Poznámky/skripta/něco z čeho se dá učit. Vytvořeno při opakování na zkoušku. Hlavní zdroje jsou poznámky MJ a tabule z přednášek. Občas jsem k nějakým bodům přidal své příklady, doplnil info ze stejnojmenného předmětu pro matematiky, nebo přidal dovysvětlení z internetu.

Ke konci se dost rozpadly, protože jsem nestíhal. To, co tu je, by mělo být relativně správně. Ale neručím, že tu je vše. Pls cross-referencujte.

Contents

1. Přednáška

> **POZN:** Cryptographic primitives: symmetric and asymmetric ciphers, hash functions, random generators. Protocols and roles. Kerckhoffs principle. Simple protocols: multi-party communication, signatures (symmetric and asymmetric) hybrid ciphers, challenge-response authentication. Designing an auction protocol: padding, nonces, sequence numbers, signatures, session IDs. Basic types of cryptographic attacks. Security level.

Kryptografická primitiva

Rychlé naznačení definic

> **DEF** (polopaticky) Symetrická šifra
>
> Obě strany mají sdílené tajemnství a mají krabičky na zašifrování a dešifrování zprávy pomocí tohoto sdíleného klíče.

> **DEF** (polopaticky) Asymetrická šifra
>
> Klíče jsou vždy v páru veřejný a privátní. Veřejným klíčem může kdokoliv šifrovat (zámek), privátním dešifrovat (klíč).
>
> Při podepisování se role klíčů často obrací. Privátním zašifrujeme, veřejným lidé dešifrují a kontrolují, že vyšlo to, co má.

> **DEF** (polopaticky) Hešovací funkce
>
> Jednosměrná kolizivzdorná funkce, která z libovolně dlouhého vstupu vytvoří výstup fixní délky.

> **DEF** (polopaticky) Asymetrický podpis
>
> Naznačeno v def asymetrické šifry. Zašifruji nějaký hash ze zprávy, toto je podpis.
>
> Protistrana ověřuje tak, že si hash spočítá a porovná s rozšifrovaným podpisem.

> **DEF** (polopaticky) Symetrický podpis (MAC)
>
> Obě strany mají sdílené tajmenství. Podepíšu tak, že spočítám hodnotu ze zprávy a klíče. Tu přidám ke zprávě.

> Protistrana ověřuje tak, že ze zprávy a klíče spočítá stejnou hodnotu a porovná s tou, kterou obdržela.

> **DEF** (polopaticky) Hybridní šifra
>
> Zprávu šifruju sym. šifrou s náhodně generovaným klíčem.
>
> Asymetrickou šifrou šifruju vygenerovaný klíč.
>
> Dvojici posílám.
>
> Používá se kvůli pomalosti asym. šifer.

Kerckhoffův princip
Tajný by neměl být algoritmus šifry, ale klíč.

Kategorie kryptografických útoků
Dělí se podle toho, kolik toho má útočník k dispozici.

Ciphertext-only (COA)
Útočník dostane jenom šifrové texty.

Known-plaintext (KPA)
Útočník má nějaké dvojice šifrových a otevřených textů. Snaží se dešifrovat jiný šifrový text, nebo lépe, získat klíč.

Např. se v protokolu šifruje nějaká veřejně známá inicializace, nebo něco odhadneme.

Chosen-plaintext (CPA)
Útočník si může vyžádat šifrové texty pro nějaké zprávy. Následně musí dešifrovat zprávu, kterou si nenechal zašifrovat, nebo lépe, získá klíč.

> **POZN:** Ještě se může dělit podle toho, jestli si útočník může vybírat v průběhu útoku, nebo pouze předem. (adaptive)

Chosen-ciphertext (CCA)
Útočník si může vyžádat otevřené texty pro jakékoliv šifrové texty. Následně musí dešifrovat zprávu, kterou si nenechal zašifrovat, nebo lépe, získá klíč.

> **POZN:** Můžeme dělit podobně jako CPA

Cíle útočníků

Odhalení klíče

Odhalení otevřeného textu

Rozlišení
Útočník dokáže s vyšší pravděpodobností říct, že je něco šifrovaný pomocí nějaké specifické šifry.

Jetě jsou varianty, že dokáže říct něco o typu otevřeného textu, nebo že hraje takovou hru, kdy si nechá zašifrovat dva texty a má říct jaký šifrový text je od jakého textu. Při zašifrování dobrou šifrou by měl mít šanci 50/50.

Security level
Pokud na prolomení šifry potřebujeme 2^l operací, pak je security level l.

Pozorování: $l \leq$ velikost klíče. Vždycky můžeme vyzkoušet všechny možnosti.

Základní útoky

ATTK Replay attack

Zprávu mohu poslat znovu. Záleží, jestli prostistrana uzná, že se jedná o validní zprávu.

Challenge-response protokoly se musí tomuto vyhnout tak, že úlohu vytvoří dostatečně náhodně.

Z toho plyne název ***nonce*** (Number used only ONCE).

Výplň / padding

Ke zprávě můžeme přidat jeden bit = 1, pak kolik 0 je potřeba.

Nebo pokud musíme přidat n bytů, tak přidáme n bytů s hodnotou n. Přirozeně je tento padding omezen na velikost 255.

Nějaký padding na zprávě musí být vždycky.

2. Přednáška

Narozeninové útoky

ATTK Narozeninový útok - kolize

Máme nějaký challenge-response protokol, který používá nonce, které mají n možností.

Po kolika výměnách se nonce zopakuje? Útočník si všechny předchozí pamatuje.

$\approx \sqrt{n}$

DKZ

Máme m výměn.

Funkce $f : [m] \to [n]$ určuje pro každou výměnu nonce.

Kolize nenastane $\Leftrightarrow$ f je prostá.

$$\begin{aligned} P[f \text{ je prostá}] &= \frac{n \cdot (n-1) \cdot (n-2) \cdots (n-m+1)}{n^m} \\ &= 1 \cdot \left(1 - \tfrac{1}{n}\right) \cdot \left(1 - \tfrac{2}{n}\right) \cdots \left(1 - \tfrac{m-1}{n}\right) \\ &\approx 1 \cdot e^{-\frac{1}{n}} \cdot e^{-\frac{2}{n}} \cdots e^{-\frac{m-1}{n}} \text{(vycházíme z toho, že } e^{-x} \approx 1 - x \text{ pro malé } x) \\ &= e^{-\frac{1+2+3+\cdots+(m-1)}{n}} \\ &= e^{-\frac{m(m-1)}{2n}} \approx e^{-\frac{m^2}{2n}} \end{aligned}$$

Chceme, aby $P[f \text{ je prostá}] = \frac{1}{2}$, útok pak bude mít pravděpodobnost 50%, že uspěje.

$e^{-\frac{m^2}{2n}} = \frac{1}{2}$

$-\frac{m^2}{2n} = \ln\left(\frac{1}{2}\right) = -\ln(2)$

$m^2 = 2\ln(2)n$

$m = \sqrt{2\ln(2)} \cdot \sqrt{n}$

m teda musí být $\approx \sqrt{n}$

Pokud bychom chtěli větší jistotu, změní se nám pouze konstanta, která násobí $\sqrt{n}$.

Abychom tedy rozbili nějaký protokol s nonce pomocí replay / narozeninového útoku, tak nám stačí zapamatovat si řádově $\sqrt{n}$ výměn. **Security level je tím pádem *poloviční*.**

> **ATTK** Výměna paměti za čas (meet in the middle)
>
> Máme challenge-response protokol. Sesbíráme s různých dvojic.
>
> V *první fázi*: sesbíráme s různých dvojic (úloha x, řešení).
>
> V *druhé fázi*: vyžádáme si t úloh a hledáme, jestli už jí nemáme.
>
> $P[x \text{ neznáme}] = 1 - \frac{s}{n}$
>
> $P[\forall x \text{ z } t \text{ úloh} : x \text{ neznáme}] = \left(1 - \frac{s}{n}\right)^t \approx e^{-\frac{st}{n}}$
>
> Aby byl útok úspěšný $st \approx n$. Chceme si teda pamatovat kolem $\sqrt{n}$, aby jsme to měli vyvážený.
>
> **Opět security level je *poloviční***, provádíme pouze $s + t$ operací $= 2\sqrt{n}$.
>
> Platíme ale pamětí $\Theta(s)$.

Vernamova šifra (One Time Pad)

Text seXORuji se stejně dlouhým klíčem.

Dá se zobecnit na obecnou grupu $(G, +, 0, -)$:

$E_k(x) = x + k$

$D_k(x) = x - k$

OTP je případ pro $G = \mathbb{Z}_2^n$.

Absolutní bezpečnost

Šifra je absolutně bezpečná, když text zašifrovaný náhodným klíčem je náhodný šifrový text.

