1° problem:

Secret Sharing
(core constructions and techniques)

Trivial Secret Sharing

("parole")

Goal

dealer:
crea il segreto

prob = $\frac{1}{256}$

Secret: 00101101

• potrei dividerlo in 2 parti?

How to share a secret among two persons such that it is revealed if BOTH reveal their part?

split segreto tra 2 persone

Very first idea

Secret: 0010.1101

Alex: 0010----

1/16

Bob: ----1101

ho indebolito il segreto

Wise idea???

Bad idea!

Which probability to guess secret right?

- before seeing anything : 1/256
- after seeing one share : 1/16

Secret WEAKENED when one share revealed

Alex: 0010----

Bob: ----1101

Unavoidable?

Can we do better?

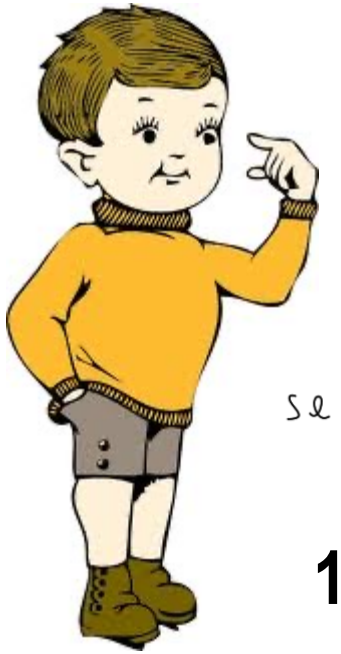
A better solution

Secret: 0010.1101

Generate random sequence, ex. Key: 1011.0100
(truly random)

XOR Secret & random one time pad : 1001.1001

≈ Vernam Cipher



Give alex the random sequence 1011.0100

se lo rubo? è random, svantaggi!

Give Bob the XOR value 1001.1001

se rubo lui!? UGUALE! (perfect secrecy)

Secret = Ashare XOR Bshare

1011.0100 ⊕ 1001.1001 = 0010.1101

Is this secure? Solo con entrambi

Security / A

Eve gets to know Alex' SHARE:
NO INFORMATION ON SECRET REVEALED

Alex share is a random sequence!
No secret included in his share!!

Alex: 1011.0100

Divido l'informazione in SHARES,
solo insieme rivelano il segreto.

Security / B

Eve gets to know Bob' SHARE:

Bob share is an XOR between random sequence and the secret…

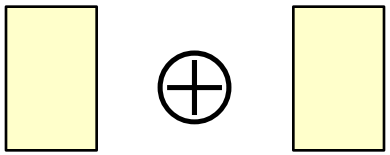
Bob:1001.1001

Does this leak information?

Perfect secrecy

➔Of course no information leaked

⇨Remember one time pad…



Secret bit

Probability:
0=p
1=(1-p)

Random bit

Probability:
0=1/2
1=1/2

Secret bit	Random bit	XOR result bit	
0	0	0	p/2
0	1	1	p/2
1	0	1	(1-p)/2
1	1	0	(1-p)/2

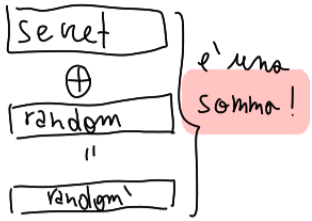
Probability to guess secret after seeing ONE share is the same as a guess without seeing any share

XOR not strictly needed: Can use (modular) sums

SECRET $0010.1101_2 \rightarrow 45_{10}$

RAND mod 256: (0,255) (poiché vorrei stare in:) 180

S – RAND mod 256: 121



$P(S=45 \mid S \ominus R=121) \equiv P(D=45)$

S⊖R è uguale!

180

121

"Se range (0, 99) non posso prendere 100 poiché NON PRIME, ma prendere il large prime più piccolo, 101, e il sistema rimane uncond. secure! (Real Shamir scheme)"

Secret = Ashare + Bshare

180+121 mod 256 = 45

Module not necessary, but practical (e.g. 32 bit numbers)

180 + 121 = 301, 301 - 256 = 45

Trivially extends to n parties

S → 45

R1 → 135

R2 → 6

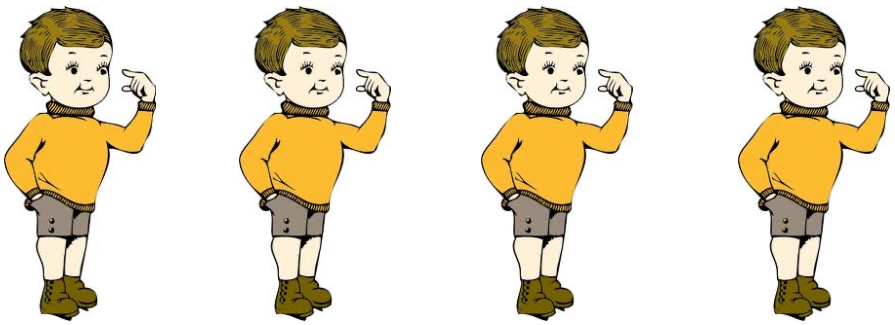
R3 → 67

quantità truly-random

|S-R1-R2-R3| → 93] 4° valore

R_1 = 135 R_2 = 6 R_3 = 67 $S-R_1-R_2-R_3$ = 93

↳ senza modulo leak info (se $< \emptyset$ allora $S < R_1+R_2+R_3$)



Reconstruction:

$(135+6+67+93) \bmod 256 = 45$

Conoscere ($S-R_1-R_2-R_3$, R_1 e R_2 $\Rightarrow$ $S-R_3$, $\emptyset$ information poiché non ottengo 'S' (TRIVIAL SECRET SHARING)

Perfect secrecy up to n-1 shares revealed

adversary knowing n-1 shares has still (initial!) 1/256 prob to guess secret

Shamir Secret Sharing

trivial secret share

persone che servono per rivelare il segreto!

parti in cui è diviso (eccetto deale)

Further goal

➔(n,n) secret sharing scheme

- ⇨Secret revealed only if ALL n parties provide a share
- ⇨The previous trivial scheme

➔(t,n) secret sharing scheme

- ⇨Secret revealed when ANY t OUT of n parties provide a share
  - ➔$1 \leq t \leq n$ (caso interessante)
  - ➔If t=1, every partner has secret (trivial case)
  - ➔ If $t = n$, torno al caso precedente

utile perché se uno delle persone aventi uno "shares" non è disponibile non posso ricostruire 's'.

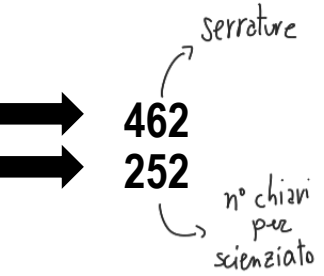
se t=1 basta una persona per risolvere il segreto, cioè tutti hanno il segreto e basta prendere una persona.

Why (t,n) schemes?

