BPC-AKR

Aplikovaná kryptografie

Otázky ke státnicím

Bakalářský obor Informační bezpečnost, FEKT VUT

https://github.com/VUT-FEKT-IBE/BPC-IBE-SZZ

Text: kámen u cesty, jedla
Korektura: czechbol

28. května 2023

Obsah

1 Prvočísla – popište způsob generování a uveďte příklad pravděpodobnostního testu a skutečného testu

Jsou extrémně důležitá pro téměř všechny kryptosystémy: asymetrické šifrování, digitální podpisy, ustanovení klíčů i vyšší systémy (TLS, PKI).

Prvočísel je nekonečně mnoho, počet $\pi(n)$ menší než n lze pro velké hodnoty aproximovat jako $\pi(n) = \frac{n}{\ln(n)}$. Současné kryptosystémy využívají prvočísla o velikosti 2048–4096b (600–1200 cifer).

1.1 Typy prvočísel

Mersennova prvočísla: $M_p = 2^n - 1$ (např. 31).
Sophie-Germain prvočísla: p pokud $2p + 1$ je také prvočíslo (např. 11).
Bezpečná prvočísla: $B_p = 2p + 1$ (např. 23). Jsou generována SG prvočísly.

1.2 Generování

Velmi neefektivním způsobem je Eratosthenovo síto. Sestaví se seznam všech čísel od 2 do n. První se ze seznamu vyřadí a prohlásí se za prvočíslo, všechny jeho násobky se ze seznamu vyřadí. Takto se pokračuje dokud nedosáhneme $\sqrt{n}$, kdy zbývají pouze prvočísla.

Prakticky se buď volí bezpečná prvočísla, nebo se náhodně vygeneruje číslo požadované velikosti a otestuje se, zda se o prvočíslo jedná.

1.3 Pravděpodobnostní testy

Fermatův test je založen na modulárním mocnění. Zvolíme libovolné $a : 1 \leq a \leq n - 1$, spočteme $a^{n-1} \stackrel{?}{\equiv} 1 \pmod{n}$. Pokud kongruence platí, opakujeme pro jiné a, pokud neplatí, n prvočíslo není. Čím více pokusů, tím větší šance že se o prvočíslo jedná. Nedokáže odlišit prvočísla a Carmichaelova čísla[1].

Miller-Rabinův test je spolehlivější než Fermatův; složená čísla projdou iterací s maximálně 25% pravděpodobností.

1.4 Skutečné testy

Lucas-Lehmerův test funguje pouze pro Mersenova prvočísla ($M_p = 2^s - 1$).
$u_0 = 4$, $u_{n+1} = (u_n^2 - 2)$, $\ldots$, $u_{s-2} \stackrel{?}{=} 0 \pmod{M_p}$.

[1] Carmichaelovo číslo splňuje kongurenci $b^{n-1} \equiv 1 \pmod{n}$ pro všechna b nesoudělná s n.

0	$n = 2^s r + 1$ kde r je liché
1	$a : 1 \leq a \leq n - 1$
2	$\begin{cases} a^r \equiv 1 \pmod{n} & \text{nová iterace} \\ a^r \not\equiv 1 \pmod{n} & \text{bod 3} \end{cases}$
3	Platí $a^{2^i r} \equiv n - 1 \pmod{n}$ pro všechna $i : 0 \leq i \leq s - 1$ $\begin{cases} \text{ano :} & \text{nová iterace} \\ \text{ne :} & \text{je složené} \end{cases}$

2 Teorie čísel, algebraické struktury – popište účel a způsob výpočtu Eulerovy funkce. Popište požadavky na grupu a způsob generování grupy pro algoritmy založené na problému DL.

Teorie čísel je část matematiky zkoumající vlastnosti (zejména) celých čísel. Prvočíslo je $p \geq 2$, které nemá jiné dělitele než triviální. Každé celé číslo ≥ 2 je buď prvočíslo nebo se dá zapsat jako součin prvočísel. Pokud číslo není prvočíslo, nazýváme ho číslem složeným.

V kryptografii se nejčastěji setkáme s dělitelností (zbytek a modulo), největším společným dělitelem (GCD, výpočet jako násobek společných prvočinitelů s jejich nejmenší mocninou), nejmenším společným násobkem (LCM, výpočet jako násobek všech prvočinitelů s největší mocninou). Jestliže jsou výsledek GCD roven 1, jedná se číslo nesoudělné a bude se nejspíše jednat o prvočíslo.

Eulerova funkce $\phi(n)$ je často používaná v kryptografii. Udává počet celých čísel, která jsou nesoudělná s n.

Tabulka 1: Výpočet Eulerovy funkce ϕ

prvočísla	$\phi(n) = n - 1$
mocniny prvočísel	$\phi(p^k) = (p-1)p^{k-1}$
složená čísla	$\phi(a \cdot b) = \phi(a) \cdot \phi(b)$
obecná čísla	$\phi(n) = n \cdot (1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_r})$
	kde $p_1, p_2, \ldots, p_r$ jsou prvočísla ze kterých se n skládá, nebo
	$\phi(n) = (p_1 - 1)p_1^{k_1 - 1} \cdot (p_2 - 1)p_2^{k_2 - 1} \cdots (p_r - 1) \cdot p_r^{k_r - 1}$
	kde platí $n = p_1^{k_1} \cdot p_2^{k_2} \cdots p_r^{k_r}$.

Algebraické struktury představují množiny prvků s operacemi a různými vlastnostmi. V kryptografii jsou nejpoužívanější grupy, tělesa a konečná tělesa.

Grupou (G, $\circ$) nazýváme množinu G spolu s binární operací $\circ$ (operace se dvěma prvky a, b kterým přiřazuje prvek c stejné grupy) na množině G, která splňuje:

- uzavřenost (pro všechny prvky a, b v G je i $a \circ b$ prvkem G)
- asociativita $(a \circ b) \circ c = a \circ (b \circ c)$
- existence neutrálního prvku (existuje prvek e v G takový že pro všechna a v G platí $a \circ e = e \circ a = a$)
- existence inverzního prvku (pro každý prvek a existuje prvek b takový že $a \circ b = b \circ a = e$, tj. že jejich složení je rovno neutrálnímu prvku. Prvku b se říká inverzní.