OTP je absolutně bezpečný. Dokazuje se přes rozbor případů jednoho znaku ot. textu společně s jedním znakem klíče.

$$x \begin{cases} 1 \begin{cases} k=1 \Rightarrow y=0 \\ k=0 \Rightarrow y=1 \end{cases} \\ 0 \begin{cases} k=1 \Rightarrow y=1 \\ k=0 \Rightarrow y=0 \end{cases} \end{cases}$$

Je to rovnoměrně rozdělené.

Nepraktické. Klíč musí být stejně dlouhý, nebo delší, než otevřený text.

> **VĚTA** O počtu klíčů a zpráv pro absolutní bezpečnost
>
> Pokud je množina zpráv větší, než množina klíčů, pak šifra nemůže být absolutně bezpečná.
>
> > **DKZ**
> >
> > Mějme šifrový text $y \in C$.
> >
> > Pak existuje nějaké $x \in P$, že $E_k(x) = y$.
> >
> > Mějme $x \neq x' \in P$. Aby se zašifroval na y, musí být jeho klíč k' jiný. Šifra by nebyla jinak prostá.
> >
> > Každé $x \in P$ potřebuje unikátní k, aby se zašifrovalo na y. k je ale méně, než x, takže budou existovat x, které se nemůžou na dané y zašifrovat.

> Pravděpodobnost, že y je nějaký otevřený text tedy není rovnoměrně rozdělená a šifra tak není absolutně bezpečná.

Shamirovo schéma pro sdílení tajemství

Teorie:

Jakýkoliv polynom s kořeny $\alpha_1, ..., \alpha_n$ lze rozepsat jako
$p(x) = (x - \alpha_1)(x - \alpha_2) \cdots q(x)$, kde $q(x)$ je polynom bez kořenů

Nenulový polynom stupně d má d, nebo méně kořenů.

Pokud p, q jsou polynomy stupně $< d$ a $p(x) = q(x)$ v alespoň d různých bodech, pak $p = q$.
Polynom $p(x) - q(x) = 0$ v alespoň d bodech. Zároveň je ale pořád stupně menšího d. Z předchozího tvrzení musí být nulový. $\Rightarrow p = q$

> **VĚTA** Lagrange
>
> $\forall x_1, ..., x_n$ navzájem různá, a $\forall y_1, ..., y_n$
>
> Existuje polynom stupně $< d$, že $\forall i : p(x_i) = y_i$
>
> Tento polynom je jednoznačný podle před. tvrz.

Schéma:

- rozdělí tajemství na k částí,
- pomocí l z nich ho lze zpět sestavit,
- žádných $l - 1$ částí nic neprozradí.

Staví na principu, že polynom stupně $l - 1$ je jednoznačně určen l body.

Budu sestavovat polynom.

$p(0) =$ tajemství

$p(1$ až $l - 1) =$ náhodné

Máme jednoznačně určený polynom stupně $< l$.

Dohledám si zbylých $k - l + 1$ bodů $(l, ..., k)$, abych měl k částí.

Rozdám body 1 až k.

Z jakýchkoliv l bodů dokážu sestavit polynom a vypočítat $p(0)$ (Lagrange).

Pokud mám pouze $l - 1$ bodů, pak pro každý bod $(0, y)$ existuje polynom, který nimi prochází. O tajemství tedy nic nevím.

3. Přednáška

> **POZN:** Introduction to symmetric ciphers. Block ciphers: trivial examples, an attempt to define security. General constructions: iterated ciphers, substitution-permutation networks, Feistel networks. DES: history, structure, critique, work-arounds (3-DES). AES a.k.a. Rijndael: history, structure, critique.

> **DEF** Symetrická šifra
>
> Množina plaintextů P, ciphertextů C, klíčů K.
>
> Pro $k \in K$ symetrická šifra je **prostá** funkce (musí být invertibilní)

$$E_k : P \to C$$

K dešifrování používáme E_k^{-1}, značíme D_k.

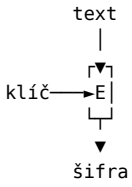
$\forall x \in P : x = D_k(E_k(x))$

Dělíme na **blokové** a **proudové**.

DEF Bloková šifra

Bloková šifra je $E : \{0,1\}^b \times \{0,1\}^k \to \{0,1\}^b$

Vždy vezme blok, s klíčem ho zamíchá a vydá stejně dlouhý blok.

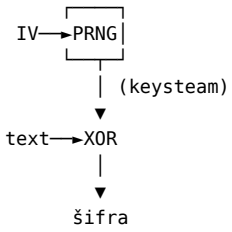


Bloková šifra je permutace na bloku:
Je prostá a množina vzorů je stejně velká jako množina obrazů. Musí být tedy na. Množiny jsou stejné ⇒ je to permutace.

DEF Proudová šifra

Mám nějaké IV, které následně posílám do náhodného generátoru.

Generátor mi vytváří dlouhý *keystream*, který následně XORuji s otevřeným textem.



Jedná se o OTP, ale dlouhý klíč je náhodně generovaný z IV.

Proudové šifry jsou inverzní samy k sobě. Zároveň komutují.

Většina šifer jsou sudé permutace, protože většina primitivních operací, které na číslech děláme, jsou sudé permutace. Skládáním pouze sudých perm. vytvoříme taktéž sudou perm. ⇒ měli bychom být schopní rozlišit šifru od náhodné permutace.

DEF Konfuze a difuze

Konfuze: Bity šifrového textu by měly být netriviálně závislé na co nejvíce bitech klíče.

Difuze: Každý bit otevřeného textu by měl ovlivnit co nejvíce bitů šifrového textu. Malá změna v ot. textu by měla mít možnost změnit co nejvíce šifrového textu.

DEF

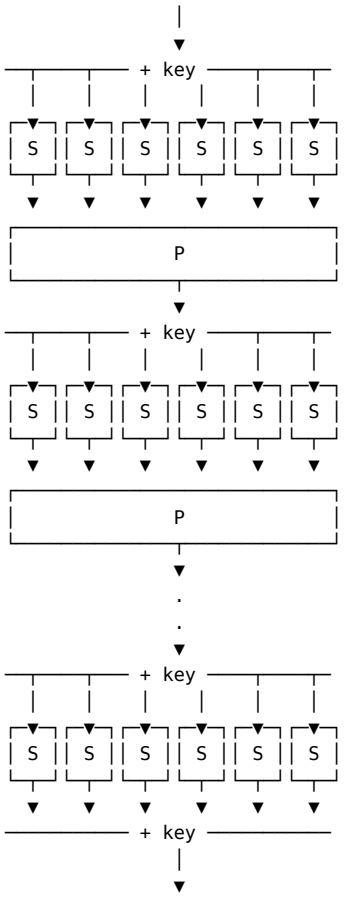
DEF Substituční-permutační sítě (SPN)

SPNky se tvoří z jednotilivých rund. Jedna runda vypadá následovně:
1. přidání rundovního klíče

2. substituce částí textu podle substitučních tabulek (s-box)
3. zpermutováním textu

Poslední runda je speciální: permutace je k ničemu, protože jí můžeme i po přidání klíče odstranit. Zároveň musíme jako poslední přidat klíč, jinak by i s-box byl k ničemu. Z toho vyjde, že poslední runda by měla být přidání klíče, s-box, přidání klíče.

Tabulky i permutace jsou známé (Kerckhoff), pouze rundovní klíče jsou tajné.



Permutace a přidání klíče je vždy invertibilní. Aby SPN byla celá dešifrovatelná, pak je potřeba udělat s-boxy takové, aby taktéž byly invertibilní – musí být bijekce.

Dešifrování SPNky je také SPNka. Protože operace permutace a přidání klíče komutují, tak po jejich prohození nám vznikne SPNka v opačném směru.

```
šifrování:    K -> S -> P -> K -> S -> P -> K -> S -> K
dešifrování:  K <- S <- P <- K <- S <- P <- K <- S <- K
dešif. SPNka: K <- S <- K <- P <- S <- K <- P <- S <- K
```

Občas ale nepotřebujeme, aby SPNky byly invertibilní, např. v f ve Feistelově schématu. Pak je ok nemít invertibilní s-boxy.

Přidání klíče a s-boxy zařizují **konfuzi**.

Permutace pak hlavně **difuzi**.

DEF Feistelovo schéma

Udělal pan Feistel při navrhování DESu.

