Curs 13 + 14

Zero - Knowledge Proofs

Peggy = prover, Victor = verifier

Peggy vrea să îl convingă pe Victor că știe **ceva**, fără să ca Victor să afle **exact** cât știe Peggy.

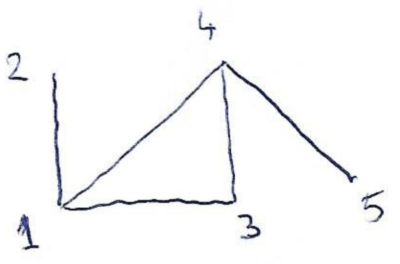
Exemplu clasic: Graph Isomorfism Problem.

G_1, G_2 grafuri, $|G_1| = |G_2| = n$ același nr de vârfuri.

$$\varphi : G_1 \to G_2 \text{ izomorfism}:$$

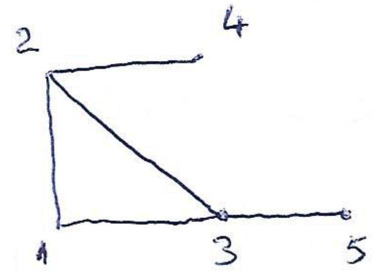
φ **bijectie** și xy muchie în G_1 $\iff$ $\varphi(x)\varphi(y)$ muchie în G_2

G_1



$\sigma = (1,2,4,3)$ $\longrightarrow$

G_2



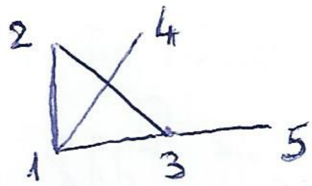
G_1, G_2 **publice**

Peggy știe un izomorfism φ (secret)

Peggy vrea să îl convingă pe Victor că știe un izomorfism, fără să îi dezvăluie tot izomorfismul.

<u>Procedură</u> Peggy ia G_2 și aplică o permutare <u>secretă</u> ψ pe vârfurile lui G_2. $\Rightarrow$ alt graf H

(dacă $\psi = (1\ 2)$

H = ) commitment graph (angajament)

Peggy cunoaște următoarele izomorfisme

$$\varphi : G_1 \to G_2$$

$$\psi : G_2 \to H$$

$\psi \circ \varphi$: $G_1 \to H$, notat $\varphi\psi$

Victor lansează o provocare: alege $b \in \{1, 2\}$ și întreabă despre izomorfismul grafurilor H și G_b. (challenge). Peggy dă un <u>răspuns</u> trimițând $X = \psi$ sau $X = \varphi\psi$.

$$\varphi\psi = (1\ 2)(1\ 2\ 4\ 3) = (2\ 4\ 3)$$

$P \to V : H$

$V \to P : b$

$P \to V : X$

$$X = \begin{cases} \psi & \text{dacă } b = 2 \\ \varphi\psi & \text{dacă } b = 1 \end{cases}$$

Dacă Peggy nu cunoaște φ, poate răspunde corect doar cu probabilitate de 50 %. Prin repetarea protocolului minciuna iese la iveală...

Protocolul de identificare al lui Schnorr

Secretul lui Peggy : $x = \log_g y$, $\langle g \rangle = G$ are ordin q.
(y public)

$P \longrightarrow V : r = g^k$, k random

$V \longrightarrow P : e$

$P \longrightarrow V : s = k + x e$, unde numai Peggy știe k și x

V calculează $g^s y^{-e}$ și trebuie să-i dea r.

$$g^s y^{-e} = g^s (g^x)^{-e} = g^{k+xe-xe} = g^k = r \quad \text{O.K.}$$

Dacă Peggy trișează, probabilitatea ei de reușită este $\frac{1}{q}$, mult mai bine decât $\frac{1}{2}$!

<u>Obs</u> Nu putem rula protocolul de două ori, fiindcă Victor îl află pe x dacă r se repetă :

(r, e, s) $\qquad$ (r, e', s')

$$r = g^s y^{-e} = g^{s'} y^{-e'}$$

$$s + x(-e) = s' + x(-e') \bmod q$$

$$x = (s - s')(e - e')^{-1} \bmod q$$

(deci trebuie evitată alegerea unui aceluiași k)

Definiție $R(x,k)$ = compute the commitment r
x = secretul, k = o alegere random.

c = provocarea (challenge)

$S(c,x,k)$ = algoritmul folosit de Peggy pentru a crea un răspuns s

$V(r,c,s)$ = algoritm de verificare folosit de Victor.

$S'(c,s)$ = algoritmul de simulare care creează o valoare r a.î. (r,c,s) este un protocol valid.

