Foundations of theoretical cryptography

lect. 7

* Digital Signatures Elektronický podpis *

- autentizace dat: Jak zajistíme, že zpráva přišla opravdu od Alice? Nebylo se zprávou manipulováno?
- klasicky – hlas, písmo, podpis
- Encryption neposkytuje autentizaci! Utajování (privacy) a autentizace jsou rozdílné problémy
- Ciphertext lze změnit tak, že plaintext je také pozměněn (malleability)
- řešení: ke zprávě přidáme tag / digitální podpis

1) Private key := message authentication codes (MACs)
   - Alice a Bob sdílejí klíč pro autentizaci zpráv
2) Public key = elektronický digitální podpis (digital signatures)
   - kdokoliv může ověřit

Vlastnosti: a) veřejně ověřitelné

b) přenositelné: – ~~lze~~ příjemce může ukázat třetí straně, která ověří zprávu

c) nepopiratelnost: z definice pouze Alice mohla vytvořit digitální podpis

Definice: Schéma elektronického podpisu se skládá ze tří algoritmů (G, S, V):

1) Generátor klíčů G vrací ~~[illegible]~~ veřejný a soukromý klíč $(pk, sk) \leftarrow G(1^n)$
2) Podpisový algoritmus S je pravděpodobnostní alg., který pro soukromý klíč sk a zprávu m vrací podpis $\sigma = S_{sk}(m)$.
3) Ověřovací alg. V je deterministický alg., který pro veřejný klíč pk, zprávu m a podpis σ vrací $V_{pk}(m, \sigma) \in \{accept, reject\}$.

Korektnost
- Požadujeme $V_{pk}(m, S_{sk}(m)) = accept$ pro všechny $(pk, sk) \leftarrow G(1^n)$ a $m \in \{0,1\}^*$

→ oproti PKE odesílatel má sk. Alice posílá Bobovi zprávu zašifrovanou pk Boba podepsanou pomocí sk Alice.

→ na rozdíl od PKE není nutná randomizace

→ zprávy nemají předem danou strukturu: řetězce z $\{0,1\}^*$. Pokud schéma požaduje strukturu, pak ~~[illegible]~~ vložíme kontrolky

* Security: *

Definice (existential unforgeability under adaptive chosen message attack)

Schéma elektronického podpisu je secure pokud $\forall$ PPT A existuje neglig. ε takové, že $\Pr[A^{S_{SK}(\cdot)}(PK) \text{ podílá}^{\text{podpis}}] \leq \varepsilon(k) \quad \forall k$,

kde pst je přes $(PK, SK) \leftarrow G(1^k)$ a náhodné mince A.

"A podílá podpis" $\equiv$ A vytvoří (m, σ), pro které a) $V_{PK}(m, \sigma) = \text{accept}$
b) m je různá od všech dotazů na S_{SK}

Silná definice:

a) A může poslat dotazy na libovolné zprávy a dostane podpis

b) A podílí podpis, i když m je "nesmyslná"

– není specifické pro kontext

– a) si lze představit i v praxi: notář podepíše dokument bez ohledu na obsah

Aplikace: 1) k budování Public-Key Infrastructure bez veřejného adresáře.

- Jediná důvěryhodná strana (Verisign, certifikační autorita) má PK, který znají všichni uživatelé. Verisign pak podepíše zprávu tvaru

$$m = \text{'Verisign ověřil, že } PK_{Alice} \text{ je veřejný klíč Alice'}.$$

$$\sigma_{Alice} = S_{SK_{Verisign}}(m)$$

- na začátku komunikace pošle Alice certifikát $(PK_{Alice}, m, \sigma_{Alice})$
- lze i hierarchicky Verisign $\rightarrow PK_{CUNI} \rightarrow PK_{MFF} \rightarrow PK_{STUDENT}$.

2) Lze použít k ověření vlastností $\rightarrow$ např. integrita software.

3) Ke konstrukcím encryption schemes s lepší security (Chosen Ciphertext Attack)

Příklady: **Insecure**

1) f OWF:

a) $y = f(x)$ pro $x \leftarrow \{0,1\}^n$. $SK = x$, $PK = y$

b) $S_{SK}(m) = x$

c) $V_{PK}(m, \sigma) = 1$ iff $f(\sigma) = y$.