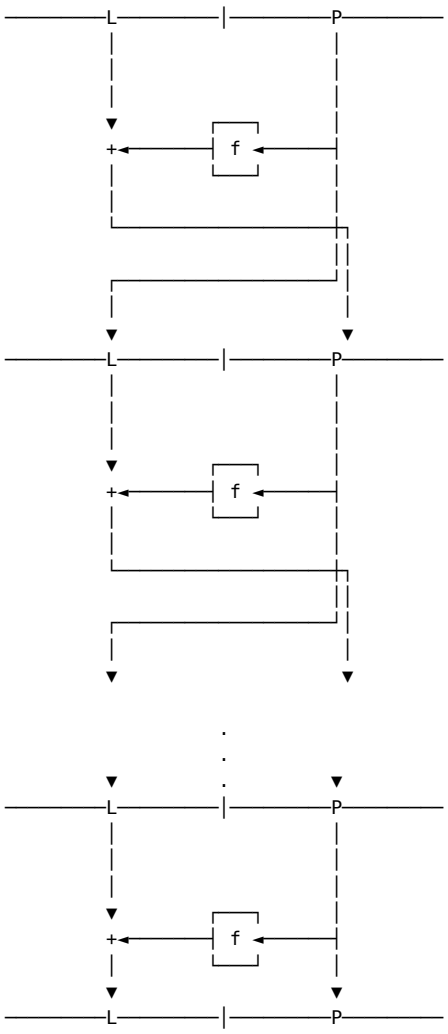
1. Otevřený text rozdělíme na levou a provou část.

2. K levé přiXORujeme f(pravou)
3. Strany prohodíme, začínáme novou rundu na kroku 2

Po poslední rundě už strany prohazovat nemusíme, je to k ničemu.

Feistelova funkce f nemusí být invertibilní.

Dešifrování F. schématu je opět F. schéma.



DES (Digital Encryption Standard)

Feistelovo schéma s 16 koly a 32-bitovými stranami. Na začátku a na konci je permutace navíc. Není tam kvůli kryptografii, ale kvůli efektivní hardwarové implementaci. V softwaru je to jenom otravný.

DESovská feistelova funkce:

- expanze 32-bit půlbloku na 48 bitů
- přidání rundovního klíče
- 8 s-boxů - berou 6 bitů, vrací 4
- permutace (ta je zpátky zase na jen 32 bitech $4 \cdot 8$ sboxů)

Key schedule – generuje rundovní klíče:

- vezme 56 bitů klíče, zpermutuje je
- rozdělí na dvě půlky
- pro každou rundu:

- každou půlku cyklicky zrotuje o 1 nebo 2 pozice
- vrátí výběr zpermutovaný výběr 48 bitů (tolik je potřeba do feistelovy funkce)
- a znovu

Kritika:

- $E_{\overline{k}}(\overline{x}) = \overline{E_k(x)}$

  Doplňky se ve feistelově funkci vyruší. Zůstává jenom negace levé strany, která se ale v další rundě zase s klíčem vyruší. Takhle se to propaguje dál.

  $$\begin{aligned} f\left(\overline{R_i}, \overline{k_i}\right) &= P\left(S\left(\overline{k_i} \oplus E\left(\overline{R_i}\right)\right)\right) \\ &= P\left(S\left(\overline{k_i} \oplus \overline{E(R_i)}\right)\right) \\ &= P(S(k_i \oplus E(R_i))) = f(R_i, k_i) \end{aligned}$$

- Krátké klíče. Jsou 64 bitové, ale 6 bitů je k ničemu (kvůli NSA). Pouze 56-bit klíče.
- Diferenciální kryptoanalýza 2^{47} chosen plaintextů. Lineární 2^{43} chosen plaintextů.

2-DES, 3-DES

Iterace DES, s jinými klíči. U 2-DES se může udělat meet-in-the-middle útok $\Rightarrow 2^{56} + 2^{56} = 2^{57} \Rightarrow$ security level je meh.

3-DES je 3x iterovaná DES, ale uprostřed se používá dešifrování DES, aby když to použijeme s 3x stejným klíčem, tak se to chovalo jako normální DES. Není tam žádná újma na bezpečnosti.

Zase meet-in-the-middle útok nám to zmenší na security level 2-DES, ale bez meet-in-the-middle. V podstatě děláme meet-in-the-third :) (sec level 112)

U 3-DES lze taky udělat variantu s dvěma klíči. První a poslední iterace mají stejný klíč. Uprostřed jiný. Klíč má pak celkově 112 bitů. (sec level ≈ 80)

AES (Advanced Encryption Standard)

AES je oficiální název pro vítěze soutěže od NIST pro symetrickou šifru. Původně se šifra jmenovala Rijndael.

Je to SPNka, která pracuje na bajtech. Ty jsou reprezentovány prvky v tělese $\mathrm{GF}(2^8)$.

1 runda vypadá takto:
1. ByteSub
2. ShiftRows
3. MixColumns (není v poslední rundě)
4. AddRoundKey

Před první rundou se pouští AddRoundKey.

Je několik variant podle délky klíče AES-128 (10 rund), AES-192 (12 rund), AES-256 (14 rund).
Počet rund se mění, protože existují útoky, který závisí jenom na počtu rund. Počet se tedy zvyšuje, aby to bylo podobně obtížné, jako bruteforcovat klíč.

Operace se provádějí na vnitřním stavu šifry, který je reprezentovaný maticí 4×4.

$$\begin{pmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_1 & B_5 & B_9 & B_{13} \\ B_2 & B_6 & B_{10} & B_{14} \\ B_3 & B_7 & B_{11} & B_{15} \end{pmatrix}$$

ByteSub

Každý byte se prožene stejným s-boxem.

S-box je inverze x v $\mathrm{GF}(2^8)$ a poté affiní transformace.

> **POZN:** Inverze je totiž silně nelineární. S-box je pak odolný vůči lineární a diferenciální kryptoanalýze. Affiní transformace zařizuje další alg. složitost.

ShiftRows

Cyklické posunutí i-tého řádku o i pozic doleva.

MixColumns

Na každý sloupec se aplikuje lineární transformace. Každý bajt v sloupci ovlivní každý další bajt v sloupci.

V poslední rundě se vynechává.

AddRoundKey

XOR s rundovním klíčem.

AES má velmi dobrou difuzi. ByteSub zpropaguje změnu na celý byte, ShiftRows jí propaguje mezi sloupci a MixColumns změnu hned zpropaguje na celý sloupec. Během dvou rund by měla být zpropagována po celém vnitřním stavu.

Rozvrh klíčů
Rozdělíme nejdříve klíč na 32-bit slova

$$w_0, w_1, w_2, w_3$$

4 pro 128-bit klíč.

Následně vytváříme n (v tomto případě $n = 4$) dalších slov následovně:

$w_4 = w_0 \oplus \mathrm{SubWord}(\mathrm{RotWord}(w_3)) \oplus \mathrm{Rcon}_1$

$w_5 = w_1 \oplus w_4$

$w_6 = w_2 \oplus w_5$

$w_7 = w_3 \oplus w_6$

Atd:

$w_8 = w_4 \oplus \mathrm{SubWord}(\mathrm{RotWord}(w_7)) \oplus \mathrm{Rcon}_2$

$w_9 = w_5 \oplus w_8$

…

(viz expanze klíče *v prezentaci zde*)

SubWord je použití s-boxů na bajty. RotWord je rotace o jeden byte nahoru/doleva. Rcon_i jsou konstanty.

Pro AES-256 se ještě dělají nějaké SubWordy uprostřed. I guess aby to bylo míň lineární.

Kritika
- jednoduchá algebraická struktura… útok řešením rovnic, ale zatím se neumí
- malý počet rund. Prý se dostáváme hodně blízko k tomu, aby to nestačilo.

- bloky moc malé → kolizní útoky po 2^{64} blocích. fix je měnit klíč po 2^{32} blocích.

4. Přednáška

> **POZN:** How to (mis)use a block cipher: padding, modes ECB, CBC, CTR, OFB. Padding oracle attacks on CTR and CBC. Information leaks in CBC and CTR modes. Ciphertext stealing in ECB mode (see here for CBC version). Stream cipher sketches: LFSR-based constructions, eSTREAM project, Trivium, ChaCha20.

Padding (výplň)

Zarovnání zprávy na násobek bloku. Musíme umět odstranit.