Existența lui S' este necesară pt a ne convinge că Victor nu află x prin derularea protocolului. x secret, $y = g^x$ public

$R(x,k)$: $r = g^k$

$S(c,x,k)$: $s = k + cx \mod q$

$V(r,c,s) = \underline{true} \iff g^s = r y^{-c}$

$S'(c,s)$: $r = g^s y^c$

Într-adevăr, S' îl regăsește pe r fără a îl ști pe x.

Deci este zero-knowledge pentru Verifier.

Alt exemplu folosește Pedersen commitments

$$B_a(x) = h^x g^a$$

unde $G = \langle g \rangle$ grup ciclic de ordin prim q

$h \in G$, $\log_g h$ necunoscut.

x valoare angajată

a random.

Caz particular important $x \in \{-1, 1\}$.

Persoana care face angajamentul trebuie să demonstreze că

$$x \in \{-1, 1\}$$

fără a dezvălui valoarea angajamentului!

- Peggy alege random d, r, w modulo q.
- Peggy publică $B_a(x)$ și

$$\alpha_1 = \begin{cases} g^r (B_a(x) h)^{-d} & \text{dacă } x = 1 \\ g^w & \text{dacă } x = -1 \end{cases}$$

$$\alpha_2 = \begin{cases} g^w & \text{dacă } x = 1 \\ g^r (B_a(x) h^{-1})^{-d} & \text{dacă } x = -1 \end{cases}$$

- Victor trimite c = random challenge.
- Peggy răspunde cu

$$d' = c - d$$
$$r' = w + a d'$$

și trimite

$$(d_1, d_2, r_1, r_2) = \begin{cases} (d, d', r, r') & \text{dacă } x = 1 \\ (d', d, r', r) & \text{dacă } x = -1 \end{cases}$$

- Victor verifică dacă:

$$c = d_1 + d_2$$
$$g^{r_1} = \alpha_1 \left(B_a(x)\, h\right)^{d_1}$$
$$g^{r_2} = \alpha_2 \left(B_a(x)\, h^{-1}\right)^{d_2}$$

Un sistem de vot electronic

m votanți, n centre de numărare, <u>referendum</u> (da sau nu)

- Numai persoane autorizate votează
- Fiecare votant votează doar <u>o dată</u>
- Niciun votant nu poate afla cum a votat altcineva
- Votul cuiva nu se poate duplica
- Rezultatul se calculează corect
- Toți participanții pot verifica corectitudinea
- Protocolul funcționează chiar dacă se încearcă trișarea lui.

<u>Setup</u> Fiecare centru de vot are o cheie publică E_i

G fixat, ciclic, de ordin q.

g, h alese, g generator, $h = g^x$, nimeni nu cunoaște x.

Fiecare alegător are o cheie publică de signatură. (7)

Votare Fiecare alegător din m alege $v_j \in \{-1, 1\}$

- alege $a_j \in \mathbb{Z}_q$ random
- publică buletinul lui de vot $B_j = B_{a_j}(v_j)$.
- publică un certificat

$$C = H(\alpha_1 \| \alpha_2 \| B_a(x)) \quad , H \text{ hash function}$$

care arată că votul lui $x \in \{-1, 1\}$ fără a dezvălui val

- Buletinul de vot + certificatul sunt semnate cu alg. public de signatură.

Distribuirea voturilor

Cu Shamir secret sharing se distribuie valoarea lui a_j și v_j în centrele de numărare.

Fiecare alegător j alege random polinoame modulo q de grad $\in \mathbb{F}_q$

$t < n$

$$R_j(X) = v_j + r_{1,j} X + \dots + r_{t,j} X^t$$

$$S_j(X) = a_j + s_{1,j} X + \dots + s_{t,j} X^t$$

Alegătorul calculează

$$(u_{i,j}, w_{i,j}) = (R_j(i), S_j(i)) \qquad 1 \le i \le n$$

- Alegătorul ~~calculează~~ cifrează perechea

$$(u_{i,j}, w_{i,j}) = (R_j(i), S_j(i))$$

folosind cheia publică E_i a centrului de numărare.

și îl trimite ~~lui~~ centrului de numărare E_i.

