $P \dots P$ (P) } předpokládáme orákulum, které nám řekne, jestli dešifrovaná zpráva má správný padding
- měníme bity v posl. bytu Y_{n-1} → to mění jednak X_{n-1}, ale hlavně odpovídající bity X_n

→ jinak nechceme útok selhat

  - pokud $P \neq 01$: právě jedna změna vede na korektní padding (totiž $P' = 01$)
  - víme tedy P → umíme ho nastavit na 02
  - najdeme předposl. byte, se kterým bude padding OK ⇒ to musí být 02 ⇒ víme, jaký byl původně
  - teď nastavíme posl. 2 byty na 03 03 a pokračujeme ...

... až rekonstruujeme celý posl. blok, zprávu zkrátíme a pokračujeme → nakonec vyluštíme vše kromě 1. bloku (ten jen pokud můžeme ovlivňovat IV)

→ složitost útoku = 256 · délka zprávy.

to může být dlouhá doba nebo nějaký postranní kanál (třeba čas)

↓

uvidíme útoky tohoto typu na SSL/TLS

Prosakování informací z módů blok. šifer

ECB: $X_i = X_j \Leftrightarrow Y_i = Y_j$

CBC: Pokud $Y_i = Y_j$: $E_k(X_i \oplus Y_{i-1}) = E_k(X_j \oplus Y_{j-1})$

$X_i \oplus Y_{i-1} = X_j \oplus Y_{j-1}$

$X_i \oplus X_j = Y_{i-1} \oplus Y_{j-1}$

} jednou za průměrně $2^{b/2}$ bloků vyzradím b bitů ($Y_0 = IV$)

Naopak pro $Y_i \neq Y_j$ dostanu nerovnost XORů.

CTR: Všechny bloky keystreamu $C_1 - C_m$ jsou navzájem různé?

Takže $Y_i \oplus Y_j = (X_i \oplus C_i) \oplus (X_j \oplus C_j) = (X_i \oplus X_j) \oplus (C_i \oplus C_j)$

$\neq X_i \oplus X_j$.

→ vyzradím: pro každý pár X_i, X_j

párů → $\binom{m}{2} \cdot (b - \log(2^b - 1))$

$\log \frac{2^b}{2^b - 1} = \log\left(1 + \frac{1}{2^b - 1}\right) \doteq \frac{1}{2^b - 1} \doteq 2^{-b}$

informace z párů nemusí být nezávislé → je to horní odhad

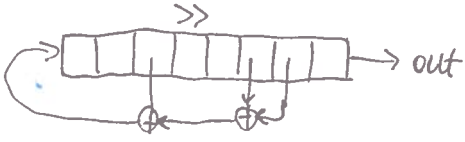
pro max $2^{b/2}$ že to konst. # bitů

⇒ chci šifrovat méně než $2^{b/2}$ bloků, abych prosakování minimalizoval.

PROUDOVÉ ŠIFRY

- Známe jich celkově málo
- eSTREAM project – evropský projekt hledající nové proudové šifry
  - začal v roce 2004, finále 2008
  - Profile 1 (SW): 4 šifry
  - Profile 2 (hw): 3 šifry

LFSR Linear-Feedback Shift Registers



$y_{n-1} - y_0 \rightarrow y_n - y_1$

kde $y_n = \bigoplus_{i=0}^{n-1} c_i \cdot y_i$ (c_i = klíč)

Neznáme klíč a počáteční stav registru.

- pro vhodně zvolené klíče má perioda $2^n - 1$
  ↑ to lze hezky popisovat pomocí algebry polynomů
  ... ale snadno podlehne known-plaintext útoku:
  - z prvních n bitů výstupu přečteme iniciální stav
  - z dalších n bitů sestavíme lineární rovnice pro klíč
    ... pro max. periodu vždy vyjde regulární soustava

Pokusy o nápravu:
- nelineární feedback (je těžké zaručit dlouhou periodu)
- nelineární výstup (kombinujeme víc bitů registru)
- nelin. kombinace výstupů různých registrů (perioda se prodlužuje na LCM)
- výstup jednoho registru řídí hodiny jiného

Šifra A5/1 v GSM (prolomená)

Trivium - eSTREAM hw profile
- 3 registry různých délek, celkem 288 bitů
- nelineární zpětné vazby (kombinace AND a XORů)
- lineární generování výstupu
- init: registry naplním 80b klíče + 80b IV + konstanty a provedu 1152 kroků naprázdno
- zatím není známý útok složitosti menší než 2^{80}, ale některé zkrácené varianty (rychlejší init) lze prolomeny
- trik: 3 prvních 65 bitů každého registru nic nevede ⇒ výpočet lze paralelizovat.

RC4 (Rivest 1987) - šifra založená na permutacích, vhodná pro SW

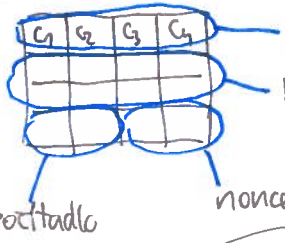
Stav: S[0...255] permutace na 0...255
indexy i, j

Kroky:
$i \leftarrow (i+1) \bmod 256$
$j \leftarrow (j + S[i]) \bmod 256$
$S[i] \Leftrightarrow S[j]$
output $S[(S[i] + S[j]) \bmod 256]$

Init: 256 kroků
k j navíc přičítám i-tý znak klíče (cyklicky)

Ještě nedávno dost populární ... netrpěla na útoky na padding (16)
- statistické útoky: stav se nepromíchává dostatečně, z korelací mezi byty jde spočítat klíč
  → 2015: použití v TLS rozbito za 75 hodin
  použití ve WPA-TKIP za 1 hodinu
  (mnohem dřív: použití ve WEP rozbito kvůli related keys)

ChaCha20 (odstupce Salsa20 z eSTREAMu, SW profil)
rund šifry (Bernstein 2008)

- 256-bit. klíč, 64b počítadlo bloku, 64b nonce } pozor, různé verze mají různě rozdělen. bitů mezi počítadlo a nonci
  funguje skoro jako CTR mód blokové šifry
- stav je matice 4×4 32b čísel
- init: 

  konstanty: ASCII "expand 32-byte k"
  tzv. ARX-šifra
- čtvrtina rundy (QR): kombinace XORů, a rotací aplikovaná na 4 políčka matice
  sčítání
- sudá runda: QR na sloupce
  lichá runda: QR na toroidní diagonály
- má elegantní implementaci vektorovými instrukcemi
- výstup se nakonec přičte k poč. stavu (po složkách)