Analog ručního podpisu. Ověřuje, že Alice umí podepsat.

- není secure $\rightarrow$ lze jednoduše zkopírovat a připojit k jakékoli zprávě m.
- ukazuje, že naše definice je silnější, než klasický podpis.

2) $\mathcal{F} = \{f_i : D_i \to D_i\}_{i \in I}$ je kolekce <u>TDPs</u>

a) $PK = i$, $SK = t$

b) $S_{SK}(m) = f_i^{-1}(m)$

c) $V_{PK}(m, \sigma)$: ověří, že $f_i(\sigma) = m$.

<u>Není secure</u>: f_i může být snadno invertovatelná pro specifickou m

- RSA: pro $m = 1$ $S_{N,d}(1) = 1$.
- obecně: zvol $\sigma \in D_i$, $m := f_i(\sigma)$ a pošli (m, σ).

3) <u>~~Lamport~~ základní RSA</u>

a) $PK = (N, d)$, $SK = (N, e)$

b) $S_{SK}(m) = m^e \bmod N$

c) $V_{PK}(m, \sigma) = 1$ accept iff $\sigma^d = m \bmod N$

<u>Není secure</u>: Pro zprávu m a $PK = (N, d)$, může A požádat o σ_1, σ_2 pro m_1 a $m_2 = m_1^{-1} \cdot m$, kde $m_1 \leftarrow \mathbb{Z}_N^*$. Potom $\sigma_1 \cdot \sigma_2 \bmod N$ je validní podpis pro m. S dobrou pstí je také m různé od m_1, m_2.

- jsme schopni prolomit digitální podpis i bez znalosti SK.

* <u>Konstrukce</u> *

- lze konstruovat z jednosměrných fcí (komplikovaně)
- ukážeme myšlenky konstrukcí z collision-resistant hash funkcí
  1) one-time signature pro zprávy pevné délky ($\mathcal{M}_k = \{0,1\}^k$) z OWFs
  2) hash-then-sign technika pro zprávy neomezené délky
  3) key-refreshing pro převod z one-time → many-use.

* One-time signatures (Lamport '78) *

Nechť f je jednosměrná fce ~~$f: \{0,1\}^k$~~ OWF

Single bit zprávy

1) $SK = \{x_0, x_1\}$ pro $x_0, x_1 \leftarrow \{0,1\}^k$.
   $PK = \{y_0, y_1\}$ pro $y_0 = f(x_0)$, $y_1 = f(x_1)$

2) $S_{SK}(b) = x_b$

3) $V_{PK}(b, x) = \text{accept}$ iff $f(x) = y_b$.

- není moc efektivní, ~~ale~~ podpis vydá polovinu SK
- pokud vidí pouze jeden podpis, pak nemůže efektivně padělat. (x_0 a x_1 jsou zvoleny nezávisle)

zprávy délky ℓ

$$PK = \begin{pmatrix} \boxed{y_{1,0}} & y_{2,0} \cdots y_{\ell,0} \\ y_{1,1} & \boxed{y_{2,1}} \; \boxed{y_{\ell,1}} \end{pmatrix}$$

1) $G(1^k)$: pro $i = 1, \dots, \ell$ zvol $x_{i,0} \leftarrow \{0,1\}^k$, $x_{i,1} \leftarrow \{0,1\}^k$. $SK =$
   $y_{i,0} = f(x_{i,0})$ a $y_{i,1} = f(x_{i,2})$

   $PK = \{y_{1,0}, y_{1,1}, \dots, y_{\ell,0}, y_{\ell,1}\}$ $SK = \{x_{1,0}, x_{1,1}, \dots, x_{\ell,0}, x_{\ell,1}\}$

2) $S_{SK}(m)$ pro $m \in \{0,1\}^\ell$: vrať $\sigma = (\sigma_1, \dots, \sigma_\ell)$, kde $\sigma_i = x_{i,m_i}$ $i = 1, \dots, \ell$

3) $V_{PK}(m, \sigma) = \text{accept}$ iff $f(\sigma_i) = y_{i,m_i}$ pro všechna $i \in \{1, \dots, \ell\}$

~~Věta~~ $\boxed{\text{Tvrzení}}$ Předešlé schéma je ~~secure one-time signature~~ one-time secure (tj. pokud adversary ~~udělá~~ pošle pouze jeden dotaz na S_{SK}) pro ~~message~~ prostor zpráv $\{0,1\}^\ell$, pokud f je jednosměrná funkce, pak

Důkaz: Redukcí na ~~pro~~ inverzi f. Nechť existuje PPT ~~padělatel~~ A, ~~pro~~ který padělá s non-negligibilní pstí ε. Zkonstruujeme B, který invertuje f s non-negligibilní pstí.