Start of Shamir's paper *- Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present.*

What is the smallest number of locks needed? → **462**

What is the smallest number of keys to the locks each scientist must carry? → **252**



➔ Compelling real world scenarios

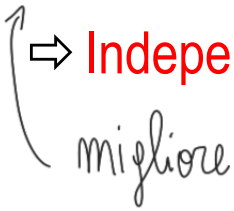
⇨ two-out-of-tree nuclear weapon control in Russia in the early 1990s

➔President, Defense Minister, and Defense Ministry [Beimel 2010].

➔Basic building block in several cryptographic constructions

➔ Shamir 1979, Blakley 1979

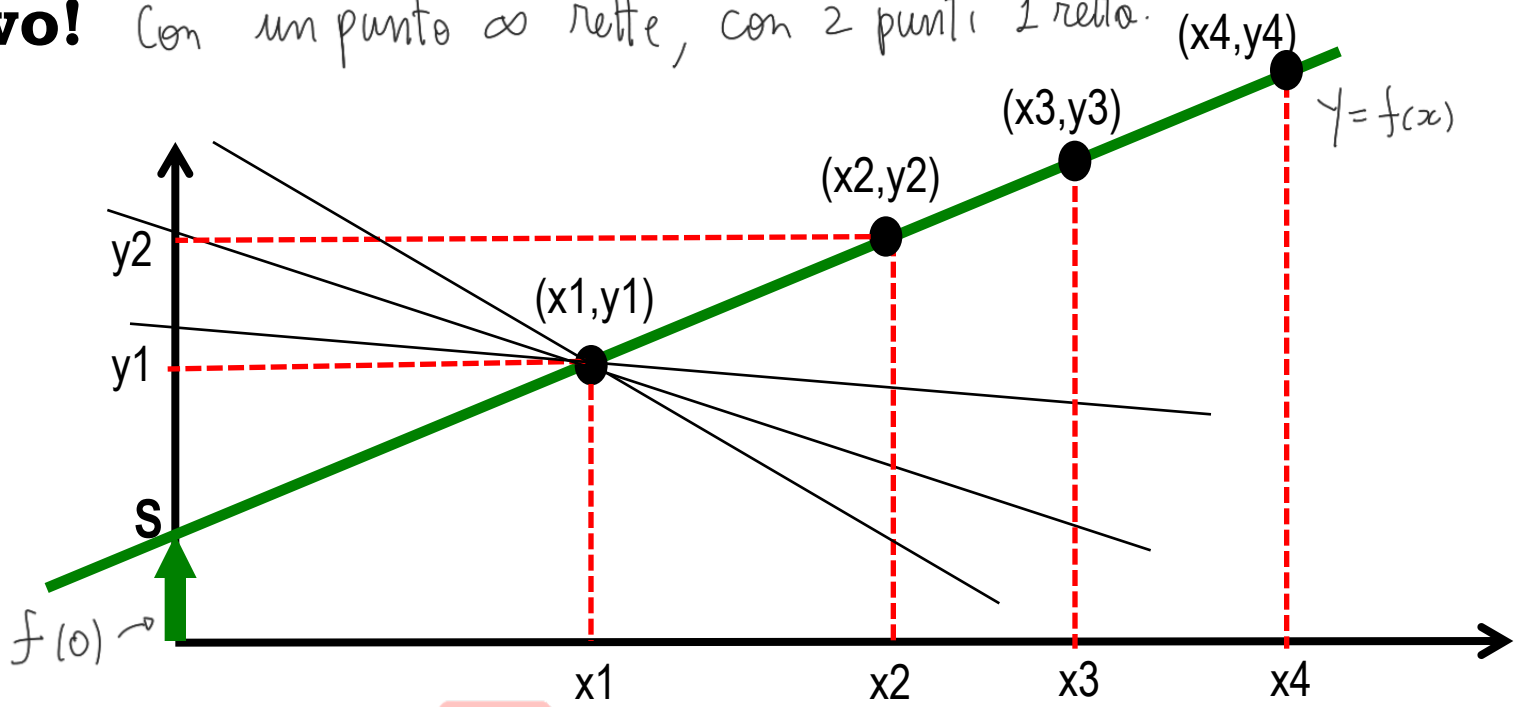
⇨ Independently, different solutions



A (2,n) scheme: idea

➔**How many points uniquely define a line?**

➔**Two!** Con un punto ∞ rette, con 2 punti 1 retta.



➔ **Secret S = y-value at x=0**

➔ **Share for user i: point (x_i,y_i)**

- ⇨ One share: S remains unknown
- ⇨ Two+ shares: Permit to UNIQUELY determine S!

→ quanti punti voglio!

- nel grafico ho 4 partecipanti!
- S nascosto, non lo ha nessun "user".

Procedure: dealing (2, n)

- **Dealer: construct line s. t.**
  - coefficient a: randomly chosen
  - Secret S: known term

partecipanti non la conoscono!

$$y = S + a \cdot x$$

↳ truly random

- **Example: a=15, S=39** inventato → $y = 39 + 15 \cdot x$

- **Distributes shares to n participants**
  - x_i randomly chosen or a priori known
    - Participant 1: $x_1 = 1$ → share = (1, 54)
    - Participant 2: $x_2 = 2$ → share = (2, 69)
    - Participant 3: $x_3 = 3$ → share = (3, 84)
    - …

Lo share sarebbe (x_i, y_i) ma in pratica il segreto è solo in "y_i"

Procedure: reconstructing

➔**Receive two shares** $P_i = (x_i, y_i) \quad P_j = (x_j, y_j)$

➔**Interpolate points (equation of line)**

$$\frac{y - y_j}{y_i - y_j} = \frac{x - x_j}{x_i - x_j} \quad \Rightarrow \quad y = y_j + \frac{x - x_j}{x_i - x_j}(y_i - y_j)$$

$\frac{y - y_2}{y_1 - y_2} \quad \frac{x - x_2}{x_1 - x_2}$

➔**Set x=0 for deriving y=S**

$$S = y_j + \frac{0 - x_j}{x_i - x_j}(y_i - y_j) = y_j \frac{-x_i}{x_j - x_i} + y_i \frac{-x_j}{x_i - x_j}$$

➔ **Example: Pi=(3,84), Pj=(1, 54)**

$$\frac{y - 54}{84 - 54} = \frac{x - 1}{3 - 1} \quad \Rightarrow \quad y = 54 + 15(x - 1) \quad \Rightarrow \quad S = 54 + 15(0 - 1) = 54 - 15 = 39$$

$$S = y_1 \frac{-3}{1 - 3} + y_3 \frac{-1}{3 - 1} = 54 \cdot \frac{3}{2} - 84 \cdot \frac{1}{2} = 39$$

Extension to (t,n)

- **Polynomial of degree t-1 is uniquely defined by t points**
  - ⇨Line → 2 points
  - ⇨quadratic → 3 points
  - ⇨Cubic → 4 points
  - ⇨…
- **Need formula to interpolate t shares**
- **Most convenient: Lagrange interpolation**