Grupa (G, $\circ$) je **cyklická**, jestliže existuje prvek g v G takový, že pro každé b v G existuje celé číslo i takové, že $b = g^i$. Takový prvek se nazývá generátor grupy.

Řád prvku a je definován jako nejmenší kladné celé číslo t takové že $a^t = 1$ pokud takové t existuje.

Řád generátoru grupy se rovná řádu grupy. Počet generátorů lze u prvočíselných multiplikativních grup zjistit jako $\phi(\phi(n))$.

Příklady praktických grup pro kryptografii:

- Množina celých čísel $\mathbb{Z}$ spolu s operací sčítání (+) tvoří abelovskou grupu.
- Množina $\mathbb{Z}_n$ s operací sčítání modulo n tvoří grupu řádu n.
- Množina $\mathbb{Z}_n$ s operací násobení modulo n ($\mathbb{Z}_n^*$) netvoří grupu, protože inverzní prvek neexistuje ke všem prvkům ze $\mathbb{Z}_n$.
- Množina $\mathbb{Z}_p^*$ s operací násobení modulo prvočíslo p tvoří grupu, protože inverzní prvek existuje ke všem prvkům ze $\mathbb{Z}_p$.

Příklad pro $\mathbb{Z}_{17}^*$:

- Grupa obsahuje prvky 1–16 celkem $\phi(17) = 16$.
- Neutrální prvek je $e = 1$.
- Každý prvek a má inverzní prvek a^{-1}, protože pro každé a platí GCD(a,17) = 1.
- Generátorem grupy jsou čísla $3, 5, 6, 7, 10, 11, 12, 14$ a celkem je jich $\phi(16) = 8$

Nalezení generátoru:

- Zvolení náhodného prvku g z $\mathbb{Z}_n^*$.
- Výpočet $\phi(n)$.
- Nalezení všech prvočíselných dělitelů $\phi(n)$ označených jako p_i.
- Pro každého prvočíselného dělitele se vypočte $g^{\phi(n)/p_i} \equiv 1 \mod n$.
- Pokud předchozí rovnice neplatí pro žádné p_i jedná se generátor.

2.1 Volba parametrů grupy pro algoritmy založené na problému DL

Předem je zvolená bitová délka prvočísla, je vygenerováno prvočíslo p o této bitové délce. Poté je potřeba zvolit řád prvku q. Víme, že $q|\phi(p)$. Některé algoritmy (např. DSA) navíc vyžadují, ať je q také prvočíselná hodnota. Poté je hledán prvek g o řádu q. Následně je vygenerován soukromý klíč x, $x < q$ a je zveřejněn veřejný klíč y, $y \equiv g^x \mod p$.

3 Modulární aritmetika – popište algoritmus Square and Multiply a Čínskou větu o zbytcích.

Modulární aritmetika pracuje s celými čísly v daném intervalu. Pro čísla platí:
– **reflexivita**: $a \equiv a \pmod{n}$
– **symetrie**: $a \equiv b \wedge b \equiv a \pmod{n}$
– **tranzitivita**: $a \equiv b \wedge b \equiv c \Rightarrow a \equiv c \pmod{n}$

Malá Fermatova věta urychluje modulární mocnění:
$a^p \equiv a \pmod{p}$ (také „$a^p - a$ je dělitelné p“).

Eulerova-Fermatova věta umožňuje získat inverzní prvek:
$a^{-1} = a^{\phi(n)-1} \bmod n$. Obecně platí $a^{\phi(n)} = 1 \bmod n$

Rozšířený Euklidův algoritmus umožňuje získat inverzní prvek (zde $5^{-1} \bmod 12$):

$12 = n$		$\boxed{\overset{?}{\pm}\, 5} = 2 \cdot 2 + 1 \pmod{12}$
$5 = x$	$2 = \lfloor 12/5 \rfloor$	$2 = 2 \cdot 1 + 0$
$2 = 12 - (5 \cdot 2)$	$2 = \lfloor 5/2 \rfloor$	1
$1 = 5 - (2 \cdot 2)$	$2 = \lfloor 2/1 \rfloor$	0
$0 = 2 - (1 \cdot 2)$		

Ukázka výpočtu inverzního prvku použitím Rozšířeného Euklidova algoritmu.

Počítání probíhá nejprve v prvních dvou sloupcích, poté zespoda nahoru ve třetím. Po získání výsledku je nutné zkontrolovat znaménko (tj. je možné, že bude nutné výsledku přidat mínus a přičíst n).

3.1 Square & Multiply

Urychlení modulárního mocnění příkladů typu „ $a^k \bmod p$“. Při řešení příkladu „$5^{11} \bmod 17$“ lze exponent přepsat do binární podoby: $11_{10} = 1011_2$. Jednotlivé bity se vypíší do sloupečku tabulky odspoda nahoru (na prvním řádku je LSB). Postupuje se zeshora dolů, a pokud platí $k_i = 1$, vykoná se krok b.

k_i	A	$b = 1$
1	$5 = a$	$5 = 1 \cdot 5$
1	$8 = 5^2 \bmod 17$	$6 = 5 \cdot 8 \bmod 17$
0	$13 = 8^2 \bmod 17$	
1	$16 = 13^2 \bmod 17$	$\boxed{11} = 6 \cdot 16 \bmod 17$

Ukázka výpočtu pomocí Square & Multiply.