- Intuice: (m, σ) dává půl klíče a A musí vyrobit padělaný podpis pro $m \neq m'$, kde m a m' se liší alespoň na jedné pozici i. $\Rightarrow$ musí invertovat. B se pokusí uhodnout pozici pro kterou A invertuje

Invertor $B(y)$

1, zvol $j \leftarrow \{1, \dots, \ell\}$, $c \leftarrow \{0,1\}$, $y_{j,c} = y$.

2, $\forall (i,b) \neq (j,c)$ zvol $x_{i,b} \leftarrow \{0,1\}^k$, $y_{i,b} = f(x_{i,b})$

3, $PK = \{y_{1,0}, y_{1,1}, \dots, y_{\ell,0}, y_{\ell,1}\}$

4, simuluj $A(PK)$

- pro podpis na zprávu m požadovaný A
  Pokud $m_j \neq c$, odpověz správně pomocí ~~[illegible]~~ $\sigma = S_{sk}(m)$
  Jinak, ukonči a B vrátí **fail**.
- pokud A vrátí padělaný podpis (m', σ') a $m'_j = c$, vrať $x = \sigma'_j$
  jinak vrať **fail**.

Pokud A uspěje, potom B uspěje s pstí $\geq \frac{1}{2\ell}$

(pokud pro (j,c) platí $m_j \neq c$ a $m'_j = c$, pak σ'_j je inverze pro y). ∎

* Elektronický podpis pro ~~multiple~~ ~~mnoho~~ dlouhé zprávy: *

hash-then-sign (příště)

* Pro více zpráv *:

- myšlenka: generujme nové klíče za běhu a autentizujme předešlými
- mějme One-time el. podpis (G, S, V) a zkonstruujme schéma
  - veřejný klíč: pk_0, soukromý klíč: sk_0 z prvního schématu
  - zpráva m_1: $(pk_1, sk_1) \leftarrow G(1^k)$
    $\sigma_1 \leftarrow S_{sk_0}(m_1, pk_1)$
    vrať $\sigma'_1 = (1, \sigma_1, m_1, pk_1)$
  - zpráva m_2: $(pk_2, sk_2) \leftarrow G(1^k)$
    $\sigma_2 \leftarrow S_{sk_1}(m_2, pk_2)$
    vrať $\sigma'_2 = (2, \sigma'_1, \sigma_2, m_2, pk_2)$
- velice neefektivní → použijme strom (každý klíč autentizuje dva klíče)
  → konstrukce za použití OWF (komplikovaná)

* Praktické digitální podpisy: *

* Hash-then-sign pro RSA *

- $pk = (N, e)$, $sk = (N, d)$, $ed \equiv 1 \pmod{\varphi(N)}$
- $S_{sk}(m) = (H(m))^d \bmod N$ $\quad$ H je SHA-1 nebo MD5 (SHA-2)

→ intuice: adversary nemá kontrolu nad $H(m)$, nemůže podělat $H(m)$, že ~~vybere~~ zvolí nejdříve σ nebo invertovat jednoduché vstupy pro RSA.

* RSA PKCS #1: $H(m) = 001\,FF \ldots FF00 \,||\, SHA\text{-}1(m)$

→ nemá zdůvodnění

→ není a priory založení na jednosměrnosti RSA, protože $H(m)$ má velmi specifickou strukturu

* Random Oracle Model:

- pokud by H byla plně náhodná fce, $H: \{0,1\}^* \to \mathbb{Z}_N^*$, pak lze dokázat z jednosměrnosti RSA
- žádná praktická funkce H však není plně náhodná.

