13/12/21

Verifiable Secret Sharing

- Chi mi assicura che i voti di un docente siano nel range corretto? (se $0 \leq voto \leq 5$, e uno vota '11', come lo capisco?
- Come prevengo il Cheating?

Limitations, so far

- **Everybody must be honest!**
  - Dealer, Parties
  - Honest-but-curious model

attaccante che segue le regole ma viola il segreto!
⇒ attaccante non usa cheat (non è attive attack!)

Se attaccante usa cheats?

- **No way to check that protocol rules are followed**
  - i.e., detect/block cheaters

→ sono i dealer, possono dare shares non validi (danno più shares)
→ sono i player, possono esporre un fake share!

- **Need for secret sharing protocols where:**
  - Party can verify whether dealed share is consistent
    - → Detect malicious dealer
  - Parties can verify whether revealed share is consistent
    - → Detect cheating parties

QUANDO lo verifico??

Verifiable Secret Sharing

(verifico qualcosa che non è possibile dire!)

- **VSS**
  - First (non interactive) scheme proposed in 1987
    - Feldman VSS scheme
    - Several improvements since then

- **Frequently used as basis for distributed schemes**
  - More later on this ☺

- **Basic approach presented next**
  - Special case of Feldman VSS (using discrete logs)
  - Security limitations

Feldman scheme: dealer

➔ **Start with an ordinary Shamir scheme**

* (t, n) secret sharing

⇨ Generate random p(x) with degree (t-1)*, with P(0)=s

$$p(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-2} x^{t-2} + a_{t-1} x^{t-1}$$

⇨ Distribute one share to each of the n parties

$$(x_i, y_i) \quad y_i = p(x_i)$$

In practice, it is convenient to use x_i=i

➔ **Further step**

⇨ For each coefficient of the polynomial as well as secret, further broadcast (pubblicare in chiaro) → li "committo", e quindi non posso cambiare s' (anche se segreto, sennò cambio tutto!)

· leak info?

con large prime, e $s, \alpha_1, \alpha_2 \ldots$ → large field NO! (DLOG op.)

$$c_0 = g^s ; c_1 = g^{a_1} ; \cdots c_{t-1} = g^{a_{t-1}} \quad \text{mod } p$$

⇨ These are special type of "**commitments**" (NON CAMBIABILI, scelti attentamente!)

➔ Discrete log: hard to compute exponent

Ho creato dealer che fa bind <shares, pubblica commitments> ma non rileva i coeff. del polinomio!

What is a commitment?

COMMIT phase

voglio dare segreto senza mostrarlo subito!

a

"in un box"

COMM = Commit(a)

non posso cambiarlo

proprietà #1

Hiding:

Receiver should NOT learn anything about "a"

REVEAL phase

CANNOT cheat
"I sent a' ≠a"

(può essere il valore stesso!)

OPEN= verification key

proprietà #2

Binding:

Commitment CAN ONLY open to the value "a", no way for sender to cheat

Verification algorithm:
VERIF(COMM,OPEN,a) = YES/NO

!!

va bene SOLO se il segreto è preso da uno spazio grande, non deve essere ristretto!

; per capire: se ho pw super cifrata, ma essa è "123456" NON SONO SICURO.

$c=g^x$ is a commitment

mod p

spazio segreti piccolo

➔(computationally) Hiding:

⇨Given $c=g^x \bmod p$, computationally bounded receiver cannot know x

- $C=H(x)$ realizza hiding (H^{-1} difficile) ma solo computazionalmente (piccolo range, posso provarli!)
  il binding? solo computazionale! (posso trovare collisione: $H(x')=H(x)$). ⇒ g^x è meglio?

➔(perfectly) Binding:

g, p properly chosen

⇨Sender cannot find any other x' such that $g^{x'}=c$

porchè

➔Note: x' cannot be greater than group order

➔otherwise trivial (why? ☺)

Se ho $g^x \bmod p = 100441$ con $g=3$, $x=2345$, $p=104729$, riesco a trovare altri valori che lo generano? SI

uso $\phi(104729)=104728$, $\sim 3^{2345+104728\cdot 2} \bmod 104729 = 100441$, l'esponente = 211801 è valore committabile <=> x nel range $(1, p-1)$

Con Perfect Binding non trovo $1<x'<p-1$

Feldman scheme: verifier

➔ **Party i receives share (x_i,y_i)** ($y_i = p(x_i)$)

⇨ Other parties can do this, when party i reveals its share

⇨ Check that party i is honest

GB → $\langle x_{GB}, y_{GB} \rangle$

Verifier: (IDEA)
. prendo x_{GB}, faccio all'esponente homomorphic computation di $p(x_{GB})$ (non vedo l'addizione)
. vedo y_{GB}, è uguale a $g^{p(x_{GB})}$?

➔ **Verifies that share is valid (hence dealer honest)** uso "x_i" per computare.

⇨ by computing (mod p)

$$\underbrace{C_0 \cdot C_1^{x_i} \cdot C_2^{x_i^2} \cdot C_3^{x_i^3} \cdot \ldots \cdot C_{t-1}^{x_i^{t-1}}}_{\text{mod } p} =$$

$$= \left(g^{s}\right) \cdot \left(g^{a_1}\right)^{x_i} \cdot \left(g^{a_2}\right)^{x_i^2} \cdot \left(g^{a_3}\right)^{x_i^3} \cdot \ldots \cdot \left(g^{a_{t-1}}\right)^{x_i^{t-1}} =$$

$$= g^{s} \cdot g^{a_1 x_i} \cdot g^{a_2 x_i^2} \cdot g^{a_3 x_i^3} \cdot \ldots \cdot g^{a_{t-1} x_i^{t-1}} =$$

$$= g^{s + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \ldots + a_{t-1} x_i^{t-1}} = g^{p(x_i)}$$

($\text{mod } p-1 = \text{mod } q$ se $g \in QR$)

← computato da me, confrontato con lo share comunicato.

➔ **And checks that this is equal to** g^{y_i}

➔ Where y_i is the party's share

Caveats (in realtà NON funziona sempre!)

- **See exercise for**
  - Proper moduli (p,q)
  - C0 may leak info about secret (Legendre Symbol)
- **Aftermath:**
  - use p=2q+1
  - Use subgroup order q

An annoying issue…

- **Shamir Scheme is unconditionally secure** → $g^x \bmod p$
- **But Feldman's VSS is "only" computationally secure** (per via del binding)

- **What does this practically means?**
  - ⇨ $c_0=g^s$ leaks information about secret s!!
  - ⇨ If s is small (e.g. in [1…1000]) trivial "dictionary" attack (enumeration attack)
    - → For x:1…1000 repeat g^x until $g^x=c_0$ → s known!