3.2 Čínská věta o zbytcích

Používá se pro řešení soustav rovnic v modulární aritmetice. $m_1, \ldots, m_k$ jsou nesoudělná čísla. $a_1, \ldots, a_k \in \mathbb{Z}$. Systém

$$\begin{aligned} x &\equiv a_1 \,(\mathrm{mod}\, m_1) \\ x &\equiv a_2 \,(\mathrm{mod}\, m_2) \\ &\ldots \\ x &\equiv a_k \,(\mathrm{mod}\, m_k) \end{aligned}$$

má výsledné modulo $M = m_1 \cdot m_2 \cdots m_k$. Řešení lze vypočítat vzorcem

$$x = \sum_{i=1}^{k} a_i N_i L_i \bmod M$$

kde $M = m_1 \cdot m_2 \cdots m_k$, $N_i = M/m_i$, $L_i = N_i^{-1} \bmod m_i$.

$$\begin{aligned} x &\equiv 3 \pmod 6 (= a_1 \pmod{m_1}) \\ x &\equiv 2 \pmod 7 (= a_2 \pmod{m_2}) \\ M &= m_1 \cdot m_2 = 6 \cdot 7 = 42 \\ N_1 &= M/m_1 = 42/6 = 7 \\ N_2 &= M/m_2 = 42/7 = 6 \\ L_1 &= N_1^{-1} \bmod m_1 = 7^{-1} \bmod 6 = 1 \\ L_2 &= N_2^{-1} \bmod m_2 = 6^{-1} \bmod 7 = 6 \\ x &= (a_1 N_1 L_1 + a_2 N_2 L_2) \bmod M = 3 \cdot 7 \cdot 1 + 2 \cdot 6 \cdot 6 \bmod 42 = \boxed{9} \end{aligned}$$

Ukázka výpočtu čínské věty o zbytcích.

4 Symetrická kryptografie – proudové šifry, synchronní a asynchronní proudové šifry.

Výhodou symetrické kryptografie je velká rychlost a malé šíření chyb, nevýhodou nízká úroveň difuze a náchylnost k úmyslným falzifikacím. Symetrické šifry jsou vhodné pro šifrování velkých objemů dat. Klíč pro šifrování a dešifrování je stejný a je nutné ho držet v tajnosti (bývá často uložen v hardware tokenech: čipová karta, SIM, TPM).

V proudových šifrách je symbol otevřeného textu ihned převáděn na symbol šifrového textu, bez znalosti kompletní zprávy. Základem je *keystream generator*, který z předsdíleného klíče generuje data pro XOR operaci.

Proudové šifry bývají rychlejší než blokové a mívají nižší hardwarové nároky, ale jsou náchylné k určitým typům útoků. Pro žádné dvě šifrované zprávy nesmí být použit stejný *seed*.

Aby byla šifra bezpečná, musí operovat nad velkým modulem a z výstupu musí být nemožné získat originální klíč nebo vnitřní stav. Nemělo by být možné odlišit náhodný šum od výsledného proudu šifrovaných dat, a to i pokud útočník zná *plaintext* či *ciphertext* nebo ho může volit.

4.1 Synchronní proudové šifry

Proud pseudonáhodných čísel je generován nezávisle na zprávě a je s ní slučován pomocí operace XOR. Poškození části zprávy má za následek ztrátu synchronizace a zpráva se stává nečitelnou – proto se do *ciphertextu* můžou přidávat značky či odsazení, aby šlo přečíst zbývající data. *Bit flip* ale ovlivní pouze jeden znak šifrované zprávy, a změna jednoho bitu šifrovaného textu se projeví změnou bitu textu dešifrovaného, aktivní útočník tak může změnit význam.

4.2 Asynchronní proudové šifry

K vygenerování proudu se využívá n předchozích znaků zašifrovaného textu, tyto šifry se také nazývají *samosynchronizační*. Lépe lze detekovat ztrátu nebo přidání bitu, chyba na jednom bitu ovlivní n znaků dešifrovaného textu. Na tomto principu fungují například blokové šifry v CFB módu.

4.3 Příklady

RC4 (a kvůli „obcházení“ patentu také ARC4) generuje *keystream* pomocí dvou ukazatelů i, j a permutační tabulky S o 256 bajtech. RC4 je náchylná na *bit flipping* útoky a při špatném použití z *ciphertextu* můžou unikat informace o klíči.

Salsa20 a příbuzná **ChaCha20** jsou postaveny na pseudonáhodné funkci ARX (*add-rotate-XOR*); klíč má velikost 256b, nonce a čítač 64b, a mapují se na bloky o velikosti 512b. Díky této konstrukci je možné začít číst z jakékoliv pozice *ciphertextu* v konstatním čase. Využívá se 20 rund a k současné době je odolná vůči všem typům útoků. ChaCha je aktualizací Salsa20, využívá větší klíče a nonce a obsahuje rychlejší difuzi změn. ChaCha20 je využitá s křivkou Poly1305 v TLS a OpenSSH; linuxový kernel 4.8+ ji využívá ke generování dat v `/dev/urandom`. Je rychlejší než AES (na CPU, které nenabízí AES akceleraci, tedy například ARM).

5 Blokové šifry – Product Ciphers, konstrukce, Feistelova síť, DES, AES, základní módy blokových šifer.

Na rozdíl od proudových šifer operují na blocích dat o dané velikosti pomocí symetrického klíče. Algoritmy pro šifrování i dešifrování mají na vstupu blok o velikosti n a klíč velikosti k, na výstupu je n-bitový blok *ciphertextu*. Formálně lze šifrování zapsat jako $E_K(P) := E(K,P) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ a platí $\forall K : D_K(E_K(P)) = P$.

Produktová šifra kombinuje více transformací vstupu za účelem ztížení kryptoanalýzy. Využívá transformace jako substituce (S-box), permutace (P-box) a modulární artitmetiku. Výraz **SPN** (*Substitution-permutation network*) se využívá k popsání série S- a P-boxů v blokových šifrách. S-box zajišťuje konfuzi (vazbu *ciphertextu* na více částí klíče), P-box difuzi (změna jednoho bitu *plaintextu* změní polovinu *ciphertextu*, tzv. lavinový efekt).