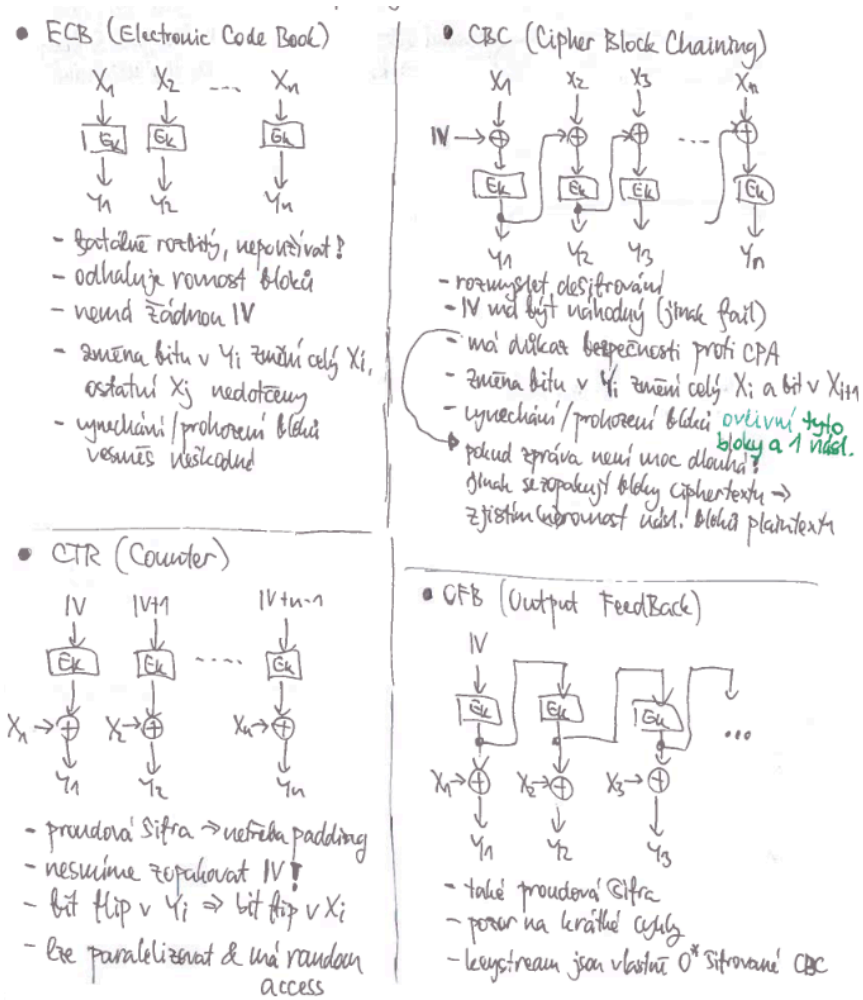
Např. přidání nul nefunguje – je ta nula padding, nebo součástí zprávy?

Hodně způsobů, tady nějaké:

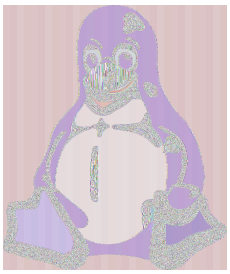
- na konec zprávy přilepit konstatní byte + kolik nulových bytu je potřeba na zarovnání.
- pokud potřebujeme n bytů na zarovnání, tak přidáme n bytů s hodnotou n. Toto je limitované na bloky velikosti 255 bytů. Pokud je zpráva už zarovnaná, přilepíme celý nový jeden blok. Padding tam musí být vždycky.
- na konec zprávy přilepit jeden bit = 1, pak samé nuly. Ekvivalent toho prvního u šifer, které pracují na bitech.

Módy blokových šifer

Způsob jak šifrovat blokovou šifrou delší zprávy.



ECB (Electronic Code Book)
Šifruji každý blok samotný.

> **ATTK** Prostě špatnej mód tbh
>
> Pokud se mi bloky otevřeného textu opakují, budou se opakovat i v šifrovém ⇒ leakuju strukturu.
>
> 
>
> **NEPOUŽÍVAT** jakože vůbec. Vlastně není moc důvod…

CBC (Cipher Block Chaining)
K bloku otevřeného textu přiXORuji předchozí blok šifrového.

> **ATTK** IV se nemění
>
> Zprávy, které začínají stejně, budou mít stejný začátek. Ups.
>
> **IV se musí měnit**

> **ATTK** IV je předvídatelný
>
> Dostal jsem zprávu (IV, Y_1, Y_2, …), mám přístup k šifrovadlu a vím, jaké bude další IV'.
>
> Mám teorii o tom, že Y_2 je zašifrované X.
>
> Pak zkusím, jestli $Y_2 = E(\text{IV}' \oplus Y_1 \oplus X)$. Efektivně vyměním IV' za Y_1, což bylo to, co se v původní zprávě přixorovalo k OT, když jsme šifrovali Y_2.
>
> **IV musí být nepředvídatelné**

CTR (Counter)
- tvoří proudovou šifru → není potřeba padding
- pokud zopakujeme IV, dostaneme úplně stejný keystream → ups

> **ATTK** Zopakované IV
>
> Pokud dostanu dva šifrové texty se stejným IV, pak vím, že keystream byl totožný u obou zpráv.
>
> ⇒ když je zXORuju, dostanu jejich rozdíl. GG.
>
> **IV musí být nonce**

OFB (Output Feedback)
- opět proudová šifra
- **je možné, že se keystream zacyklí**, (navíc:) střední hodnota periody je 2^{b-1} (b je velikost bloku), musíme tedy měnit klíč.

Stejně jako u CTR, IV musí být nonce.

Padding oracle

Pokud máme orákulum, který nám řekne, jestli je padding ok, tak můžeme byte po bytu hádat plaintext. Tohle se mi nechce rozepisovat, soráč.

Prosakování informací (z módů blok. šifer)

ECB je prostě done – $X_i = X_j \Leftrightarrow Y_i = Y_j$

CBC: $Y_i = Y_j$:

$$
\begin{aligned}
E_k(Y_{i-1} \oplus X_i) &= E_k\big(Y_{j-1} \oplus X_j\big) \\
Y_{i-1} \oplus X_i &= Y_{j-1} \oplus X_j \\
Y_{i-1} \oplus Y_{j-1} &= X_i \oplus X_j
\end{aligned}
$$

Z narozeninového paradoxu tohle nastane tak jednou za $\sim 2^{\frac{b}{2}}$ bloků. Vyzradíme potom b bitů (jednoznačne určíme jeden z bloků, pokud máme druhý).

CTR:

Zakladní myšelnka je, že žádné dva bloky keystreamu se nerovnají. Potom:

$$
\begin{aligned}
Y_i \oplus Y_j &= (C_i \oplus X_i) \oplus \big(C_j \oplus X_j\big) \\
&= X_i \oplus X_j \oplus C_i \oplus C_j \\
&\overset{C_i \oplus C_j \neq 0}{\Rightarrow} \quad Y_i \oplus Y_j \neq X_i \oplus X_j
\end{aligned}
$$

Kolik bitů ale leakujeme?

Jeden blok má $\log 2^b = b$ bitů entropie.

Když ale známe nerovnost, tak jeden z těch bloků je omezený. Právě jedna možnost totiž nerovnost poruší.

Možností je tedy $2^b - 1$, což znamená, že má pouze $\log\big(2^b - 1\big)$ bitů entropie.

Pro každou dvojici teda leakujeme

$b - \log\big(2^b - 1\big) = \log 2^b - \log\big(2^b - 1\big) = \log \frac{2^b}{2^b - 1} = \log\big(1 + \frac{1}{2^b - 1}\big) \approx \frac{1}{2^b - 1} \approx 2^{-b}$

bitů (hodně málo).

Ale pro každou dvojici!

Pokud máme m bloků, tak to je $\binom{m}{2} \cdot 2^{-b}$. To je $\approx m^2 \cdot 2^{-b}$.

A pokud $m \approx 2^{\frac{b}{2}}$, tak to je ≈ 1.

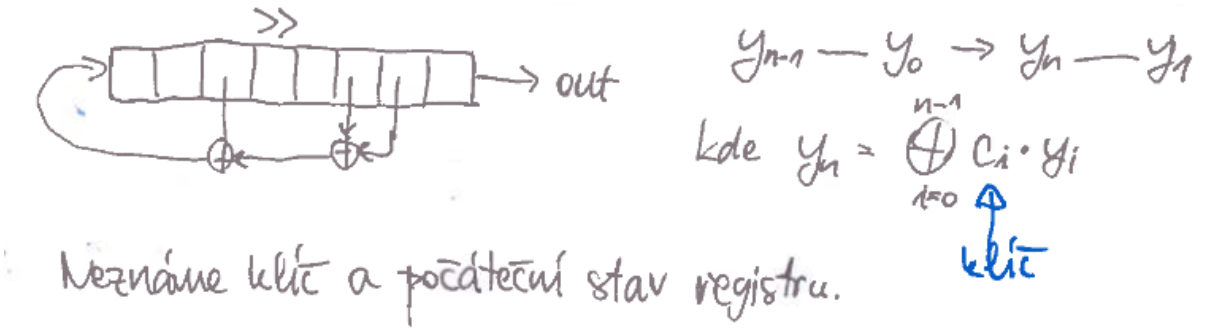
Je to celý jenom horní odhad, protože informace, kterou dostáváme z dvojic, nemusí být vždy nezávislá. Ale jednou za $2^{\frac{b}{2}}$ bloků leakujeme cca 1 bit infa.

Proudové šifry

Definice viz 3. přednáška.

Existuje eSTREAM projekt, který hledá nové proudové šifry.

LFSR (Linear-Feedback Shift Register)



LFSR je posuvný registr který při každém kroku spočítá nový bit pomocí lineární rekurentní rovnice a vmáčne ho na začátek. Bity, které z registru vypadávají, tvoří keystream.

