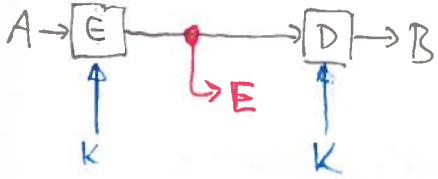
Úvodem

- Cíle: • přehled o kryptografii teoretické i praktické
  - → kryptografická primitiva
  - → protokoly
  - → implementační otázky
  - cílem je rozumět existujícím protokolům … a vědět dost o návrhu vlastních, abychom to nechtěli dělat :o
- nebudeme budovat hlubokou teorii (→ Foundations of Theor. Cryptl. & lectures 2 MI1B … ale občas nějakou větu dokážeme :o
- o obtížnosti návrhu bezpečných systémů (a útoků se obecně)
- prerekvizity (tak trochu): algoritmy, architektura HW, algebra, složitost … ↓ náhodný uživatel vs. zlomyslný uživatel ↓ nechci kódy, attack trees

Primitiva

Scénář: • Alice chce posílat Bobovi šifrované zprávy
(to obecně nejsou konkrétní osoby, ale role v protokolu → posílání oběma směry (prohození rolí) → Bob může být třeba Alice v budoucnosti)

- Eva poslouchá [< eavesdropping]
- Mallory data upravuje (↑ (ale o tom později) The Man in the Middle)



- hodí se E a D parametrizovat klíčem
- Kerckhoffsův princip: tajný má být klíč, ne algoritmus!

Rationale: ② je-li šifra veřejně známá, bývá lépe otestovaná ②
③ vyměnit kompromitovaný klíč je snazší než algoritmus
① dobrých šifer je málo a je těžké je vytvořit

→ symetrická šifra (E i D používají stejný klíč)

šifra obvykle nemůže utajit délku zprávy.
↓
typ. sym. šifra délku zachovává
↓
formálně:
$E: \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$
$D:$ — " —
$\forall K \; \forall X \; D(x, E(x,K)) = X$
& pro náhodný klíč se $E(-,K)$ chová jako náhodná permutace na $\{0,1\}^n$

příklad: Caesarova šifra

② Asymetrická šifra

- dvojice šifrovací a dešifrovací klíč
- typické aplikace:
  - → N lidí komunikujících navzájem
    - šifrovací klíč je veřejný
    - dešifrovací je tajný
    - problémy s distribucí klíčů!
  - → digitální podpisy
    - šifrovací klíč tajný, dešif. veřejný
    - každý může podpis ověřit, ale jen 1 osoba vytvořit

③ Hešovací funkce: $\{0,1\}^* \to \{0,1\}^b$ (třeba $b = 256$)

Chceme:
- nemožnost inverze
- nemožnost nalezení kolize

"dostatečně náhodná"

- typické aplikace:
  - → kompaktnější podpisy (nechceme kolize!)
  - → Message Auth Code (symetrická verze podpisu)

④ Náhodné generátory

Chceme:
- nepredikovatelnost
- neovlivnitelnost

- aplikace:
  - → hybridní šifra ze symetrické a asymetrické
  - → challenge–response autentikace

Společné cvičení: protokol pro aukci (viz Šmrž)
- → padding
- → timestamps / seq. numbers (proti replayování)
- → nonce (proti porovnávání šifrovaných zpráv)
- → session ID (proti replayi jiné instance protokolu)

Modely útoku - proti komu se bráníme ← DÚ: házení mincí po telefonu
- jak dlouho musí tajemství vydržet

↓ commitment pomocí hes. fcí

Typy útoků
- known ciphertext (chceme plaintext)
- known plaintext (chceme klíč)
- chosen plaintext } též chceme klíč
- chosen ciphertext & known plaintext
- rozlišovací útoky

Jak měřit obtížnost útoku? → security level

"Narozeninové" útoky

① Challenge-response autentikace, n různých nonců
… kolik pokusů v průměru potřebujeme, než se nonce zopakuje?

$\Pr[\text{náhodná } f \text{ z } [m] \text{ do } [n] \text{ je prostá}]$ (m … pokusy, n … nonce)

$$= \frac{\#\text{prostých fcí}}{\#\text{všech fcí}} = \frac{n^{\underline{m}}}{n^m} = 1 \cdot \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right) \cdot \_\_ \cdot \left(1-\frac{m-1}{n}\right)$$

Jelikož $1-x \approx e^{-x}$, aproximujeme $1 \cdot e^{-\frac{1}{n}} \cdot e^{-\frac{2}{n}} \cdot \_\_ \cdot e^{-\frac{m-1}{n}}$

$$= e^{-\frac{1+2+\dots+m-1}{n}} = e^{-\frac{m(m-1)}{2n}}$$

Chceme $\Pr[\text{kolize}] = \frac{1}{2} \rightarrow e^{-\frac{m(m-1)}{2n}} = \frac{1}{2} \Rightarrow \frac{m(m-1)}{n} = -2\ln\frac{1}{2} \doteq 1.38$

$\Rightarrow$ přibližně $m \approx \sqrt{n}$ $\Rightarrow$ security level je poloviční

② Protokol: • A zvolí náhodný klíč (a pošle ho zašifrovaného tajným šifrou) (4)
↓ z množiny velké n
• A pošle uvítací zprávu podepsanou klíčem
• další zprávy podepsané stejně

Útok: E si předpočítá podpisy uvítací zprávy pro a náh. klíčů
Pak poslouchá m relací a čeká, až se objeví předpočítaný klíč

Model:

$$Pr[\text{f se strefí do } A] = \left(1-\frac{a}{n}\right)^m \approx e^{-\frac{am}{n}}$$

... to je konstanta pro $n \approx am$.

⟶ trade-off mezi časem na předvýpočet a délkou útoku.

! Pozor, security level je někdy 2x menší, než bychom čekali !

Jednorázové klíče – Vernamova šifra (a.k.a. One-time Pad)

- zpráva $x \in \{0,1\}^n$, klíč náhodný $k \in_R \{0,1\}^n$ → $x \oplus k \in \{0,1\}^n$
  $E(x,k)$
  - E a D jsou tatáž funkce
  - výsledek je posloupnost n nezávislých náhodných bitů!
    ... ošem korelovaných s klíčem
- Jiná podobná konstrukce: $x \in \mathbb{Z}_{2^n}$, $k \in \mathbb{Z}_{2^n}$, $E(x,k) = x+k$, $D(y,k) = y-k$ } v $\mathbb{Z}_{2^n}$
  ↑ funguje v jakékoli grupě

∀y ∀x ∃! k: $E(x,k)=y$

⇒ $Pr_k[D(y,k)=x]$ je pro všechna x stejná

⇒ y nenese žádnou informaci o x (kromě délky)

Def. perfektní bezpečnosti

→ k čemu je to dobré? → code books

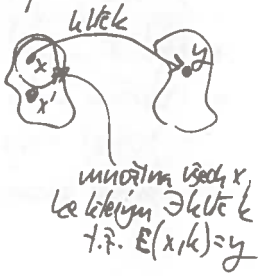
Ale pozor!

- nesmíme nikdy zopakovat klíč (viz Soneti ve IW2)
- útočník může zprávu triviálně měnit

Věta: Pokud #klíčů < #zpráv, šifra není perfektně bezpečná.

Důk: Nechť $y \in \{0,1\}^n$

Pak $\exists x, x' \in \{0,1\}^n : \exists k : E(x,k) = y$
ale $\forall k' : E(x',k') \neq y$



Proto $Pr_k[D(y,k) = x] > 0$,
ale $Pr_k[D(y,k) = x'] = 0$
⇒ rozdělení není rovnoměrné.

Dělení tajemství (aneb o silnějších generátorech)

① $x \to x^1, x^2$ t.ž. samotné x^i mi neřekne nic o x (kromě délky),
ale x^1, x^2 dohromady určí x jednoznačně.

Řešení: x^1 náhodné, $x^2 := x \oplus x^1$

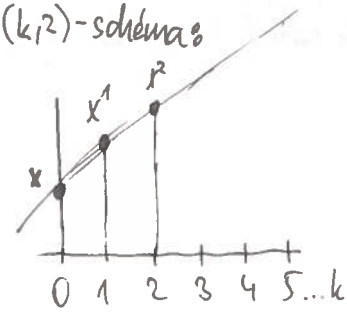
② $x \to x^1, \_, x^k$ t.ž. všech k částí určí x jednoznačně,
žádných $k-1$ nic neprozradí.

Řešení: $x^1 \_ x^{k-1}$ náhodné, $x^k := x \oplus \bigoplus_{i=1}^{k-1} x^i$.

Obecně: (k,l)-prahové schéma rozděluje zprávu na k částí tak, že
- libovolných l částí určí celé x,
- žádných $l-1$ nic neprozradí.

⇒ pomocí ③ sestrojíme (k,k)-schéma.

③ $(k,2)$-schéma:



Vezmu $f(t) = at + b$
t.ž. $f(0) = x$ } tahací f
$f(1)$ je náhodné } existuje právě 1

a pak volím
$x^1 = f(1), \_, x^k = f(k)$

aby to bylo dobře def., počítám v konečném tělese (dost velkém)

👁 libovolná dvě x^i, x^j jednoznačně určí f,
ale pokud znám jen x^i, všechna x jsou stejně pravděpodobná (každému odpovídá právě jedna f).

④ obecné (k,ℓ) - schéma

- f bude polynom stupně menšího než ℓ nad konečným tělesem
- $f(0)=x$, $f(1)$ až $f(\ell-1)$ volím náhodně } to jednoznačně určí f (& všechny f jsou stejně pravděpodobné)
- rozdám části $f(1)$ až $f(k)$

→ pokud znám ℓ částí, určím jednoznačně f a najdu $f(0)$

→ pokud znám $c<\ell$ částí: pokud liberálně nastavím dalších $\ell-c-1$ částí, každá volba x určí právě jeden f
→ všechna x jsou stejně pravděpodobná

Lemma: Pokud p je polynom s kořeny $\alpha_1 - \alpha_t$, pak
$p(x) = (x-\alpha_1)\cdot\ldots\cdot(x-\alpha_t)\cdot q(x)$ pro nějaký polynom q bez kořenů.

Věta: Polynom stupně d má nejvýše d kořenů.
(nenulový)

Důsledek: Pokud p,q jsou polynomy stupně menšího než d a $p(x_i)=q(x_i)$ pro navzájem různá (nejaká) $x_1 - x_d$, pak $p=q$.

Věta (Lagrange): $\forall$ $x_1 - x_d$ navzájem různá $\forall$ $y_1 - y_d$
$\exists p$ polynom stupně $<d$ t.ž. $\forall i$ $p(x_i)=y_i$.

↑ z předchozího víme, že je jednoznačný.
⇒ máme bijekci mezi polynomy stupně $<d$ a vektory $(f(x_1),\ldots,f(x_d))$ pro libovolné čísla navzájem různé $x_1 - x_d$.

SYMETRICKÉ ŠIFRY

Dva základní druhy → proudové (stream ciphers)
→ blokové (block ciphers)

proudové: generují keystream, se kterým se data xorují

(vlastně Vernamova šifra s pseudonáhodným generátorem)
- D = E (inverzní sama k sobě)
- nesmíme opakovat nonci
- znegování y_i zneguje x_i
- E_K, $E_{K'}$ komutuje

Více pozdějí.

blokové:
- šifrují bloky pevné délky b
  $E: \{0,1\}^b \times \{0,1\}^k \to \{0,1\}^b$
  často značíme $E_K: \{0,1\}^b \to \{0,1\}^b$
- E_K musí být invertibilní: je to permutace na $\{0,1\}^b$
- delší zprávy šifrujeme po blocích (TODO)

Triviální příklady

- Caesarova šifra má 1znakové bloky, permutace je cyklický posun abecedy o klíč.
  - Bug #1: málo klíčů → triviální brute-force útok
  - Bug #2: krátké bloky, nulová interakce mezi nimi → frekvenční analýza
- Vigenèrova šifra: víceznakové bloky, opět přičítání klíče.
- obecné permutace abecedy nebo větších bloků

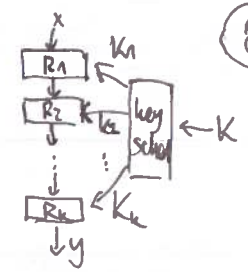
na tohle se dá dívat i jako na proudové šifry

Bezpečnost blok. šifer

- Těžké definovat formálně (buď to umí útoky obejít, nebo definici nesplňuje žádná rozumná šifra)
- Idea: šifru nelze efektivně rozlišit od náhodné permutace
  - verifikátor dostane orákulum buď s E_K pro náhodný K, nebo s náhodnou permutací
  - má odpovědět, které orákulum dostal
  - může položit více dotazů
  - chceme, aby nešla dosáhnout $Pr[\text{úspěch}] \geq 2/3$ s lepší složitostí než $\sim 2^{\text{security level}}$
    ↳ co to je?
- Tohle nepokrývá chosen-key/related-key útoky!
- časem prostudujeme další algoritmy

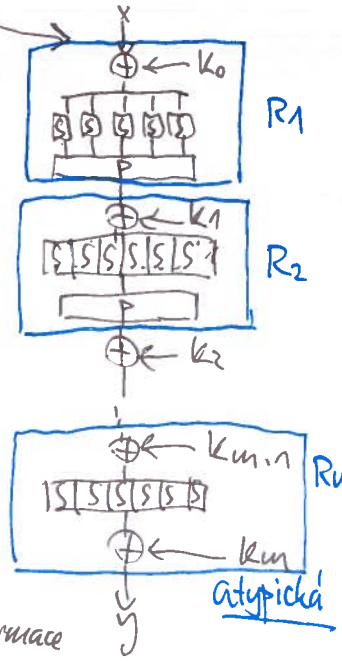
! Reálné šifry jsou prakticky vždy sudé permutace

DES (Digital Encryption Standard):



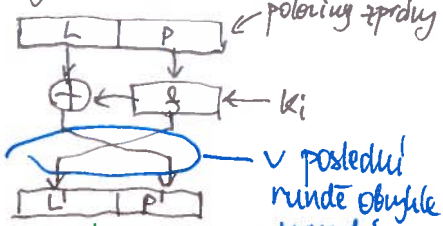
Odbočka: o způsobech konstrukce blok. šifer