Feistelova síť je konstrukt iterativního míchání bloků:

$$L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

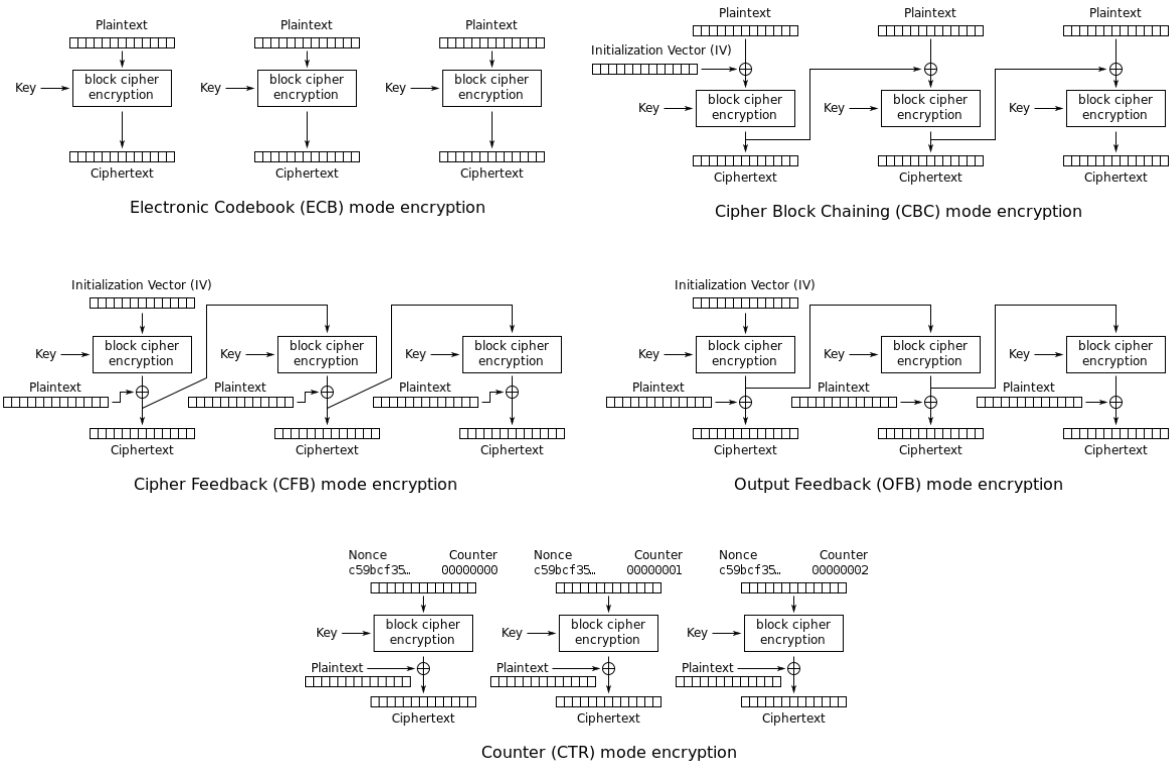
kde k_i je podklíč použitý v i-té rundě (tj. rundovní klíč) a f je libovolná funkce. Altenativou je Lai-Massey schéma, které se konstruktem velmi Feistelově síti podobá, ale rundovní funkce f nemusí být invertovatelná.

Blokové šifry velmi často využívají algoritmus ARX (*modular addition*, *byte rotation*, XOR), konfuzi a difuzi (DES, AES) nebo modulární násobení (IDEA).

5.1 Módy operací

Šifrovaná data jsou rozdělena na bloky, které se šifrují postupně. Nejprimitivnější kódování, ECB, bloky šifruje nezávisle – stejný *plaintext* tak má při stejném klíči stejný *ciphertext* a v zašifrované zprávě se tak mohou objevit vzorce vedoucí k dešifrování zprávy útočníkem.

ECB	$c_i = f(p_i, k)$
CBC	$c_i = f(c_{i-1} \oplus p_i, k)$
CFB	$c_i = f(c_{i-1}, k) \oplus p_i$
OFB	$c_i = f(f(c_{i-1}, k), k) \oplus p_i$
CTR	$c_i = f(\text{nonce}, \text{CTR}_i) \oplus p_i$
MAC	obdoba CBC, šifrování a autentizace
GCM	základem je inkrementace CTR_0, XORování se vstupem



Schémata některých módů operací. Převzato z Block cipher mode of operation.

5.2 DES

Jedna z prvních globálně používaných blokových šifer. Využívá 56b klíče a pracuje s 64b bloky v šestnácti rundách. Využívá Feistelovo schéma (pro „míchání“ dvou 32b polovin bloku). Funkce f se skládá z expanze (32b půlblok na 48b), XOR s rundovním klíčem (ten je získán pomocí *Key scheduleru* různými bitovými posuny), S-boxu a P-boxu.

Není dostatečně bezpečný (brute-force útoky, diferenciální a lineární kryptoanalýza). Poprvé zlomen 1997, v roce 2016 zlomený během patnácti dnů na GTX 1080 Ti.

5.2.1 3DES

Také Triple DES, TDES. Využívá DES třikrát za sebou za účelem zvýšení jeho bezpečnosti. 3DES využívá tři 56b klíče a *ciphertext* je získán výpočtem $c = E_{K3}(D_{K2}(E_{K1}(m)))$[2]. 3DES je náchylný na *Meet-in-the-Middle* útok a některé *chosen-plaintext*/*known-plaintext* útoky, OpenSSL ho neobsahuje od roku 2016 a NISTem byl prohlášen za nedostatečně bezpečný v roce 2017.

[2]Šifrování prvním klíčem, dešifrování druhým a šifrování třetím. 3DES totiž nabízí možnost použít i dva klíče, kdy $K3 = K1$.

5.3 Rijndael

Ke konci 90. let bylo třeba vyměnit nedostatečně silné DES/3DES, Rijndael[3] je vítězný kandidát pro šifru AES, je rychlý na hardware i sofware. AES využívá klíče o délce 128b/192b/256b a 128b bloky.

Rundovní operace (na bloku 4x4B): substituce (S-box): šestnáct bijektivních operací, v software řešeno překladovou tabulkou, *row shift* (bajtová permutace): rotace i-tého řádku o $i-1$ pozic doleva, *column mix* (lineární transformace): vynásobení matic, *key addition*: XOR s rundovním klíčem. V první rundě se provádí pouze přidání klíče, v poslední pouze substituce, *row shift* a přidání klíče.

I když existují útoky na oslabené verze AES (nižší počet rund a menší velikost klíčů), jde o bezpečný protokol který je využitý v TLS, OpenSSH a většině ostatních současných protokolů.

[3] Rijndael, holandská výslovnost ['rɛindaːl]

6 Asymetrické algoritmy – RSA, Diffie-Hellman, ECDH systém a jejich využití pro digitální podpis.

Problém diskrétního logaritmu[4]:

$$c = m^n \bmod p$$

je pro velká čísla neřešitelné.