- **Can we have perfectly hiding scheme?**

NON $\exists$ [binding Perfect + hiding Perfect], but $\exists$ commitment che fa perfect hiding e computational binding (Pedersen)

enumeration attack non funziona

Pedersen commitment

- **Feldman approach works because:**
  - $g^a \bmod p$ is a Commit(a)
  - And it is homomorphic:
    - Commit(a+b) = Commit(a) · Commit(b)
    - Indeed: $g^{a+b} = g^a \cdot g^b \bmod p$

≈ linear (ma NON è lineare, è più flessibile)

- **We need a perfectly hiding commitment scheme which is homomorphic…**
  - Found by Pedersen, 1991 - given g and h (public):
  - Commit(a; r) = $g^a \cdot h^r \bmod p$, funzione di 'a' (segreto) ne' 'r'

truly random di solito — basi NOTE

double DLOG construction

homomorphic → $Commit(a+b; r_a + r_b) = g^{a+b} h^{r_a + r_b} =$

perfectly hiding

$= g^a h^{r_a} \cdot g^b h^{r_b} = Commit(a, r_a) \cdot Commit(b, r_b)$

∀ 'a' trovo 'r' : trovo c, il pair è facile se conosco legame(g,h), semmai è hard → parametri large prime → computational binding, se ho tempo / conosco legame(g,h) lo rompo in termini di binding

Pedersen commitment

→ dove contenere tutto il range iniziale

➔ **(perfectly) Hiding:**

⇨ $c=g^a h^r \bmod p$ may be a commitment for **any possible value**!

➔ Formally, for any a'≠a, we can find unique r' such that

$$commit(a';r')=g^{a'}h^{r'} = g^a h^r = commit(a;r)$$

esempio:
a = KKK a' = QQQ
r = 34 r' = 137
$g^a \cdot h^r = 1345 = g^{a'} \cdot h^{r'}$
se vedo lui, non so se nascondo KKK o QQQ !

➔ **(computationally) Binding:**

⇨ Sender should NOT be able to find such an a' (and related r')!

⇨ Which is the case if sender does **NOT** know $\log_g h$

➔ a trapdoor committment scheme!

↳ finché è difficile calcolare $w = \log_g h$ allora è robusto, se la trovo rompo tutto!

➔ **What about a perfectly hiding + perfectly binding committment?**

⇨ Impossible, you may have either one or another, **not both**

Se voglio PRIVACY non uso hash ma commitment, che è meglio!!

Proof (base of)

let $h = g^w$ i.e. $w = \log_g h$, non lo conosco, MA deve esserci!

we know a, r, we are given a', and we look for r' s.t.

$$g^a h^r = g^{a'} h^{r'} \Rightarrow g^a g^{wr} = g^{a'} g^{wr'} \Rightarrow$$

$$\Rightarrow g^{a+wr} = g^{a'+wr'} \Rightarrow$$

$$\Rightarrow a + wr = a' + wr' \quad \langle \mathrm{mod}\, q \text{ (group order, primes.t. } 2q + 1 = p) \rangle \Rightarrow$$

$$\Rightarrow r' = w^{-1}(a - a' + wr) \quad \text{mod 'qualcosa'}$$

SKIP

i.e., given w, we can compute ANY "colliding" committment, hence we would not commit to anything!! (no binding)

Pedersen VSS - dealer

(Feldmann Scheme)

commitment (s, r)

- (t, n), s secret,
- $\alpha_1, \alpha_2 \ldots$ random
- s può essere SMALL, altri NO!
- uso LARGE MODULUS

➔Generate TWO random polynomials

- uno per hide segreto $f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-2} x^{t-2} + a_{t-1} x^{t-1}$
- uno per hide Key commitment $f'(x) = r + b_1 x + b_2 x^2 + \cdots + b_{t-2} x^{t-2} + b_{t-1} x^{t-1}$

$b_1, b_2, \ldots$ random

➔Give party i "double" share (x_i, y_i, z_i)

Shamir Share (coordinata x_i) $y_i = f(x_i) = s + a_1 x_i + a_2 x_i^2 + \cdots + a_{t-1} x_i^{t-1}$

Share random Key commitment $z_i = f'(x_i) = r + b_1 x_i + b_2 x_i^2 + \cdots + b_{t-1} x_i^{t-1}$

➔Publish commitments

$c_0 = g^s h^r$ ← Pedersen commitment

$c_1 = g^{a_1} h^{b_1}$

...

$c_{t-1} = g^{a_{t-1}} h^{b_{t-1}}$

non devo conoscere relazione g, h, ma non devono per forza essere grandi!

$h = g^{(w)}$ non devo rivelarlo

anche per lui uso p strong prime = 2q+1
e per f(x) e f'(x) mod q,
con g, h elementi subgroup

Pedersen VSS - verifier

➔ **Party i receives share (x_i, y_i, z_i)**

➔ **Verifies (mod p) that:**

x_i value; y_i share of secret; z_i share of commit

$$c_0 \cdot c_1^{x_i} \cdot c_2^{x_i^2} \cdot \ldots \cdot c_{t-1}^{x_i^{t-1}} = \text{come Feldmann!! (homomorphic proprierty)}$$

$$= \left(g^s h^r\right) \cdot \left(g^{a_1} h^{b_1}\right)^{x_i} \cdot \left(g^{a_2} h^{b_2}\right)^{x_i^2} \cdot \ldots \cdot \left(g^{a_{t-1}} h^{b_{t-1}}\right)^{x_i^{t-1}} =$$

$$= g^s \cdot g^{a_1 x_i} \cdot g^{a_2 x_i^2} \cdot \ldots \cdot g^{a_{t-1} x_i^{t-1}} \cdot h^r \cdot h^{b_1 x_i} \cdot h^{b_2 x_i^2} \cdot \ldots \cdot h^{b_{t-1} x_i^{t-1}} =$$

$$= g^{\underbrace{s + a_1 x_i + a_2 x_i^2 + \ldots + a_{t-1} x_i^{t-1}}_{\text{polynomial per } x_i}} \cdot h^{\underbrace{r + b_1 x_i + b_2 x_i^2 + \ldots + b_{t-1} x_i^{t-1}}_{\text{polynomial per } x_i}} =$$

$$= g^{y_i} h^{z_i} \text{ (calcolati)}$$

$$\| \quad g^{y_i z_i} \text{ (dati all'inizio)}$$

esempio

$p = 83$, $g = 5$

- $5^x \bmod 83 = \{1, 2, 3, 4, 5, 6, 7, 8, \ldots 79, 80, 81, 82\}$ ← SPAWN di tutti i valori!!

generator of the group

- definisco $s = 16$, $a_1 = 23$, $a_2 = 13$ → $f(x) = [s + a_1 x + a_2 x^2] \bmod 83$
- $f(15) = 49$, $f(16) = 60$, $f(17) = 14$ dove 15, 16, 17 sono i partecipanti (non per forza 1, 2, 3!)
- calcolo Λ coefficient $\Lambda_{15} = 53$, $\Lambda_{16} = 77$, $\Lambda_{17} = 37$, ovviamente $(\Lambda_{15} \cdot y(15) + \Lambda_{16} \cdot y(16) + \Lambda_{17} \cdot y(17)) \bmod 83 = s$
- calcolo i commitments:

$C_0 = g^s \bmod p = 65$; $C_1 = g^{a_1} \bmod p = 19$; $C_2 = g^{a_2} \bmod p = 47$

- per $x = 15$, $y = 49$ → $g^{49} \bmod p$, verifico se è corretta!