- iterované šifry, rundy, rozvrh klíčů
- substitučně-permutační sítě (SPN)
  - S-boxy: malé tabulky, musí být invertibilní
  - počáteční a koncový XOR: whitening (omezuje kontrolu útočníka nad vstupy do S-boxů
  - P-box: obecná permutace na pozicích v bloku
  - díky atypické R_n je inverze k SPN zase SPN (někdy volíme S, P jako involuce → tatáž SPN, jen obrátíme rozvrh klíčů)
  - confusion vs. diffusion
  - upgrade: kromě P zavádět invertibilní lin. transformace



Hranice rund se posunou, P a ⊕ komutují, pokud permutujeme rundový klíč

- Feistelovy sítě
  - konstrukce s nevertibilními S-boxy
  - runda obecně vypadá takto:
  - $f(P, K_i)$ může být libovolná funkce (typ. postavená z S/P-boxů)
  - inverze je zase Feistelova síť, jen se obrátí pořadí rundových klíčů

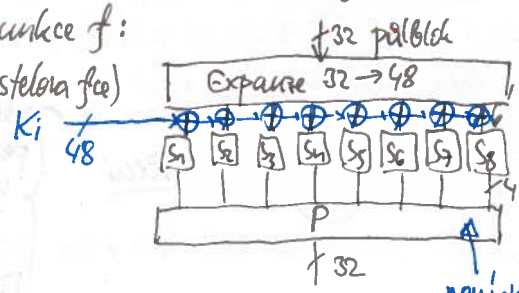


Historie DESu:

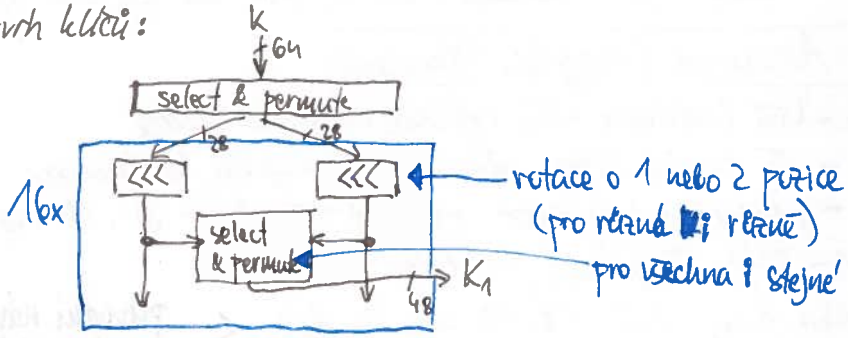
- vyvinut začátkem 70. let v IBM na zakázku NBS (Nat. Bureau for Standards), do vývoje mluvila i NSA
- 56-bitový klíč (technicky 64, ale v byte má paritní bit)
  - původní verze byla silnější
- NSA na poslední chvíli vyměnila S-boxy – krajně podezřelé?
  - dnes už víme, že tím šifru zeslabila
- 64-bitové bloky

Struktura DESu:

- Feistelova síť s 16 rundami pracujícími s 32-bitovými půlbloky
- Navíc počáteční a koncový P-box (zcela zbytečné)
- Funkce f:



- Rozvrh klíčů:



Kritika DESu

- Pokud $K=0^{56}$, všechny k_i jsou $0^{48} \rightarrow E_K = D_K$ } 4 tzv. slabé klíče
  podobně pro $K=1^{56}$ a 2 další klíče.
- 6 dvojic klíčů (K, K'), takové, že $K \rightarrow (k_1, k_2, k_1, k_2, \ldots)$ a $K' \rightarrow (k_2, k_1, k_2, k_1, \ldots)$
  Pak: $E_K(E_{K'}(x)) = x$ pro všechna x.
- $E_{\overline{K}}(\overline{x}) = \overline{E_K(x)}$ } komplementárnost
- Příliš krátké klíče!
  - už v roce 1977 se odhadovalo, že za 20 M$ jde postavit stroj, který vyzkouší všechny klíče za 1 den
  - 1997: RSA Security Inc. DES Challenge – cena 10k$
    → crackнuto distrib. výpočtem v idle čase 78k počítačů
  - ⋮
  - 2012: deska s 48 FPGA prohledá celý prostor za 26 hodin
    (pronajímají jako službu?)

- Krátké bloky – kolize bloků jednou za 2^{32} bloků !
- Útoky na strukturu:
  - diferenciální kryptoanalýza: stačí 2^{47} chosen plaintextů
  - lineární kryptoanalýza: stačí 2^{43} známých plaintextů
  - → dost na to, abychom šifru považovali za rozbitou

Pokusy o záchranu DESu (90. léta)

- 2-DES – nepoužije ? → sec. level jen 57 → cvičení
- 3-DES – $E_{K_3}(D_{K_2}(E_{K_1}(x)))$ → 168-bit. klíč, sec. level 112
  - někdy se potká varianta s $K_1 = K_3$, pozor, má sec. level jen cca 80

nebezpečí: permutace mohou tvořit grupu [DES je netvoří]

AES – Advanced Encryption Standard

- 1997 – NIST (nástupce NBS) vypisuje otevřenou soutěž
  - 15 návrhů šifer, několik kol veřejného hodnocení
  - kriteria: bezpečnost, rychlost + snadnost SW i HW implementací
- 2001 – šifra Rijndael prohlášena za AES
- 128-bit. bloky, klíč 128, 192 nebo 256 bitů ← původní návrh uměl i delší klíče a větší bloky

Struktura:

10 rund 12 14 rund

- není to feistelovská šifra, ale SPN s lineární transformací navíc
- bajtově orientovaná (pro efektivní implementaci v SW)
  ↳ s bajty zacházíme jako s prvky $GF(2^8)$
- Stav šifry (předávaný mezi rundami) je matice 4×4 bytů, rundový klíč má stejný tvar.
- Runda:
  - ByteSub – bajty stavu prožene me identickými S-boxy $8 \to 8$ [S-box je inverze v $GF(2^8)$ + afinní transf. + rotací a XORů]
  - ShiftRow – 1-tý řádek rotujeme o i bytů doleva
  - MixColumn – na ∀ sloupec (coby 4D vektor) aplikujeme stejnou invertibilní lin. transformaci
  - AddRoundKey – XOR s klíčem

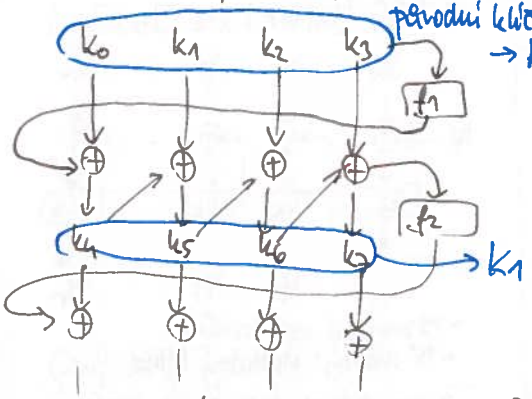
v poslední rundě není

- Před 1. rundou AddRoundKey

- Inverzní runda:
  - AddRoundKey(Ki) } komutuje, pokud Ki nahradíme jeho mixem
  - Inv MixColumn
  - Inv ShiftRow } komutuje
  - Inv ByteSub

  } těchto prohodíme lazením posunem rund
  ↓
  DK vypadá skoro jako EK, jen máme jiné S-boxy a jiné mixování

- Rozvrh klíčů: pracuje po 32-bit. slovech



fi je postavena z S-boxu (téhož jako v rundě), rotace o 1 byte a přimíchání rundové konstanty

add. verze pro 128 bit. klíč → pro 192 bit. je to jen širší, pro 256 bit. je na prostředních ⊕ ještě jedna nelinearita (aplikace S-boxu) ← ki

- Vylepšení implementace na 32-bit. CPU
  - 4 tabulky 8→32 kombinující S-box s částí MixColumns (1 sloupec je XOR 4 lookupů v tabulkách) } 4 kB

Kritika

- jednoduchá algebraická struktura (útok řešením rovnic ... zatím se nedaří)
- příliš malá rezerva v # rund
- zarovnaná na bajty

→ ale zatím se žádný zajímavý útok nedaří. [kromě implementačních — viz později]

→ nějaké related-key útoky, ale stejně mají velkou složitost a tolik rel. keys se těžko shání

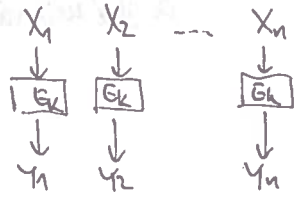
- 128-bit. klíč není bezpečný proti kvantovým počítačům (Groverův alg.)
- 128-bit. bloky hrozí kolizními útoky po 2^{64} blocích
  → obejdeme změnami klíče po ~2^{32} blocích (v protokolu)

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)

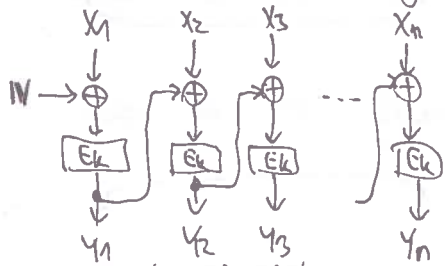
(?) chceme kontrolovat, že padding má správný formát? (nebo náhodné bity)

- ECB (Electronic Code Book)



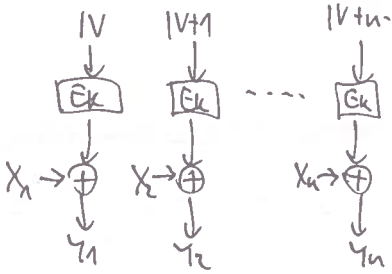
- ~~bezpečné~~ rozbitý, nepoužívat!
- odhaluje rovnost bloků
- nemá žádnou IV
- změna bitu v Y_i změní celý X_i, ostatní X_j nedotčeny
- vynechání / prohození bloků nevnese nežádoucí

- CBC (Cipher Block Chaining)



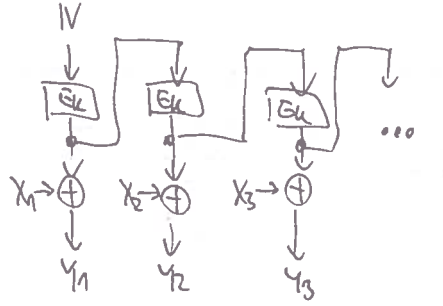
- rozumný šifrování
- IV má být náhodný (jinak fail)
- má důkaz bezpečnosti proti CPA
- změna bitu v Y_i změní celý X_i a bit v X_{i+1}
- vynechání / prohození bloků ovlivní tyto bloky a 1 následující
- pokud zpráva není moc dlouhá!
  jinak se opakují bloky ciphertextu → zjištění (ne)rovnost některých 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

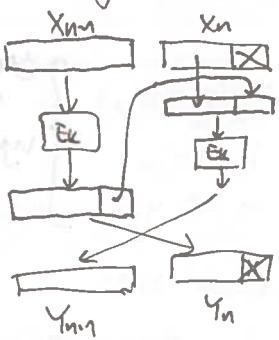
- CFB (Output FeedBack)



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

- Ciphertext stealing – jak se vyhnout paddingu

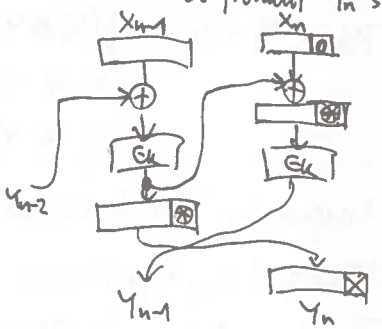
u ECB:



→ část dat šifrujeme 2x

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

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



Další zajímavé blokové šifry z finále AES:

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

Padding oracle attacks

- ukážeme pro CBC s paddingem typu $P \ldots P$ ($\underbrace{\quad}_{P}$) } 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 bit 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 umíme ovlivňovat IV)

→ složitost útoku = 256 · délka zprávy.

to může být chybová hláška nebo nějaký postranní kanál (třeba čas)

↓

uvidíme útok 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 páru nemusí být nezávislé → je to horní odhad

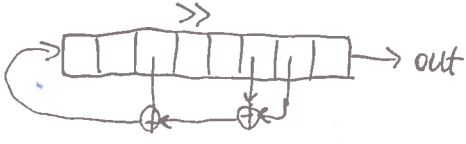
pro $m \approx 2^{b/2}$ je to konst. # bitů

⇒ 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 $y_n = \bigoplus_{i=0}^{n-1} c_i \cdot y_i$ (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 podléhne 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: z prvních 65 bitů každého registru nic nevede ⇒ výpočet lze paralelizovat

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

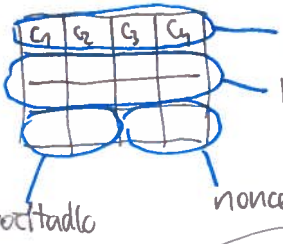
Stav: S[0...255] permutace na 0...255
indexy i, j

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 (nástupce Salsa20 z eSTREAMu, SW profil)
rund — šifry (Bernstein 2008)

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

- čtvrtina rundy (QR): kombinace XORů, a rotací aplikovaná na 4 políčka matice — tzv. ARX-šifra
  (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)
  ↳ to je nutné, protože QR je invertibilní
  (jinak by ze známého páru plaintext + ciphertext šel získat klíč!)

HEŠOVACÍ FUNKCE

Cíl: funkce $h: \{0,1\}^* \to \{0,1\}^b$

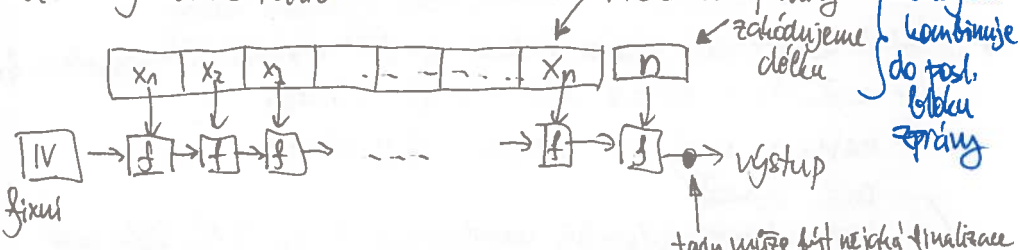
- ideálně nerozlišitelná od náhodné funkce
  … to ale nemůžeme nadefinovat, neboť h nemá klíč
- Typické požadavky: ① Nesmíme najít kolizi: $f(x)=f(x')$ pro $x \neq x'$
  ② Nesmíme najít druhý vzor: pro x nenajdeme x': $f(x)=f(x')$
  ③ Nesmíme invertovat: pro y nenajdeme x t.ž. $f(x)=y$.
- 👁 ③ ⇐ ② ⇐ ①
  ↑ pokud známe $y=f(x)$, inverze y by nejspíš našla jiné x (typicky má nekonečně možností)

Aplikace: Commitment

Merkleova-Damgårdova konstrukce

- pořídíme si kompresní funkci $f: \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^b$ (viz dále)

Pak h vyrobíme takto:



(někdy se kombinuje do posl. bloku zprávy)

- Pokud f je odolná proti kolizím, h je též odolná.