Reminder: Lagrange basis polynomials and Lagrange Interpolation

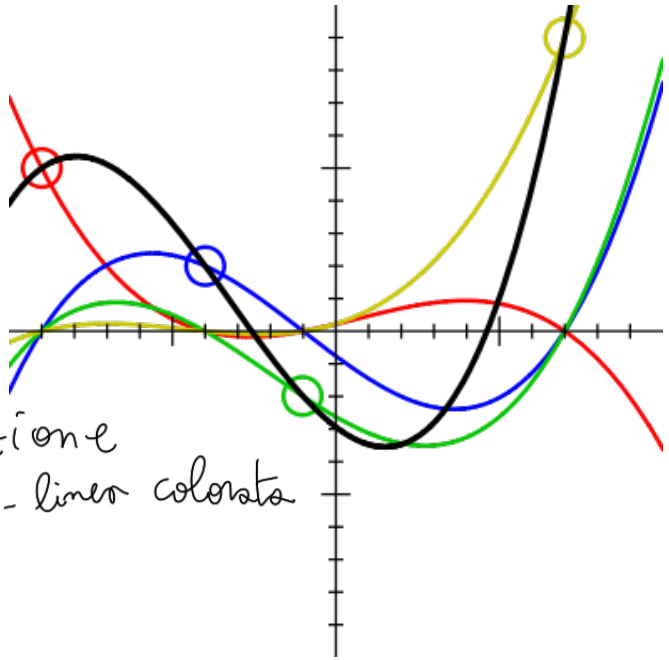
➔ **Any polynomial of degree t-1 with t known points (x_1,y_1), …, (x_t,y_t) can be decomposed as**

$$y = \sum_{i=1}^{t} y_i \Lambda_i(x)$$

Lagrange Base Polinomial

$\Lambda_i(x)$ → value polinomial in position x

cioè linea nera = ∑ punti intersezione linea nera - linea colorata



➔ **Where $\Lambda_i(x)$ is the basis polynomial**

$$\Lambda_i(x) = \prod_{m=1, m\neq i}^{t} \frac{x - x_m}{x_i - x_m} = \frac{(x-x_1)}{(x_i-x_1)} \cdots \frac{(x-x_{i-1})}{(x_i-x_{i-1})} \frac{(x-x_{i+1})}{(x_i-x_{i+1})} \cdots \frac{(x-x_t)}{(x_i-x_t)}$$

➔ **Clearly a basis:** $\Lambda_i(x_i) = 1; \quad \Lambda_i(x_m) = 0 \quad for\ m \neq i;$

Posso precalcolare le lambda e poi applicare le shares(Y)

Lagrange Composition

(t,n) scheme

se t=3

t-1=2 cioè x²

➔Dealer:

- ⇨Generate random polynomial p(x) with degree (t-1)
- ⇨Set secret s as known term in the polynomial

Secret

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

- ⇨Distribute one share to each of the n parties

(parabola)

solo se n>t

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

Se n=t uso trivial

➔Reconstruction

- ⇨Collect t shares out of the n available
- ⇨ Compute secret using Lagrange interpolation @ x=0

$$s = \sum_{shares\ x_i} y_i \Lambda_{x_i} \quad with \quad \Lambda_{x_i} = \Lambda_{x_i}(0) = \prod_{shares\ x_k \neq x_i} \frac{-x_k}{x_i - x_k}$$

value

function

Example: (3,4) scheme

4 shares (4 people)

Secret: value between 1 and 100

$$\text{dealer}: s = 32$$

secret, random, 3 out on 'n'

$$\text{dealer}: p(x) = 32 + 52x + 3x^2$$

$$\text{shares}: (1,87), (2,148), (3,215), (4,288)$$

$32+52\cdot 1+3\cdot 1^2$ $\quad$ $32\cdot 52\cdot 2+3\cdot(2)^2$

4 partecipanti

Collected shares: 1,2,3 (4 missing), cambia a seconda dei punti!

$$\Lambda_1 = \frac{-x_2}{x_1 - x_2}\cdot\frac{-x_3}{x_1 - x_3} = \frac{-2}{1-2}\cdot\frac{-3}{1-3} = 3$$

$$\Lambda_2 = \frac{-x_1}{x_2 - x_1}\cdot\frac{-x_3}{x_2 - x_3} = \frac{-1}{2-1}\cdot\frac{-3}{2-3} = -3$$

$$\Lambda_3 = \frac{-x_1}{x_3 - x_1}\cdot\frac{-x_2}{x_3 - x_2} = \frac{-1}{3-1}\cdot\frac{-2}{3-2} = 1$$

non serve calcolare tutto il polinomio, ma solo queste lambda. Se avessi 1,2,4 le lambda cambiano! Precomputo solo quando so gli shares da collezionare

$$\text{secret}: s = y_1\Lambda_1 + y_2\Lambda_2 + y_3\Lambda_3 = 87\cdot 3 + 148\cdot(-3) + 215\cdot 1 = 32$$

What about secrecy?

Security solo per "2 out n" scheme, NO "3 out n", "4 out n"...

➔**Unconditionally secure scheme:**

⇨knowledge of any t-1 or fewer shares leaves secret s undetermined

➔in the sense that all its possible values are equally likely

➔**Is this true???**

➔**NO! Scheme presented so far is flawed!!**

Example: shares 1 and 3 collected; can we say something about s??

$$\text{compute as before}: \Lambda_1 = 3; \Lambda_2 = -3; \Lambda_3 = 1$$

$$\text{share 2 unknown}: \text{set to D}$$

$$\text{secret}: s = y_1\Lambda_1 + D\Lambda_2 + y_3\Lambda_3 = 87 \cdot 3 + D \cdot (-3) + 215 \cdot 1 = 476 - 3D$$

vedo che se D =126 s=98; se D=127 s=95
poichè D intero, tra 126 e 127 non c'è nulla , allora
s NON è nè 97 nè 96, quindi c'è leak di info

D=integer ➔ s=98 (D=126), 95, 92, 89, 86, 83, … 2/3 of values s EXCLUDED!

Knowledge of shares 1 and 3 permits to triplicate guess probability!

The real Shamir scheme!!

➔ **Use modular arithmetic** con prime modulos
- ⇨ instead of real arithmetic
- ⇨ Secret & polynomial in prime field F_p
  - ➔ Interpolation formula still holds mod p

real Shamir è molto simile, tranne dove al posto della divisione devo fare modular inverse, e posso farlo solo con numeri coprimi.

$$\prod_{x_k \neq x_i} \frac{-x_k}{x_i - x_k}$$

➔ **Result: unconditionally secure scheme!**

➔ **Which condition on prime p?**
- ⇨ Just greater than the domain for the secret!
  - ➔ E.g. if secret in (0,100), p=101 perfectly OK
- ⇨ Does NOT strictly need to be large!
  - ➔ Security is NOT computational (related to some hard problem) but is information theoretic!