- Alegătorul publică angajamentele lui relativ la polinoamele $R_j(X)$, $S_j(X)$ sub forma

$$B_{\ell,j} = B_{s_{\ell,j}}(r_{\ell,j}) \quad \text{pt } 1 \le \ell \le t.$$

Verificarea consistenței

Fiecare centru de numărare i trebuie să verifice că valoarea lui

$$(u_{i,j}, w_{i,j})$$

primit de la alegătorul j este consistent cu angajamentul făcut de alegător:

$$B_j \prod_{\ell=1}^{t} B_{\ell,j}^{i^\ell} = B_{a_j}(v_j) \prod_{\ell=1}^{t} B_{s_{\ell,j}}(r_{\ell,j})^{i^\ell} =$$

$$= h^{v_j} g^{a_j} \prod_{\ell=1}^{t} \left(h^{r_{\ell,j}} g^{s_{\ell,j}}\right)^{i^\ell} = h^{\left(v_j + \sum_{\ell=1}^{t} r_{\ell,j} i^\ell\right)} \cdot$$

$$\cdot g^{\left(a_j + \sum_{\ell=1}^{t} s_{\ell,j} i^\ell\right)} = h^{u_{i,j}} g^{w_{i,j}}$$

Numărarea voturilor

Fiecare din cele n centre de numărare postează suma voturilor primite

$$T_i = \sum_{j=1}^{m} u_{i,j}$$

și suma codificărilor de valori alese random:

$$A_i = \sum_{j=1}^{m} w_{i,j}$$

Alte centre și alegători pot verifica consistența:

$$\prod_{j=1}^{m} \left(B_j \prod_{l=1}^{t} B_{l,j}^{j^l} \right) = \prod_{j=1}^{m} h^{u_{i,j}} g^{w_{i,j}} = h^{T_i} g^{A_i}$$

Fiecare parte poate calcula rezultatul considerând t+1 dintre valorile T_i și rezolvând un sistem sau o interpolare:

$$T_i = \sum_{j=1}^{m} u_{i,j} = \sum_{j=1}^{m} R_j(i) =$$

$$= \underbrace{\left(\sum_{j=1}^{m} v_j \right)}_{V} + \underbrace{\left(\sum_{j=1}^{m} r_{1,j} \right)}_{W_1} i + \dots + \underbrace{\left(\sum_{j=1}^{m} r_{t,j} \right)}_{W_t} i^t$$

Se rezolvă sistemul $\Rightarrow$ se află $\sum_{j=1}^{m} v_j$

$\Rightarrow$ se află câți de +1 și câți de -1.

Secure multi-party computation

Un grup de milionari la restaurant.
Cel mai bogat trebuie să plătească consumația.
Ei nu doresc să se afle câți sunt de bogați, dar vor să afle cine este cel mai bogat.

$$f(x_1, \ldots x_n) = i \iff x_i > x_j \ \forall i \neq j$$

Cazul bipartit

A și B au inputs x și y
A vrea să calculeze $f_A(x,y)$, B vrea să calculeze $f_B(x,y)$.
B nu trebuie să știe nimic despre x, nici A despre y.

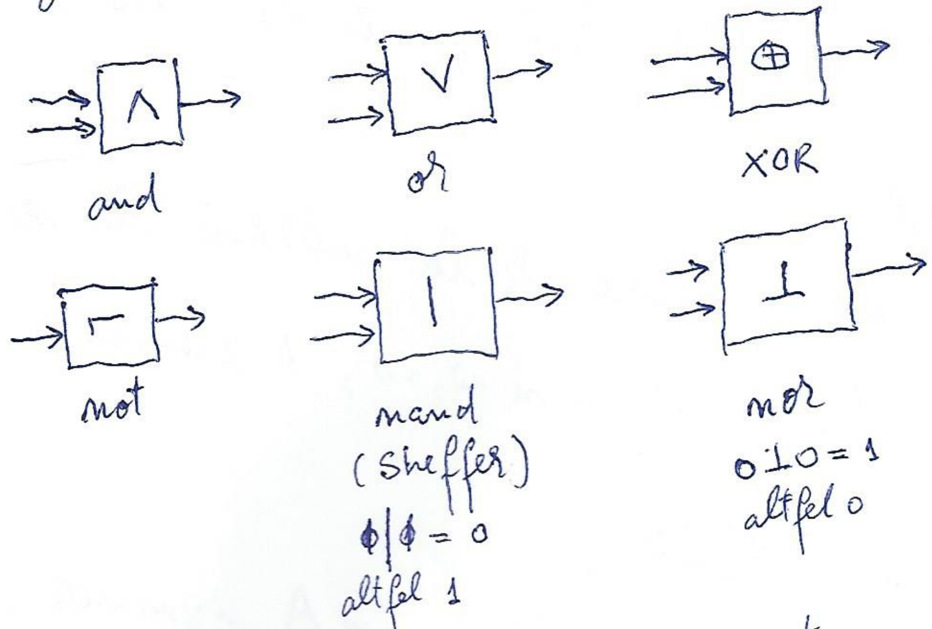
A să aibă un extra input K, atât de lung cât $f_A(x,y)$.
Dorim un protocol în care B află valoarea funcției

$$f(x,y,K) = \left(K \oplus f_A(x,y),\ f_B(x,y)\right)$$

apoi B trimite $K \oplus f_A(x,y)$ lui A, care îl decriptează.
Deci este suficient ca o singură funcție să fie calculată,
și B are voie să citească rezultatul.