  [dává smysl, pokud kompr. fce má vstupy různě dlouhé → máme větší vnitřní stav než bloky]

  Dk: Nechť $h(x_1 \| \dots \| x_n) = h(x'_1 \| \dots \| x'_{n'})$
  ① Pokud $n \neq n'$, máme kolizi v posl. volání f.
  ② Pokud $n = n'$: — posl. bloky se nerovnají ⇒ kolize v f
  rovnají ⇒ kolize kratších zpráv ⇒ iterujeme

- Kdybych nepřihešovával délku:
  - Při kolizi h najdu postupem pozpátku buď kolizi f nebo inverzi $f^{-1}(IV)$ … ale to jsem nepředpokládal, že jde (z odolnosti f to neplyne).
- Length extension – v tom se liší od náhodné funkce!

Odolnost proti kolizím

- Řádově $2^{b/2}$ vstupů (jakýchkoli – mohou to být smysluplné zprávy) stačí na "narozeninovou" kolizi, ale potřebují paměť $2^{b/2}$
- Málo paměti: $x_{i+1} = h(x_i)$, želva a zajíc se honí po kružnici
  → Jak to udělat se smysluplnými zprávami?
  Pořídím si parametrizované zprávy (bulist, kde si mohu vybrat mezi 2 smysluplnými variantami)
  a pak volím x_{i+1} jako zprávu parametrizovanou $h(x_i)$.
  → Varianta s umožňujícím narozeninovým útokem: (ten už zas potřebuje paměť)
  - "Oběť" je ochotna podepisovat "hodné" zprávy,
    já chci podepsat "zlou" zprávu
  - Pořídím si parametrizovanou hodnou a zlou zprávu
  - Vygeneruji hodně $2^{b/2}$ hodných a $2^{b/2}$ zlých zpráv
  - S velkou PP ∃ průnik obou množin hodně.
- U M-D konstrukce umím v čase $k \cdot 2^{b/2}$ vyrobit 2^k-násobnou kolizi
  - najdu x_1 a x_1' t.ž. $f(IV, x_1) = f(IV, x_1') =: y_1$
  - najdu x_2 a x_2' t.ž. $f(y_1, x_2) = f(y_1, x_2') =: y_2$
    atd. k-krát
  - zprávu mohu libovolně kombinovat z x_i a x_i', vždy vyjde stejný hash.
    → Z toho útok na kaskádování dvou různých hešů
    Z každé části každý je M-D

Kde selhnat kompresní funkci?

- Daviesova-Meyerova konstrukce z blokové šifry:
  $f(a,b) = E_a(b) \oplus b$
  - Proč $\oplus b$? Bez toho: $E_a(b) \to y$, $D_{a'}(y) \to b'$ ... pak $f(a,b) = f(a',b')$.
  - Pozor, rozbije se pro DES: $f(\bar{a}, \bar{b}) = E_{\bar{a}}(\bar{b}) \oplus \bar{b} = \overline{E_a(b)} \oplus \bar{b} = E_a(b) \oplus b = f(a,b)$
  - Věta: Je-li E/D ideální šifra, f je odolná proti kolizím.
    Přesněji: útočník, který zavolá E/D q-krát, najde kolizi s pstí $\le q^2/2^b$
    Dk: Bílý útočník nevyhodnocuje E/D redundantně. pro $q < 2^{b/2}$
    Pokud se zeptá na $E_x(y)$, dozví se $f_1(x,y) = E_x(y) \oplus y$.
    Pokud na $D_x(y)$, dozví se $f(x, D_x(y)) = y \oplus D_x(y)$.

Při i-tém pokusu nastane kolize, pokud se strefím do některé z $i-1$ předchozích hodnot ... pro $\forall$ cílovou hodnotu to nastane pro právě 1 volbu výsledku $\in D$. Proto:

$$\Pr[\text{dvojice se shodne}] \leq 1/(2^b-(i-1)) \leq 2^{-(b-1)}$$

triv. odhad pomocí 2^b nefunguje, neb $\in$ není náhodná

$$\Pr[\text{najdu kolizi}] \leq \Pr[\text{dvojice se shodne}] \cdot \#\text{dvojic} \leq q^2/2^b$$

$\leq q^2/2$

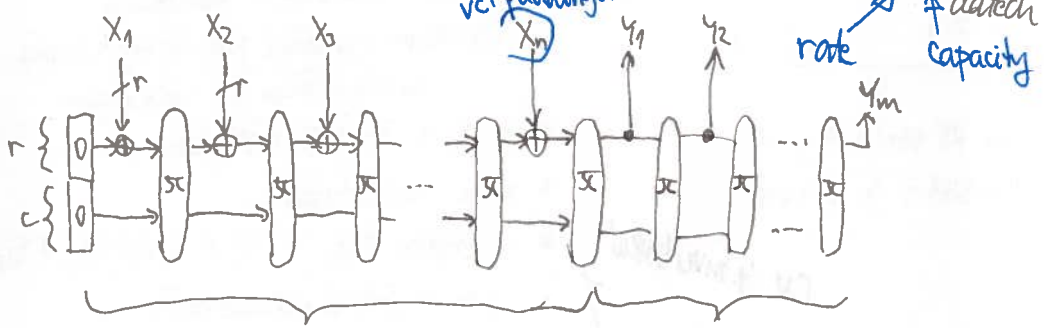
max. $i-1$ hodnot je už obsazenc z předch. dotazů

Šce, když náhodná permutace!

- Pozor, jde najít $\beta(a,b)=b$ ... stačí položit $b = D_a(0)$ → to by pro náhodnou β mělo být těžké (ale nevadí to...)

- MD5: 128b výsledek, od roku 2004 známé kolize (i chosen-prefix) (Rivest 1992)
  - to je málo
  - ↳ ale invertovat ji nedokáže
- SHA-1: 160b výsledek, 2015 kolize v kompres. funkci, 2017 plná kolize (zatím dost náročná) (NSA)
  - konstrukce podobná Daviesovi-Meyerovi (příslušné šifře se občas říkal SHACAL)
- SHA-2: verze s 224, 256, 384, 512 bity výsledku
  - mohutnější, ale podobná struktura
  - zatím nejsou rozbité
- SHA-3 (veřejná soutěž NISTu, finále 2015)

"Houbovitá funkce" ... uvažujme permutaci π na $w=(r+c)$-bit. (width)



- Odolnost proti kolizím je ovlivněna velikostí vymačkávaného výstupu, kapacitou c (interní kolize)

- Interní kolizní útok

kruhům samými nulami

Po řádově $2^{c/2}$ krocích najdu $b_i = b_j$, $i<j$.
Zprávy 0^i a $0^{j-1}(a_i \oplus a_j)$ vedou na totožný interní stav
⇒ vyrušká se stejný výstup. [mohu pak dolepit stejné pokrač. za obě prefixy a zase mám kolizi...]

⇒ security level je nejvýše $c/2$.

- SHA-3 je houba s permutací Keccak šířky 1600 bitů.
Standard definuje:

SHA3-224	r=1152	c=448	} vždy c = 2·délka výstupu
⋮			
SHA3-512	r=576	c=1024	
SHAKE128	r=1344	c=256	} XOF = extendable output functions (výstup lib. velikosti) ↳ PRNG
SHAKE256	r=1088	c=512	

vynechává se jedinými krokem

- Jak vypadá Keccak?

Vnitřní stav je kvádr

... 25 slov šířky w
(v SHA-3 je w=64)

Provádíme $12 + 2\log w$ rund, v každé:

- ke každému sloupečku přixoruji paritu 2 okolních sloupečků
(to dělám paralelně pro všechny sloupce v 1 svislém řezu bit-slicingem
- každé z 25 slov zrotujeme
- slova permutujeme
- v každém řádku: $x_i \leftarrow x_i \oplus (\neg x_{i+1} \,\&\, x_{i+2})$
[to je jediná nelinearita]
- přimíchám k 1 slovu rundovou konstantu

CV: je invertibilní

- Různé módy použití (SHA-3, SHAKE, další budoucí) mají různý padding ⇒ jsou rozlišitelné

Merkleovy stromy

- listy hešují bloky vstupu (lze počítat paralelně)
- vnitřní vrcholy hešují výsledky svých synů
- pozor, kořen je potřeba odlišit! Jinak bych mohl vyříznout podstrom,
  - uložím hloubku
  - k vrcholu přihešuji flag
- → kódování Sakura (součást specifikací kolem SHA-3)
- výhody: - paralelizace
  - random-access ověřování i update

MESSAGE AUTHENTICATION CODES (MACs, symetrické podpisy)

- obecně: funkce na generování a ověřování podpisu
  - ↳ pokud je deterministická, ověření je memcmp
- můžeme si představit jako keyed hash
  - pro náhodný klíč se má chovat jako náhodná funkce
- příklad: $MAC(k,X) = hash(k \| X)$
  - ! rozbité pro hešе typu Merkle-Damgård → extension attacks!
    - ↳ ale pro ideální hash funguje a pro SHA-3 také (spec. kMAC)
- co třeba $hash(X \| k)$?
  - to není bezpečné proti obecným kolizním útokům na h (většina výpočtu nezávisí na klíči)

→ konstrukce $HMAC_h(k,X) := H(k \oplus c_{out} \| H(k \oplus c_{in} \| X))$

- s klíčem i na začátku je to pro SHA-1, jakž takž OK i dnes
- HMAC nad SHA-1/2 je běžný v internetových protokolech
- konstanty
- jako třeba zde

myšlenka: skládám 2 funkce:
- vnitřní je odolná proti kolizím a bere $\{0,1\}^*$
- vnější je bezpečná MAC, ale stačí fixní délky

Bezpečnost (CPA):
- útočník dostane podepisovací orákulum
- má vyprodukovat korektní podpis pro zprávu, na kterou se neptal orákula.

- Kombinace šifra + MAC:
  1. nezávisle obě (auth & encrypt): MAC prozradí info o plaintextu (třeba rovnost)
  2. encrypt-then-MAC: bezpečné
  3. MAC-then-encrypt: často míval padding orákula

- **CBC-MAC** – šifruji pomocí CBC, MAC = poslední blok zašifr. textu
  - nutná konstantní IV (jinak děravé – možno měnit 1. blok zprávy a kompenzovat změnou IV)
  - nesmíme vyhradit jiné zašifrované bloky (jinak truncate/reassemble)
  - umíme dokázat bezpečnost, pokud:
    ① šifra je ideální, ② množina zpráv je bezprefixová [konst. délky / délka na začátku atd.]

! Nesmíme použít stejný klíč pro šifrování a MAC ☹ (jinak reassemble kombinací 2 zpráv)

- **Shannonovsky bezpečný MAC** – předpokládejme one-time klíč
  - pořídíme si rodinu 2-nezávislých heš. funkcí (↑ ne v kryptograf. smyslu!)

$$\mathcal{H} := \{h_k \mid k \in \mathcal{K}\},\ \forall k\ h_k : X \to Y$$
$$\forall x, x' \in X,\ x \neq x',\ \forall y, y' \in Y\ \ Pr_{h \in \mathcal{H}}[h(x)=y \ \&\ h(x')=y'] = \frac{1}{|Y|^2}$$

je to také integer $|K|$ $\Rightarrow |K| \geq |Y|^2$

Příklad: $X = Y = \mathbb{Z}_p$, $K = \mathbb{Z}_p^2$, $h_{a,b}(x) = ax + b$ } pro x, x', y, y' dostanu soustavu 2 lineárních rovnic pro a, b $\Rightarrow \exists! a,b \Rightarrow Pr = 1/p^2$

  - podepisuji pomocí h_k s náhodným klíčem $k \in \mathcal{K}$
  - jaká je Pr, že po pozorování dvojice $(x, \underbrace{h(x)}_{y})$ mi vyjde tip (x', y')?

$$Pr[\text{útok vyšel}] = Pr_h[h(x')=y' \mid h(x)=y] = \frac{Pr[h(x)=y \ \&\ h(x')=y']}{Pr[h(x)=y]} = \frac{1/|Y|^2}{1/|Y|} = \frac{1}{|Y|}.$$

↑ počítáním Pr z definice $\mathcal{H}$ přes všechna y'

→ nepřekonám náhodné tipnutí podpisu.

klíč musí být aspoň 2x delší než zpráva – cvič.

- **Praktická implementace**: pořídím si nonci, z té šifrováním taj. klíčem odvodím pseudonáhodný klíč pro $\mathcal{H}$

- **Aproximace**: $(2,c)$-nezávislost … $Pr[= \_] \leq c/|Y|^2$
  … např. $((ax+b) \bmod p) \bmod m$ je $(2,4)$-nezávislé.

Jak se změní Pr úspěšného útoku?

$$\frac{Pr[h(x)=y \ \&\ h(x')=y']}{Pr[h(x)=y]} \leq \frac{c/|Y|^2}{1/|K|} = \frac{c \cdot |K|}{|Y|^2}$$

tato je moc slabé, více méně konstanta!

↑ tohle je $\leq c/|Y|$, ale to nám je ve jmenovateli navíc … ovšem také je to $\geq 1/|K|$ ☹

- **Polynomiální MAC** ... opět nad tělesem $\mathbb{Z}_m$, klíče $\in \mathbb{Z}_m^2$

$h_{a,b}(x_1 \dots x_n) = x_1 \cdot a^n + x_2 \cdot a^{n-1} + \dots + x_n \cdot a^1 + b$ } snadno spočteme Hornerovým schématem

Falšování pro 2 zprávy téže délky.

$Pr_{a,b}[h_{a,b}(x) = y \;\&\; h_{a,b}(x') = y']$

... odečtením rovnic: $(x_1 - x_1')a^n + (x_2 - x_2')a^{n-1} + \dots + (x_n - x_n')a^1 = y' - y'$

čili a musí být kořenem nějakého polynomu stupně nejvýš n $\Rightarrow$ takových a je max. n

... pro každé takové a $\exists!\, b$ takové, že platí i druhá rovnice.

$\Rightarrow Pr \le n/m^2$.