$(C_0 \cdot C_1^x \cdot C_2^{x^2}) \bmod p \rightarrow 21$, come è possibile? è diverso da 49.

Feldman verifier rotto!?

Spiegazione

Nella slide "Feldman Verifier" noto che:

$C_0 \cdot C_1^{x_1} \cdot \ldots$ e $g^{P(x_i)}$ è mod p, mentre la ricostruzione $g^{s + a_1 x_1 \ldots}$ non è mod p, bensì mod $\phi(p)$ (come RSA!) ← homomorphic computation

Prime for dummies

- group $(G, \circ)$ ≐ set di elementi (anche numeri) con "$\circ$" gruppo di operazioni, esso ha 4 proprietà:
  - chiusa: $g_1, g_2 \in G \Rightarrow g_1 \circ g_2 \in G$
  - identità: $\exists$ 'I': $g_i \circ I = g_i \ \forall g_i$
  - inverso: $\forall g \ \exists \ g^{-1}: \ g \cdot g^{-1} = I$
  - associativa: $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$

; se commutativa ⇓ GRUPPO ABELIANO

The Z_p^* group

p ← integer mod p

$*$ ← multiplicative group (NO +, NO −)

- elementi: $\{1, \dots, p-1\}$, sono "p-1", 0 escluso (no inverso), gruppo finito
- proprietà:
  - chiusura: ✓ (ho modulo p)
  - associatività: ✓
  - identity: ✓ (ho "1")
  - commutativa: ✓

  GRUPPO ABELIANO
  - inverso: · x ha inverso $\iff$ $\gcd(x,N)=1$ (COPRIMI), allora $\exists\ x^{-1}: x \cdot x^{-1} \bmod n = 1$
    · se $N = p$ prime → tutti hanno inverso

esempio: Z_{11}^*, 11 prime, #elementi $= 10 = p-1$ $\{1,2,3,4,5,6,7,8,9,10\}$

- ho inversi: $1 \to 1$ $2 \to 6$ $3 \to 4$ $4 \to 3$ ... $10 \to 10 \iff (p-1)(p-1) \bmod p = 1$
- per gruppi ampi extended euclidean algorithm

Nota bene: $x^Y = \underbrace{x \circ x \circ x \circ x \dots}_{Y \text{ volte}}$, allora ho anche exponentiation. Posso generare il gruppo con questa operazione?

$\exists\ g: \{g^0, g^1, g^2 \dots g^{m-1}\}$? Se "m" = prime, ogni membro è generatore (tranne l'identità)

Problema: Z_p^* e p PRIME → $|Z_p^*| = p-1$ NON PRIME, non è il nostro caso!

esempio: in Z_{11}^* ho generatori: $g=2$ $\{2,4,8,5,10,9,7,3,6,1\}$ ✓ $g=3$ $\{3,9,5,4,1,3,9,5,4,1\}$ ✗ (SOTTOGRUPPO ordine 5)

$g=10$ ha sottogruppo ordine 2! ✗ #attenzione: $10 = 5 \cdot 2$, cioè ordini 2 o 5, NO 3 o 4!

ogni "g" è un generatore, ma può generare un sottogruppo di cardinalità = fattore di $|G|$. Tutti sottogruppi sono CICLICI

STRONG PRIMES

Se $p = 101$ $\#Z_{101}^* = 100 = 2 \cdot 5 \cdot 5 \cdot 2$, dipende da "g" !! Uso STRONG PRIME: $p = 2q+1$: p è prime, q anche; "p-1" $= 2q$ PARI

esempio: $p = 83$, $p-1 = 82 = 2 \cdot 41$, 41 è prime, 83 anche .... anche Z_{11}^* era STRONG!

Con STRONG prime: $\forall x \in G$ tranne "1" e "p-1" genero l'intero gruppo o sottogruppo ordine "q".

Se p, q large ⇒ gruppi LARGE (ne uso solo large p → p-1 fattorizzabile in piccoli numeri)

Quadratic residue subgroup

Se $g=6 \in G$, posso scriverlo come $x \cdot x = 6 \bmod p$? Se $p=11$ NO: $2\cdot 2=4$, $3\cdot 3=9$, $4\cdot 4=5$... i valori esprimibili come xx creano Sottogruppi!

esempio $2\cdot 2=4 \rightarrow g=4$ genera $\{4,5,9,3,1,4,5,9,3,1\}$ sottogruppo ordine 5, 4 fa parte di Quadratic residue subgroup

In questo gruppo valgono le proprietà dei gruppi, di ordine $\frac{p-1}{2}=9$ (mapping $2\rightarrow 1$), cioè è Prime order group

Per sapere se $g \in QR$ vedo se $g^{\frac{p-1}{2}} \bmod p = 1$, se $g \notin QR$ $g^{\frac{p-1}{2}} \bmod p = "-1"$.

esempio: $p=11$, $\frac{p-1}{2}=5$, $g=5 \in QR$? $5^5 \bmod 11 = 1$! "$g=5$" genera $\{5,3,4,9,1,5,3..\}$

$g=2 \in QR$? $2^5 \bmod 11 = 10$ NO! $g=2$ genera: $\{2,4,8,5,10...\}$ (1° 2° 3° 4° 5°)

Quadratic residue subgroup

- $x \in Z_p^*$ **is a quadratic residue if it admits square root in** Z_p^*
  - ⇨ i.e., there exists a such that $a^2 \bmod p = x$
- **QR form subgroup of order** $\frac{p-1}{2}$
  - ⇨ 2→1 mapping: $x \rightarrow x^2$, $p-x \rightarrow x^2$ Indeed, $(p-x)^2 \bmod p = p^2 - 2px + x^2 \bmod p = x^2$
- **QR test: legendre symbol**
  - ⇨ $a \in QR$ if $a^{\frac{p-1}{2}} \bmod p = 1$ (otherwise -1)
- **Example for** Z_{11}^***:** $QR_{11} = \{1,3,4,5,9\}$
  - ⇨ If p strong prime, QR_p has prime order q!