6.1 RSA

Jde o relativně pomalý algoritmus, proto se s ním nešifrují přenášená data samotná, ale pouze klíče pro symetrickou kryptografii.

Vygenerování privátních prvočísel	p, q
Vypočítání privátního čísla	$\phi(n) = (p-1)(q-1)$
Vypočítání veřejného čísla	$n = p \cdot q$
Vypočítání veřejného klíče	$k_{\text{pub}} \in [1, n], \text{GCD}(k_{\text{pub}}, \phi(n)) = 1$
Vypočítání privátního klíče	$k_{\text{priv}} = k_{\text{pub}}^{-1} \bmod \phi(n)$

Parametry p, q by měly být podobně velké, i když s trochu rozdílnou velikostí (pro ztížení útoků). Jejich společné faktory by měly být co nejmenší, pokud to prvočísla nejsou. Místo funkce ϕ se v praxi používá Carmichaelovu funkci λ, protože ϕ může generovat čísla větší než je třeba.

šifrování	$c = m^{k_{\text{pub}}} \bmod n$
dešifrování	$m = c^{k_{\text{priv}}} \bmod n$
podepisování	$\text{Sig}_m = m^{k_{\text{priv}}} \bmod n$
ověření	$\text{Sig}_m^{k_{\text{priv}}} \stackrel{?}{=} m \bmod n$

6.2 Diffie-Hellman výměna

Veřejnými prvky jsou prvočíslo p (s velikostí nad 2048b), prvočíslo q (větší než 224b) a generátor $z \in \mathbb{Z}_p^*$ velikosti q. Jde o systém náchylný na MitM, klíče nejsou nijak autentizované.

Alice		Bob
$1 \leq a \leq p-2$		$1 \leq b \leq p-2$
$A = g^a \bmod p$	$\stackrel{A}{\rightarrow}\ \stackrel{B}{\leftarrow}$	$B = g^b \bmod p$
$k = B^a \bmod p$		$k = A^b \bmod p$

Znázornění DH výměny.

[4] *Discrete Logarithm Problem*, DLP

6.3 Diffie–Hellman výměna nad eliptickými křivkami

Vlastnosti ECDH jsou totožné s DH, DLP je pouze nahrazen jeho verzí nad křivkami (ECDLP). Násobení na eliptických křivkách je aplikováno jako opakované přičítání bodu. Asymetrické kryptosystémy nad eliptickými křivkami nabízí stejnou bezpečnost při mnohem kratší délce klíče (3072b DH $\approx$ 160b ECDH).

Alice		Bob
a		b
$A = aP \bmod p$	$\overset{A}{\rightarrow}\ \overset{B}{\leftarrow}$	$B = bP \bmod p$
$k = aB \bmod p$		$k = bA \bmod p$

Znázornění ECDH výměny.

7 Hašovací funkce – vlastnosti, princip, použití, kolize, odolnost proti kolizím, příklady.

Hashovací funkce slouží k vyjádření reprezentace dat pomocí krátného řetězce. Jejími požadavky jsou **jednosměrnost** (z hashe nelze získat původní data), **bezkoliznost** (vytvořit dva soubory se stejným otiskem je velmi obtížné až nemožné) a **rychlost**.

Bezkoliznost prvního řádu (*preimage resistance*) zabraňuje nalezení vzoru, bezkoliznost řádu druhého (*second preimage resistance*) zabraňuje ke zprávě m_1 nalézt m_2 takovou, aby platilo $h(m_1) = h(m_2)$.

7.1 Konstrukce

Jedním ze způsobů je rozdělení na n-bitové bloky, které jsou zpracovány XOR. Také je možné bloky řetězit: $h_i = E(m_i, h_{i-1})$, použít kompresní funkci ($h_i = f(m_i, h_{i-1})$). Před vstupem do iteračního procesu se zarovnává poslední blok a přidává se za ním blok s délkou zprávy (tzv. Damgard-Merklovo zesílení).

7.2 Využití

Zkrácení zprávy na její otisk (pro digitální podpisy), **autentizace**, zrychlení vyhledávání v databázích, **hesla**, PRNG, ...

7.3 Kolize

Na běžné současné hashovací funkce neexistují způsoby, jakými by šla původní data získat. Proto je nutné použít útok hrubou silou a zkoušet jeden vstup za druhým.

Pro kolizi prvního řádu (nalezení vzoru) je složitost $O(2^n)$, kde n je délka původní zprávy. Pro kolizi druhého řádu (nalezení kolizního souboru) je složitost $O(2^{n/2})$, což vyplývá z tzv. narozeninového paradoxu[5].

7.4 Příklady

128b **LM Hash** je stará funkce využívaná v OS Windows od osmdesátých let do verze Vista/Server 2008, v nejnovějších systémech je možné ji zapnout. Dva (*null-padded*) bloky po sedmi bajtech jsou zašifrovány dvěma DES klíči a spojeny zpět k sobě.

[5]Stačí 23 náhodně vybraných lidí k tomu, aby se s pravděpodobností 50 % našla dvojice, která má narozeniny ve stejný den, u skupiny třiceti lidí je pravděpodobnost 70 %. Pravděpodobnost, že nikdo shodný datum narozenin mít nebude, je $\bar{p}(n) = \frac{365!}{365^n(365-n)!}$. Obrácené tvrzení $p(n) = 1 - \bar{p}(n)$ překračuje hranici 0.5 pro $n = 23$. Tomuto útoku se říká narozeninový útok a narozeninovou hranici lze spočítat právě jako $2^{n/2}$.

128b **MD5** aplikuje 64 rund (XOR, bit shift, posouvání subbloků, ...). Poprvé plně prolomena v roce 2004, od roku 2010 oficiálně prohlášena za nedostatečně bezpečnou. Stále je možné ji používat jako kontrolu integrity.
224b/256b/384b/512b **SHA-2** se sada hashovacích funkcí využívajících XOR, *bit shift* a modulární sčítání. Existují útoky na oslabené verze (méně rund), všechny funkce z této rodiny jsou považovány za bezpečné.
224b/256b/384b/512b **SHA-3** využívá algoritmus Keccak („princip houby“) a je součást SHA rodiny od roku 2015.