Basta un prime "leggermente più grande" (unconditionally secure) non sto in RSA dove necesito large prime (computational secure)

(unconditionally secure) ↔ perfectly secure scheme

(computational secure) ↔ not perfectly secure scheme

Example – mod p=101

Secret: value between 1 and 100

$$\text{dealer}: s = 32$$ (segreto da nascondere, $0 \le S \le 100$)

$$\text{dealer}: p(x) = 32 + 52x + 3x^2 \mod p = 101$$

$$\text{shares}: (1,87), (2,47), (3,13), (4,86)$$

random, generati tra 0 a 100, poiché opero in modulo 101

Collected shares: 1,2,3 (4 missing)

$$\Lambda_1 = 3; \Lambda_2 = -3; \Lambda_3 = 1;$$

$$s = y_1\Lambda_1 + y_2\Lambda_2 + y_3\Lambda_3 = 87 \cdot 3 + 47 \cdot (-3) + 13 \mod 101 = 32$$

$$s = 87 \cdot 3 + D \cdot (-3) + 13 \mod 101 = 72 - 3D \mod 101$$

S uniformly distributed in F_p!

esempio esame

Q8 - A Shamir Secret Sharing scheme uses a non-prime modulus p=55 (if you need modular inverses see table on the right). Of the 5 participating parties $P_1,\ldots,P_5$, with respective x coordinates $x_i = \{1,2,3,4,5\}$, parties P_1, P_3 and P_5 aim at reconstructing the secret.

a) compute the Lagrange Interpolation coefficients for parties 1,3,5;

b) Reconstruct the secret, assuming that the shares are:

$P_1 \rightarrow 46$
$P_3 \rightarrow 51$
$P_5 \rightarrow 2$

c) Prove that the system is NOT unconditionally secure, by showing that the knowledge of the two shares P_3 and P_5 leak information about the secret – specifically, after knowing shares P_3 and P_5 which would be the only possible remaining secret values?

[Answer:
Lambda1=50, lambda3=40, lambda5=21
Secret = 37;
set of possible secrets: the 11 possible values which satisfy 47+50x mod 55 →
→ {42,37,32,27,22,17,12,7,2,52,47}

x	1/x mod 55
1	1
2	28
3	37
4	14
6	46
7	8
8	7
9	49
12	23
13	17
14	4
16	31
17	13
18	52
19	29

Q5 - Consider an RSA digital signature based on a (2,2) secret sharing, and assume all following operations are based on modulo n, with n being the RSA parameter. The tag $H(m)^d$ is reconstructed by:

- ○ **a)** Summing the tags constructed using the two shares
- ○ **b)** Multiplying the tags constructed using the two shares
- ○ **c)** Interpolating the tags constructed using the two shares using Lagrange coefficients
- ○ **d)** Using a special approach proposed by Shoup.

b) $P_1 = 46$, $P_3 = 51$, $P_5 = 2$

$$[50 \cdot 46 + 40 \cdot 51 + 21 \cdot 2] \bmod 55 = 37 = S$$

c) Ho note P_3 e P_5

$$[50 \cdot d + 40 \cdot 51 + 21 \cdot 2] \bmod 55 = [2082 + 50d] \bmod 55$$

$= 47$, ottengo: $[47 + 50d] \bmod 55$, vado a tentativi:

$d = 0 \rightarrow 47$
$d = 1 \rightarrow 42$
$d = 2 \rightarrow 37$
$d = 3 \rightarrow 32$

fattore 5, fattore di 55, "ricircola" nel set di 11 punti, cioè:

d in range [0,54] ottengo:

{47, 42, 37, 32, 27, 22, 17, 12, 7, 2, 52, 47, ...} è sottoperiodo (subset) di 11 elementi su 54 che sono realmente validi, questo è UNCONDITIONALLY SECURE

NB: Con p primo, avrei avuto come sequenza i valori da 0 a 58 se p=59 (periodo di p primo = 59)

Se usassi H(s), e non 's', in condizioni computational secure non basterebbe, sempre nel range (0,100) sono!
(enumeration attack)

Ideality

(se secret in 8 bit, share al max 8 bit)

➔Can share be smaller than secret?

- ⇨No: share at least as large as secret
- ⇨Intuition: Given t-1 shares, no information can be determined about secret. Thus, final share must contain as much information as the secret itself.

(se t-1 shares dà 8 bit di info, allora l'ultimo share deve avere 8 bit di info!
Posso avere shares lower than the secret, MA perdo security proprierty.

➔Shamir scheme: ideal!

- ⇨|share|=|secret|
- ⇨Few proposed schemes NON ideal
  - ➔share larger than secret (e.g. t-times, n-times)
  - ➔E.g Blakley scheme shares are t-times secret

Secret Sharing for Secure Multiparty Computation (basics)

Homomorphic property (di Shamir)

genera

Secret S_F ed inventa

Polynomial f(x)

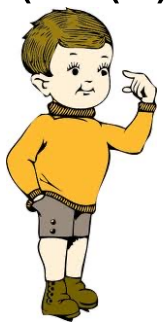
$S_F \neq S_G$, $f(x) \neq g(x)$

generati in modo diverso!

- se degree $(f(x)) \leq x^2$ → (3,4) scheme

[lo faccio valere anche per g(x)]

(1,f(1))



(2,f(2))

(3,f(3))

(4,f(4))

4 shares

Homomorphic property:

SUM of the SHARES = Share of the SUM S_F+S_G

↑ obiettivo

f(i), g(i) → f(i)+g(i) → → (f+g)(i)

(1,g(1)) (2,g(2)) (3,g(3)) (4,g(4))

Secret S_G ed inventa

Polynomial g(x)

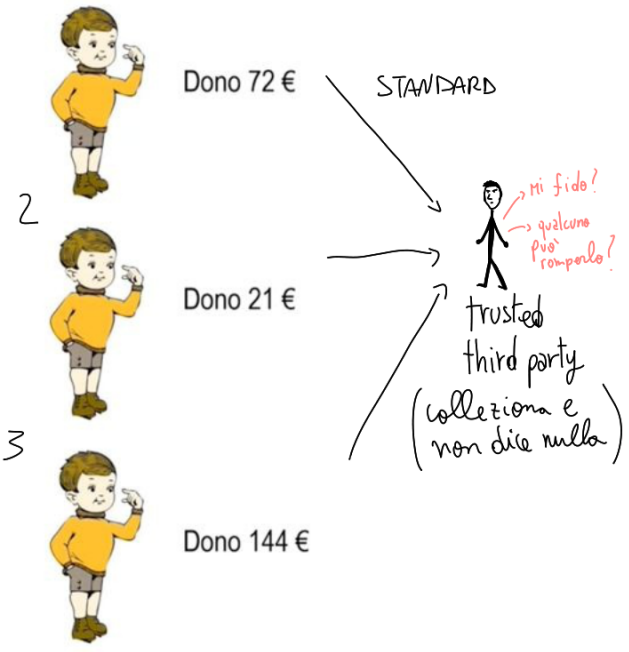
posso fare $S_F + S_G$ senza conoscerli =