Giuseppe Bianchi

Riprendo l'esempio con $p=83$ (STRONG PRIME) e $g=5$ (generator?) PROVO:

$5^{\frac{83-1}{2}} \bmod 83 = 82 \doteq -1 \rightarrow$ è GENERATOR. Se faccio $(5^2)^{\frac{83-1}{2}} \bmod 83$ genererà QRG (poiché $25=5\cdot 5$), la cardinalità del sottogruppo è PRIME NUMBER

Se g è quadratic residue (come 5) $\Rightarrow$ ordine subgroup è PRIME.

Nell'esempio di Feldman: cosa uso come modulo? p Prime, $q=\frac{p-1}{2}$ prime, uso modulo q (usato all'esponente), alla base ho modulo p

ma invece di prendere $G \in$ generator group, $G \in QR$ (sottogruppo) $\Rightarrow$ nell'esempio: $p=83$, $q=41$ entrambi prime $\rightarrow f(x)=[s+a_1x+a_2x^2] \bmod q$ (tra 1 e 40)

$f[7]=35$, $f[19]=21$, $f[22]=8$ Shares $(\bmod q)$

- calcolo Λ coefficient $\Lambda_7=21$, $\Lambda_{19}=39$, $\Lambda_{22}=23$, ovviamente $(\Lambda_7 \cdot y(7) + \Lambda_{19} \cdot y(19) + \Lambda_{22} \cdot y(22)) \bmod 41 = s$ ($q=41$)
- calcolo i Commitments: (uso $g=5$ ok, ma $g=25$ (SQUARE): $25^9 \bmod p = 1$ crea 'sottogruppo', più preciso!, il range riparte ad un certo punto)

$C_0 = g^s \bmod p = 65$; $C_1 = g^{a_1} \bmod p = 29$; $C_2 = g^{a_2} \bmod p = 51$

esempio $[C_0 \cdot C_1^x \cdot C_2^{x^2}] \bmod p$

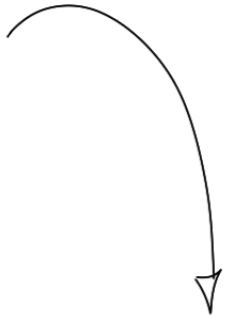
- $x=7 \rightarrow 16 = 25^{f(7)} \bmod p$
- $x=19 \rightarrow 78 = 25^{f(19)} \bmod p$
- $x=22 \rightarrow 65 = 25^{f(22)=8} \bmod p$

ora va bene!!

Sappiamo che Pedersen of finché non è noto w, infatti $g^w = h \rightarrow w = \log_g(h)$

- ∃ schemi con Public Key $= g^x$, $SK = x$, ∃ casi in cui 'x' segreta ma deve essere ricostruibile, con g, h NOTI, w NON NOTA ma ricostruibile.

Come la realizzo?



Distributed Key Generation

by Pedersen

Question

➔ **Dlog-based cryptosystems**

- ⇨ **Private key: x; Public key: g^x**
- ⇨ Many important cases: El Gamal, Identity Based Encryption, …

➔ **Can we generate (Pub_K, $Priv_K$) pair s.t. all know Pub_K but NOBODY knows $Priv_K$?**

➔ **Useful when:**

- ⇨ $Priv_K$ to be revealed later on, when needed
  - →E.g. encrypt message than noone can open, unless special conditions occur
- ⇨ $Priv_K$ NEVER revealed, but functions of it are needed
  - →This is extremely important in IBE (more later)

➔ **Other scenarios**

- ⇨ e.g. generate $h=g^w$ in pedersen VSS
- ⇨ And many and many other creative constructions

Answer: yes!

➔**Schemes called DKG**

- ⇨Distributed Key Generation
- ⇨Fully distributed, NO authority or TTP

➔**Basic one: Pedersen DKG**

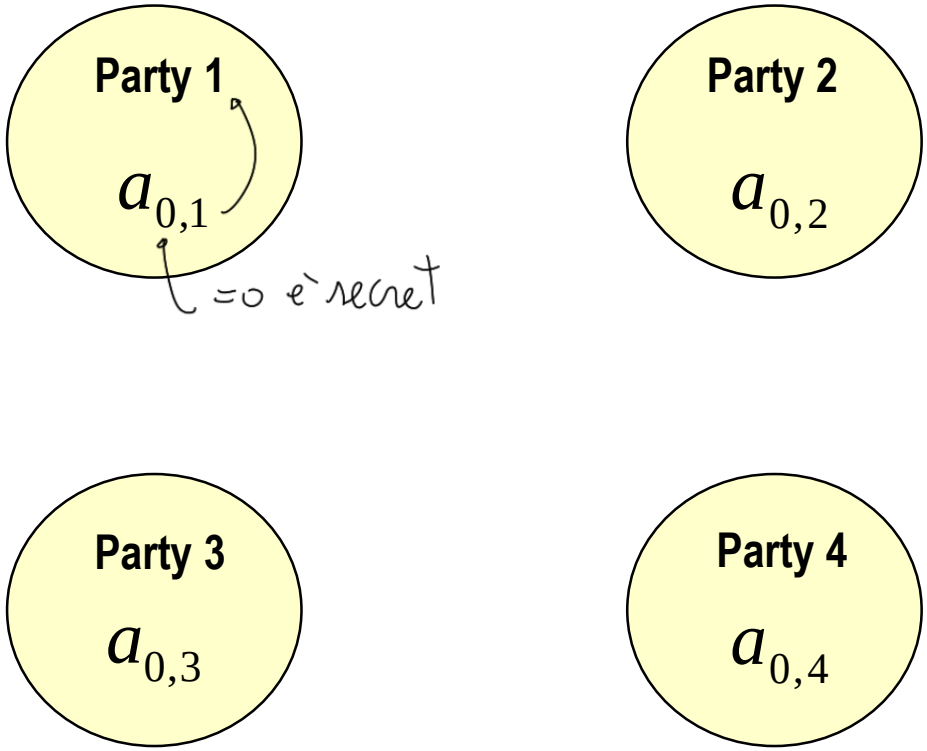
- ⇨1991
  - ➔"simple" application of what we saw so far
- ⇨improved and made more secure when cheating parties
  - ➔Gennaro++ (various works from 1999 to 2007)
- ⇨We stick to basic approach
  - ➔Limitations are quite subtle even with cheating parties
  - ➔Pedersen DKG still OK in several practical applications

(IDEA)

Best seen by example (3,4)

Step 1: each party independently generates random secret $a_{0,i}$

4 persone,
- Voglio (x, g^x) (x: confidential finché 3 su 4 persone vogliono ricostruirlo; g^x: public)
- No dealer !!