$Pr_{a,b}[h_{a,b}(x) = y] = 1/m$ ... pro každé a $\exists!\, b$, pro které to platí.

$\Rightarrow Pr[\text{útok uspěje}] \le n/m$. – tedy chci $m \ge n^2$, abych Pr stlačil pod úspěšnost tipování podpisu

- **Mód blokových šifer GCM** [Galois/Counter Mode], populární s AES
  - pracuje v tělese $GF(2^{128})$: sčítání je XOR, násob. je CLMUL
  - autentikuji nezašifrovaná data $A_1 \dots A_m$ a $Y_1 \dots Y_n$ ($Y_i = X_i \oplus E_k(IV+i)$), za to přilepím blok kódující m, n a blok $E_k(IV)$.
  - výsledné bloky dají koeficienty polynomu, ten vyhodnotím v bodě $E_k(0)$

$\Rightarrow$ je to polynomiální MAC s klíčem generovaným blokovou šifrou

- **Poly1305** [Bernstein 2005]
  - poly-MAC v tělese $GF(2^{130} - 5)$ ... o něco větší rozsah než 128b bloky, výsledek nakonec moduluje 2^{128} (prvočíslo)
  - každý blok se paduje (to se vždy vejde ☺)
  - potřebuji nonci a tajný klíč (k, r) [k je 128b, r má nějaké bity navíc → jen 106b]
  - polynom vyhodnotím v bodě r a přičtu $E_k(\text{nonce})$
    (a mod 2^{128}; tohle už mod 2^{128})

to zjednoduší implementaci aritmetiky

to je potřeba i zde

☺ Jelikož přičítám "one-time pad", z žádné odchycené zprávy se nedozvím nic o r $\Rightarrow$ není třeba pokaždé jiné r

$Pr[\text{útočník uspěje}] \le$ délka zprávy $/ 2^{106}$, což je ok pro zprávy rozumných délek.

- původně specifikován s AES, dnes se často kombinuje s ChaCha20.

NÁHODNÉ GENERÁTORY

Požadavky: Útočník ani se znalostí předchozího výstupu nedovede (efektivně) předpovídat budoucí výstup.
[to speciálně implikuje statistickou rovnoměrnost]

Možná řešení: → pseudonáhodný generátor (třeba šifra v CTR módu)

→ HW generátor náhodnosti
- šum na odporu / diodě apod.
- radioaktivní zářič
- průchod/odraz fotonu na polopropust. zrcátku
- lávové lampy
- rádiový šum
- kruhový oscilátor
- timing kláves / disku / sítě ...

} pozor, útočník může tyhle jevy také měřit a případně ovlivňovat

→ kombinace obojího – /dev/random a spol. } máme-li reálnou náhodu, použije námi; pokud ne, stále je to PRNG
- RDRAND v procesoru (jak moc důvěryhodný?)

Problémy: (metastabilní oscilátor, automatická kontrola anomálií, konec PRNG založ. na AES)
- Je-li vnitřní stav kompromitován, potřebujeme přidat hodně entropie najednou, jinak útočník probere všechny možnosti a určí nový stav
  → pooling, odhadování entropie (šarlatánství ...)
- Inicializace po bootu → ukládání stavu, riziko rollbacku (zálohy, snapshoty VM)

Fortuna [Ferguson, Schneier 2003] ... elegantní PRNG, který nepotřebuje odhadování entropie zdrojů

- Generátor
  - používá AES s 256b klíčem
  - šifruje 128b počítadlo (nikdy nepřeteče)
  - po vygenerování nejvýše 2^{16} bloků (nebo požadovaného množství dat) vygeneruje nový klíč (CTR nepakuje bloky, na to by se časem přišlo), ale neresetuje počítadlo (tím rozbije potenciální krátké cykly)
- Akumulátor
  - sbírá externí náhodnost do kyblíčků $P_0 - P_{31}$, každý zdroj náhodnosti přidává j-tý vzorek do $P_{j \bmod 32}$.

- jakmile P_0 naakumuluje dost vzorků (ale ne častěji než jednou za 100 ms), přihešuji jeho obsah ke klíči generátoru a v i-tém kroku ještě všechny P_j pro $2^j \setminus i$. Použité kyblíčky vysypu.

  $\Rightarrow$ z kompromitovaného stavu se časem vzpamatuji (čas závisí na rychlosti přitékání entropie) ... přesněji:
  - nechť za 1 krok reseedování přiteče ϱ bitů entropie
  - 128 bitů určitě stačí k zotavení
  - pokud $\varrho \geq 128$, zotavíme se příštím reseedem (P_0 použijeme vždy)
  - jinak se zotavím po reseedu z P_i takového, že

$$128 \leq 2^i \cdot \varrho/32 < 256$$

(pod $2^i \cdot \varrho/32$: přítok do 1 kyblíčku; u 256: jinak mohu zmenšit i)

  - čili chci $2^{12} \leq 2^i \cdot \varrho \leq 2^{13}$

$$\frac{2^{12}}{\varrho} \leq 2^i \leq \frac{2^{13}}{\varrho}$$

(2^i: # kroků na zotavení)

BEZPEČNÝ KANÁL
(příklad z Practical Crypto)

- Alice a Bob mají unikátní tajný klíč (pro ∀ kanál jiný)
- chtějí obousměrně komunikovat
- jeden pošle $m_1 - m_r$, druhý přijme podposloupnost (a ví, kterou)
- zkombinujeme:
  - šifru (třeba AES/CTR)
  - MAC (after encrypt)
  - sekvenční číslo

    ↳ po 2^{32} zprávách chceme měnit klíče

  } pro každý směr zvlášť
  - odvozování klíčů (2x šifra, 2x MAC) hešovací funkcí z hlav. klíče

---

Pozn. k Linuxovému /dev/(u)random
- pooly míchá pomocí CRC (tady stačí inf.-teor. mixování)
- PRNG založen na ChaCha20
- entropy estimators

- složitost aritmetiky pro b-bit. čísla: add/sub $\mathcal{O}(b)$, mul $O(b^2)$, dělení $O(b)$ [ale s obří konst.], div/mod ... o trochu horší než mul, rozhodně $O(b^2)$
  - modulární umocňování: $a^k \bmod N$ pomocí $O(\log k) \times$ mul → pro b-bit. čísla $O(b^3)$
- Euklidův algoritmus: $O(b)$ průchodů → celkem $O(b^3)$
  - lepší analýza / bindrul GCD → $O(b^2)$
  - značení: $\gcd(x,y)$, $x \perp y \Leftrightarrow \gcd(x,y)=1$ (nesoudělnost)
  - rozšířený E.a.: spočte $u,v \in \mathbb{Z}$: $ux + vy = \gcd(x,y)$ — Bézoutovy koeficienty
- počítání mod N: $\mathbb{Z}_N$ je okruh ... kdy je prvek invertibilní?
  - řešíme kongruenci $ax \equiv b \pmod N$ ... a,b známe; x hledáme
  - ekvivalentně: $\exists y \in \mathbb{Z}: ax - Ny = b$
    1. pokud $b = \gcd(a,N)$: Bézoutovy koef. dají x,y
    2. pokud $b = c \cdot \gcd(a,N)$: jako předtím, nakonec vynásobíme c
    3. jinak nemá řešení: levá strana je dělitelná $\gcd(a,N)$, pravá ne
  - a je invertibilní $\Leftrightarrow a \perp N$
    - ↳ $\mathbb{Z}_N^*$: multiplikativní grupa mod N [je to grupa]
  - pokud N je prvočíslo, invertibilní je vše kromě 0 ⇒ $\mathbb{Z}_p$ je těleso.
  - inverze umíme počítat efektivně
- Malá Fermatova věta: pokud $x \perp p$, pak $x^{p-1} \overset{p}{\equiv} 1$.
  (díky tomu x^{p-2} je inverze x, to dává jiný efektivní alg.)
  - ↳ zobecnění: Eulerova věta: pokud $x \perp N$, pak $x^{\varphi(N)} \overset{N}{\equiv} 1$
    - zde $\varphi(N) = |\mathbb{Z}_N^*|$, tedy $\#a \in \mathbb{Z}_N: a \perp N$
    - Def: $x^0, x^1, x^2, \dots, x^{k-1}$ je nějaká podgrupa $\mathbb{Z}_N^*$, říkejme jí třída H
      - ↳ x^k bude první 1 kromě x^0
      - Lagrangeova věta: je-li G konečná grupa a $H \leq G$, pak $|H| \setminus |G|$. (nám stačí pro komutativní grupy)

→ podle Lagrange: $|H| \setminus |\mathbb{Z}_N^\times| = \varphi(N)$
(tohle je k)

→ $\varphi(N) = k \cdot c$ pro nějaké c

→ $x^{\varphi(N)} \equiv (x^k)^c \equiv 1^c \equiv 1$.

- **Čínská zbytková věta** (CRT): Pokud $N_1 - N_k$ navzájem nesoudělné a $N = \prod_i N_i$, pak $\mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k} \cong \mathbb{Z}_N$
  (↑ obrovský isomorfismus, naše efektivní)

Dk: Bůno $k=2$, dál se pokračuje indukcí (triv.)

① Nekonstruktivně: $f(x) := (x \bmod N_1, x \bmod N_2)$
je zobrazení z $\mathbb{Z}_N$ do $\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$
→ je prosté ⇒ je také na (vede mezi stejně velkými množinami)

② Konstruktivně: Chci vzor dvojice (a_1, a_2).
Najdu čísla u_1, u_2 t.ž. $f(u_1) = (1,0)$, $f(u_2) = (0,1)$
↳ pak stačí položit $x := a_1 u_1 + a_2 u_2$ (vše mod N)
↳ kde je vzít: $f(N_2) = (q, 0)$ ... pokud $q=1$, vyhrál jsem a mám u_1
... jinak násobím N_2 inverzí q mod N_1 (vím, že $q \neq 0$)
$q \perp N_1$ díky $N_1 \perp N_2$
→ podobně u_2.

- **Výpočet $\varphi(N)$:**
  - $\varphi(p) = p-1$ [viz výše]
  - $\varphi(p^k) = (p-1) \cdot p^{k-1}$
  - pro $x \perp y$ máme $\varphi(xy) = \varphi(x) \cdot \varphi(y)$ ... to je vidět z CRT

⇒ $\varphi(N)$ umíme spočítat, pokud známe faktorizaci N

- **Faktorizace vs. prvočíselnost**

považuje se za těžkou:
- přímočaré alg. jsou exponenciální
- umí se různě subexponenciální (čím dál lepší)
- kvantové počítače umí polynomiálně [Shor 1994]

snadná...
- rychlé pravděpodobnostní testy s 1-stranou chybou
- poly alg. [Agarwal et al. 2002] ... zatím nepříliš prakticky

- **Pravděpodobnostní testy prvočíselnosti**

  - **Fermatův test**: pro náhodné $x \in \mathbb{Z}_N$ spočítáme $x^{N-1} \bmod N$.
    → pokud nevyjde 1, N je složené [x je Fermatův svědek]
    ↳ buď proto, že $x \perp N$, nebo díky F. věť

    - Jaká je Pr, že složené číslo projde testem? Kolik je svědků?
      - bohužel existují **Carmichaelova čísla** (nejmenší je 561)
        pro něž $\forall x \in \mathbb{Z}_N^* \; x^{N-1} \overset{N}{\equiv} 1$ ... mají jen Euklidovy svědky a těch je málo
        → Carm. čísel je nekonečně mnoho [Alford et al. 1994]
      - pokud N není Carm., už to dopadne dobře:
        $H := \{ x \in \mathbb{Z}_N^* \mid x^{N-1} \overset{N}{\equiv} 1 \}$ je podgrupa $\mathbb{Z}_N^*$
        ... přitom $H \neq \mathbb{Z}_N^*$, takže podle Lagrangeovy věty $|H| \leq \frac{|\mathbb{Z}_N^*|}{2}$
        $\Rightarrow$ Pr[x je svědek] $\geq 1/2$.

  - **Rabinův-Millerův test**:

    1. $x \in_R \{1 \dots N-1\}$
    2. pokud $\gcd(x, N) \neq 1$: SLOŽENÉ (Euklidův svědek)
    3. spočítáme $x^{N-1} \bmod N$ ← pokud není 1: SLOŽENÉ (Fermatův svědek)
       [pozpátku] $x^{\frac{N-1}{2}} \bmod N$
       $x^{\frac{N-1}{2^i}} \bmod N$
       Pokud jsou 1, pokračujeme.
       Pokud −1: PRVOČÍSLO
       Jinak SLOŽENÉ (Riemannův svědek)

    zastavíme se, až bude exponent lichý → odpovíme PRVOČÍSLO

    Uvo: Pokud odpovím SLOŽENÉ, je to pravda

    Věta [Rabin]: Pr[PRVOČÍSLO | x složené] $\leq 1/4$

    Věta [Miller]: Pokud platí zobecněná Riemannova hypotéza, $\exists$ svědek $\in O(\log N)$.

- **Generování velkých prvočísel**: náhodně tipujeme a testujeme, hustota prvočísel je cca $1/\ln N$.

- **Diskrétní logaritmus**
  - Věta: $\mathbb{Z}_p^*$ je cyklická grupa $\rightarrow \exists g$ (generátor) t.ž. $\{g^0, g^1, \dots, g^{p-2}\} = \mathbb{Z}_p^*$
  - Jinými slovy $\mathbb{Z}_p^* \cong (\mathbb{Z}_{p-1}, +)$
    - isomorfismus je v jednom směru umocňování, v druhém diskrétní log.
      - ↓ poly. alg.
      - ↳ podobně těžké jako faktorizace – subexp. / kvantově poly.
  - Jak ověřit, zda g je generátor? pro $g \in \mathbb{Z}_p^*$
    Pokud není, pak $H := \{g^0, g^1, \dots\}$ je nějaká podgrupa $\mathbb{Z}_p^*$ $\Rightarrow |H| \setminus \varphi(p) = p-1$
    $\rightarrow g^{\frac{p-1}{k}} \equiv 1$ pro nějaké přirozené k ... dokonce stačí prvočíselné k.
    $\rightarrow$ jakmile zvládnu faktorizaci $p-1$, umíme to otestovat (dost rychle, protože faktorů je $O(\log p)$).
  - Jak najít generátor? Náhodně vybereme a testujeme...
    Kolik je generátorů?
    $\rightarrow$ pokud g je gen., pak g^k je gen. $\Leftrightarrow k \perp p-1$
    $\rightarrow$ # generátorů = $\varphi(p-1)$ ... to je dost (nepočítáme přesnou Pr...)
- **Diskrétní odmocniny**
  - v $\mathbb{Z}_5$: $1^2 = 4^2 = 1$, $2^2 = 3^2 = 4$ $\Rightarrow$ 1, 4 mají 2 odmocniny
    2, 3 nemají žádnou
    0 má právě 1
    "kvadratické zbytky" (QR)
  - Obecně: kromě 0 má polovina čísel 2 odmocniny, zbytek žádnou.
    mod p
    - nejvýše 2: jsou to kořeny kvad. polynomu
    - pokud $x^2 = a$, pak také $(-x)^2 = a$
      ... kromě $x = -x$ (jen pro $x = 0$) je odmocnin sudý počet
    - nechť g je generátor $\mathbb{Z}_p^*$: g^{2k} má 2 odmocniny } takových čísel je $\frac{p-1}{2}$
      ↳ na čísla g^{2k+1} už žádné odmocniny nezbyly
    $\Rightarrow$ sudost/lichost $d\log(x)$ prozradí, zda x je QR.