Foundations of Theoretical Cryptography — lecture 9

Bitcoin → distribuovaný konsensus (2009) Satoshi Nakamoto

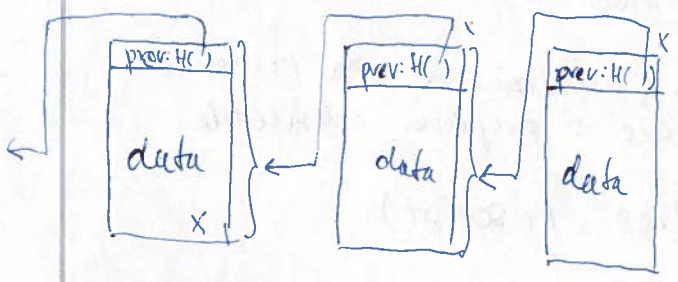
Search Puzzle (Proof-of-Work)

- hash funkce $H: \{0,1\}^* \to \{0,1\}^n$
- puzzle $ID \leftarrow \mathcal{D}$, kde $\mathcal{D}$ je distribuce s vysokou min-entropy
- cílová množina $Y \subseteq \{0,1\}^n$

– Řešení: x t.ž. $H(id \| x) \in Y$

Hash pointers (Blockchain)

- datová struktura, která zamezuje tamperingu



- pro změnu dat, musí změnit hashe

- Bitcoin používá SHA-256 jako hashovací fci
- Digitální podpis → Elliptic Curve Digital Signature Algorithm (ECDSA)
  - private key: 256 bitů
    public key: 512 bitů (257 bitů compressed)
    délka zprávy: 256 bitů
    délka podpisu: 512 bitů
- Uživatel se zaregistruje vygenerováním (pk, sk) pk funguje jako address (hash(pk))

Konsenzus:

- problém Byzantských generálů (Lamport, Shostak, Pease '82)
  - generálové se chtějí dohodnout zda zaútočit nebo se stáhnout
  - musí se dohodnout na jedné strategii
  - někteří generálové jsou zrádci a můžou posílat různé zprávy (stáhnout i zaútočit)
  - lze dosáhnout, pokud loajální generálové jsou ve většině
  - nutné znát identity všech generálů (účastníků)
- pokud neznáme identity, pak lze použít tzv. Sybil attack — permissionless setting
  - můžu vytvořit mnoho identit

Konsenzus v Bitcoinu

1, transakce jsou posílány uživatelům

2, každý uživatel vybere ~~tar~~ nové transakce a vytvoří blok

3, jeden uživatel je vybrán náhodně a jeho blok je poslán ostatním

4, ostatní přijmou blok, pokud obsahuje validní transakce

5, ~~blok~~ hash bloku bude přidána do nového bloku

---

transakce

- pro převod peněz musí existovat někde v historii

inputs : - ~~pointe~~ seznam hash. pointerů na předchozí transakce s podpisem odesílatele

outputs : seznam adres (+ script)

1	Inputs : ∅ Outputs : 12.5 → Alice
2	Inputs : 1[0] Outputs : 10 → Bob, 2.5 → Alice sign(Alice)

- **Script je** stack-based jazyk, který umožňuje vytvářet "smart contracts"
  - není Turing-complete
  - žádné cykly → **terminace v konečném čase**
  - umožňuje uplatnit transakci při podpisu dvou uživatelů (escrow)

Block

- header + data
  - header : 1, prev : H(previous_block)
    2, mrkl_root : MT(transactions)
    3, nonce : 0x7a83 ∈ $\{0,1\}^{32}$
    4, hash : hash bloku musí mít t leading zeros
  - transactions : 1, coinbase 12.5 BTC + fees
    2, vybrané transakce

Mining:

1, sesbíráme transakce a spočítáme mrkl_root

2, vybereme předchozí blok a odkážeme pomocí hashe (bereme poslední block z nejdelšího řetězce) block-chain

3, vybereme nonce a spočítáme hash

a, pokud validní, pošleme ostatním

b, pokud ne, inkrementujeme nonce a zkoušíme znovu

difficulty nastavíme tak, abychom našli blok zhruba každých 10 min.

$$next_difficulty = previous_difficulty \cdot \frac{2016 \cdot 10}{time_last_2016_blocks}$$