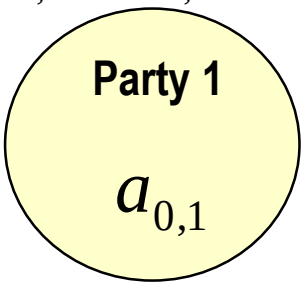


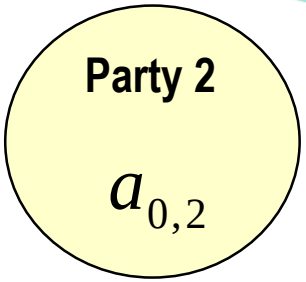
Best seen by example (3,4)

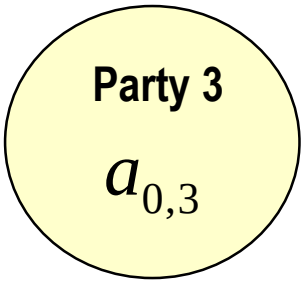
Step 2: generates random polynomial of degree (t-1) with known term $a_{0,i}$

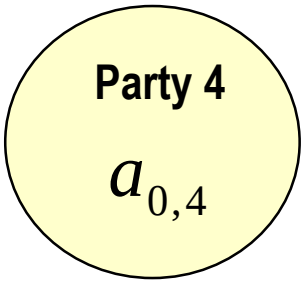
$$p_1(x) = a_{0,1} + a_{1,1}x + a_{2,1}x^2$$