- množina všech QR tvoří podgrupu $\mathbb{Z}_p^*$. (1 je QR, QR·QR je QR)
- Testování QR: x je QR $\Leftrightarrow x^{\frac{p-1}{2}} \equiv 1$. (Eulerovo kritérium)

  Dk: $(g^{2k})^{\frac{p-1}{2}} \equiv g^{k(p-1)} \equiv 1^k \equiv 1$
  $(g^{2k+1})^{\frac{p-1}{2}} \equiv g^{k(p-1)} \cdot \underbrace{g^{\frac{p-1}{2}}}_{\text{tohle je } \sqrt{1},\ \text{tedy } -1} \equiv -1$

  $x^{\frac{p-1}{2}}$ je tedy homomorfismus $\mathbb{Z}_p^* \to (\{-1,1\},\cdot)$ indikující QR (Legendreův symbol)
- Jak počítat $\sqrt{x}$?
  - pokud $p = 4t+3$: $\left(x^{\frac{p+1}{4}}\right)^2 \equiv x^{\frac{p+1}{2}} \equiv \underbrace{x^{\frac{p-1}{2}}}_{1 \text{ dle Eul. kritéria}} \cdot x \equiv x$
  - pro $p = 4t+1$: randomizovaný alg. [Tonelli 1891, Shanks 1973]
- Odmocniny mod složené n: Pokud umíme n faktorizovat, použijeme CRT, jinak těžké.

RSA [Rivest, Shamir, Adleman 1978; GCHQ 1973, tajně public 1997]

Klíč: $n = p \cdot q$ p, q dvě různá velká prvočísla [modulus]
$\varphi(n) = (p-1)(q-1)$
e t.ž. $e \perp \varphi(n)$ [šifrovací exponent]
d t.ž. $ed \equiv 1 \pmod{\varphi(n)}$ [dešifrovací exponent]
→ Šifrovací klíč (e,n), dešifrovací klíč (d,n). (1 veřejný, 2. tajný)
idea: bez faktorizace n je těžké z jednoho klíče spočítat druhý

Šifra: $E(x) = x^e \bmod n$
$D(x) = x^d \bmod n$
} zprávy jsou prvky $\mathbb{Z}_n$

Korektnost: $(x^e)^d \equiv x^{ed} \equiv x^{k\cdot\varphi(n)+1} \equiv \underbrace{\left(x^{\varphi(n)}\right)^k}_{1} \cdot x \equiv x.$

! Tohle selže, pokud $x \pm n$
→ mohu důkaz opravit pomocí CRT (dokáži zvlášť mod p a mod q)
→ ale pokud se do takového x trefím, mám jiné problémy 😉

Efektivita: Poly, ale pomalé... často stavíme hybridní šifru z RSA a sym. šifry

Triky na zrychlení: - volíme malý e (veřejný) (třeba 3 nebo 17)
- dešifrování pomocí CRT (menší čísla ⇒ rychlejší aritmetika) (to utajíme stejně jako modul)

Důležité vlastnosti: - Komutuje: $E_{k_1}(D_{k_2}(E_{k_1}(E_{k_2}(x)))) = x$ (pro klíče se stejným modulem)
- klíče lze prohodit (ale nelze bezpečně používat oba směry v jednom protokolu)
- homomorfní šifra: $E(x \cdot y) \equiv E(x) \cdot E(y)$
  ↳ to je většinou spíš k neštěstí, ale má to i hezké aplikace:

Slepé podpisy (viz protokoly na digitální peníze)
- Alice podepisuje libovolné zprávy (šifruje je tajným e)
- Bob si chce nechat podepsat x, ale nechce, aby ho A. znala
- Bob vygeneruje $r \in_R \mathbb{Z}_n^*$, pošle Alici $x \cdot r^d \bmod n$.
- Alice spočítá $(x \cdot r^d)^e \equiv x^e \cdot r^{ed} \equiv x^e \cdot r$
- Bob výsledek vynásobí inverzí r a získá x^e.
- Alice (až na případy $x \notin \mathbb{Z}_n^*$) nezjistí o x nic.

Útoky:
- pokud $x < n^{1/e}$, stačí spočítat odmocninu v $\mathbb{Z}$, což je poly.
- známe-li $\varphi(n)$, můžeme faktorizovat n: $n = pq$, $\varphi(n) = (p-1)(q-1) = pq - p - q + 1$ } soustava rovnic, snadno řešitelná!
- je-li $d < n^{1/4}$, lze ho spočítat z e [Wiener 1990]
  ⇒ malý si můžeme dovolit jen veřejný exponent
- z d lze spočítat $\varphi(n)$ randomizovaným alg. (viz Stinson & Paterson)
- **Meet in the middle:**

... máme $c = m^e$ (známe, neznáme)

→ našli jsme u, v: $u^e \equiv c \cdot v^{-e}$

Jak velká u, v potřebujeme?

$\Pr[\exists u,v \leq n^{\frac{1}{2}+\varepsilon} : uv \equiv m] \geq \text{const.}$

$u^e \cdot v^e \equiv c$
$(uv)^e \equiv c$
$uv \equiv m$

⇒ útok hrubou silou stihneme asi na $\sqrt{n}$ pokusů?

- **Podobné zprávy**: pokud známe $c \equiv m^e$, $c' \equiv (m+\delta)^e$ a ... m je spol. kořen polynomů $p(x) = x^e - c$, $p'(x) = (x+\delta)^e - c'$
  → pokud je $\gcd(p, p')$ lineární, známe m.
  (nastane s velkou pstí; potřebuji malé e)
- **Částečně známé zprávy**: stačí hádat neznámé bity (netriviální!)
- **p a q blízko u sebe** → faktorizace:
  Nechť $q = p + 2d$. Potom $n = pq = p(p+2d) = p^2 + 2dp$,
  takže $n + d^2 = p^2 + 2dp + d^2 = (p+d)^2$
  ⇒ mohu zkoušet různá d a odmocňovat $n + d^2$.
- **Více klíčů používá stejný modul**: jednak může každý majitel priv. klíče faktorizovat n, jednak při posílání téže zprávy více příjemcům může Eva dešifrovat. [Ba důkazu, viz Stitson, cvič. 6.17]
- **Tatáž zpráva zašifrovaná klíči s různými moduly**: (toto je lepší)
  ukážeme pro $e=3$ a 3 odchycené zprávy.
  Víme: $x^3 \equiv c_1 \pmod{n_1}$, $x^3 \equiv c_2 \pmod{n_2}$, $x^3 \equiv c_3 \pmod{n_3}$
  Nyní: $N := n_1 \cdot n_2 \cdot n_3$
  Jistě $x^3 < N$
  ... ale díky CRT je toto x^3 jednoznačně určeno zbytky c_1, c_2, c_3
  → umíme najít x^3 v $\mathbb{Z}$
  → stačí odmocnit v $\mathbb{Z}$.

- **Chyba při výpočtu**
  - Nechť Alice podepisuje tajným klíčem s optimalizací pomocí CRT
  - Opakovaně podepisujeme 1 zprávu x, dostáváme $x^e \bmod n$.
  - Pokud A. udělá chybu při výpočtu BUNO zbytku mod p, vydá výsledek lišící se o násobek q: $\gcd(\text{výsledek} - x^e \bmod n, n)$ prozradí q!

⇒ útočník se může snažit chyby umělé vyvolávat.

Sémantická bezpečnost RSA

↳ žádná vlastnost plaintextu (efektivně testovatelná) nemá jít efektivně zjistit z ciphertextu

- RSA zachovává Jacobiho symbol (to je CCA 1 bit informace)

Def:
- Legendreův symbol $\left(\frac{a}{p}\right)$ pro p prvočíslo $\equiv a^{\frac{p-1}{2}} \bmod p$
  - +1 pokud a je QR, −1 není-li QR, 0 pro $p \mid a$
- Jacobiho symbol to generalizuje pro liché složené $n = \prod_{i=1}^{k} p_i^{\alpha_i}$:

$$\left(\frac{a}{n}\right) := \left(\frac{a}{p_1}\right)^{\alpha_1} \cdot \_ \cdot \left(\frac{a}{p_k}\right)^{\alpha_k}$$

($\left(\frac{a}{n}\right)$ je hom. ze $\mathbb{Z}$ do $\{-1,0,+1\}$)

  - 0 pokud $\gcd(a,n) > 1$, jinak je to $\pm 1 \to -1 \Rightarrow a$ není QR; $+1 \Rightarrow$ může, ale nemusí
  - $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\cdot\left(\frac{b}{n}\right)$ [nejprve dokážeme pro L. symbol]
  - existuje poly alg. pro výpočet $\left(\frac{a}{n}\right)$, který nepotřebuje faktorizaci n využívá $\left(\frac{a}{n}\right) = \left(\frac{a'}{n}\right)$ pro $a' \equiv a \pmod n$

$$\left(\frac{a}{n}\right)\cdot\left(\frac{n}{a}\right) = (-1)^{\frac{a-1}{2}\cdot\frac{n-1}{2}}$$

[Gaussův zákon kvadratické reciprocity – netriviální]

a pak je podobný Euklidovu alg.

↳ pro $\alpha \mid n$ je $Pr\left[\left(\frac{a}{n}\right)=1\right] = \frac{1}{2} \equiv \left(\frac{a}{n}\right)$ nese 1 bit informace

… ale to je jediné známé prosakování informace!

- Def. $half(x) := \lfloor D(x) \geq n/2 \rfloor$, $parity(x) := D(x) \bmod 2$ (↑ indikátor 0/1)

Věta: Umíme-li efektivně spočítat $half(x)$, umíme i ~~všechno~~ počítat $D(\dots)$. (umíme-li na to orákulum; tedy invertovat RSA)

Dkz: ~~Zapíšeme x jako $n\cdot\alpha$, kde $\alpha \in [0,1)$, nějaké~~ ~~$half$. Nějaké $c = m$~~

Nějaké $y = x^e$. Známe y, chceme zjistit x.

Jisté je $x = n\cdot\alpha$ pro nějaké $\alpha \in [0,1)$, které zapíšeme binárně.

$half(y)$ nám řekne nejvyšší bit α

$half(y\cdot 2^e)$ nám řekne nejvyšší bit $2\alpha \bmod 1$,

$D(y\cdot 2^e) = 2x$ což je 2. nejvyšší bit α

… atd. a za $\log n$ pokusů známe α': $|\alpha - \alpha'| < \frac{1}{2n}$

$\Rightarrow \alpha'\cdot n$ po zaokrouhlení dá x.

Analogicky pro parity(x). Ono totiž platí:

- $half(x) = parity(x \cdot 2^e)$ ⇒ pomocí orákula pro paritu můžeme počítat half v předchozím důkazu.

Padding – nechceme pomocí RSA šifrovat surovou informaci (hrozí, že bude moc malá apod., stejně tak RSA je deterministické ⇒ není CPA-bezpečné)
↳ a hrozí multi-modulový útok

PKCS #1 v1.5

Public-key Crypto. Std. od RSA Security Inc.

- random: aspoň 8 náhodných bytů různých od 00
- část se nevyužije, pokud |log n| není násobek 8
- typ paddingu (00 a 01 jsou deterministické)

Bleichenbacherův útok [1998]: zneužití paddingového orákula (řekne, zda dešifrovaná zpráva má správný formát paddingu – to může být třeba časový postranní kanál)

Nechť x je správně opadovaná zpráva, známe $y = x^e$.
↳ $2B \le x < 3B$ pro nějakou mocninu dvojky B (daná pozicí 02 v paddingu)

Ptáme se orákula na $y \cdot s^e$ pro různá s ... tedy zda $x \cdot s$ je správná.
Odhadujeme Pr, že to tak bude (heuristicky – předpokládáme náhodnost ☺)

① $Pr[2B \le \underbrace{xs \bmod n}_{\text{nejvyšších max. 16 bitů má správný tvar}} < 3B] \ge 2^{-16}$

② $Pr[\text{za 00 02 je 8 nenul a pak aspoň 1 nula}] = \left(\frac{255}{256}\right)^8 \cdot \left(1 - \left(\frac{255}{256}\right)^{k-10}\right) \ge 0.18$
(k bytů; pro $k \ge 64$ (aspoň 512b klíč))

$Pr[\text{obojí najednou}] \ge 2.8 \cdot 10^{-6}$
⇓
cca za průměrně 10^6 pokusů se to povede

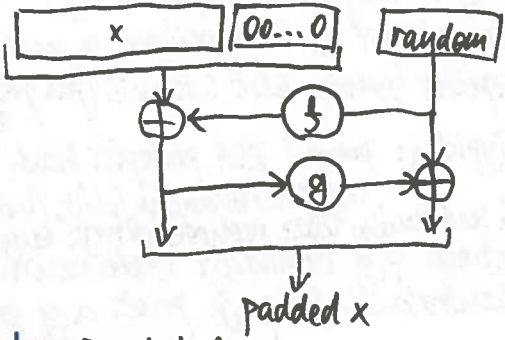
Co se dozvíme o x?

- $2B \le \underbrace{xs \bmod n}_{\text{mod } n} < 3B \Rightarrow \exists r: 2B \le xs - rn < 3B$
$\Rightarrow \exists r: \left\lceil \frac{2B + rn}{s} \right\rceil \le x \le \left\lfloor \frac{3B - 1 + rn}{s} \right\rfloor$

To je nějaký interval ... ale my neznáme r ⇒ spousta intervalů ($\sim 2^{16}$)

Velmi zhruba: začneme intervalem $[2B, 3B)$,
každý další půlus ho protne se sjednocením intervalů.
Heuristicky: intervalů dlouhodobě ubývá a zkracují se
$\Rightarrow$ časem je x jednoznačně určeno.