Revealing t composite shares reveals SUM of secrets but does NOT reveal individual secrets

Estremamente pratiche?! Un semplicissimo esempio!

dealer 1

$\emptyset$ ≤ range donazioni ≤ 1000

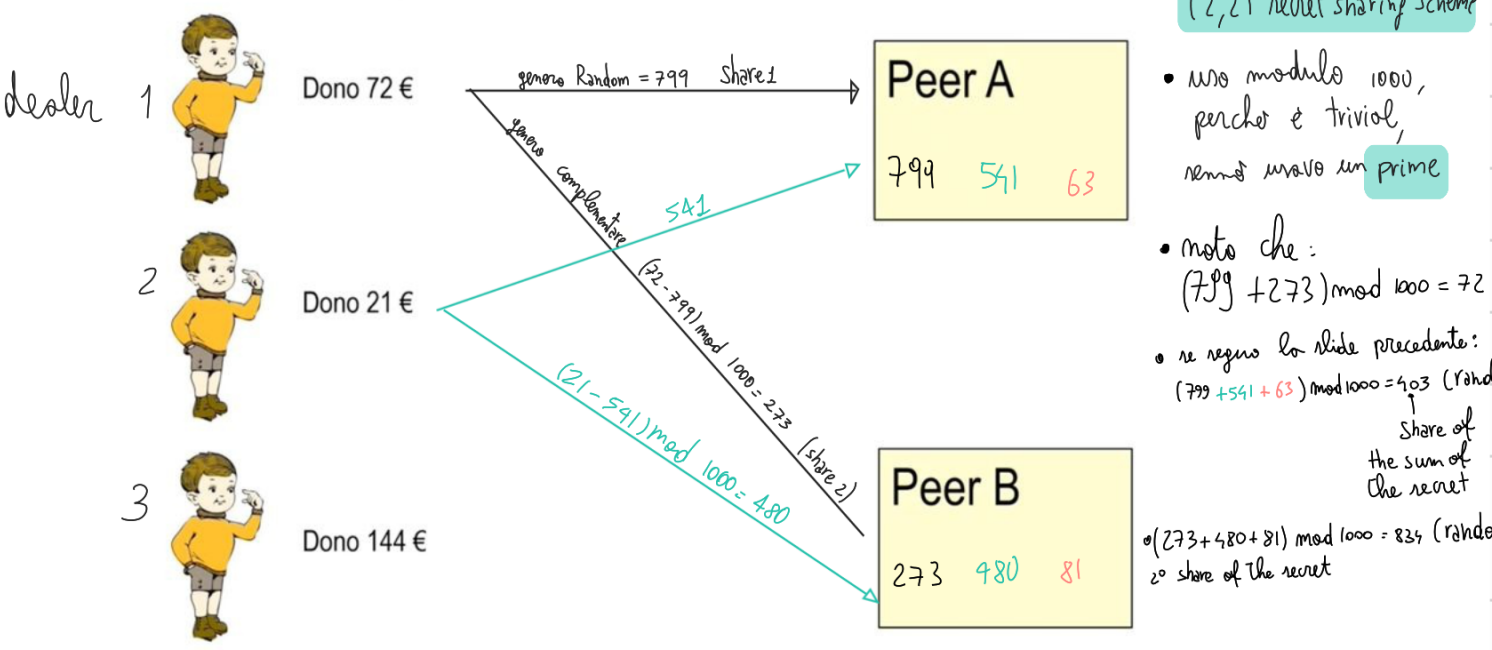


Posso senza third party?

Giuseppe Bianchi

Estremamente pratiche?! Un semplicissimo esempio!

$\emptyset$ ≤ donazioni ≤ 1000



(2,2) secret sharing scheme

- uso modulo 1000, perché è trivial, nemmeno usavo un prime
- noto che: (799 + 273) mod 1000 = 72
- se seguo la slide precedente: (799 + 541 + 63) mod 1000 = 403 (random) ↑ share of the sum of the secret
  (273 + 480 + 81) mod 1000 = 834 (random) 2° share of the secret

Giuseppe Bianchi

72 + 21 + 144 = 237, inoltre (403 + 834) mod 1000 = 237, cioè operazioni senza rilevare gli input value, facendo una somma senza usare le donazioni!

In termini tecnici, parlo di una branca della criptografia nota come:

What is SMC?

➔ **Secure Multiparty Computation**

⇨ Compute the result of a function without revealing input data

➔ **Several use-case scenarios**

⇨ Business/financial, security, medical, traffic monitoring, etc

⇨ Largely underrated in the real-world, IMHO

➔(also for historical reasons: considered "cumbersome")

➔ **Main facts**

⇨ Born as 2-party computation (Yao's millionaire problem, 1982)

⇨ VERY general theorems available → estendibile a più parties

➔ANY ARBITRARY 2-party (Yao, 1986) and multiparty (Goldreich, Micali, Widgerson, 1987) computation can be made secure

➔But cost of construction may be non practical

Dati 2 milionari, posso dire chi è il più ricco senza rivelare i loro patrimoni? (GARBLED CIRCUITS)

≈ operare su dati criptati è possibile!