> **POZN:** Na obrázku je trochu nešikovně namalovaný. Sice tam je 8-bitový LFSR, ale ve skutečnosti, vzhledem k tomu, že poslední bit nijak do počítání nového bitu nepřispívá, tak se chová jako 7-bitový.

Vlastnosti

- může se poměrně rychle zacyklit, ale jde vytvořit klíč, který zaručí periodu $2^n - 1$ (n délka LFSR)

Pokusy o opravu

Bylo jich dost, asi nejpodstatnější je ten, který využívala šifra v GSM (telefony) A5/1. Jednak mají regisry 3 a jejich výstup XORují, druhak se navzájem ovlivňují při "tikání". Občas se nějaký LFSR neposune. Každopádně je ta šifra pořád hodně moc rozbitá.

Pokud těch LFSR máme více a mixujeme jejich výstup dohromady, tak je jejich perioda potom NSN z jednotlivých period.

> **ATTK** Known-Plaintext útok na LFSR
>
> Máme LFSR o n bitech. Neznáme klíč, ale známe dvojici otevřeného a šifrového textu o $2n$ bitech.
>
> Chceme najít klíč, který se tvoří z n bitů. Máme tedy n neznámých.
>
> $\mathrm{OT} \oplus \check{\mathrm{S}}\mathrm{T} = \mathrm{keystream}$
>
> Víme, že prvních n bitů keystreamu vytvořilo $n + 1$ bit pomocí lin. rovnice.
>
> Takto to uděláme pro zbytek a máme n rovnic.
>
> Zacvičíme lingebru a máme klíč.
>
> Pokud by známý kus zprávy nebyl na začátku, tak registr pak zpětně odkorkujeme na začátek a máme tím pádem i počáteční stav (a taky zbytek zprávy).
>
> Ups.

Trivium

LFSR šifra s 288 bity interního stavu. Má tří LFSR. Musí se odkrokovat kolem 1.5k bitů jako inicializace.

Má security level 80, takže nic moc, ale na lightweight rychlou šifru fajne.

Chacha20

Předchůdce Salsa20. 20 je počet rund.

Stav je matice 4×4 32 bit čísel.

$$\begin{pmatrix} c_1 & c_2 & c_3 & c_4 \\ c_5 & c_6 & c_7 & c_8 \\ c_9 & c_{10} & c_{11} & c_{12} \\ c_{13} & c_{14} & c_{15} & c_{16} \end{pmatrix}$$

Na začátku:

$c_1, c_2, c_3, c_4 =$ "`expand 32-byte k`" v ASCII

$c_5, c_6, c_7, c_8 =$ klíč

$c_9, c_{10}, c_{11}, c_{12} =$ klíč pokračování

$c_{13}, c_{14} =$ počítadlo

$c_{15}, c_{16} =$ nonce

Skládá se z čtvrtrund (:D) (Quarder Round). V jedné rundě se vezmou čtyři prvky a nějak se zašmodrchají pomocí Addition Rotation Xor ($\Rightarrow$ jméno designu šifer ARX).

V lichých rundách se to dělá na sloupce, v sudých na diagonály.

Na konci se přičítá stav před čtvrtrundou. Čtvrtrunda je jinak lineární a dá se invertovat. Měli bychom pak stejný problém jako u LFSR, u kterého můžeme zpětně odkrokovat až ke klíči.

5. Přednáška

> **POZN:** Hash functions: requirements, Merkle-Damgård construction by iterating compression function, length extension attacks. Birthday attacks on hash functions: the tortoise, the hare, and the turtle. How to obtain a compression function: Davies-Meyer construction from a block cipher. Practical hash functions: MD5 (broken), SHA1 (broken), and the SHA2 family.

> **DEF** Hashovací funkce
>
> Funkce $h : \{0,1\}^* \to \{0,1\}^b$.
>
> **Požadavky**
>
> 1. Neumíme najít první vzor (neumíme invertovat):
>
>   pokud dostaneme $y := h(x)$, pak nedokážeme najít x.
>
> 2. Neumíme najít druhý vzor:
>
>   pokud dostaneme x a $h(x)$, pak nedokážeme najít $x' : h(x') = h(x)$.
>
> 3. Neumíme najít kolizi:
>
>   nedokážeme najít x a x', že $h(x) = h(x')$.
>
> Pokud nedokážeme dosáhnou ani 3. požadavku, pak je funkce *kolizivzdorná*.
>
> (logicky z 3. $\Rightarrow$ 2. $\Rightarrow$ 1.)

Závazky

Jedna z aplikací hashovacích funkcí je dělání závazků (commitment). Když někomu pošleme hash, tak se efektivně zavážeme k tomu z čeho jsme hash vypočítali. Nemůžeme mu teď poslat něco jiného a tvrdit, že to byla ta původní zpráva. Plyne to z 2. požadavku.

Házení mincí po telefonu
Alice a Bob si hází mincí po telefonu, ale navzájem si nejvěří. Například hrají o peníze.

Schéma vypadá následovně:

1. Alice si náhodně vybere $x \in \{0,1\}^{s+1}$. Kde s je security level.
2. Alice pošle Bobovi $h(x)$.
3. Bob pošle Alici náhodný bit b.
4. Alice pošle Bobovi x.
5. Oba si spočítají výsledek hodu mincí $x_0 \oplus b$.

Je hodnota $x_0 \oplus b$ opravdu 50/50? Ano, oba bity jsou náhodné a $0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0$.

Nejde to podvrhnout?

Alice Bobovi pošle $h(x)$, tím se efektivně zavazuje k tomu, že ve 3. kroku pošle x.

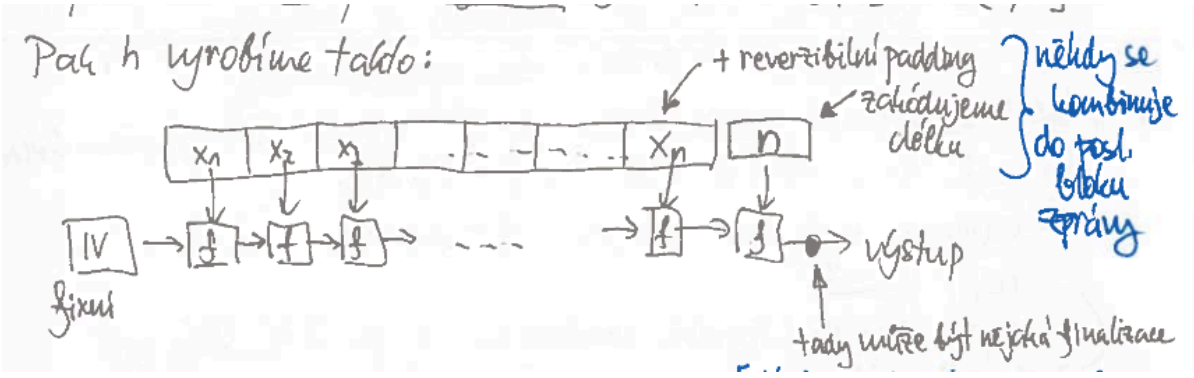
Bob z $h(x)$ nic nezjistí, protože nedokáže h invertovat. Dalších s bitů tam je k tomu, aby to nešlo spočítat hrubou silou. Nedokáže tedy nějak chytře vybrat b, aby ovlivnil hod mincí.

Bob tedy pošle svou část.

Alice teď zná výsledek mince, ale i kdyby měla prohrát, tak nemůže x změnit.

Merkelova-Damgårdova konstrukce
Konstrukce na hashovací funkce z jednodušší *kompresní funkce*.

> **DEF** Kompresní funkce
>
> $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$
>
> Z interního stavu a bloku vytvoří nový interní stav.

> **DEF** Merkelova-Damgårdova konstrukce
>
> 
>
>
> Začínáme s nějakým fixním IV jako vnitřním stavem. Pak ten stav vždy přes f aktualizujeme blokem zprávy. Poslední stav je výstup.
>
> n nakonec přidáváme, protože to jednak lehce ztěží nějaké útoky, druhak aby vycházel hezky důkaz kolizivzdornosti konstrukce.
>
> **POZN:** ten padding asi nemusí být reverzibilní…

> **DKZ** Kompresní funkce je kolizivzdorná ⇒ M-D konstrukce je kolizivzdorná

Pro spor předpokládejme $h(x_1 \cdots x_n) = h(x'_1 \cdots x'_m)$ (dvě různý zprávy, stejný hash)

Pokud $n \neq m$, nebo se nerovnají předchozí stavy, tak jsme našli kolizi v f.

Zprávy jsou tedy stejně dlouhé a kolize musela nastat někdy dřív.

V předchozím bloku? Pokud $x_i \neq x'_i$, nebo se nerovnají předchozí stavy $\Rightarrow$ kolize v f.