Pp. că f poate fi calculată în timp polinomial $\Rightarrow \exists$ circuit de mărime polinomială!

(binary circuit)



Circuit = cabluri $\cup$ porți

$W = \{w_1, \dots w_n\}$ $\quad$ $G = \{g_1, \dots g_m\}$

<u>Yao's</u> garbled <u>circuit</u> (circuit criptat)

- $\forall$ cablu w_i primește două chei K_i^0, K_i^1 : criptarea lui 0 și criptarea lui 1.
- $\forall$ cablu w_i se alege random $\rho_i \in \{0,1\}$.

  actual value $v_i \Rightarrow$ encrypted value $v_i \oplus \rho_i$
- $\forall$ poartă g_i se calculează tabela cifrată.

  $\text{Input}(g_i) = \{w_{i_0}, w_{i_1}\}$

  $\text{Output}(g_i) = \{w_{i_2}\}$

$$c_{a,b}^{w_{i_2}} = E_{K_{w_{i_0}}^{a \oplus \rho_{i_0}},\, K_{w_{i_1}}^{b \oplus \rho_{i_1}}}\left(K_{w_{i_2}}^{\sigma_{a,b}} \,\|\, \sigma_{a,b} \oplus \rho_{i_2}\right)$$

pentru $a, b \in \{0,1\}$ unde $\sigma_{a,b} = g_i(a \oplus \rho_{i_0}, b \oplus \rho_{i_1})$

Protocolul Yao

(12)

- A generează ~~protoc~~ circuitul cifrat și îi dă lui B valorile $c^i_{a,b}$
- A îi transmite lui B valorile de la cablurile lui de input. Ex: în loc de $w_1 = 0$ și $w_2 = 0$, A transmite

$$K_1^0 \| 1 \text{ și } K_2^0 \| 0$$

- Prin oblivious transfer, B îi trimite lui A inputurile lui, în așa fel, încât A să poată citi doar lucruri pe care le cunoaște dar nu și cele care îl privesc pe B.
- A îi dă lui B și pentru output wires.

Shamir Secret Sharing

În continuare seminar.

5 directori au un sistem de acces 3 din 5.

Cheia secretă $5 \in \mathbb{F}_{11}$

3 dintre ei primesc identitățile 4, 7, 9

Polinomul $5 + \underset{a}{X} + \underset{b}{X^2}$ are $f(0) = 5$

Pe stickurile lor avem

$$(4, f(4) = 3), (7, f(7) = 6), (9, 7) \; f(9)$$

$$\mathbb{Z}_{11} \begin{cases} s + 4a + 5b = 3 \\ s + 7a + 5b = 6 \\ s + 9a + 4b = 7 \end{cases}$$

$$\begin{cases} s + 4a + 5b = 3 \\ 3a = 3 \Rightarrow \boxed{a = 1} \\ 5a - b = 4 \end{cases}$$