8 PKI – certifikát X.509 struktura, certifikační autorita (základní části), časová razítka, autorita časových razítek.

8.1 Public key infrastructure

PKI neboli *Public Key Infrastructure* je souhrn technických a organizačních prostředků spojených s vydáváním, správou, používáním a odvoláváním platnosti kryptografických klíčů a certifikátů. Zabraňuje použití falešné identity. Veřejný klíč je platný pouze v případě potvrzení důvěryhodnou stranou (certifikační autoritou).

8.2 X.509

Struktura:

- verze (`0`: X.509 verze 1, `1`: X.509 verze 2, `2`: X.509 verze 3),
- sériové číslo,
- algoritmus použitý pro podpis,
- vydavatel (identifikace certifikační autority dle x.500),
- platnost od–do,
- jméno a adresa vlastníka,
- veřejný klíč,
- rozšíření certifikátu,
- digitální podpis certifikátu.

Certifikáty X.509 se ukládají ve formátu PEM (*Privacy Enhanced Mail*; hlavička a ASCII formát v base64), DER/BER (*Distinguished Encoding Rules*, *Basic Encoding Rules*; binární formát, při jehož konverzi a přidání hlavičky vznikne PEM) a ASN.1 (popis dat dobře čitelný lidmi).

8.3 Certifikační autorita CA

Základními částmi jsou třída 1, 2, 3.

V **třídě 1** ručí CA pouze za jednoznačnost certifikátu. Registrační autoritu takové CA je možné zjednodušit na program neboli žadatel vyplní formulář serveru a protokolem HTTPS jej odešle (stačí autentizace serveru a klient je tedy anonymní).

Třída 2 zahrnuje vše z třídy 1 a navíc kontroluje totožnost uživatele: k zástupci registrační autority osobně dochází žadatelé. RA ověří totožnost uživatele a odešle žádost CA. CA třídy 2 uchovává svůj soukromý klíč v bezpečném hardwaru.

Třída 3 zahrnuje vše co třída 2, ale certifikáty jsou určené pro konkrétní aplikaci a pro nic jiného se nepoužívají. Př.: certifikát lze použít k přihlášení do nějaké aplikace ale email s ním nejde podepsat. Zde se soukromý klíč také uchovává v bezpečném hardwaru.

Další části CA jsou registrační autority (RA), online RA, IVR nebo telefonní záznamník, adresářové služby, DVCS server, timestamp server a zúčtovací systém.

Registrační autorita má na starosti žádosti o certifikáty. Ověří totožnost uživatele, verifikuje žádost o certifikát a předá ho certifikační autoritě. CA ověří údaje z žádosti uživatele a údaje doplněné RA a vydá nebo nevydá příslušný certifikát.

Online RA přijímá žádosti online cestou a lze zažádat o:

- obnovení certifikátu v době platnosti starého,
- vydání nového certifikátu na základě jednorázového hesla pro vydání certifikátu,
- další certifikáty, žádost se podepisuje platným certifikátem,
- soukromý šifrovací klíč, který je archivován na CA,
- odvolání certifikátu,
- CRL (*Certificate revocation list*, seznam odvolaných certifikátů) nebo jiný certifikát vydaný CA.

IVR nebo telefonní záznamník slouží k odvolání certifikátu telefonem (jinou cestou). Pro odvolání musí proběhnout autentizace jednorázovým heslem.

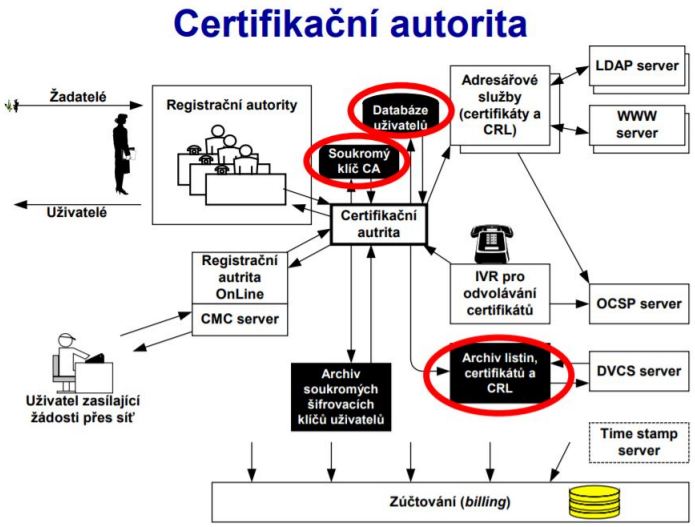
Adresářové služby nabízejí informace o uživatelích, které si uživatelé o sobě přejí publikovat (vydané certifikáty).

DVCS server poskytuje informace o platnosti certifikátu, listin a dále může poskytovat časová razítka protokolem DVCSP.

Timestamp server je speciální server poskytující pouze časová razítka a nemusí mít přístup do archívu CA.

Zúčtovací systém má za cíl vystavit fakturu za poskytnuté služby uživateli.

Důležitá aktiva CA, která se musí maximálně chránit jsou soukromý klíč CA, databázi uživatelů CA (CA nesmí vydat dvěma uživatelům stejný certifikát a obsahuje osobní údaje), archiv soukromých šifrovacích klíčů uživatelů (pokud CA poskytuje), archiv listin uložených na CA a archiv vydaných certifikátů a CRL.



8.4 Časová razítka (time stamp)

Časová razítka jsou elektronický ekvivalent časového určení a místa vlastního podpisu na listině. Řeší možné problémy vzniklé problémy odvoláním certifikátu. Zajišťuje především důkaz o existenci dokumentu v daném čase. Využívá se k poskytovaní elektronických notářských služeb a zajištění dlouhodobé archivace elektronicky podepsaných dokumentů. Časové razítko svazuje hash dokumentu s časem.

Struktura obsahuje:

- Jméno vydavatele (jméno autority pro vydávání časových razítek (TSA)).
- Jedinečné sériové číslo razítka.
- Hash dokumentu a čas.