More specifically

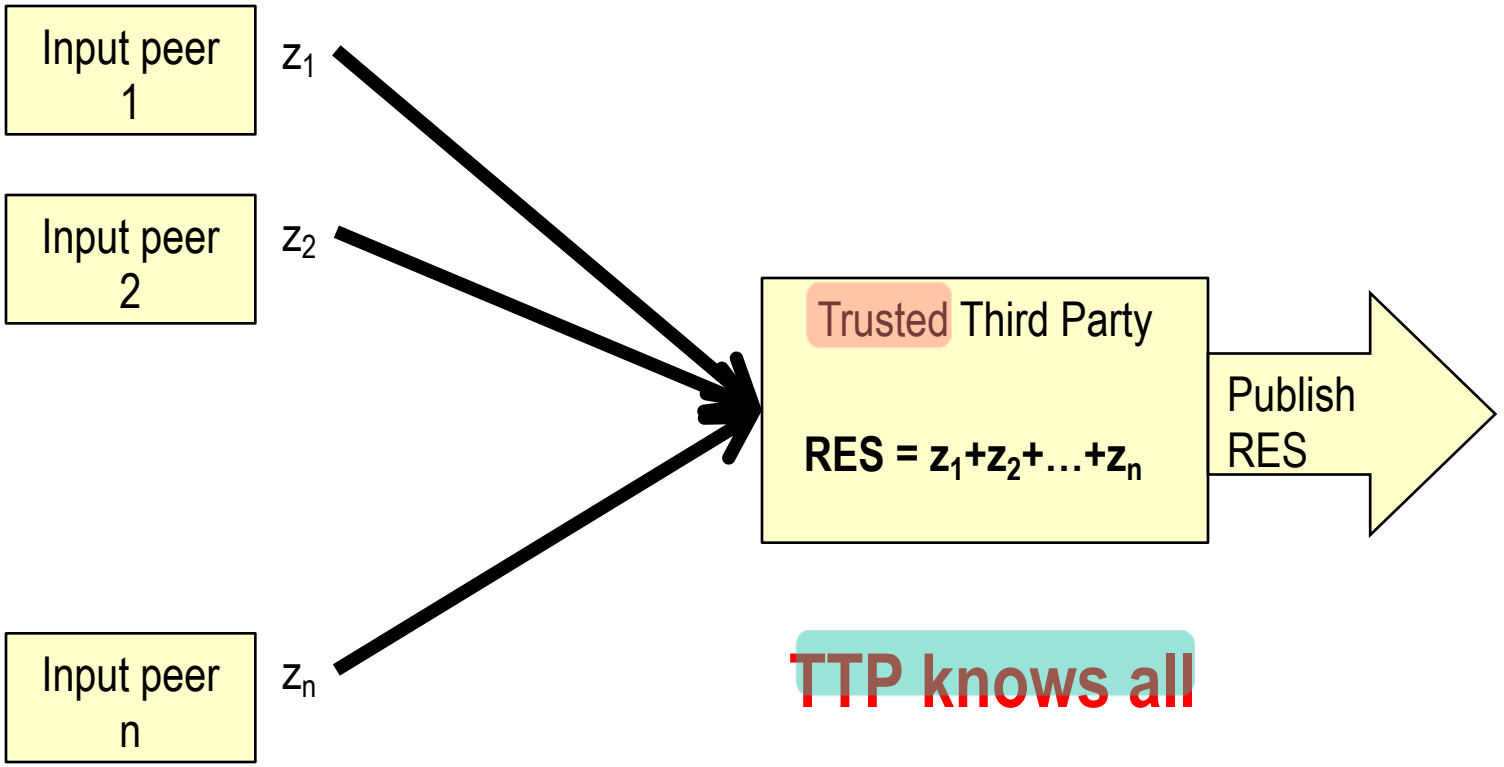
➔ **N parties P_1, P_2, …, P_n** (≈ donatori)
- ➔ Each with value z_i (≈ donazioni)

➔ **Compute function $f(z_1, z_2, \ldots, z_n)$ s.t.**
- ➔ Result is public
- ➔ But NO information is given on input values z_i

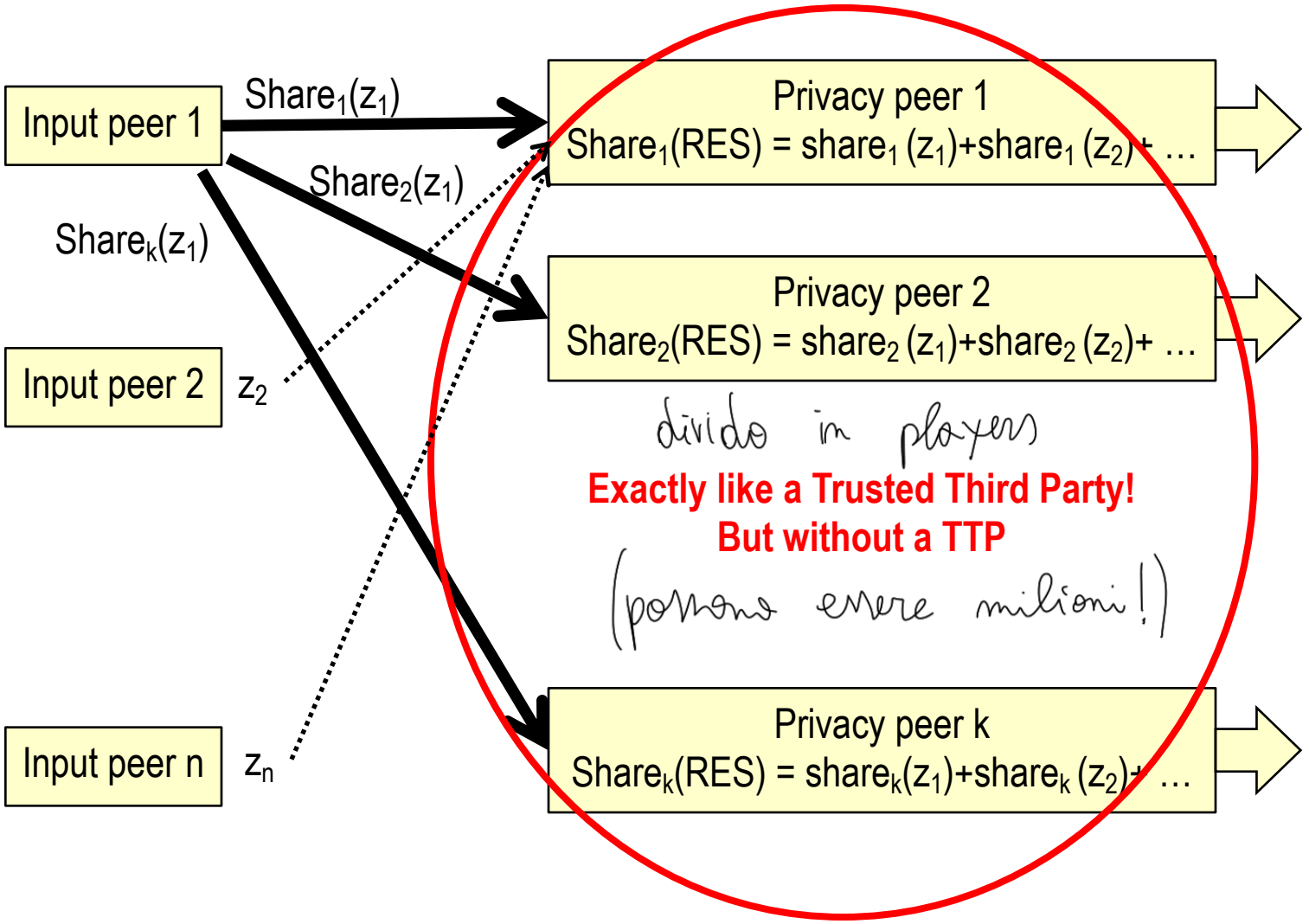
➔ **Our interest in the special case of addition**
- ⇨ Statistics aggregation
- ⇨ Actually, works also for weighted addition (z_i e z_j non moltiplicate tra loro)
  - » $f(z_1, z_2, \ldots, z_n) = \alpha_1 z_1 + \alpha_2 z_2 + \ldots + \alpha_n z_n$
- ⇨ Very frequent case in practice! More than what you imagine

➔ **If f=addition, then very efficient SMC from secret sharing schemes!**

Without SMC: Third party



With SMC: "simulated TTP"!