PKCS #1 v2.0 – protokol OAEP, netrpí těmito nedutky
– v podstatě je to Feistelova síť se 2 rundami



díky tomu je reverzibilní

f, g jsou hešovací funkce

Obecně k bezpečnosti RSA

- spoléhá na obtížnost faktorizace (ale není s ní ekvivalentní?)
- má algebraickou strukturu $\Rightarrow$ randomizujeme, hešujeme apod.
- je potřeba používat ho velmi opatrně

Rabinův kryptosystém – založený na diskrétních odmocninách

Tajný klíč: prvočísla p, q
Veřejný klíč: $n = p \cdot q$
$E(x) = x^2 \bmod n$
$D(y)$ počítá diskrétní odmocninu
- to jde se znalostí faktorizace lehko (zvlášť mod p, mod q, pak CRT)
- pozor, vyjdou 4 možná řešení, je nutno nějak zjednoznačnit (hash?)

To bohužel také ukazuje, že CCA faktorizuje n?

Bezpečnost: Pokud umíme dešifrovat, umíme i faktorizovat modul (aspoň randomizovaně):

$a \leftarrow$ náhodné ze $\mathbb{Z}_n$
$b \leftarrow D(a^2)$
Pokud $a = \pm b \Rightarrow$ FAIL
Jinak $\Rightarrow \gcd(a-b, n)$ je faktor n

b je spíš aspoň $3/4$ jiná odmocnina než a
-- s pstí $1/2$ to není ani $-a$
$\Rightarrow$ liší se o násobek p nebo q

Diffieho-Hellmanova výměna klíčů

- Veřejné parametry: prvočíslo p, generátor g grupy $\mathbb{Z}_p^*$
- Alice vygeneruje $x \in \{0 - p-2\}$ a pošle Bobovi $g^x \pmod p$
  Bob —"— $y \in \{0 - p-2\}$ —"— Alici $g^y \pmod p$
  ⇒ oba umí spočítat $g^{xy} = (g^x)^y = (g^y)^x$,
  ale Eva nikoli (leda by uměla počítat diskrétní log) ← nicméně tohle není ekvivalence
- Pozor, aktivní útočník může vstoupit do komunikace a nechat každou stranu, ať si bezpečně vymění klíč s ním 😉 (pak přeposílá zprávy)
  ⇒ nutno podepisovat! Typicky: pomocí RSA podepíši hash vygenerovaného klíče (v obou směrech)
  (dopředná bezpečnost: prozrazení soukromého klíče neohrozí dřívější komunikaci)
- Pokud se strany na parametrech p, g dohadují (nebo nevěříme tomu, kdo je stanovil), musíme kontrolovat, že p je prvč. a g generátor (obojí už umíme)
  ⇒ jinak útočník zvolí g, které generuje dost malou podgrupu, aby v ní uměl logaritmovat 😉
- Podobně by aktivní útočník mohl najít k takové, aby g^k generovalo malou podgrupu, a pak vyměnit g^x za $(g^x)^k$ a podobně g^y za $(g^y)^k$ ⇒ tím Alici i Boba zahnal do podgrupy.
  (A oni by pak spokojeně podepsali vygenerovaný klíč...
  lépe: podepisovat celý průběh protokolu)
- DH prozrazuje 1 bit: $2\left(\frac{x}{p}\right)$ se pozná lichost x, podobně lichost y
  ⇒ umíme poznat $\left(\frac{g^{xy}}{p}\right)$, tedy zda protokol vygeneruje QR.
- $\mathbb{Z}_p^*$ má určitě 2 podgrupy: $\{-1, 1\}$ a QR.
  řádu 2 řádu $\frac{p-1}{2}$

  Pokud $\frac{p-1}{2}$ je prvočíslo. (tedy $p = 2q+1$), pak už žádné jiné.
  ⇒ nikdo nás do nich nenažene.

  A pokud se budeme pohybovat v podgrupě QR, už nebudeme ani vyzrazovat informace.
  ⇒ místo generátoru tedy počítejme g^2 + testujeme, zda g^x, g^y leží v podgrupě.

- Abychom se vyhnuli velkým exponentům… Zvolíme $p = kq+1$, kde q má cca 256b, p výrazně více. Pracujeme v podgrupě generované g^k, ta má q prvků.
  $\Rightarrow$ Opět kontrolujeme, zda se držíme v této podgrupě ($a^q = 1$) a opět nás nikdo nemůže zahnat do menší.
- Obecně: DH funguje v grupách, v nichž je dlog těžký a které nemají netriviální podgrupy
- $p = 2q+1$ je obecně bezpečné ($p-1$ nesmí mít samé malé faktory, jinak umíme dlog)
- Sémantická bezpečnost: zjistit nejnižší bit je stejně těžké jako zjistit všechno (podobně jako RSA)
- DH jako asymetrická šifra: veřejný klíč je g^x, tajný x.

ElGamalův kryptosystém – asym. šifra založená na dlog

__Parametry:__ prvočíslo p, generátor g grupy $\mathbb{Z}_p^*$

__Klíče:__ $k \in_R \{0 \ldots p-2\}$ $\rightarrow$ tajný klíč k
$h = g^k \bmod p$ veřejný klíč h

__Šifrování:__ $t \in_R \{0 \ldots p-2\}$ $\rightarrow$ pošleme zprávu (g^t, y)
$s = h^t \ (= g^{kt})$ ← rovnoměrně náhodný prvek $\mathbb{Z}_p^*$
$y = x \cdot s$ ← taktéž, neboť s je invertibilní

__Dešifrování:__ $s = (g^t)^k$ … rekonstruujeme sdílené tajemství s
$x = y \cdot s^{-1}$ … pomocí s dešifrujeme

(tohle je vlastně DH: 1. krok při generování klíče, 2. krok při šifrování. Tím vyměníme s a pak jím zašifrujeme x.)

! randomizace je kritická, jinak z known plaintextu spočítáme s **!**

Podobně jako RSA prozrazuje, zda x je QR … ale to jde snadno napravit vybíráním zpráv jen z množiny QR

podmínka z veřej. klíče:

① pokud k je sudé: $\left(\frac{h}{p}\right) = \left(\frac{g}{p}\right)^k = (-1)^k = 1$
$\Rightarrow \left(\frac{h^t}{p}\right) = 1 \Rightarrow \left(\frac{x}{p}\right) = \left(\frac{y}{p}\right)$

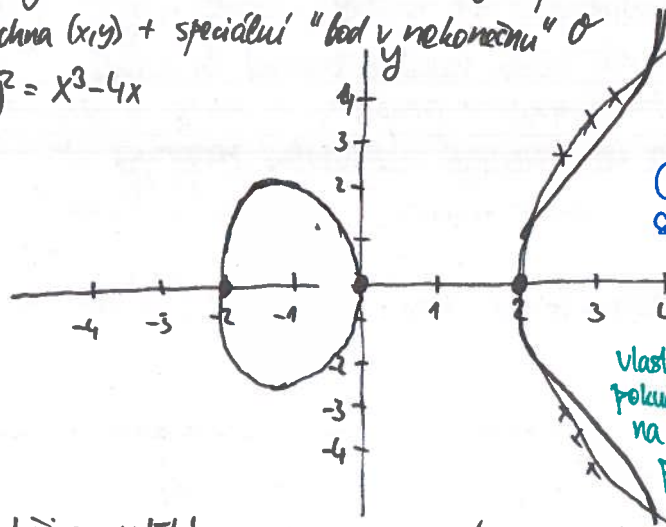
② pokud k je liché: $\left(\frac{h}{p}\right) = -1$
$\left(\frac{s}{p}\right) = \left(\frac{h^t}{p}\right) = (-1)^t = \left(\frac{g^t}{p}\right)$
$\Rightarrow \left(\frac{y}{p}\right) = \left(\frac{x}{p}\right) \cdot \left(\frac{g^t}{p}\right)$

__Obecně:__ ElG. lze provozovat v jakékoliv grupě, v níž je dlog těžký (podobně jako DH)

Na rozdíl od RSA musí být šifr. klíč veřejný a dešifr. tajný, protože z dešifr. bychom uměli udělat šifr.

Eliptické křivky – dobrý zdroj malých grup s těžkým dělogem

- Nad reálnými čísly: uvažme množinu všech $(x,y) \in \mathbb{R}^2$ t.ž.
  $y^2 = x^3 + ax + b$ (kde a,b jsou param. t.ž. $4a^3+27b^2 \neq 0$)
  $E :=$ všechna (x,y) + speciální "bod v nekonečnu" $\mathcal{O}$



- Příklad: $y^2 = x^3 - 4x$

Tohle zaručuje právě 3 průsečíky s osou x (jinak je křivka singulární a chová se jinak)

Vlastně říkáme: pokud jsou P, Q, R na spol. přímce, pak $P+Q+R=\mathcal{O}$

- Z bodů na křivce uděláme grupu s operací + a neutrálním prvkem $\mathcal{O}$
  Jak vypadá $P+Q$ pro $P=(x_1,y_1)$ a $Q=(x_2,y_2)$:
  1. $x_1 \neq x_2$: uvažme přímku PQ. Ta křivku protne ve 3 bodech: P, Q, R.
     Za výsledek prohlásíme zrcadlový obraz bodu R podle osy x.
  2. $x_1 = x_2$, $y_1 = -y_2$: výsledek je $\mathcal{O}$
  3. $x_1 = x_2$, $y_1 = y_2$: podobně jako 1., ale přímka bude tečna v bodě $P=Q$.

  Triviální: $P+Q = Q+P$, $P + (-P) = \mathcal{O}$ (-P přehlopení zrcadlením y)
  Netriviální: + je asociativní (lze dokázat pracně mechanicky nebo vybudovat hlubokou teorii)

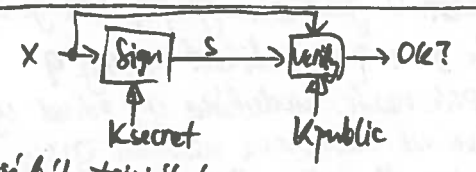
- Totéž můžeme budovat nad konečným tělesem mod p, $p>3$
  [použijeme tytéž formule pro definici +,-]
  → zase vznikne abelovská grupa (komutativní)

nebo $\mathbb{F}_{p^k}$ pro $p>3$ [pro $p=2,3$ to funguje trochu jinak]

- Věta [Hasse]: Je-li E el. křivka nad tělesem $\mathbb{F}_q$,
  pak: $q+1-2\sqrt{q} \leq |E| \leq q+1+2\sqrt{q}$.
  → existuje Schoofův alg., který $|E|$ spočte přesně v poly čase.

- **Věta:** Je-li $(\mathcal{E},+)$ eliptická křivka nad $\mathbb{F}_q$, pak $\exists n_1, n_2$:
  $(\mathcal{E},+) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, přičemž $n_2 \mid n_1$.
  → pokud $|\mathcal{E}|$ je prvočíslo nebo součin různých prvočísel, pak $n_2 = 1$, takže $(\mathcal{E},+)$ je cyklická grupa ⇒ funguje v ní DH. ← jen roli g^k hraje $k \cdot G$ ($G \in N$ řád výrazu)
  → jinak můžeme najít cykl. podgrupu velikosti n_1.
- **komprese bodů:** místo páru (x,y) stačí přenést x + 1 bit, který vybere jednu z možných druhých odmocnin (1 je sudá, druhá lichá ⇒ stačí nejnižší bit)
- **eliptický ElGamal:** DH na křivce, pak hešovací fce z křivky do $\mathbb{Z}_p$, kde provedeme maskování zprávy.
- **bezpečné křivky** (s těžkým dlog) není snadné sehnat: na mnoho typů křivek existují efektivní algoritmy na dlog?
  → https://safecurves.cr.yp.to/

Asymetrické podpisy



- Sign může využívat náhodu ⇒ verify nemusí být triviální
- útočníkova:
  - zjištění tajného klíče
  - existenční padělání (vytvoří podpis nějaké nové zprávy)
  - cílené padělání (podpis předem určené zprávy)
- možnosti útočníka:
  - zná veřejný klíč
  - zná podpisy nějakých zpráv
  - může si nechat podepsat, cokoli bude chtít různé od zadaných ze

- **podpisy pomocí RSA:** tajný e, veřejný d, $sig = x^e \bmod n$.
  - exist. padělání na základě ~~tajného~~ veř. klíče: vyberu si sig, pak $x := sig^d \bmod n$. (to je nejspíš nesmyslná zpráva)
  - ze slepých podpisů plyne cílené padělání při CPA.
  - oprava: zprávu před podepsáním hešuji.

- **ElGamalův podpis** (založený na dlogu)
  - Parametry: p, g (jako u DH)
  - Tajný klíč: $k \in_R \{2 \dots p-3\}$
  - Veřejný klíč: $a = g^k \bmod p$
  - Podpis(x): $t \in_R \{2 \dots p-3\}$ t.ž. $t \perp p-1$
    $r \equiv g^t \pmod p$
    $s \equiv (x - kr) \cdot t^{-1} \pmod{p-1}$
    $\Rightarrow$ podpis (r, s)
  - Verify(x, r, s): ① $0 < r < p$, $0 < s < p-1$
    ② $g^x \equiv a^r \cdot r^s \pmod p$
  - proč funguje: $a^r \cdot r^s \equiv g^{kr} \cdot g^{t(x-kr)t^{-1}} \equiv g^x$
  - opět nepříjemné algebraické vlastnosti $\Rightarrow$ hůřejšího
  - lze provést v jakékoliv grupě, kde je dlog těžký
    $\Rightarrow$ pak je z $p-1$ velikost grupy q
    $\Rightarrow$ r pak navíc moduline q, pokud vyjde 0, přegenerujeme t

navíc má zabudované hezkými zprávy

- **Digital Signature Algorithm** [USA 1994]
  - Parametry: $p = c \cdot q + 1$ (p je ~4Kbit, q cca 256b),
    g je generátor podgrupy $\mathbb{Z}_p^*$ velikosti q (tedy c-tá mocnina generátoru $\mathbb{Z}_p^*$)
  - Tajný klíč: $k \in_R \{1 \dots q-1\}$
  - Veřejný klíč: $a = g^k \bmod p$
  - Sign(x): $t \in_R \{1 \dots q-1\}$
    $r \equiv (g^t \bmod p) \bmod q$, pokud $r \neq q$, ↺ znovu
    $s \equiv t^{-1} \cdot (\text{hash}(x) + kr) \bmod q$, pokud $s \neq q$ ↺ restart
    $\rightarrow$ podpis (r, s) (příjemně krátký)
  - Verify(x, r, s): $s^{-1} \bmod q$
    $u_1 \equiv \text{Hash}(x) \cdot s^{-1} \pmod q$
    $u_2 \equiv r \cdot s^{-1} \pmod q$
    check $(g^{u_1} \cdot a^{u_2} \bmod p) \equiv r \pmod q$

Nelze přímo vyrobit z ElGamalovy šifry, neboť ta nemá šifrovat tajným klíčem a dešifrovat veřejným

- Proč to funguje: z $S \equiv_q t^{-1}(h(x)+kr)$

dostaneme $t \equiv_q S^{-1}(h(x)+kr) \equiv \underbrace{S^{-1}h(x)}_{u_1} + \underbrace{S^{-1}kr}_{u_2 \cdot k}$

Proto $g^t \equiv_p g^{u_1} \cdot \underbrace{g^{u_2 k}}_{a^{u_2}}$

↓ tohle dá r po modulo q

- Podobně ECDSA s eliptickou křivkou: místo $g^{\text{něco}}$ operace v grupě křivky.