Takto se dostáváme až na začátek zprávy. Někdy v průběhu musí $x_i \neq x'_i$, protože jsou zprávy různé. Nekdy v průběhu tedy nastane kolize na f.

ATTK Length-extension / Prodloužení

Jestliže mám h hashovací funkci sestavenou M-D konstrukcí a dostanu $h(x)$.

Pak dokážu vytvořit $h(x \parallel l(x) \parallel y)$ bez znalosti x.

Výstup M-D je celý stav, který z f vylezl jako poslední. Nic nám ale nebrání počítat dál hash delší zprávy. Bohužel budeme mít uprostřed délku x, protože původní $h(x)$ jí přidalo jako poslední blok.

Toto je problém, pokud bychom se snažili něco "podepsat" tak, že uděláme $h(\text{klíč} \parallel \text{zpráva})$. Útočník pak dokáže vytvořit autentizovanou delší zprávu.

ATTK Hledání kolizí pro konkatenované hashe

Když máme $h(x) = h_1(x) \parallel h_2(x)$, kde h_1 je konstruovaná M-D,

tak v čase $\frac{b}{2} \cdot 2^{\frac{b}{2}}$ umíme vytvořit $2^{\frac{b}{2}}$ kolizí pro h_1.

Máme tedy $2^{\frac{b}{2}}$ zpráv, mezi kterými bude pár, který vytvoří kolizi pro h_2.

Časová složitost je $\frac{b}{2} \cdot 2^{\frac{b}{2}}$.

Paměťová b^2.

Obecné útoky na hashovací funkce

ATTK Nalezení kolize hrubou silou (Želva a zajíc)

Z narozeninového paradoxu víme, že kolizi najdeme po cca $2^{\frac{b}{2}}$ operacích (b velikost hashe).

Pamatovat si $2^{\frac{b}{2}}$ hashů ale není reálné. Můžeme využít trik.

Začneme s náhodným hashem. Další hash vytvoříme zahashováním předchozího: $x_i = h(x_{i-1})$

Vznikne nám takovýto graf, kde vrcholy jsou hashe. V grafu musí být cyklus, protože je konečné množství hashů a z každého musí vést hrana. Zároveň bude mít kolem $2^{\frac{b}{2}}$ vrcholů.

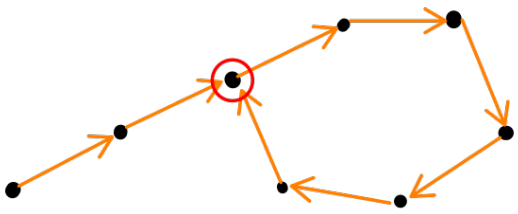


Figure 1: "Lízatkový" graf. Curtesy of https://kahann.cz/notes/kry/

Graf je tedy pořád velký. Musíme použít trik s želvou a zajícem.

Želva i zajíc se pohybují po grafu. Želva rychlostí 1, zajíc rychlostí 2. Oba začínají v rukojeti "lízátka".

Oba nakonec musí skončit v cyklu. A protože je zajíc rychlejší než želva, tak jí v cyklu někdy dožene. Řekněme, že to bylo za n kroků ve vrcholu $T_n = H_n$, kde T_i značí vrchol, ve kterém je želva v čase i a podobně pro zajíce H_i.

Nevíme ale pořád kde přesně kolize je.

Vypustíme z rukojeti tedy na scénu lenochoda L_i, který je stejně pomalý, jako želva. Po n krocích musí být na pozici L_n, ale $L_n = T_n$, protože je stejně rychlý, jako želva. Želva bude na pozici T_{2n}, ale to je pozice H_n, což je T_n.

Lenoochod se s želvou tedy potkají. Vzhledem k tomu, že začínali na různých vrcholech, pak se musely někdy jejich cesty spojit. Právě v tom bodě je kolize.

V nejhorším případě se želva zajícem potká v čase $2^{\frac{b}{2}}$ (velikost grafu). Lenochod se s želvou potká opět nejhůř za dalších $2^{\frac{b}{2}}$ kroků.

Celkově to je $2 \cdot 2^{\frac{b}{2}}$ operací. **Ale konstantní paměť**.

ATTK Hledání velkého množství kolizí u M-D konstrukce

U M-D konstrukce je možné najít 2^k kolizí v čase $k \cdot 2^{\frac{b}{2}}$.

Najdu si k následujících kolizí:

- $f(\mathrm{IV}, x_1) = f(\mathrm{IV}, x_1') =: y_1$
- $f(y_1, x_2) = f(y_1, x_2') =: y_2$
- ...

Můžeme libovolně kombinovat x_i a x_i'. Vytvoříme tak 2^k kolizí.

ATTK Hledání kolizí mezi dobrými a zlými zprávami

Nechám si od orákula udělat $2^{\frac{b}{2}}$ dobrých zpráv.

Udělám si $2^{\frac{b}{2}}$ zlých zpráv.

V průniku bude nějaká kolize.

Časová složitost $2^{\frac{b}{2}}$, paměť $b \cdot 2^{\frac{b}{2}}$.

Daviesova-Mayerova konstrukce

DEF Daviesova-Mayerova konstrukce kompresní funkce

Vyrábí se z blokové šifry:

$f(a, b) = E_a(b) \oplus b$

$\oplus\, b$ je důležité! Jinak když $y := E_a(b)$, pak vyberu a' a pak $b' = D_{a'}(y)$.

$f(a', b') = E_{a'}(b') = y = E_a(b) = f(a, b)$

VĚTA Kompresní funkce vytvořena D-M konstrukcí z ideální blokové šifry je kolizivzdorná

Kolizivzodrná v tomto kontextu znamená, že po $q < 2^{\frac{b}{2}}$ volání funkce je pravděpodobnost nalezení kolize $\leq \frac{q^2}{2^{\frac{b}{2}}}$. (Shoduje se s narozeninovým paradoxem.)

DKZ

Útočník má k dispozici šifru ze které je vytvořena kompresní funkce.

Útočník nikdy neopakuje volání šifry, když něco zašifruje, tak pak nedešifruje opačným směrem stejnou dvojici.

Šifra je ideální. Pro každý klíč k se chová jako náhodná permutace.

BÚNO může útočník mít přístup pouze k $E_k(x)$. Když něco zašifruje, má to stejný efekt, jako kdyby něco stejným klíčem rozšifroval.

Po každém volání šifry dostane $t = E_x(y) \Rightarrow f(x, y) = t \oplus y$. Tedy získá hodnotu pro pouze jednu dvojici (x, y), ale pokaždé jinou!

Vyplňuje si tedy takovou tabulku se souřadnicemi (x, y) a hodnotami $f(x, y)$. Chce najít kolizi v hodnotách tabulky.

Pokud už udělal i volání:

$$P[f(x, y) = f(x', y')] \leq \frac{1}{2^b - i} \overset{i<q<2^{\frac{b}{2}}<2^{b-1}}{\leq} \frac{1}{2^{b-1}}$$

Pravděpodobnost, že se jsou dvě hodnoty shodné by byla 2^{-b}, ale šifra není náhodná funkce, ale bijekce. Kdyby náhodou všech i volání bylo s jedním klíčem, tak je tato pravděpodobnost $\frac{1}{2^b - i}$. Proto tento horní odhad.

$$\begin{aligned} P[\exists \text{ kolize párů } (x, y)] &= \frac{1}{2^{b-1}} \cdot \text{počet párů} \\ &= \frac{1}{2^{b-1}} \cdot \binom{q}{2} \\ &\leq \frac{1}{2^{b-1}} \cdot \frac{q^2}{2} = \frac{q^2}{2^b} \end{aligned}$$

Disclaimer: poskládáno z internetu, protože od medvěda jsem to absolutně nepochopil. Myslim, že takhle to je trochu lépe vysvětleno. Např. nechápu co yappoval o tom šifrování a dešifrování. Podle mě stačilo říct, co jsem napsal k tomu BÚNO. Idk, vyhodil jsem timhle několik hodin. Kdyžtak mi někdo napište, pokud to mám blbě, actually by mě to zajímalo: as zavináč ministerstvointernetu.cz