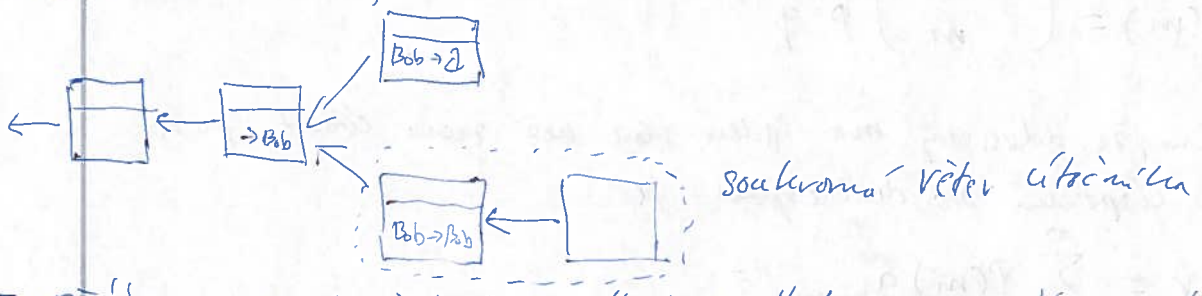
- validita: – interpretuj hash jako destinní číslo, p
  – validní blok, pokud je $p \leq difficulty$

Security

- není jasné, čeho přesně BTC dosahuje
- rozhodně neposkytuje anonymitu (pseudonymity)
- jelikož je k provedení transakce potřeba sk, je falšování transakcí obtížné
- hlavní problém: double spending / konsenzus
- pokud má adversary $\leq 1/3$ výpočetního výkonu v BTC (hashrate), pak snad secure (selfish mining)

Double spending

- attacker se může pokusit zaplatit dvakrát



- příjemce proto čeká na "potvrzení" transakce → blok je následován n bloky

částečná analýza:

adversary má q zlomek hashrate a zbytek sítě má p zlomek hash rate.

→

$z = n - m$ tj. n počet bloků sítě
m počet bloků attackera

→ nový blok nalezne síti s pstí p a adversarymu s pstí q.

→ zajímá nás, zda bude mít adversary nikoliv o blok více

$$a_z = \Pr[\text{adversary bude mít o blok víc}]$$
$$z_j = -1 \quad \text{pro nějaké } j$$

$$z_{i+1} = \begin{cases} z_i + 1 & \text{s pstí } p \\ z_i - 1 & \text{s pstí } q \end{cases}$$

pro $z < 0$ je $a_z = 1$

$$a_z = p\, a_{z+1} + q\, a_{z-1} \Rightarrow a_z = \min\left(\frac{q}{p}, 1\right)^{\max(z+1, 0)}$$

$$= \begin{cases} 1 & \text{pokud } z < 0 \text{ nebo } q > p \\ (q/p)^{z+1} & \text{pokud } z \geq 0 \text{ a } q < p \end{cases}$$

→ pokud má ~~více než~~ alespoň polovinu hashrate, pak vždy předběhne

→ pokud má méně, pak a_z klesá exponenciálně s náskokem

- zatímco síť pracuje na n blocích pro konfirmaci, adversary získá m bloků

→ (ze modelovat jako negativní binomiální distribuci pro m úspěchů (attacker bloků) než nastane n neúspěchů (bloky sítě))

$$P(m) = \binom{m+n-1}{m} p^n q^m$$

→ řekněme, že adversary má jeden blok, než začne útočit, pak pst. úspěchu pro double spend je

$$r = \sum_{m=0}^{\infty} P(m)\, a_{n-m-1} =$$

$$= \sum_{m=0}^{n-1} \binom{m+n-1}{m} p^n q^m \left(\min\left\{\frac{q}{p}, 1\right\}\right)^{n-m} + \sum_{m=n}^{\infty} \binom{m+n-1}{m} p^n q^m$$

$$= \begin{cases} 1 - \sum_{m=0}^{n} \binom{m+n-1}{m} \left(p^n q^m - p^m q^n\right) & \text{pokud } q < p \\ 1 & \text{pokud } q \geq p \end{cases}$$

→ hodnoty :

q	n = 3	n = 6	n = 10
6%	0,394 %	0,003 %	≈0%
10%	1,712 %	0,059 %	0,001 %
20%	11,584 %	2,331 %	0,316 %

Problémy

- centralizace : a) cizká skupina developerů

  b) sdružování do poolů

  → ochrana proti vysokému rozptylu pro nalezení bloku (pro nízké q je pst. nalezení bloku za jeden rok < 1/2)