8.5 Autorita časových razítek (TSA)

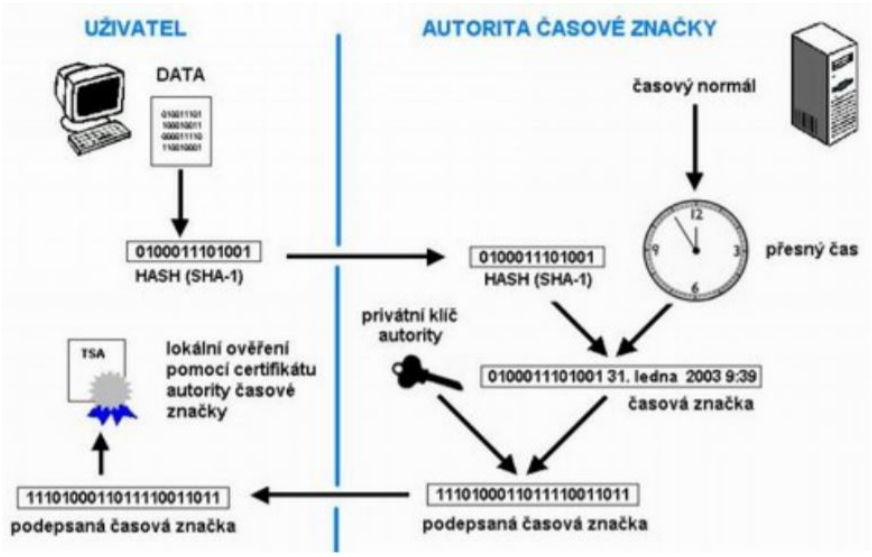
TSA musí podepisovat každé časové razítko privátnín klíčem vyhrazeným pouze k tomuto účelu. TSA musí být napojena bezpečným způsobem a v pravidelných časových intervalech synchronizována proti nejlépe třem na sobě nezávislým zdrojům. TSA využívají mezinárodní standard UTC (*Coordinated Universal Time*).

Požadavkům na zdroj času lze dosáhnout využitím PKI, zajišťující důvěrnost, integritu, nepopiratelnost a autenticitu při časové synchronizaci. Požadavky na zdroj času jsou:

- Musí pocházet z oficiálního důvěryhodného zdroje.
- Čas nesmí být možné cestou změnit.
- Vždy musí být možné zpětně dohledat zdroj času (celou hierarchii časových serverů)

8.6 Vydání časového razítka

Uživatel zažádá pomocí klientské aplikace. Klient vytvoří a odešle žádost o časové razítko ve standardizovaném formátu. Žádost je datová struktura obsahující hash dokumentu. TSA v případě kladné odpovědi odesílá odpověď obsahující časové razítko.



9 Generování náhodných čísel – kryptografické generátory, požadavky, použití, princip, testování generátorů.

Hlavními požadavky na RNG jsou **rovnoměrné rozložení** (hodnoty jsou generovány se stejnou pravděpodobností), **nezávislost hodnot** (neexistuje korelace mezi nimi), **nepredikovatelnost** (na základě znalosti čísla nebo řady čísel nelze předpovědět následující, tzv. *next-bit test*) a často také **rychlé generování** nebo odolnost vůči kompromitaci (při zjištění vnitřního stavu nelze zpětně rekonstruovat dosavadní vygenerovanou posloupnost, tzv. *state compromise*). *Next-bit* ochrana a zábrana kompromitace jsou požadavky kryptograficky bezpečných generátorů.

K vytvoření dostatečného počtu (pseudo)náhodných čísel je nutná dostatečná entropie (míra náhodnosti/nepředvídatelnosti). Když útočník následující hodnotu zná, entropie je nulová. Lze ji vyjádřit jako $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$ kde X je generovaná hodnota a $p_1, \ldots, p_n$ jsou pravděpodobnosti hodnot $X_1, \ldots, X_n$ které je generátor schopen vygenerovat.

9.1 Typy generátorů

Fyzikální generátory (TRNG)

Teoreticky jsou nedeterministické, nejsou ale známy přesné parametry, kterými by se daly popsat a modelovat. Zaručují maximální bezpečnost a neopakovatelnost. Nevýhodou je pomalé generování či problematická technická realizace. Příklady: radioaktivní rozpad, tepelný šum, kvantové generátory.

Algoritmické generátory (PRNG)

Posloupnost náhodná není, pokud útočníkovi jsou známy některé parametry generátoru. Výhodami je rychlost, nízká odchylka od poměru 1:1 nebo snadná realizace. Nevýhodami jsou bezpečnost či periodicita. Příklady: čas, teplota HW komponent, šum na *low-level* sběrnicích (USB, pohyb myši), pohyb HDD.

Smíšené generátory

TRNG z fyzikálního generátoru se spojí s PRNG výstupem pomocí XOR.

9.2 Příklady realizace PRNG

Bloková šifra v režimu čítače: náhodně se zvolí klíč a počáteční hodnota i; zvoleným klíčem se šifrují hodnoty $i, i+1, \ldots$ – perioda u n-bitové šifry je 2^n.
Hashovací funkce aplikovaná na čítač: hashuje se $i, i+1, \ldots$.
Proudové šifry: viz otázku 4.

9.3 Konkrétní příklady

Blum-Blum-Shub PRNG má formu $x_{n+1} = x_n^2 \bmod M$ (kde $M = pq$ je násobek dvou velkých prvočísel), výstup je získán pomocí bitové parity výsledku, nebo z jednoho či více nejméně významných bitů. Byl navrhnut v roce 1986. *Seed* by mělo být číslo nesoudělné s M, s vyloučením $\{0, 1\}$.

Linuxový PRNG sbírá události (myš, klávesnice, HDD, síť, *system interrupts*, systémový čas). Zdroj `/dev/random` je blokující (při nedostatku entropie se čeká), na rozdíl od `/dev/urandom` (*unblocked random*), kdy je vráceno „méně náhodné číslo“.

EGD (Entropy Gathering Daemon) využívají některé linuxové programy (OpenSSL, GPG, Apache HTTP) pokud není dostatečné množství náhodných bitů v `/dev/random`. Entropii sbírá ve stavu CPU, IO nebo sítě.

LCG – Známe z CPT, $X_{b+1} = (aX_n + c) \mod m$, ne moc dobrý PRNG

9.4 Testování

K ověření vlastností PRNG se využívají statistické testy. Odhalí nekvalitní PRNG, ale neprokážou kvalitní PRNG. Testy vrací p-hodnotu, která vyjadřuje sílu důkazů proti nulové hypotéze („Testovaná posloupnost je náhodná.“). Pokud p-hodnota překročí určitou mez, nulovou hypotézu považujeme za neplatnou.