$$p_2(x) = a_{0,2} + a_{1,2}x + a_{2,2}x^2$$







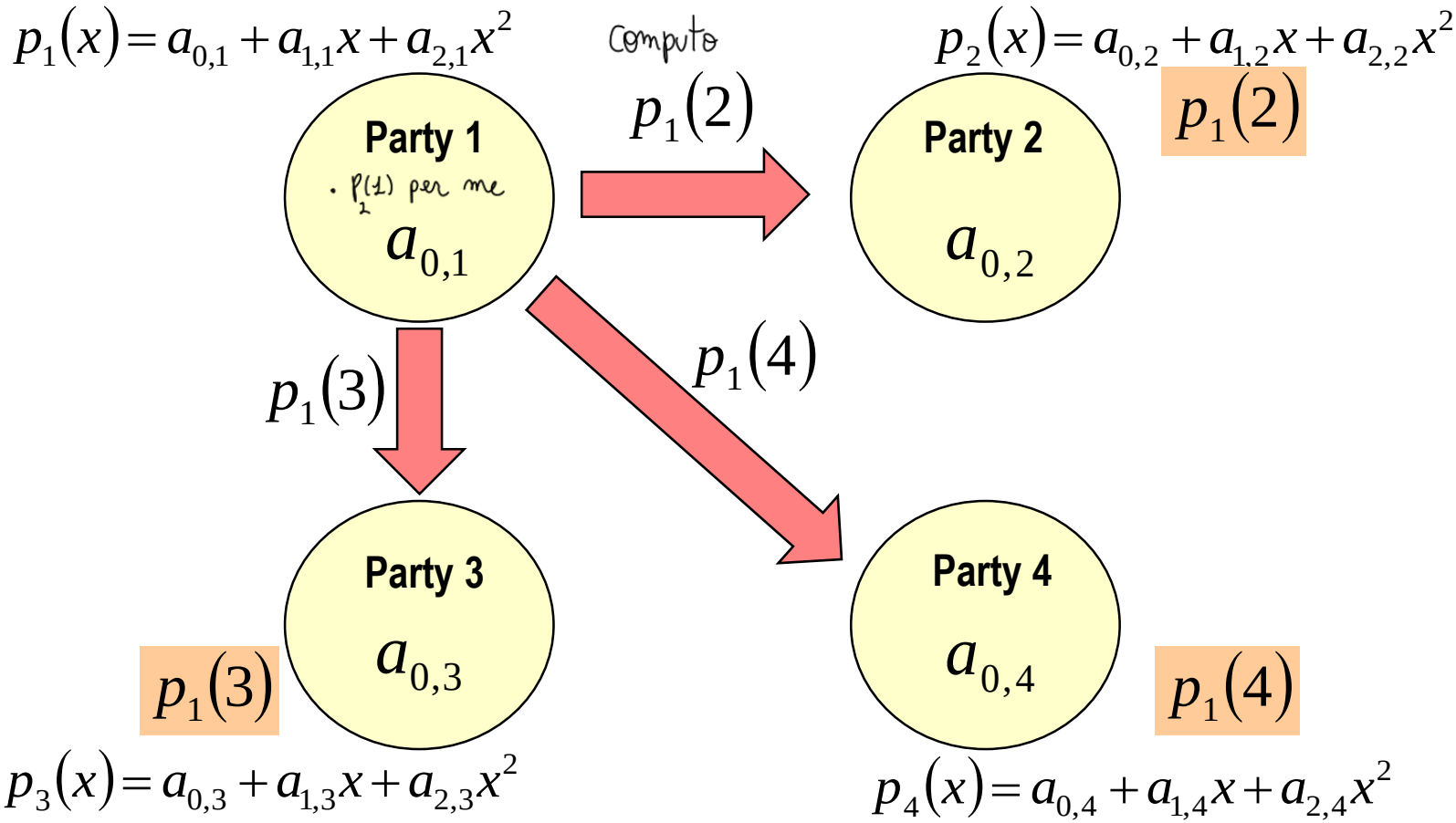


$$p_3(x) = a_{0,3} + a_{1,3}x + a_{2,3}x^2$$

$$p_4(x) = a_{0,4} + a_{1,4}x + a_{2,4}x^2$$

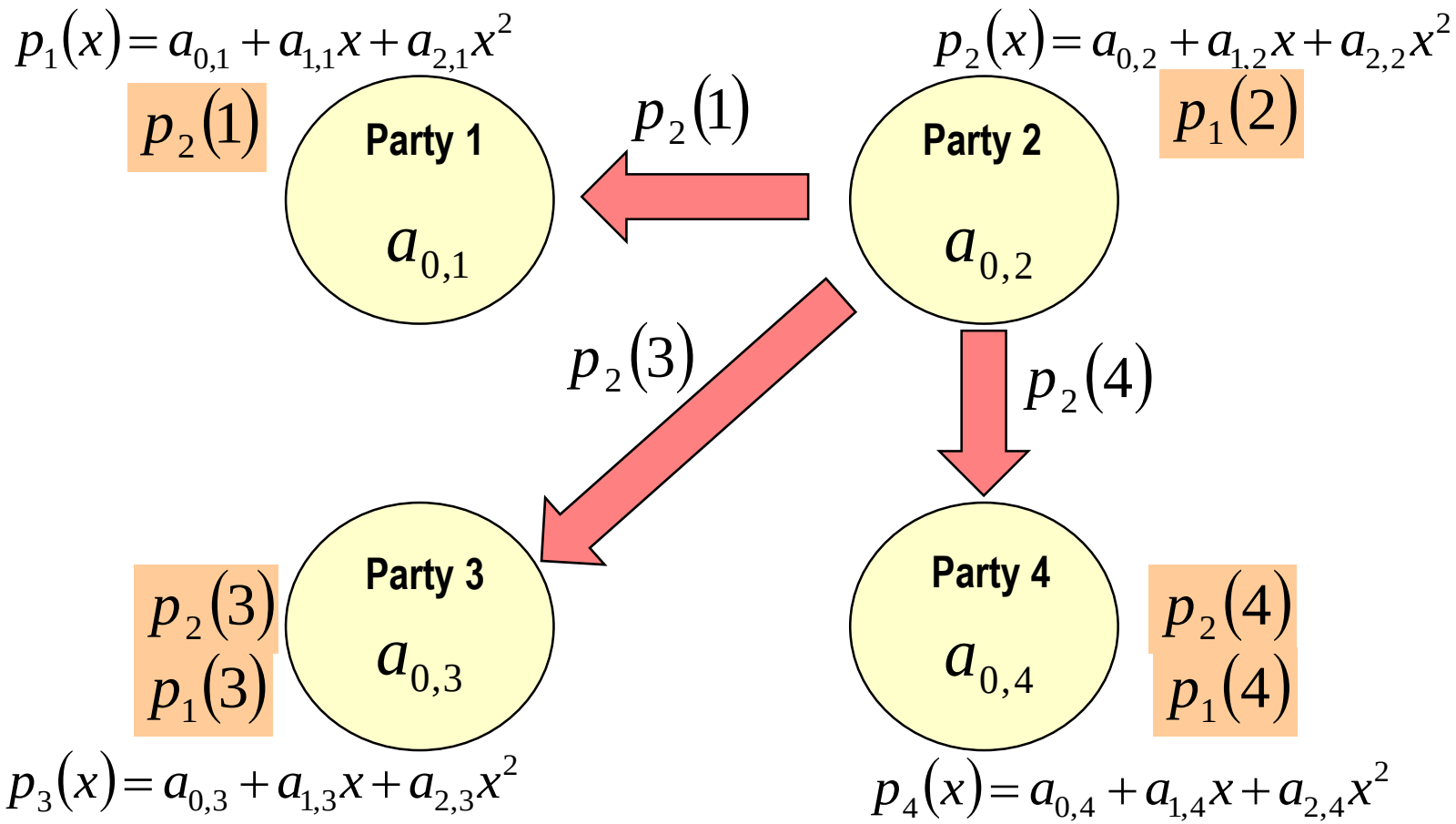
Best seen by example (3,4)

Step 3: securely exchanges share of own poly with every other party



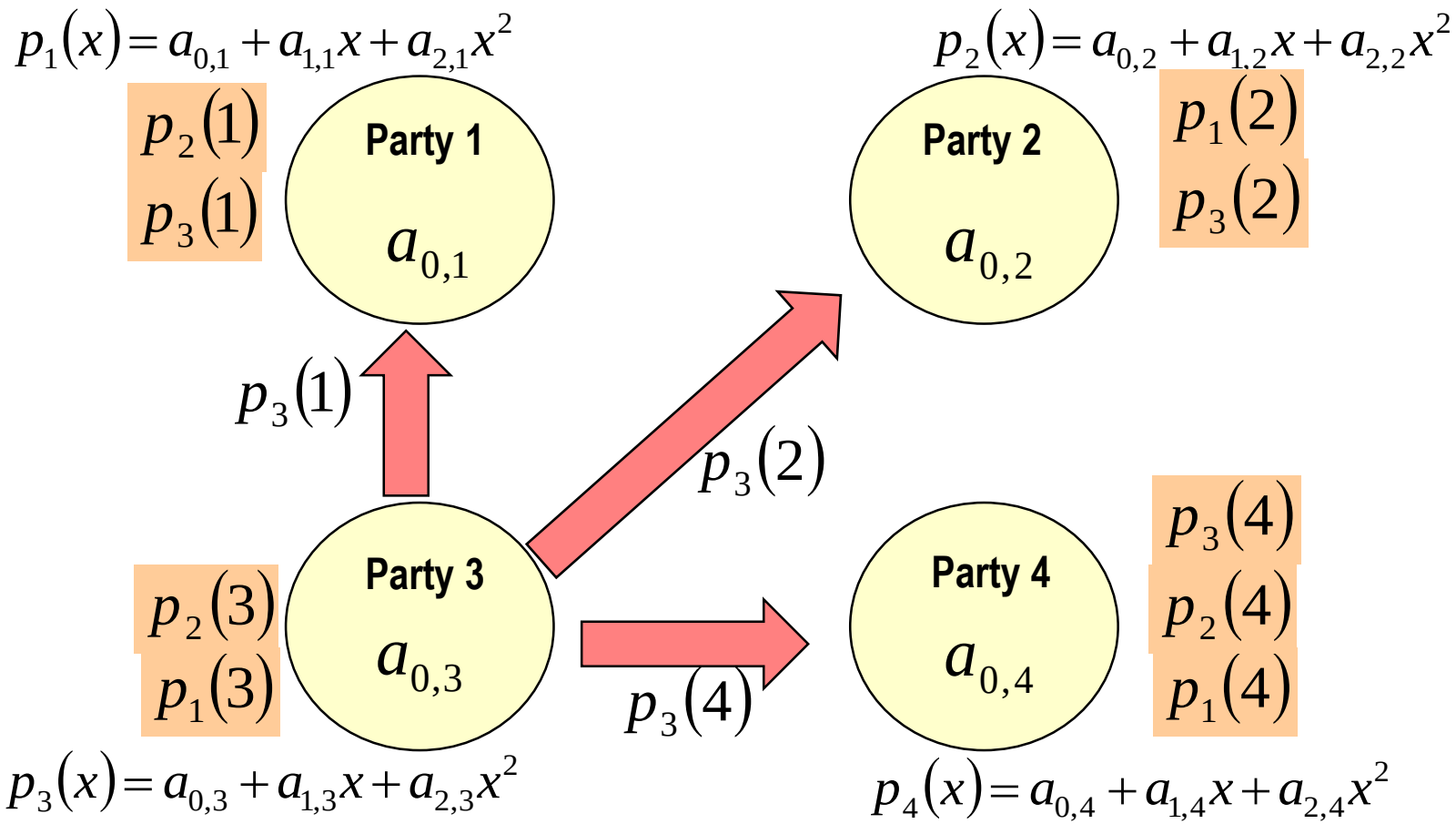
Best seen by example (3,4)