- neefektivita : je třeba konstantně počítat SHA256 hashe

  → odhadovaná roční spotřeba 32 TWh Tera = 10^{12}

  = Dánsko v roce 2014 (odhad)

  - alternativní návrhy založené na Proof-of-(Space / Stake)

    → škálovatelnost

- historie :

  Proof-of-Work : Cynthia Dwork a Moni Naor (1992)

  Ledger pomocí hash funkcí (cryptographic time-stamping) : Haber + Stornetta '91; Back '97

  eCash → anonymní elektronické peníze (David Chaum '83)

  → systém s centrální bankou

Foundations of Theoretical Crypto

Secure 2-Party Computation

- Alice a Bob chtějí zjistit, zda se navzájem milují ♡ → ♥♥
- Alice však nechce vyzradit Bobovi, že ho miluje, pokud ji také nemiluje Bob se bojí toho samého.
- Alice má bit $a \in \{0,1\}$ reprezentující, zda miluje Boba a Bob má bit $b \in \{0,1\}$.
- Chtějí zjistit, zda $a = b = 1$, tj. chtějí spočítat $f(a,b) = a \wedge b$
- Cíl: nalézt protokol, který jim umožní dozvědět se $f(a,b)$, ale nic navíc.
- Na základě rozumných předpokladů lze zkonstruovat protokol pro každou fci $f(a,b)$ (nejen pro $a \in \{0,1\}$).

Security:

- correctness: pokud postupují dle protokolu, pak lokální výstup je $f(a,b)$
- privacy: protokol nevyzradí informaci kromě toho, co se dozví z $f(a,b)$
  - pro $f(a,b) = a \wedge b$ a $a = 0$, pak $f(a,b) = 0$ pro $b \in \{0,1\}$ a Alice nesmí z view poznat, zda $b = 0$ nebo $b = 1$.
- input independence: vstup Alice nesmí záviset na vstupu Boba
  - Alice nesmí být schopna nastavit $a = b$. Jinak by byla schopna z lokálního výstupu protokolu $f(a,b) = a \wedge b$ dozvědět se Bobův vstup.
- důležité pro aplikace jako e-voting: privacy zajistí, že můj hlas zůstane anonymní
  aukce: input independence zajistí, že nemůžeme nastavit nabídku podle oponentů.

Semi-honest (honest-but-curious) Adversary:

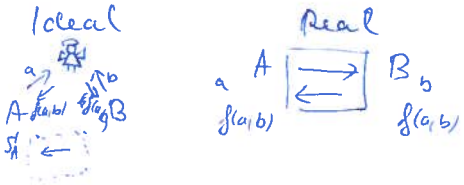
- postupuje dle protokolu, ale snaží se vyzvědět něco z view.
- i přes zdánlivě nízkou záruku je design protokolů v modelu se semi-honest adversary netriviální. (nelze jen poslat vstupy)
- obecně můžeme převést semi-honest protokol na protokol proti obecnému adversary pomocí zero-knowledge.
- máme protokol (A,B), řekneme, že (A,B) počítá $f(a,b)$, pokud platí, že lokální výstup A a B na vstupech a a b je $f(a,b)$
- $\mathrm{VIEW}_A(A(a,1^n) \leftrightarrow B(b,1^n)) = (a, r_a, m_1^B, \dots, m_n^B)$

- definice security pomocí simulátoru

Definice: Nechť $f(a,b)$ je deterministická fce. Protokol (A,B), který počítá $f(a,b)$ splňuje security v přítomnosti ~~honest~~ semi-honest adversary, pokud existují PPT algoritmy S_A a S_B t.ž. $\forall a,b$ platí:

$$\{S_A(a, f(a,b), 1^n)\}_{n\in\mathbb{N}} \approx_c \{VIEW_A(A(a,1^n) \leftrightarrow B(b,1^n))\}_{n\in\mathbb{N}},$$

$$\{S_B(b, f(a,b), 1^n)\}_{n\in\mathbb{N}} \approx_c \{VIEW_B(A(a,1^n) \leftrightarrow B(b,1^n))\}_{n\in\mathbb{N}}.$$