Deployment issues

- **Input peers n:**
  - At least 3; If 2, one party may know other party data by subtracting from final result (se dono 72, ed il totale è 73...)
  - Can be MANY
    - Thousands of end users
- **Privacy peers k**
  - At least 2, se uno si rompe? uso Shamir (3,2) da rivedere
    - But the more, the better for security
    - More robust to collusion
  - Usually small number
  - Threshold # peers: $2 \leq t \leq k$: if (t,k) scheme
- **Input peers may be privacy peers**
  - No change in construction

Construction (con Shamir)

➔ **Input peer i** (= input = dealer)

- ⇨ Input data z_i — nasconde dato nel polinomio
- ⇨ generate polynomial $p_i(x)$ of degree t-1 with z_i as known term
- ⇨ Privately send shares $p_i(1),\ldots,p_i(k)$ to privacy peer 1,…,k
  - →No coordination across input peers!!

➔ **Privacy peer m** (è il player)

- ⇨ Collect input shares $p_1(m), p_2(m), \ldots, p_n(m)$ (sono io)
- ⇨ Compute $RES(m) = p_1(m) + p_2(m) + \ldots + p_n(m)$
- ⇨ Publish aggregated share RES(m)

➔ **Public:**

- ⇨ Reconstruct RES from sufficient number of RES(m)
  - → Lagrange Interpolation

skip

Using all privacy peer? Much faster!

- **Roll back to (k,k) trivial secret sharing!**
  - Measurement z
  - Share 1: r1
  - Share 2: r2
  - …
  - Share k: s-r1-r2-..
- **Ordinary modular arithmetics, on small non-prime fields**
  - E.g. usual 32 bit CPU word
  - No need to “change” any software
  - No performance impairment at all!!

Example (donazione)

Input peers = privacy peers

non c'è autorità 3ª, opero come (3,3)