Step 3: securely exchanges share of own poly with every other party



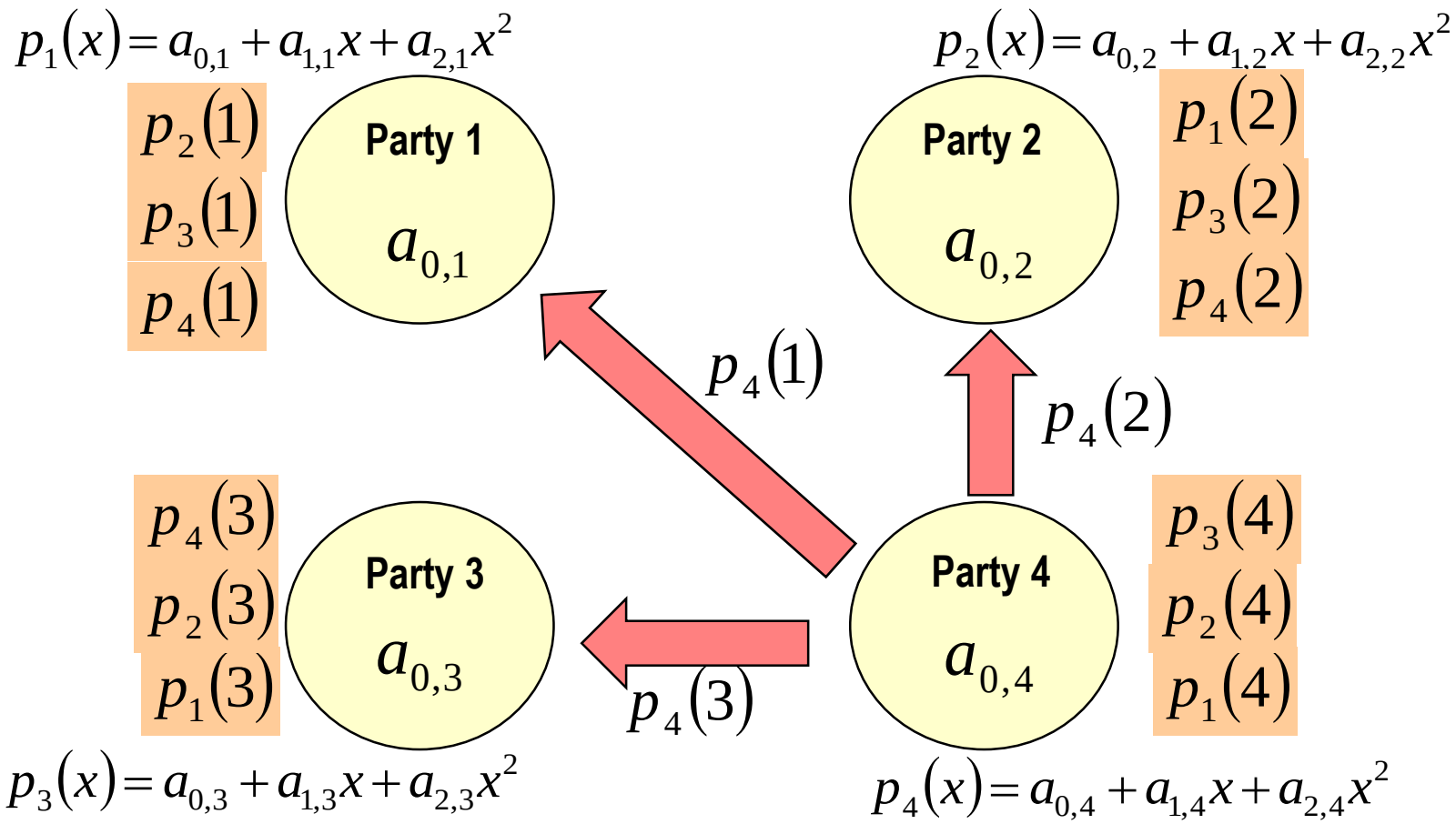
Best seen by example (3,4)

Step 3: securely exchanges share of own poly with every other party



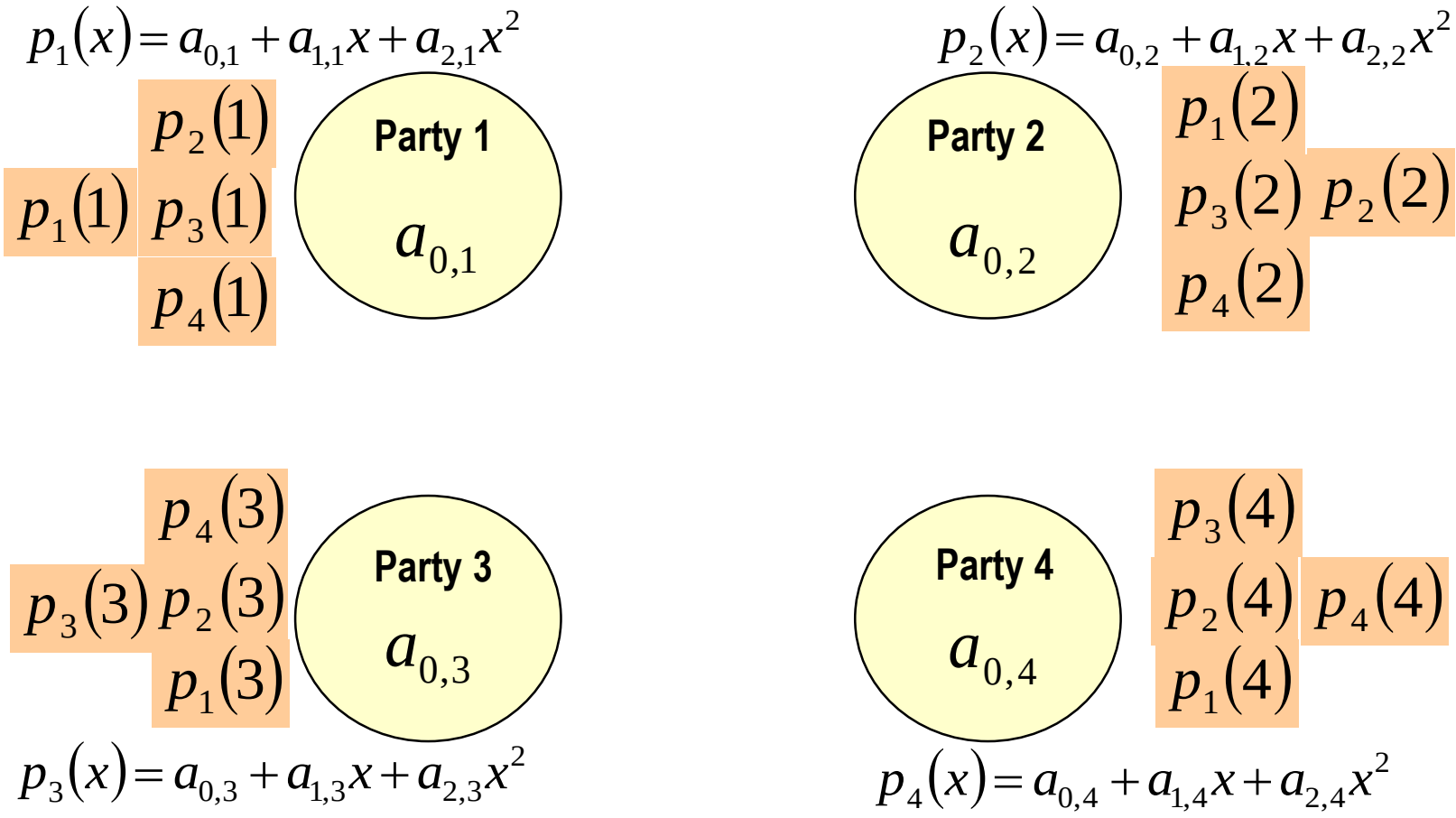
Best seen by example (3,4)