! Ve všech variantách DSA / ElGamala nesmíme opakovat t

Pro obyč. ElGamala:

① Pokud se prozradí t, pak: víme $S \equiv (x-kr)\cdot t^{-1} \pmod{p-1}$ (vše kromě k známe)

$tS \equiv x - kr$

$kr \equiv x - ts$

$k \equiv (x-ts)\cdot r^{-1}$ a máme taj. klíč

② Pokud zopakujeme t, zopakuje se i r ⇒ pro zprávy x_1, x_2 máme podpisy $(r, s_1), (r, s_2)$.

z definice: $g^{x_1} \equiv_p \underbrace{a^r}_{g^{kr}} \cdot \underbrace{r^{s_1}}_{g^{ts_1}}$ ⇒ $x_1 \equiv_{p-1} kr + ts_1$

⇒ $x_2 \equiv_{p-1} kr + ts_2$

⇓

$x_1 - x_2 \equiv_{p-1} t(s_1 - s_2)$

⇒ kdykoliv $s_1 - s_2 \perp p-1$, umíme zjistit t ⇒ ①

Oprava (dnes již běžná): t generuji PRNG současným zprávou x a klíčem (třeba pomocí hašovací funkce)

Typické protokoly

typicky pak symetrická šifra a MAC

① vyjednáme si klíč (master secret)
② pošlu zašifrovaný veř. klíčem protistrany
③ oba podepíšeme dosavadní průběh protokolu
④ ostatní klíče odvodíme z hlav. klíče

① vygeneruji náh. klíč (master secret)

nebo DH výměna klíčů

⇒ pak získám dopřednou bezpečnost (i když vyzradím taj. klíč, nemůžu dešifrovat staré zprávy)

0 vymyslíme si nonce (proti replay)

Implementační záležitosti

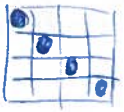
- nemůžeme navrhovat bezpečný SW ... největším nepřítelem je komplikovanost
- nemůžeme ho ani implementovat
  - testování bezpeč. chyby neodhalí (fuzzing trochu pomáhá)
    ... ale při výuce na to obvykle spoléháme
  - píšeme v Céčku ⇒ buffer overruns
  - nepíšeme v Céčku ⇒ side channels je nemožné kontrolovat
- příliš mnoho závislostí: SW i HW
  ... navíc většina z nich není optimalizovaná na bezpečnost
- rozdíl útočník má k dispozici víc informací a možností zasahovat než útočník teoretický ☹ (různé postranní kanály)
  - útočník po síti: - časovací útoky (memcmp, padding oracle, ...)
    - data-dependent instruction timing
    - generování nekorektních dat / nejednoznačných
      → útoky na parser
      → v jakém formátu jsou data?
      (JPEG ∩ Java, UTF-8 malformed ...)
      zero-terminated / counted strings
      - zahlcení (JSON každý parsuje trochu jinak
        příjemce, zahlcování sítě
        » XML také, třeba <!--->

    Chceme, aby primitiva trvala vždy stejně dlouho
  - v téže místnosti: - měření spotřeby (modular exponentiation v RSA)
    - elmag. záření
    - zvuk (klávesy, pískání měničů) → lze snímat chvění okna
    - tepelné stopy (třeba po hesle)
    - ovlivňování výpočtu elmag. pulsy, napájením atd.

- fyzický přístup k počítači:
  - "cold boot attack" (viz. tekutého dusíku)
  - přidání špehujícího HW
  - otevřený ve vypnuté paměti
- jiný program na tomtéž stroji: (třeba Javascript)
  - HW side-efekty (třeba keše) ← zejména s HyperThreadingem
  - CPU bugs

Příklad: Keřový útok na AES [Bernstein 2005] (128b klíč) ... synchronní verze (na totožném stroji běžícím těsně před/po AES), chosen plaintext

Typ. implementace rundy: ① xor rundového klíče (v první rundě xor s vnitřkem tabulky) 256 × 32b tabulky, takto se snadno útočí na decrypt

② 

$\rightarrow (T_0(a_0) \oplus T_1(a_1) \oplus T_2(a_2) \oplus T_3(a_3))$

1. sloupec ~~řádek~~ využečkan

a podobně pro další 3 ~~sloupce~~ posunuté diagonálky, týmiž tabulky

↓

10 rund, poslední atypická (jiné 4 tabulky)

Keš má 64B bloky ⇒ 16 položek tabulky na blok ⇒ z čísla bloku poznáme 4 bity stavu ⇒ pro známý plaintext 4 bity klíče

ovšem v dalších rundách se do keše otisknou další přístupy do tabulek ... a vícemémě náhodné

Nechť j je blok, v T_i, do kterého se neotiskne 1. přístup

... a dalších provedeme ještě $9 \cdot 4 - 1 = 35$ přístupů do téže tabulky

$\Rightarrow Pr[\text{žádný z nich se netrefí do } j] = (1 - 1/16)^{35} \doteq 0.1$

$\Rightarrow$ při malém počtu náhodných pokusů izolujeme jediný řádek, ke kterému se přistoupí vždy

- z první rundy máme 64 bitů klíče, trochu sofistikovanější útok na 2. rundu pak dává zbylých 64 ☺
- stačí měřit rychlost AES v závislosti na vstupu, funguje i pro known plaintext místo chosen
- Kyjovský nejlepší: řádové šetření pokusů bez znalosti plaintextu (!)
- Obrana: bit-slicing implementace S-boxů (toto je problém každé šifry s velkými S-boxy)
  HW implementace AES (v CPU)

Ukládání tajemství

- klíče v paměti: mohou skončit na disku (swap, core dump, ...) nebo na síti (po uvolnění paměti někdo alokuje paket a inicializuje jen částečně)
- klíče v registrech: mohou skončit v paměti (typ. zásobník)
- best practices: - mlock & vypnutí core dumpů
  - po použití smazat
  - případně dělení tajemství na 2 části

- na da na disk (x dnu 4

P na rague

ealo adných

st na le

zat pro nu

staý disky se ter šifr pa su Error

ter pr ndlovd

W

cast proof ovedem ylaz apajem ada

/ sm leny - jedno dy det nu dy proceso

- lf-destr otevení

ap - andom elana pr

dné al se

lez f nar ch ad ndu sou)

st | - , aby NG apd nesmí era

prcel lo power Ll oven zálohy, first boot

|To |

není dokonalé, vše jde překonat...

pr ujme se fyzickou bezpečností:

- cena útoku > cena chráněných dat
- útočníka chceme zdržet, aby byl chycen (=> motivace neútočit)
- logy, monitorování, ..., aby bylo chycení snazší
- "požární cvičení"
- penetration testing

Hesla

Problémy:
- jednoduchá hesla lze snadno uhodnout (brute-force, slovníky...)
- složitá hesla se těžko pamatují
- uživatelé jsou lidé ⇒ papírky s hesly, totéž heslo na více místech
- jak nastavit politiku hesel?

A co když ze serveru unikne DB uživatelů s hesly? ~ problém totéž

- hesla hešujeme ... ale útočník stále může provést brute-force útok offline
- předpočítané tabulky hešů, 1. pokus:
  - heslo → hash → jiné heslo
    - případ fce f
    - funkce vzorkující prostor nadějných hesel
  - řetězky vzniklé iterováním f: start → → → → konec
    - N kroků
    - prostor hesel pokryjí řetězky
    - z každého si uložím začátek a konec
    - z neznámého hše zkusím N kroků, než se trefím do konce řetězku; pak najdu začátek a od něj ... předchůdce hše.
    - v ideálním případě zmenším paměťové nároky tabulky N-krát za cenu N-krát pomalejšího hledání
  - problém: řetězky srůstají ⇒ k 1 konci může patřit víc začátků, R řetězků pokryje $\ll N \cdot R$ hesel.
- duhové tabulky (Rainbow tables)
  - v každém kroku řetězku používají jinou vzorkovací fci – "barvy"
  - pokud se 2 řetězky potkají, brzy se zase rozejdou ⇒ téměř dlouhé.
  - při hledání potřebují zkoušet všechny možné pozice v řetězku ⇒ hledání zpomalují: N^2-krát, paměť redukují cca N-krát.

Příklad: project-rainbowcrack.com
pro SHA-1, ASCII, 1-8 znaková hesla: 460 GB tabulka

Obrana proti brute-force útokům:
- "solení" hesel: hesla hešují s nonci, tu uložím spolu s hešem
  ⇒ z hešů nepoznám, že 2 hesla jsou stejná
  ⇒ duhové tabulky nepomohou

- iterování hešů: nám 1000× zpomalení ověřování hesla neradší útočníkovi zato zásadně :(
  - tzv. key stretching – iterováním hešů také můžeme vyrobit PRNG seedovaný heslem a získat tak z hesla klíč vhodné délky
- jiný přístup: neukládáme část nonce, takže musíme zkoušet :(
  - tzv. key strengthening

→ Příklad: PBKDF2 (Password-Based Key Derivation Function)

- odvozená z libovolné PRF (pseudonáhodná fce s klíčem, typicky HMAC)
- výstup: $B_1 B_2 B_3 \dots$ (podle požadované délky výstupu)

přičemž: $B_i = U_1^i \oplus U_2^i \oplus \dots \oplus U_c^i$ ← # iterací

$U_{j1}^i = PRF(password, salt \| i)$

$U_{j,n}^i = PRF(password, U_{j,n-1}^i)$

↳ pokud je moc dlouhé, tak jeho heš (tím vznikne z ekviv. hesla)

Další vývoj: snažíme se zkomplikovat paralelizaci na GPU/FPGA
⇒ typicky zvýšením požadavků na paměť
(to vše jen kvůli slabým heslům, silná nepotřebují ani iterace)

↓

Argon2 (nástřelek)

Parametry: M = množství paměti
P = stupeň paralelismu
T = # iterací

1KB bloky tak, aby jich bylo celkově M paměti

Vstupy: heslo
sůl
(taj. klíč, asociovaná data)

ne moc blízkým, aby fungovala paralelizace

↓ nepovinné

komprimuje blok dolera od akt. (cyklicky) s vybraným předch. blokem.

Varianty:
- deterministicky (podle PRNG)
- v závislosti na datech z předch. bloku

- na začátku vyplníme první z sloupce hešů vstupů a parametrů
- v iterace postupně vyplní všechny sloupce, výsl. přixoruje k pův. obsahu
- používá kompresní fci (1KB, 1KB) → 1KB odvozenou z Blake2 (to je heš odvozený z ChaCha20 ze soutěže SHA-3)

! potřebujeme synchronizované hodiny aspoň ± pár minut
↳ jak to udělat bezpečně?
po které si pamatujeme pakety

Ve skutečnosti: (Kerberos v5)

- TGS je služba jako každá jiná ⇒ klient pro ni má také tiket (TGT)
- místo klíčů z tiketů (které mohou mít dlouhou životnost) používáme autentikátory pro specifické sessions:

$A_{A,B} = \{$ A, A_1, timestamp, session key$\}_{K_{A,B}}$ ← z tiketu

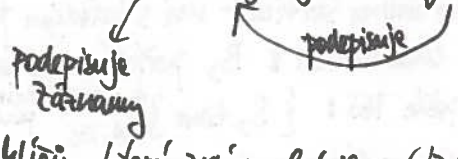
↳ na 1 použití - během replay okna si pamatuji všechny, co jsem viděl

- klient se může prvotně autentikovat heslem: ~~autorizační~~ autentifikační server umí vydat TGT zašifrovaný hešem hesla
- abychom zabránili offline útokům na hesla: preautentikace - pošlu auth. serveru timestamp zašifrovaný hešem hesla

---

DNSSEC - Secure Domain Name System

- DNS: - strom doménových jmen
  - ve vrcholech záznamy různých typů (A, AAAA, MX, ...)
  - delegace subdomén → NS záznamy, zóny vs. domény
- každá zóna obsahuje klíč (záznam DNSKEY)
- klíčem podepisujeme záznamy (pro jméno + typ → záznam RRSIG)
- nadřazená zóna podepíše klíč podřízené (záznam DS, kde je NS)
- můžeme používat více klíčů: - rotace klíčů
  - zone-signing key vs. key-signing key



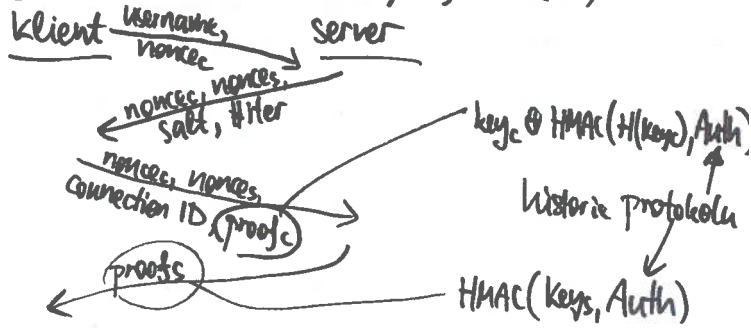
- "root of trust" - množina klíčů, které známe lokálně (třeba od root zóny)
- zónu stačí podepsat offline a pak jen servírovat hotové RRSIGy
- Jak se podepisuje neexistence záznamu? Podepisujeme díry!