R1 = 34
R2 = 89

Party 1
14$

opera come dealer per "14", ma lo manda come P_1, P_2, P_3 come players

R1 = 66
R2 = 11

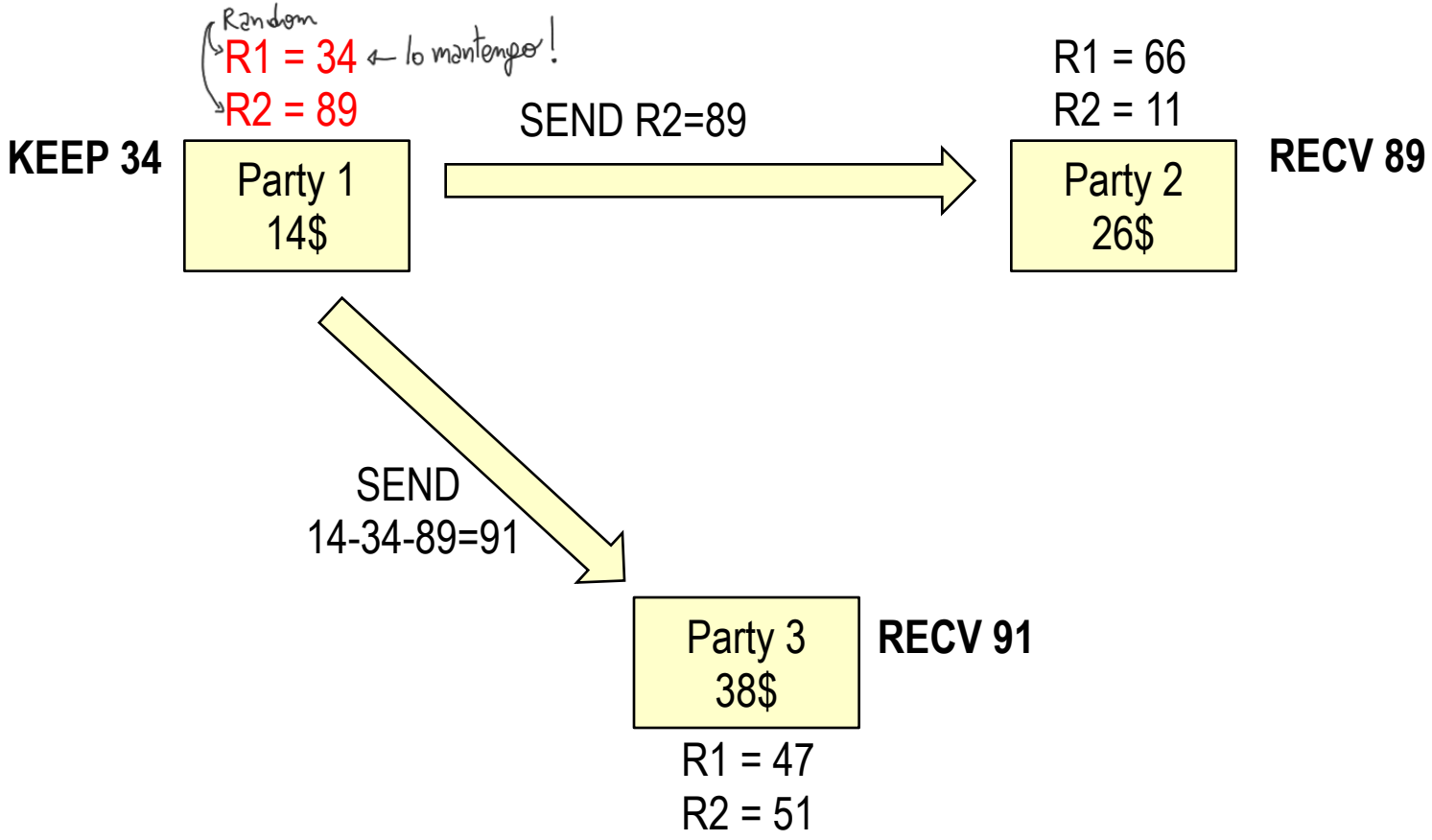
Party 2
26$

Party 3
38$

R1 = 47
R2 = 51

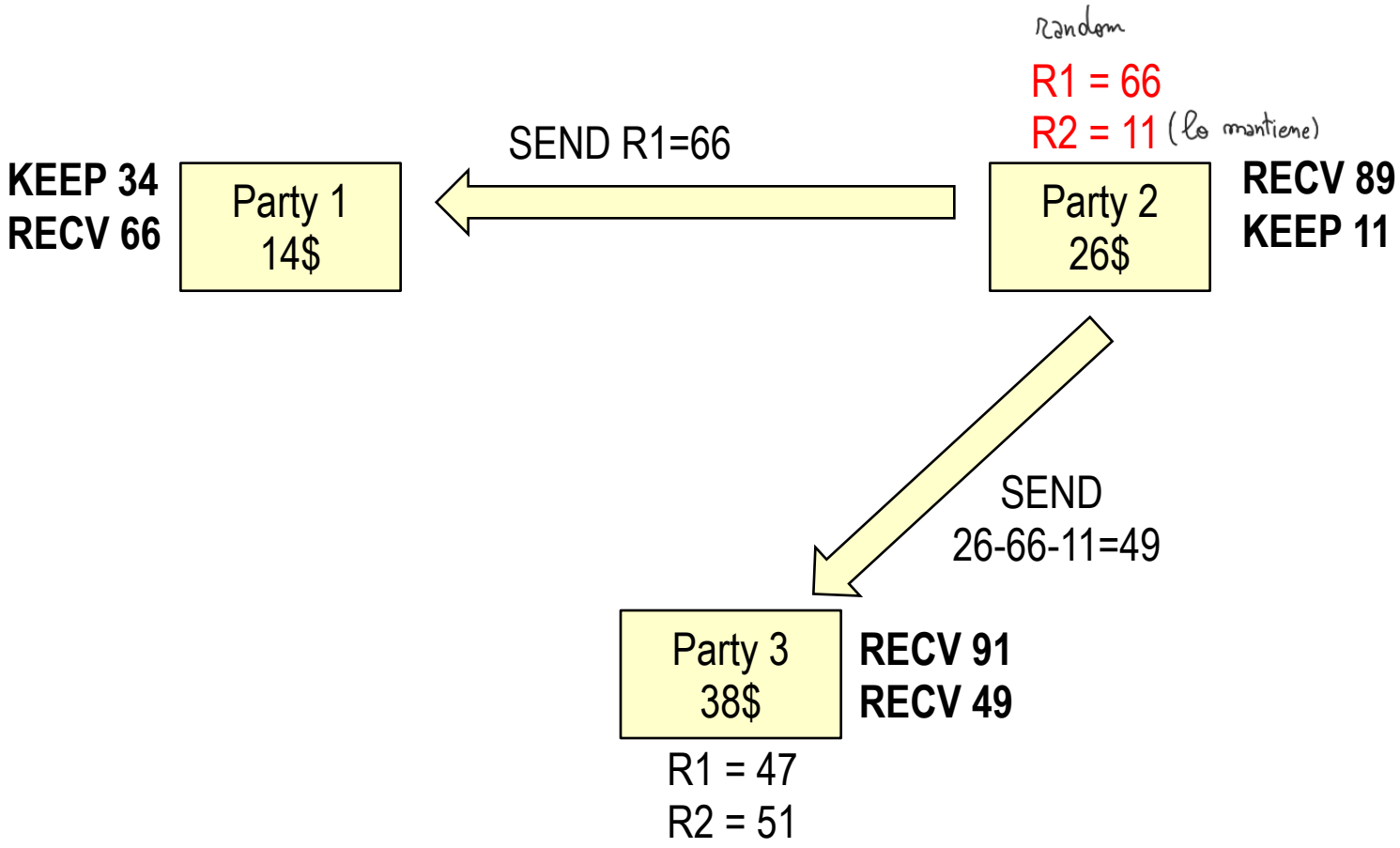
Assume modulo 100 arithmetics

Example
Imput peers = privacy peers



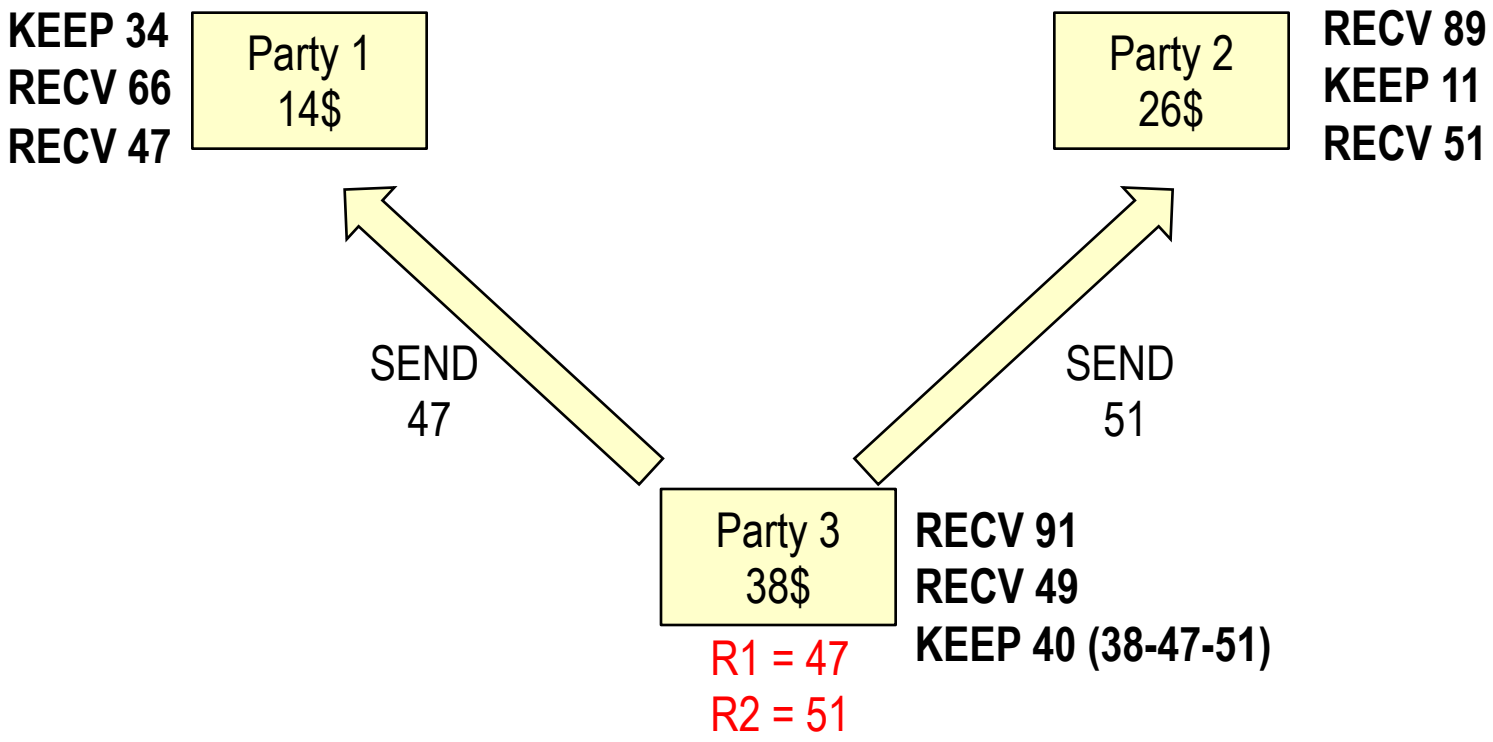
Assume modulo 100 arithmetics

Example
Imput peers = privacy peers



Assume modulo 100 arithmetics

Example
Imput peers = privacy peers



Assume modulo 100 arithmetics

Example
Imput peers = privacy peers

Valido $\forall$ Secret Sharing Scheme

KEEP 34
RECV 66
RECV 47

TOT=47

Party 1
14$

Party 2
26$

RECV 89
KEEP 11
RECV 51

TOT=51

Public: 51+47+80 = 78

No input data revealed in the run

Party 3
38$

RECV 91
RECV 49
KEEP 40

TOT=80

Assume modulo 100 arithmetics