Frekvenční test: Obsahuje testovaná posloupnost bitů přibližně stejný počet nul a jedniček? **Runs test**: Je počet a délka řetězců po sobě jdoucích stejných bitů na úrovni náhodné posloupnosti? **Test hodností matic**, **spektrální test**.

10 Bezpečnostní architektura RM OSI – služby bezpečnosti, mechanizmy bezpečnosti, útoky na bezpečnost, příklady implementace bezpečnostních mechanizmů v jednotlivých vrstvách.

10.1 Architektura bezpečnosti v RM OSI

K zabezpečení se používá doporučení ITU-T X.800, ISO 7498-2 ISO/OSI Security Architecture. Obsahuje mechanismy bezpečnosti (security mechanism), útoky na bezpečnost (security attacks) a službu bezpečnosti (security services), kde jsou definované postupy pro zabezpečení informačních systémů.

10.2 Implementace bezpečnostních funkcí ve vrstvách RM OSI

Pro implementaci jsou nejvhodnější vrstvy 7. (aplikační protokoly), 4. (transport dat) a 3. (směrovaní). Bezpečnostní mechanismy jsou zabudovány do aplikačních programů a operačních systémů (7. a 4. vrstva) a propojovacích zařízení (3. vrstva), ale existují i způsoby zabezpečení, které využívají další vrstvy.

10.3 Služby bezpečnosti

Služba bezpečnosti je realizovaná protokolem příslušné vrstvy. Existuje 5 kategorií služeb, kterými jsou autentizace (authentication), řízení přístupu (access control), zabezpečení důvěrnosti dat (data confidentiality), zabezpečení integrity dat (data integrity) a ochrana proti odmítnutí původní zprávy (non-repudiation).

Autentizace je proces ověřovaní identity uživatele (entity). Existuje autentizace uživatelů (peer entity authentication), které nezabraňují útoky zopakováním zpráv. Dále existuje autentizace zdroje dat (data origin authentication), která provádí autentizaci všech dat a zabraňuje útokům zopakováním zpráv.

Řízení přístupu: kontrola přístupu (možnost povolit nebo odepřít použití určitého zdroje určitému subjektu, řízení přístupu k materiálním, logickým, nebo digitálním zdrojům; **neplést si s autorizací**). Umožňuje přístup do systému k službám a dalším. Chrání před neautorizovaným přístupem, kde se nejčastěji nachází v aplikaci nebo OS.

Zabezpečení důvěrnosti dat je ochrana obsahu dat, ochrana toku dat při přenosu proti analýze (zjištění odesílatele, adresáta, ...).

Obsahuje služby pro:

– Důvěrnost přenosu zprávy.
– Důvěrnost spojení (ochrana důvěrnosti v rámci navázaného spojení).
– Důvěrnost toku dat (chrání informace na zálkadě atributů toku dat).
– Selektivní důvěrnosti (ochrana pouze určených částí informace).

Zabezpečení integrity dat se zabývá zabezpečením proti neautorizované modifikaci (autorizace je přiřazení oprávnění pro práci v systému, specifikují se činnosti). Spadají sem služby integrity přenosu zpráv (ochrana integrity všech přenášených zpráv), služba integrity spojení (ochrana přenosů v rámci určitého navázaného spojení) a služby selektivní integrity spojení a selektivní integrity zpráv. Integritu rozdělujeme na slabou a silnou.

Slabá integrita slouží pro objevování útoků (modifikace zprávy šumem, náhodná změna pořadí paketů, náhodná duplicita . . .) pomocí aplikací kontrolních součtů, CRC, pořadová čísla paketů atd.

Silná integrita zabezpečuje proti úmyslným, aktivním útokům (subjektivní útoky) jako jsou podvržení zprávy nebo úmyslné pozměnění zprávy. Celkově silnou integritu tvoří prostředky slabé integrity a kryptografické prostředky

Ochrana proti odmítnutí původu zprávy zajišťuje důkaz o původnosti dat, prokazuje původ (příjemce, odesílatel) a prokazuje doručení (odesílaní, přijetí).

Celkově by měla být zajištěna autentizace (vím s kým komunikuji) a nepopiratelnost (vím s kým komunikuji a lze to dokázat).

10.4 Mechanizmy bezpečnosti

Mechanizmy bezpečnosti jsou šifrování, digitální podpis, řízení přístupu, integrita dat, výměna autentizační informace, padding (výplň), řízení směrovaní a ověření třetím subjektem.

10.5 Útoky na bezpečnost – model hrozeb

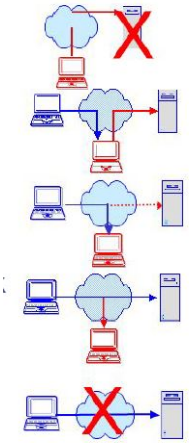
Destruction (útok na dostupnost) zničení dat či síťových zdrojů.

Corruption (útok na integritu) neautorizovaná modifikace aktiv nebo dat.

Removal (útok na dostupnost) krádež, odebrání či ztráta informací nebo jiných zdrojů.

Disclosure (útok na důvěrnost) neautorizovaný přístup k aktivům nebo datům.

Interruption (útok na dostupnost) přerušení služeb, spojení začně být nepoužitelné.



10.6 Příklady implementace

Na síťové vrstvě se o bezpečnost stará hlavně IPsec. Zajišťuje důvěrnost a integritu přenášených dat. O integritu se stará autentizační hlavička IP datagramu. Důvěrnost se zajišťuje pomocí mechanismu zapouzdření.

Na transportní vrstvě je bezpečnost zajištěna pomocí TLS (SSL bylo prohlášeno za nedostatečné v roce 2015) protokolu pro protokoly na aplikační vrstvě. Zajišťuje autentizaci mezi serverem a klientem, šifruje se spojení. Tímto zajišťuje autentizaci a důvěrnost.

Na aplikační vrstvě několik protokolů jako HTTPS, SSH a další – oba tyto zmíněné protokoly zajišťují důvěrnost, autentizaci a integritu.