Příklady hashů MD5, SHA-1, rodina SHA-2

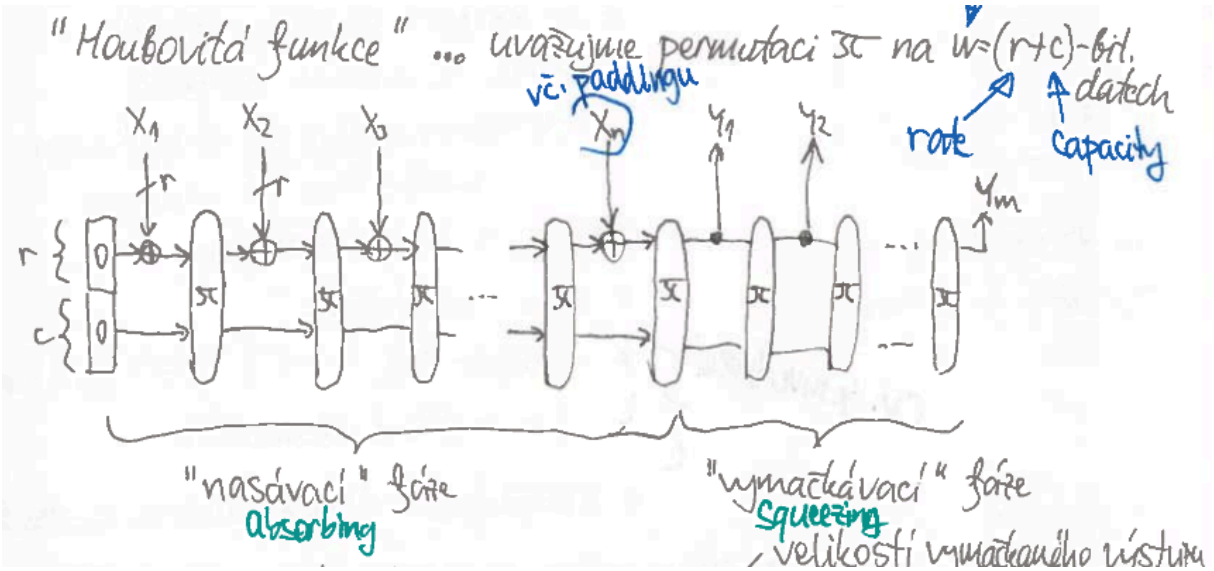
MD5: b=128, robitý

SHA-1: b=160, rozbitý

SHA-2: b=224 až 512

Houbovité konstrukce

Houbovitá konstrukce vypadá takto:



1. začínáme s interním stavem $\{0\}^{r+c}$
2. k r bitům přiXORujeme r-bitový blok vstupu
3. celý stav pošleme do π
4. opakujeme – "nasávací" fáze
5. ze stavu vytáheneme r bitů
6. celý stav pošleme do π
7. opakujeme – "vymačkávací" fáze

Nelze dělat length extension útok, protože neznáme nikdy celý stav.

> **ATTK** Interní kolize na kapacitních bitech houbovité konstrukce
>
> Do houbovité konstrukce posíláme neustále nulové bloky a hledáme kolizi na c kapacitních bitech (ke kterým se neXORují bloky).
>
> Narozeninový paradox: po $2^{\frac{c}{2}}$ krocích kolize. Máme tedy dva stavy se stejnými kapacitními bity:
>
> $(r_i, c_i), (r_j, c_j), c_i = c_j$
>
> Kolizi pouze na kapacitních bitech jsme si mohli dovolit, protože teď můžeme změnit r_2 na r_1.
>
> V dalším kroku:
>
> $(r_{i+1}, c_{i+1}) = \pi(r_i \oplus 0, c_i)$
>
> $(r_{j+1}, c_{j+1}) = \pi(r_j \oplus (r_j \oplus r_i), c_j) = \pi(r_i, c_j) = (r_{i+1}, c_{i+1})$
>
> Jedna zpráva má délku $i+1$ a druhá $j+1$, ale mají stejný hash.

Bezpečnost: je známé, že security level je $\min(\frac{r}{2}, \frac{c}{2})$, když π je náhodná permutace

SHA-3 standard

Houbovitá konstrukce s permutací Keccak, kde $w = 1600$ bitů.

Existují různé varianty (SHA3-224, SHA3-512) s různými r a c.

Také existují tzv. **XOF**, které nelimitují velikost "vymáčklého" výstupu. SHAKE-128, SHAKE-256.

6. přednáška

> **POZN:** Sponge hashes: analysis for random permutations, Keccak, SHA3-n, SHAKE-n. Parallel hashing: Merkle trees and Sakura. Symmetric signatures (MACs): construction from hash functions (KMAC and HMAC), different ways of combining a MAC with a cipher. CBC-MAC. Information-theoretic MAC from 2-independent linear functions or from polynomials. Practical polynomial MACs: GCM mode, Poly1305 (sketch).

Merkleovy stromy

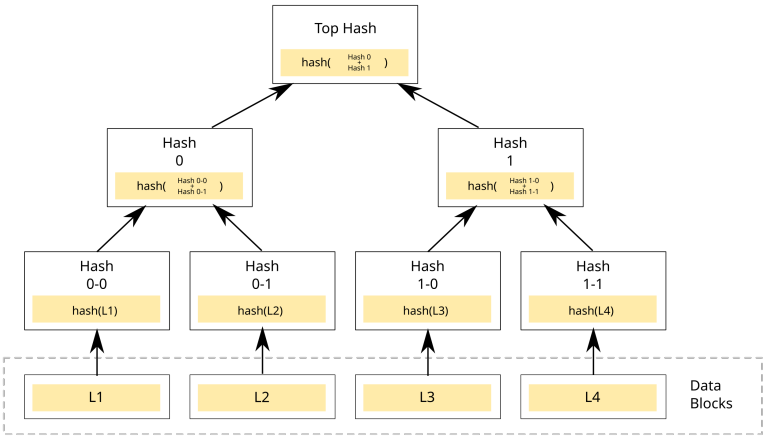


Figure 2: Z Wikipedie

Velké množství dat je těžké po každé změně přehashovávat. Můžeme tedy data rozdělit do bloků a vytvořit Merkleův strom. Listy drží hashe jednotlivých datových bloků, vnitřní vrcholy pak hash konkatenací hashů svých potomků.

Pokud dokážeme u stromu udržet logaritmickou hloubku, pak při změně bloku dokážeme přepočítat hash celé struktury v $\log n$ volání hashovací funkce.

Je nutné nějak označit kořen, jinak můžeme spočítat hash nějakého stromu obsahující původní strom jako podstrom, bez znalosti obsahu původního stromu.

Sakura kódování dokáže do hashe zakódovat strukturu celého stromu. Není pak možné vytvářet hashe bez znalosti celého stromu.

MAC (Message Authentication Code)

> **DEF** Message Authentication Code
>
> MAC(k, x) je funkce na vytvoření tagu (symetrického podpisu) x klíčem k.
>
> Pro libovolný klíč k se chová jako náhodná funkce.

Špatné MACy:

- MAC$(k, x) = h(k \parallel x)$. Pro Merkle-Damgard konstrukce rozbité kvůli length extension útoku. Pro ideální hash funkci ok a pro SHA-3 taky (KMAC).
- MAC$(x, k) = h(x \parallel k)$. U M-D opět blbé, protože pokud najdeme $h(x) = h(x')$, tak mají stejný podpis. Tuto vlastnost bychom radši neměli. Zároveň většina výpočtu nezávisí na klíči.

HMAC pro hashovací funkci h: $\text{HMAC}_h(k, x) = h(k \oplus \text{opad} \parallel h(k \oplus \text{ipad} \parallel x))$

opad a ipad jsou konstanty.

Když chci přidat tag ke zprávě, mám to udělat před, nebo po šifrování?

Doporučuje se po (encrypt-then-MAC), protože to zabrání jakémukoliv útoku na šifru skrz upravování šifrového textu (jako např. padding oracle útok).

CBC-MAC
Provádím normální CBC a MAC je poslední blok šifrového textu.

IV musí být fixní, protože $E_k(\text{IV} \oplus x_1) = E_k((\text{IV} \oplus d) \oplus (x_1 \oplus d))$. Pokud můžu měnit IV, tak můžu libovolně měnit první blok.

Šifrové bloky jiné, než posledn musí být utajené, jinak známe podpis pro kratší zprávu. Obecně to je špatná konstrukce, protože hrozně rychle podlehne na known-message útok.

Shannonovsky bezpečné MACy

> **POZN:** Další suprčupr skoro-useless bezpečné schéma.

Co chceme: Pokud útočník uvidí $(x, h(x))$, pak se nic nedozví o $h(x')$ jakéhokoliv $x' \neq x$.

Předpoklady:
- klíč bude použit pouze jednou
- Máme systém funkcí $H := \{h_k; k \in \mathcal{K}\}$ ($\mathcal{K}$ je množina klíčů) z X do Y, které jsou *2-nezávislé.*

  To znamená:

  $\forall x, x' \in X, x' \neq x, \forall y, y' \in Y : P_{h \in H}[h(x) = y \wedge h(x') = y'] = \frac{1}{|Y|^2}$

Důkaz bezpečnosti:

$$\begin{aligned} P_k[h_k(x') = y' \mid h_k(x) = y] &= \frac{P_{h \in H}[h(x) = y \wedge h(x') = y']}{P_{h \in H}[h(x) = y']} \\ &= \frac{\frac{1}{|Y|^2}}{P_{h \in H}[h(x) = y']} \\ &= \frac{\frac{1}{|Y|^2}}{\sum_{y'} P_{h \in H}[h(x) = y \wedge h(x') = y']} \\ &= \frac{\frac{1}{|Y|^2}}{\sum_{y'} \frac{1}{|Y|^2}} = \frac{\frac{1}{|Y|^2}}{\frac{|Y|}{|Y|^2}} = \frac{1}{|Y|} \end{aligned}$$

Pravděpodobnost je stejná, jako když bychom tipovali podpis.