X.Y.Z NSEC X'.Y.Z (typy záznamů pro X.Y.Z)

↳ nejbližší další jméno v kanonickém pořadí

Interaktivní autentikace heslem (4A)

- **challenge-response:** server pošle nonci, klient hash(heslo || sůl || nonce) a sůl (pro neexistující username si ji vymyslí)
  - ... riziko: server si musí pamatovat plain-textová hesla
    - ... nebo jejich hashe, ale ty pak musí poznat i klient ⇒ nepoužije
- **protokol SCRAM** (Salted Challenge-Response Authentication Mechanism)
  - určím sůl a #iterací
  - z hesla odvodím K_c = HMAC(heslo, "Client key") K_s = HMAC(heslo*, "Server key")
  - server si pamatuje: username, sůl, #iterací, K_s, hash(K_c)
  - průběh:



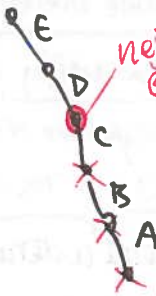
  - pokud znám H(key$_c$), mohu rekonstruovat key$_c$ (pak ho chci rychle zahodit...)

Kerberos [MIT 198x] – distribuovaná správa klíčů pomocí sym. kryptografie

- protokolu se účastní "principálové" – klienti a servery
  - Ticket Granting Service – má s každým principálem společný taj. klíč
  - když A chce mluvit s B, pořídí si tiket T_{AB}:
    - A pošle TGS: $\{B, time\}_{K_{A,TGS}}$ ← zašifrováno pomocí
    - TGS vyrobí session key $K_{A,B}$ a pošle A: $\{K_{A,B}\}_{K_{A,TGS}}$, $T_{A,B}$
      kde tiket $T_{A,B} = (B, \{A, \text{adresa } A, \text{time range}, K_{A,B}\}_{K_{B,TGS}})$
    - A přepošle $T_{A,B}$ ⟶ B
    - B případně pošle $\{time\}_{K_{A,B}}$, aby se také autentikoval
    - dále už šifrujeme pomocí $K_{A,B}$ zprávy mezi A, B.

- Neexistence je ale složitější kvůli hraničním zón a wildcardům.

Pro A.B.C.D.E



Vygeneruji:
1. NSEC pro D.E dokazující, že D.E existuje a není NS
2. NSEC pro díru pokrývající celé jméno
3. NSEC dokazující neexistenci *.D.E

- Drobná vada na krásе: ~~Výpis~~ záznamy v zóně jde pomocí NSEC enumerovat
  → NSEC3: místo jmen používá jejich hеše → díry mezi hеši.
  Nutno řešit i všechny předky uvnitř zóny, kterých v ② uměl dokázat ∄ C.D.E

Ověřování identity aneb kdo to je na druhém konci „drátu"?
- typicky znám veřejný klíč protistrany, ale nevím, komu patří

- PKI - Public Key Infrastructure
  - Certifikační autorita (CA): všichni jí věří a znají její veřejný klíč
    → ke každému klíči vydá certifikát = podepsanou zprávu s hešem veřejného klíče a identitou (a nějakým omezením platnosti časovým)
  - protistrana pak dodá podpis, veřejný klíč a certifikát
    ↑ ověřím veř. klíčem ↑ ověřím certifikátem ↑ ověřím veř. klíčem CA
  - Výhody: - CA vezná soukromé klíče (oproti Kerberovi)
    - CA je offline, k navázání spojení ji není potřeba
    - PKI je univerzální, ověřuje identitu pro všechny aplikace
    - lze decentralizovat zavedením sub-CA a řetězci certifikátů
  - Problémy: - nenajdeme nikoho, komu věří všichni (ani většina)
    - co vlastně je identita? (Jan Novák, firma na Bahamách, ...)
    - revokace certifikátů (musí být aspoň částečně online)
      ↳ CRL / online protokol / krátkodobé certifikáty

- Trust On First Use - SSH, ale vlastně tak používáme i většinu webu
- Web Of Trust - PGP - vzájemné podepisování klíčů, důvěra částečně tranzitivní
  - je pro většinu uživatelů příliš složité

Co tedy funguje? - PKI specifická pro aplikaci (třeba klienti banky)
- TOFU + nezávislé ověření při FU

TLS: Transport Layer Security

- původně SSL vyvinuté firmou Netscape pro HTTPS, dnes všudypřítomné
- evoluce: SSL1 → SSL2 → SSL3 → TLS1.0 → 1.1 → 1.2 → 1.3
  - SSL1: nepublikováno
  - SSL2 až 1.1: obsoletní a deravé
  - 1.2: dnes běžné (o něm budeme mluvit)
  - 1.3: horká novinka
- v podstatě meta-protokol, skoro vše je volitelné
  - šifrování + MAC
    - proudová šifra + MAC
    - bloková šifra + MAC
    - } MAC-then-encrypt (!)
    - AEAD (autentifikovaná šifra - třeba AES-GCM)
  - PRF pro odvozování klíčů
    - ve starších verzích fixní (založená na HMAC)
    - dnes (1.2) volitelná
  - výměna klíčů: (generuje pre-master secret, z něj master-secret, z ně dalších klíče)
    - ~~RSA~~: klient generuje klíč, zašifruje veř. klíčem serveru
    - DH + RSA: fixní parametry DH v certifikátu
    - DHE + RSA: Ephemeral DH, podepisuje parametry veř. klíčem
    - ECDHE + ECDSA: podobně s eliptickými křivkami
    - DHE-anon: bez ověření protistrany (MITM!)
    - PSK: pre-shared
    - DHE + PSK: místo certifikátu použije PSK
    - Kerberos
    - (a mnoho dalších)
  - dohoda se domluví na cipher suite, jak např.:

    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

    key xchg | auth | cipher | key size | mode | MAC & PRF
  - komprese
- základem je Record Protocol:
  - posílá po TCP spojení záznamy, v nich zprávy dalších protokolů
  - záznamy mají protokol, typ a délku
  - zajišťuje šifrování a MAC domluvenými alg. (na začátku žádné)

Handshake Protocol

K → S ClientHello
- verze protokolu (max. podporovaná)
- Client random
- podporované suity a kompresní algoritmy
- seznam rozšíření (typ + délka + hodnota)

← ServerHello
- vybraná verze protokolu
- server random
- vybraná suite a mód komprese
- seznam rozšíření

← [Certificate] - certifikát serveru

← [ServerKeyExchange] - závisí na zvoleném algoritmu pro KX

← [CertificateRequest] - chceme i auth. klienta

← ServerHelloDone - deklarace, že server je hotov s KX

→ [Certificate] - cert. klienta

→ ClientKeyExchange - klientova část KX (povinná)

→ [CertVerify] - podpis dosavadních zpráv certifikovaným klíčem

→ ChangeCipherSpec - klient deklaruje přechod na novou šifru
(pozor, tohle je samostatný sub-protokol)

→ Finished - handshake complete
- podpis: PRF(master secret, "client finished" | hash(handshake messages))
  - spočítá se z premaster secretu a server/client random

← ChangeCipherSpec

← Finished - teprve tímhle se u RSA auth ověří, že server má soukromý klíč ke svému certu

zajímavá rozšíření

- session resume - ServerHello obsahuje ID, pod kterým server ukládá session
  - další spojení:
    - ClientHello požádá o resume s dřívějším ID
    - zkrácený handshake (jen 2× hello a 2× finished)
      - nové klíče
      - ~~nový master secret~~ se odvodí ze starého mast. secretu a nových náhodných dat

- session tickets – podobné, ale celý stav si pamatuje klient (zašifrovaný serverovým klíčem)
- server name – pro virtual hosting (viz Host: v HTTP)
- ALPN (app-level protocol negotiation ... třeba HTTP 1 vs. 2 vs. SPDY)
  - klient nabídne protokoly, server vybere jeden
- **Re-negotiation** – lze iniciovat opětovné spuštění dohadování (od Hello)
  - třeba po přenesení pár GB dat (chceme nové klíče)
  - nebo jsme v průběhu HTTP zjistili, že chceme klientův certifikát
  - TLS ≤ 1.2 má v re-neg zásadní bug: nepodepíše návaznost na předchozí nego. ⇒ elegantní MiTM útok
    - navážu spojení se serverem, pošlu část dat, spustím reneg.
    - pak propojím se skutečným klientem, ten nego. dokončí
    - ⇒ umím vnutit prefix relace (třeba HTTP GET, k němuž klient doplní cookies nebo svůj cert)
  - secure reneg. extension ⇒ doplňuje návaznost do podpisů
- **Close Alert** – podepsané ukončení spojení
  - klienti často ignorují ⇒ cookie cutting attack

Útoky na SSL/TLS

- BEAST (Browser Exploit Against SSL/TLS)
  - TLS ≤ 1.0 používá CBC s jednou IV – celé spojení je 1 posloupnost bloků
  - tím pádem víme, jaká IV se použije pro další zprávu
    ⇒ 1. blok další zprávy je efektivně ECB
  - CPA: umíme zjistit, zda se CP zašifruje stejně jako některý z předchozích bloků
  - navíc můžeme CP paddingem zařídit, aby předch. blok obsahoval hodně známých dat + trochu tajných (třeba 1. znak cookie)
- Compression side-channels
  - ⇒ CRIME (Compression Ratio Info-leak Made Easy)
  - ⇒ chceme vypnout kompresi

útočíme na cookies, XSRF tokens atd.

- Lucky 13 – CBC padding oracle (MAC-then-encrypt)
- POODLE (Padding Oracle On Downgraded Legacy Encryption)
  - mnoho implementací lze donutit k downgrade na SSL3

- SSL3 nekontroluje obsah paddingu
- zařídíme, aby pošl. blok obsahoval jen padding
  → poslední bajt bloku je B-1, předchozí libovolné
- zašifrovaný blok vyměníme za jiný (o němž chceme něco zjistit)
  → s Pr = 1/256 vyjde po dešifrování a XORu s předch. blokem B-1 na konci; jinak nesedí MAC a spojení se rozpadne
  → tehdy zjistíme pošl. bajt vybraného bloku (xor...)
  → pak posuneme plain-text (CPA) a iterujeme...

- DROWN (Decrypting RSA using Obsolete and Weakened eNcryption)
  - ve starších verzích funguje Bleichenbacherův útok
  - pokud server umí více verzí, použijeme starou jako orákulum na lámání nové!
    ↑ se stejným certem
- ROBOT (Return Of Bleichenbacher Oracle Threat)
  - i TLS 1.2 pořád používá PKCS #1 v 1.5, ale s work-aroundy proti Bleich. útokům
  → ještě stále se najdou varianty útoku, které fungují

- Shrnutí:
  - nechceme používat blok. šifry s CBC → buď proudové nebo GCM
  - chceme zakázat obsolétní verze a suity
  - jsou potřeba rozšíření protokolu

Internet PKI

- PKI založená na standardu ITU X.509 ← obskurní, překomplikovaný ASN.1, původně určený pro jiný svět (ISO/OSI, X.500)
- cert. autority
  - typicky komerční – co je jejich zájmem?
  - pár neziskových – hlavně Let's Encrypt
  - je jich hrozně moc (Firefox momentálně věnává 181 kořenových certů!)
    → jak pravděpodobné je, že všechny CA jsou
      a) poctivé,
      b) dostatečně důsledné?

- mezilehlé (intermediate) certifikáty
  - podepsané root klíčem, dál podepisují → cert chains
  - některé používá samo CA, jiné deleguje (?)
  - mohou mít omezení na domény / použití
- jiný distribuovaný model: 1 CA, více registračních autorit

! složité, ve validaci časté chyby

- typy certifikátů: DV = domain-validated (držitel měl podkontrolu domén)
  OV = organization-validated (legal entity)
  EV = extended validation
- certifikát obsahuje:
  - subject (x.500 DN !)
  - subject alt. names - domény, e-mail. adresy &c.
  - heš veřejného klíče
  - identifikaci vydavatele & podpis
    - vlastně vadrátovaný cert (root je self-signed)
  - použití: server / klient / code signing / CA / ...
  - časový interval validity
  - množina rozšíření
- revokace certifikátů:
  - CRL (Cert Revocation List) - velké seznamy, pomalý download → aktualizují se zřídka
  - OCSP (Online Cert Status Protocol) - cert odkazuje na OCSP responder
    - problémy se soukromím (nešifrováno, jen podpisy odpovědí)
    - pomalé a nespolehlivé → klienti dělají soft-fail ↯ (triviální MiTM útok)

  → TLS extension: OCSP reply stapling
  → cert extension: must-staple (zrušim ji klienti moc nemusí...)
  - Google: crlset
  - Mozilla: OneCRL

  } proprietární seznam, klientem průběžně stahovaný automaticky se do něj propagují revokace klíčů CA a "high-profile" sítes

  ... a Chrome dnes ani klasický OCSP nepoužívá :o

- CA/Browser fórum: stanovuje požadavky na CA
  → pravidla, povinné audity atd.
  → za vážná porušení prohlížeče CA blacklistují (už se párkrát stalo)
- opatření proti podvodně vydaným certům
  - Perspectives – pozorování certů z více míst v síti
  - public key pinning
    - Google v Chrome pinuje klíče svých domén (nečekaně úspěšné)
    - HPKP (HTTP Public-key Pinning): pin v hlavičce odpovědi
      [dosti křehké...]
    - DANE (DNSSEC Authentication of Named Entities)
      [elegantní, ale odmítané autory prohlížečů kvůli latenci]
  - CAA v DNS – záznam omezující, která CA smí vydávat certy
  - Certificate Transparency (CT)
    - veřejné logy vydaných certů – Merkleovy stromy, lze snadno kontrolovat konsistenci
      ↳ různí provozovatelé
    - vyhledávač crt.sh
    - CA/B fórum nařizuje pro EV certy a intermediates, občas také za trest 😉
    - součástí certifikátu nebo OCSP může být potvrzení o zařazení do CT logu
    - HTTP: "Expect-CT" v odpovědi
- problémy s míchaním obsahu na HTTPS a HTTP → warningy
  ... ale co míchání DV a EV certifikace?
- uživatelé často spoléhají na HTTP redirect na HTTPS
  2023: dnes už Chrome vyžaduje
  - to by také mohlo řešit DANE, ale...
  - HTTP Strict Transport Security (STS)
    - v hlavičce odpovědi: "zapamatuj si, že tady máš používat jen HTTPS"
    - ale neřeší to first use
  - plugin HTTPS Everywhere → nespolehlivé, občas je na HTTP a HTTPS jiný obsah
    ↓
    2023: dnes už prohlížeče defaultují na HTTPS