- definice je podobná semantic security
- pro semi-honest není input independence problém
- další problémy:
  - pravděpodobnostní fce
  - větší lokální výstupy $f_A(a,b)$ a $f_B(a,b)$
  - obecný adversary
  - statická vs. dynamická korupce
  - fairness

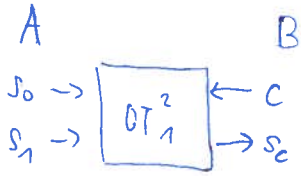
AND dvou bitů:

- základní stavební prvek ~~1 out of 2~~ 1-ze-2 oblivious transfer

Oblivious transfer:

1-ze-2 oblivious transfer (OT_1^2) je protokol mezi Alicí (sender) a Bobem (rec.)

$s_0, s_1 \in \{0,1\}$

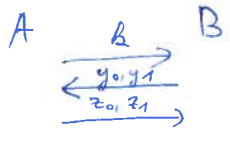


$c \in \{0,1\}$ "choice"

$$f((s_0,s_1),c) = (\perp, s_c)$$

- konstrukce pro semi-honest adversary z trapdoor permutací
- Nechť $\mathcal{F} : \{f_k : D_k \to D_k\}_{k\in K}$ je kolekce trapdoor permutací
  a $\{b_k : D_k \to \{0,1\}\}_{k\in K}$ je kolekce ~~trapdoor~~ ~~predikátů~~ hardcore bitů pro $\mathcal{F}$.

Protokol pro OT_1^2



Společný vstup: security parameter $n \in \mathbb{N}$

Vstup Alice: $s_0, s_1 \in \{0,1\}$

Vstup Boba: $c \in \{0,1\}$

1) Alice: zvolí $(k,t) \leftarrow G(1^n)$ pro $\mathcal{F}$ a pošle k Bobovi
2) Bob zvolí $x, y \leftarrow D_k$, $y_c = f_k(x)$ a $y_{1-c} = y$ a pošle y_0, y_1 Alici
3) Alice použije trapdoor t pro nalezení $x_0 = f_k^{-1}(y_0)$, $x_1 = f_k^{-1}(y_1)$.
   $z_0 = b(x_0) \oplus s_0$ a $z_1 = b(x_1) \oplus s_1$ a pošle z_0, z_1 Bobovi.
4) Bob použije $x = x_c = f_k^{-1}(y_c)$ pro výpočet $s_c = b_k(x_c) \oplus z_c$

- protokol lze použít pro OT_1^ℓ pro konstantní $\ell \in \mathbb{N}$

<u>Tvrzení</u>: Pokud F je kolekce trapdoor permutací, pak (A,B) je secure OT_1^2 protocol v přítomnosti semi-honest adversary.

<u>Myšlenka důkazu</u>: $S_A((s_0, s_1), \perp, 1^n)$:

1) zvol $(k,t) \leftarrow G(1^n)$ pomocí náh. mince r
2) zvol $y_0, y_1 \leftarrow D_k$
3) vrať $((s_0, s_1), r, (y_0, y_1))$

- řetězce y_0, y_1 ve výstupu S_A jsou identicky distribuované jako y_0, y_1 od Boba.

  Bob volí $x_c \leftarrow D_k$ $y_c = f(x_c)$, $y_{1-c} \leftarrow D_k$

$S_B(c, s, 1^n)$:

1) zvol $(k,t) \leftarrow G(1^n)$
2) zvol $x, y \leftarrow D_k$ pomocí r. Nechť $y_c = f_k(x)$ $y_{1-c} = y$
3) vypočti $z_c = b_k(x) \oplus s$ a zvol $z_{1-c} \leftarrow \{0,1\}$
4) vrať $(c, r, (k, (z_0, z_1)))$

- kromě (z_0, z_1) je výstup S_B distribuováno identicky jako Bobovo view.
- platí i pokud uvážíme z_c ($z_c = b_k(x) \oplus s$ v obou případech)
- Jediný rozdíl je v z_{1-c}, které ale nelze ~~[illegible]~~ ~~[illegible]~~ rozlišit od $z_{1-c} \leftarrow \{0,1\}$, protože b_k je hardcore bit pro F. $\boxtimes$