$$\begin{cases} s + 5b = 10 \Rightarrow \boxed{s = 5} \\ b = 1 \end{cases}$$

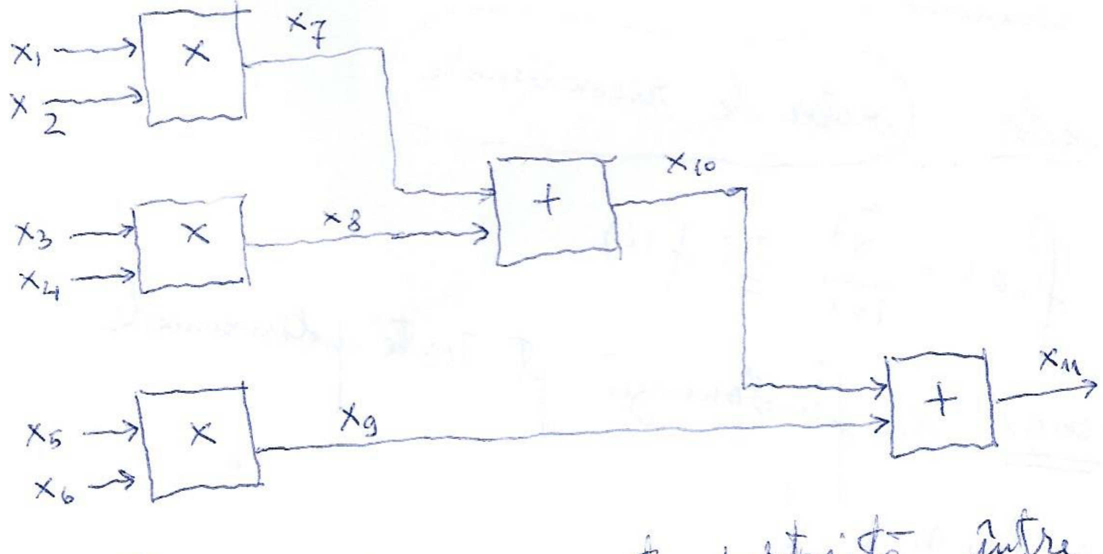
Cazul multipartit

- exemplu bazat pe secret sharing
- 6 părți $P_1, \dots P_6$ au 6 valori secrete $x_1, \dots x_6 \in \mathbb{F}_p$

(normal $p \approx 2^{128}$, în exemplul nostru $p = 101$)

- Vor să calculeze o funcție, de exemplu

$$f(x_1, \dots x_6) = x_1 x_2 + x_3 x_4 + x_5 x_6 \mod p$$



Fiecare valoare x_i este partajată între participanți

j --- primește $x_i^{(j)}$

Cum calculăm partajele de output, cunoscând partajele de input?

Adunare Două secrete a și b sunt partajate folosind polinoamele

$$f(X) = a + f_1 X + \dots + f_t X^t$$

$$g(X) = b + g_1 X + \dots + g_t X^t$$

Fiecare participant primește $a^{(i)} = f(i)$, $b^{(i)} = g(i)$

Considerăm polinomul $h(X) = f(X) + g(X)$.

El codifică secretul $c = a + b$.

$$c^{(i)} = h(i) = f(i) + g(i) = a^{(i)} + b^{(i)} \quad \text{în } \mathbb{F}_p$$

Deci fiecare trebuie să adune partele lui de secret!

Înmulțire Reamintim interpolarea Lagrange:

$f(X)$ polinom necunoscut.

$\Rightarrow$ există un vector, vector de recombinare

a.î. $f(0) = \sum_{i=1}^{n} \underline{r_i} f(i)$

și aceiași r_i funcționează pt toate polinoamele

de grad $\underline{n-1}$!

Fiecare parte are un share de secret

$$a^{(i)} = f(i) \ , \ b^{(i)} = g(i)$$

unde $f(0) = a$, $f(0) = b$.

Vrem să calculăm $c^{(i)} = h(i)$ astfel încât $h(0) = c = ab$.

- Fiecare P_i calculează $d^{(i)} = a^{(i)} b^{(i)}$
- Fiecare P_i produce un polinom $\delta_i(X)$ de grad $\leq t$ a.î. $\delta_i(0) = d^{(i)}$.
- Fiecare parte P_i distribuie părților P_j valoarea

$$d_{i,j} = \delta_i(j).$$

- Fiecare parte j calculează $c^{(j)} = \sum_{i=1}^{n} \underbrace{r_i}_{\text{vect. de recombinare}} d_{i,j}$

<u>De ce funcționează?</u>

- La pasul 1 se calculează un polinom $h'(X)$ de grad $\leq 2t$ cu $d^{(i)} = h'(i)$, și $c = h'(0)$. Problema e gradul [h' coresp pol. fg]

- <u>Obs</u> Dacă $2t \leq n-1$, $c = \sum_{i=1}^{n} r_i d^{(i)}$
- Considerăm polinomul $\delta_i(X)$ generat la pasul 2.

$$\text{Dacă } h(X) = \sum_{i=1}^{n} r_i \cdot \delta_i(X)$$

Toate $\delta_i(x)$ sunt de grad $\leq t \Rightarrow \deg h \leq t$

Din nou $h(0) = \sum_{i=1}^{n} r_i \delta_i(0) = \sum_{i=1}^{n} r_i d^{(i)} = c$

din nou dacă $2t \leq n-1$.

Dar h este exact polinomul folosit la ultimul pas:

$$h(j) = \sum_{i=1}^{n} r_i \delta_i(j) = \sum_{i=1}^{n} r_i d_{i,j} = c^{(j)}$$

Deci este suficient să luăm $t < \frac{n}{2}$ și metoda funcțione