Nevýhody:

- klíč musí být 2x tak dlouhý, jako to, co hashujeme (nedokázáno)
- klíč je jednorázový (to byl předpoklad)

7. přednáška

> **POZN:** Generating random bits: pseudo-random generators from symmetric ciphers, physical randomness (LavaRand, circular oscillators, …), combining both – Fortuna, RDRAND, /dev/random. Putting all symmetric cryptography together: a secure channel. Basics of algorithmic number theory: Complexity of arithmetics. Euclid's algorithm and Bézout's coefficients. Multiplicative group modulo N. Little Fermat's and Euler's theorem. Lagrange's theorem on subgroups (without proof).

Generátory náhodných čísel

> **DEF** Generátor náhodných čísel
>
> Zjevné z názvu. Máme pro něj ale požadavek, aby útočník, který zná celou historii výstupu, nedokázal predikovat následující bit s větší pravděpodobností, než 50%.

Typy generátorů:

1. Fyzická náhoda: např. z tepelného šumu, radioaktivity, časování I/O, nebo tzv. ring oscillator.
2. Pseudo náhoda: tvořená např výstupem z AES v CTR módu.

V praxi používáme oboje.

Aby se pseudonáhodný generátor dostal z **komprimovaného stavu**, je potřeba větší množství fyzické náhody.

Fortuna
Generátor náhodných čísel. Skládá se ze dvou částí:

1. Generátor – vytváří výstup
2. Akumulátor – sbírá fyzickou náhodu

Generátor: Šifruje počítadlo pomocí AES-256. Po nějakém množství bloků se mění klíč (asi z jeho vlastního outputu?). **Počítadlo se ale nechává**, nemůže potom dojít k zacyklení. Občas si bere náhodu z akumulátoru (*re-seeding*).

Akumulátor: Má n poolů, do kterých rovnoměrně posílá nasbíranou fyzickou náhodu. Každý pool se ale používá jinak často. Pool P_i se používá pouze každý 2^i re-seed. "Vysoké" pooly se tak používají exponenciálně méně. I když je fyzické náhody málo, tak se ve vyšších poolech nahromadí a při nějakém re-seedu se využije najednou. Generátor se tak dostane z kompromitovaného stavu.

Bezpečný kanál (Secure channel)

Později bude ještě jednou. Toto je bezpečný kanál pro obousměrnou komunikaci *s předem nasdíleným tajemstvím.*

- bloková šifra v CTR módu. *soukromí*
- MAC po šifrování. *integrita/autenticita*
- Pořadové čísla zpráv. *integrita dialogu/pořadí zpráv*
- PRNG pro nonce
- KDF pro vytvoření klíčů pro šifru a MAC z jednoho sdíleného tajemství, nejlépe separátní klíče pro oba směry. (**nepoužívat jeden klíč na víc věcí**)

Komplexita operací

Pro b-bitové číslo:

- sčítání $O(b)$
- násobení $O(b^2)$
- dělení $O(b^2)$
- mocnění $O(b^3)$, když použijeme square-multiply
- euklid (i rozšířený) $O(b^2)$

Trocha teorie čísel

> **VĚTA** Lagrange
>
> $H \subseteq G$ (podgrupa: na operaci uzavřená a splňuje všechny předpoklady grupy)

$\Rightarrow |H| \setminus |G|$ (velikost H dělí velikost G)

Důkaz není, just trust me bro.

VĚTA Eulerova

$\forall N > 0 : \forall x \in \mathbb{Z}_N^* : x^{\varphi(N)} \equiv 1 \bmod N$

$\varphi(N)$ je počet prvků v $\mathbb{Z}_N^*$, tedy který mají inverzní prvek.

DKZ

Pro x máme posloupnost

$$x^0, x^1, x^2, ..., x^k, ...$$

Ta se musí začít opakovat, protože grupa je konečná.

Překněme, že se začne opakovat na x^k.

První prvek, který se zopakoval, musel být x^0, protože jinak $x^i \equiv x^k \Rightarrow x^{-1} \cdot x^i \equiv x^{-1} \cdot x^k \Rightarrow x^{i-1} \equiv x^{k-1}$. Opakujeme, dokud se nedostaneme na x^0. Tam už to posunout nejde.

$H = \{x^0, ..., x^{k-1}\}$

$|H| \setminus |\mathbb{Z}_N^*|$ (z Lagrange. Ještě dokázat, že H je grupa, takže ověřit všechny podmínky.)

$k = |H| \setminus |\mathbb{Z}_N^*| \Rightarrow \varphi(N) = k \cdot c$

$x^{kc} \equiv \left(x^k\right)^c \equiv 1^c \equiv 1$

Všechno je moduleno N obviously.

VĚTA Malá fermatova

Euler, ale N je prvočíslo. Takže $\mathbb{Z}_N$ je potom těleso. $\varphi(N) = N - 1$ pro prvočísla.

Jako tbh tohle asi na zkoušce dokazovat nemusíme (snad).

8. přednáška

POZN: More number theory: Chinese remainder theorem and its constructive version. Computing Euler's function. Factoring vs. primality testing. Fermat and Rabin-Miller tests (R-M without proof), Carmichael numbers, note on Agarwal's deterministic test. Generating large primes. Multiplicative group is cyclic, discrete logarithms, how to find a generator. Discrete square roots, Euler's criterion.

Diskrétní logaritmy

$\mathbb{Z}_p^*$ (p prvočíslo) je cyklická grupa – má generátor g, že $\{g^0, g^1, ..., g^k\} = \mathbb{Z}_p^*$

Isomorfismus – jedním směrem močnění, druhým diskrétní logaritmus.

Disktrétní logaritmus je těžký spočítat.

g není generátor $\Rightarrow$ generuje nějakou grupu $H \Rightarrow |H| \setminus |\mathbb{Z}_p^*| = p - 1 \Rightarrow g^{\frac{p-1}{k}} \equiv 1$ pro nějaké k (podíl je $|H|$). Dokonce stačí nějaké prvočíselné k, přesněji nějaké z rozkladu $p - 1$.

Generátory hledáme náhodně.

Diskrétní odmocniny

Kromě 0 má polovina čísel 2 odmocniny, druhá žádnou. $y = x^2 = (-x)^2$

Číslům, které mají odmocninu, říkáme *kvadratické zbytky.*

VĚTA Eulerovo kritérium

x je kvadratický zbytek v $\mathbb{Z}_p^* \Leftrightarrow x^{\frac{p-1}{2}} \equiv 1 \bmod p$

DKZ

$\left(g^{2k}\right)^{\frac{p-1}{2}} \equiv g^{k(p-1)} \equiv 1^k \equiv 1$, když je mocnina lichá, tak smolík.

Počítání odmocnin pro $p = 4t + 3$: $\left(x^{\frac{p+1}{4}}\right)^2 \equiv x^{\frac{p+1}{2}} \equiv x^{\frac{p-1}{2}} \cdot x \equiv 1 \cdot x \equiv x$

Pro $p = 4t + 1$ je rand algoritmus. Ale tohle na zkoušce def nebude...

Rabin-Miller test pro prvočíselnost

Euklidův svědek x, že N není prvočíslo: $\gcd(N, x) \neq 1$

Fermatův svědek x, že N není prvočíslo: $x^{N-1} = 1 \bmod N$

Carmichaelova čísla jsou takové, které nemají Fermatovy svědky, ale nejsou prvočísla. Eukidových svědků je málo, takže to je problém.

Test:

1. Náhodné $x \in \{1, ..., N-1\}$
2. Pokud $\gcd(x, N) \neq 1$, euklidův svědek. SLOŽENÉ
3. Počítáme: $x^{N-1} \bmod N, x^{\frac{N-1}{2}} \bmod N, \ldots$ dokud není exponent lichý. Pokud někdy $= 1$, tak SLOŽENÉ
4. PRVOČÍSLO

Podle Rabina $P\left[\text{PRVOČÍSLO} \mid x \text{ složené}\right] \leq \frac{1}{4}$

Takže to opakujeme a dostáváme velmí malou pravděpodobnost, že to je fake prvočíslo.

Prvočísla generujeme tak, že si je tipneme a zkoušíme. Je jich dost $\frac{1}{\ln N}$

Čínská věta o zbytcích (CRT)

Podívej se https://kahann.cz/notes/kry/, hledej CRT. Je to dobrý.