Step 3: securely exchanges share of own poly with every other party



Best seen by example (3,4)

Step 4: each party computes "self" share



Best seen by example (3,4)

share of the sum dei sub-secret $a_{01}+a_{02}+a_{03}+a_{04}$ (che NON conosco)

Step 5: each party computes "full" share (and keeps it)

$$y_1 = p_1(1)+p_2(1)+p_3(1)+p_4(1)$$

$$p_1(x)=a_{0,1}+a_{1,1}x+a_{2,1}x^2$$

(homomorphic proprierty)

sum of shares ≡ shares of the sum

Party 1

$a_{0,1}$

$$y_2 = p_1(2)+p_2(2)+p_3(2)+p_4(2)$$

$$p_2(x)=a_{0,2}+a_{1,2}x+a_{2,2}x^2$$

Party 2

$a_{0,2}$

Chiamo $S = a_{01}+a_{02}+a_{03}+a_{04}$

$y_1\ y_2\ y_3\ y_4$ sono shares di S

Con 3 y_i faccio interpolazione Lagrange etc...

Party 3

$a_{0,3}$

$$p_3(x)=a_{0,3}+a_{1,3}x+a_{2,3}x^2$$

$$y_3 = p_1(3)+p_2(3)+p_3(3)+p_4(3)$$

Party 4

$a_{0,4}$

$$p_4(x)=a_{0,4}+a_{1,4}x+a_{2,4}x^2$$

$$y_4 = p_1(4)+p_2(4)+p_3(4)+p_4(4)$$

Posso fare g^s? voglio ($s_{nascosto}$, $g^s_{pubblico}$)

Best seen by example (3,4)

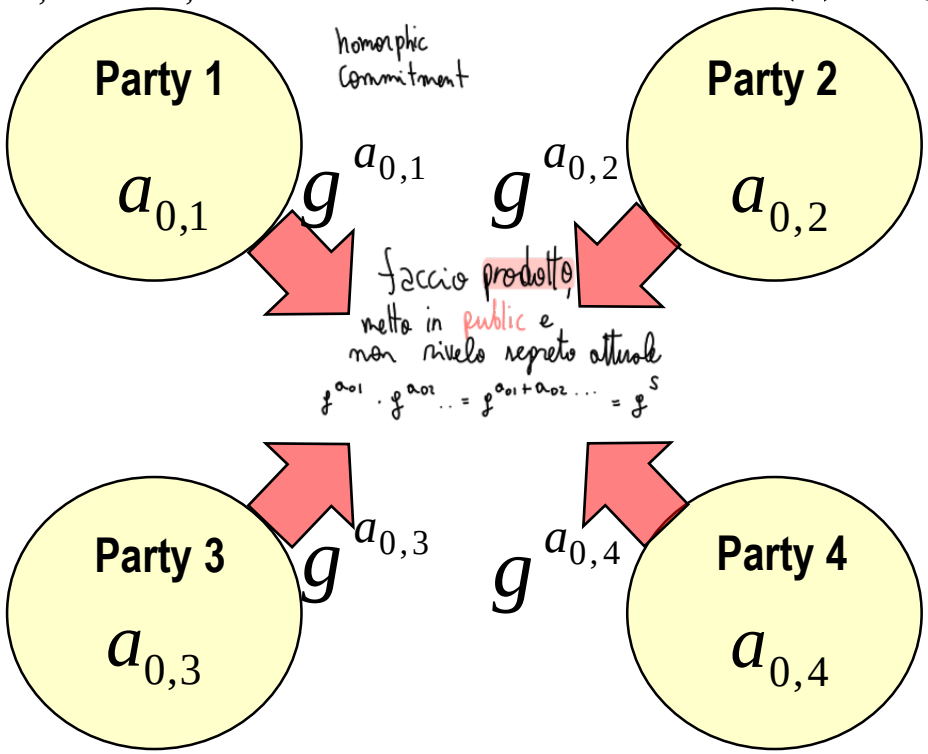
Step 6: …and broadcasts committment for $a_{0,i}$ (if all commitments, full Feldman VSS scheme)

$$y_1 = p_1(1) + p_2(1) + p_3(1) + p_4(1)$$

$$p_1(x) = a_{0,1} + a_{1,1}x + a_{2,1}x^2$$

$$y_2 = p_1(2) + p_2(2) + p_3(2) + p_4(2)$$

$$p_2(x) = a_{0,2} + a_{1,2}x + a_{2,2}x^2$$



$$p_3(x) = a_{0,3} + a_{1,3}x + a_{2,3}x^2$$

$$y_3 = p_1(3) + p_2(3) + p_3(3) + p_4(3)$$

$$p_4(x) = a_{0,4} + a_{1,4}x + a_{2,4}x^2$$

$$y_4 = p_1(4) + p_2(4) + p_3(4) + p_4(4)$$

Best seen by example (3,4)

Wrap up: what we have accomplished so far?

We have distributely built a random polynomial P(x) where a random S is unknown to ALL…

generati da diversi player

$$\begin{aligned} p_1(x) &= a_{0,1} + a_{1,1}x + a_{2,1}x^2 \\ p_2(x) &= a_{0,2} + a_{1,2}x + a_{2,2}x^2 \\ p_3(x) &= a_{0,3} + a_{1,3}x + a_{2,3}x^2 \\ p_4(x) &= a_{0,4} + a_{1,4}x + a_{2,4}x^2 \end{aligned}$$

$$\sum \quad P(x) = s + \alpha_1 x + \alpha_2 x^2$$

non servono, basta shares

Being:

$$s = \sum_{i=1}^{4} a_{0,i}$$

$$\alpha_1 = \sum_{i=1}^{4} a_{1,i}$$

$$\alpha_2 = \sum_{i=1}^{4} a_{2,i}$$

… but where each party has a valid SHARE of such random secret S…

$$y_i = p_1(i) + p_2(i) + p_3(i) + p_4(i) = P(i)$$

… and (our target!) where each party can compute g^s using the broadcast committments!

$$g^{a_{0,1}} \cdot g^{a_{0,2}} \cdot g^{a_{0,3}} \cdot g^{a_{0,4}} = g^{a_{0,1}+a_{0,2}+a_{0,3}+a_{0,4}} = g^s$$