* <u>Protokol pro AND pomocí OT</u> *

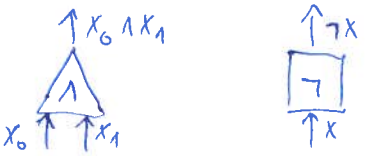
1) Alice nastaví $(s_0, s_1) = (0, a)$
2) Bob nastaví $c = b$
3) Alice a Bob použijí OT_1^2 pro vstupy $((s_0, s_1), c)$. Bob získá s_c.
4) Bob pošle Alici s_c.
5) oba vrátí $f(a,b) = s_c$

- $s_c = 1 \iff a = b = 1$

<u>Poznámka</u>: pokud je OT_1^2 secure ~~[illegible]~~ v přítomnosti semi-honest adversary, pak ~~je~~ (A,B) ~~proto~~ ~~[illegible]~~ je secure protokol pro $f(a,b) = a \wedge b$.

Protokol pro libovolnou funkci

- libovolnou fci vyhodnotitelnou v polynomiálním čase lze ~~pomocí~~ reprezentovat pomocí Booleovského obvodu polynomiální velikosti s hradly AND a NOT.
- obvod je DAG (directed acyclic graph):
  - vrcholy odpovídají hradlům
  - hrany odpovídají ~~vrcholů~~ hodnotám
  - vstupní hrany a výstupní hrany



Počítání se sdílenými vstupy:

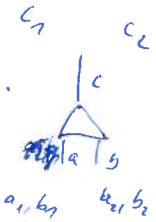
- Alice má vstupy $a^1, \dots, a^n \in \{0,1\}$, Bob má vstupy $b^1, \dots, b^n \in \{0,1\}$
- chtějí počítat $f((a^1, \dots, a^n), (b^1, \dots, b^n))$
- vymíní si "části vstupu".

Input sharing:

1) Alice zvolí $a_1 \leftarrow \{0,1\}$, položí $a_2 = a \oplus a_1$. Pošle a_2 Bobovi.

2) Bob zvolí $b_2 \leftarrow \{0,1\}$, položí $b_1 = b \oplus b_2$ a pošle b_1 Alici.

- a_1, a_2 samostatně neobsahují informaci o a, stejně b_1, b_2 neobsahují informaci o b.
- Cíl: Z sharů a_1, b_1 + a_2, b_2 získat shary pro label výstupní hrany hradla



- postupně můžeme provést pro celý obvod.
- v momentě, kdy mají shary pro výstupní hranu obvodu, si vymíní shary a zjistí $f((a^1, \dots, a^n), (b^1, \dots, b^n))$

- Výpočet NOT se sdílenými vstupy:
  - Alice a_1 share pro a Bob a_2 share pro a
  - $f(a) = \neg a$: Alice $c_1 = \neg a_1$ Bob $c_2 = a_2$ $\Rightarrow$ $c_1 \oplus c_2 = \neg a$

Počítání AND se sdílenými vstupy:

- Alice má a_1, b_1, Bob má a_2, b_2, potřebují c_1 a c_2 (náhodné) pro $c = a \wedge b$.
- Nad $\mathbb{Z}_2$ chtějí shary pro $(a_1 + a_2) \cdot (b_1 + b_2) = (a_1 \oplus a_2) \wedge (b_1 \oplus b_2)$
- Alice zvolí $c_1 \leftarrow \{0,1\}$ a sestaví tabulku pro OT_1^4:

Index	(a_2, b_2)	hodnota
1	(0,0)	$c_1 + a_1 b_1$
2	(0,1)	$c_1 + a_1 \cdot (b_1 + 1)$
3	(1,0)	$c_1 + (a_1 + 1) \cdot b_1$
4	(1,1)	$c_1 + (a_1 + 1)(b_1 + 1)$

- Bob ~~zvolí~~ použije $2a_2 + b_1 + 1$ jako index pro OT_1^4
- Bob použije jako c_2 hodnotu z OT_1^4.

Pozorování: Pokud je OT_1^4 secure protocol v přítomnosti semi-honest adv., pak (A,B) je secure v přítomnosti semi-honest adv. pro fci $((a_1, b_1), (a_2, b_2)) \mapsto (c_1, c_2)$, kde c_1, c_2 jsou náhodné t.ž. $c_1 + c_2 = (a_1 + a_2) \cdot (b_1 + b_2)$.