Protocollo di Yao

Secure Two-Party Computation

Prof. Paolo D'Arco

Università degli Studi di Salerno

Presentazione a cura di Michele Boccia e Francesco Matarazzo

made with LaTeX

12 Giugno, 2012

Contenuti

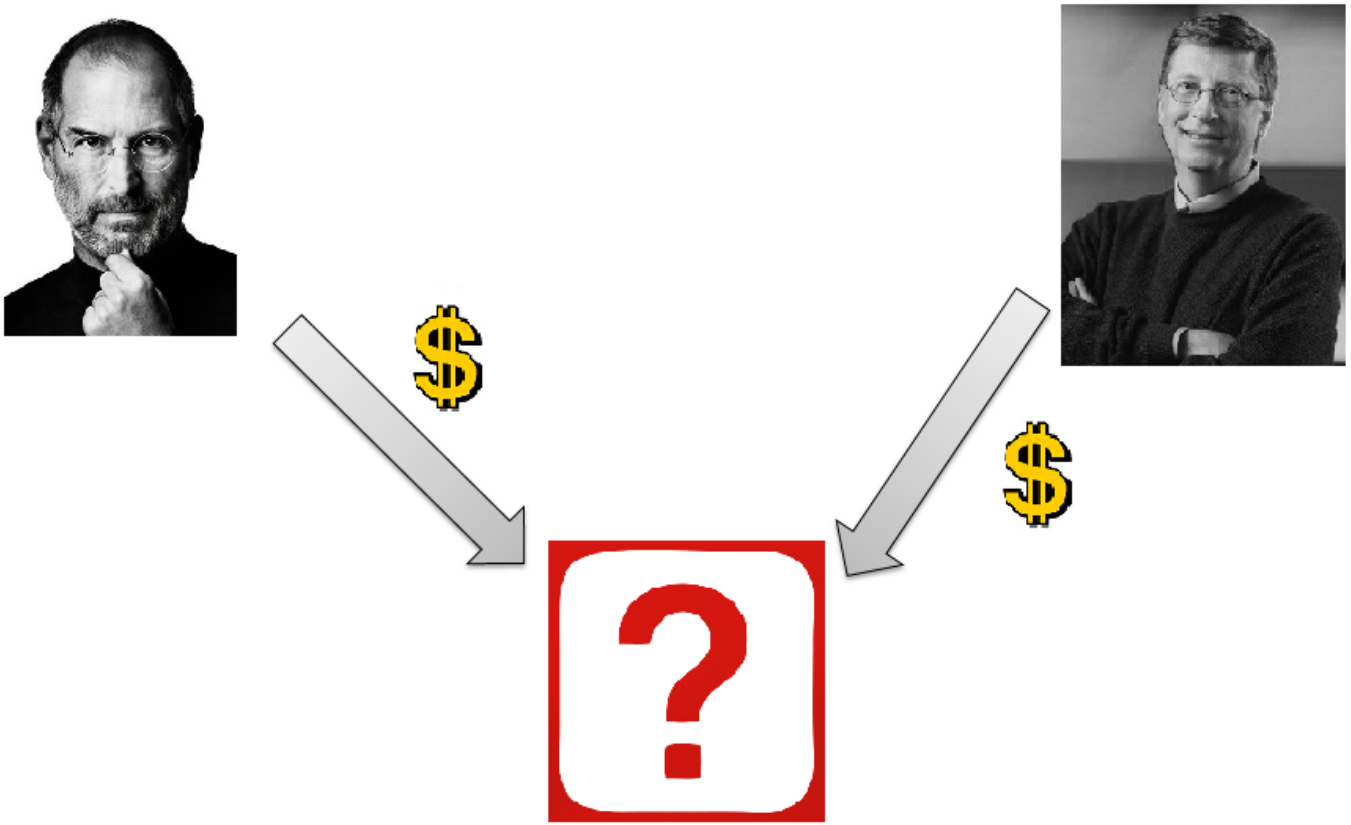
Introduzione
- Problema dei Milionari

Visione fisica
- Circuito
- Scatole
- Vasi Magici

Implementazione Digitale
- Cifratura
- Protocollo OT 1-su-2

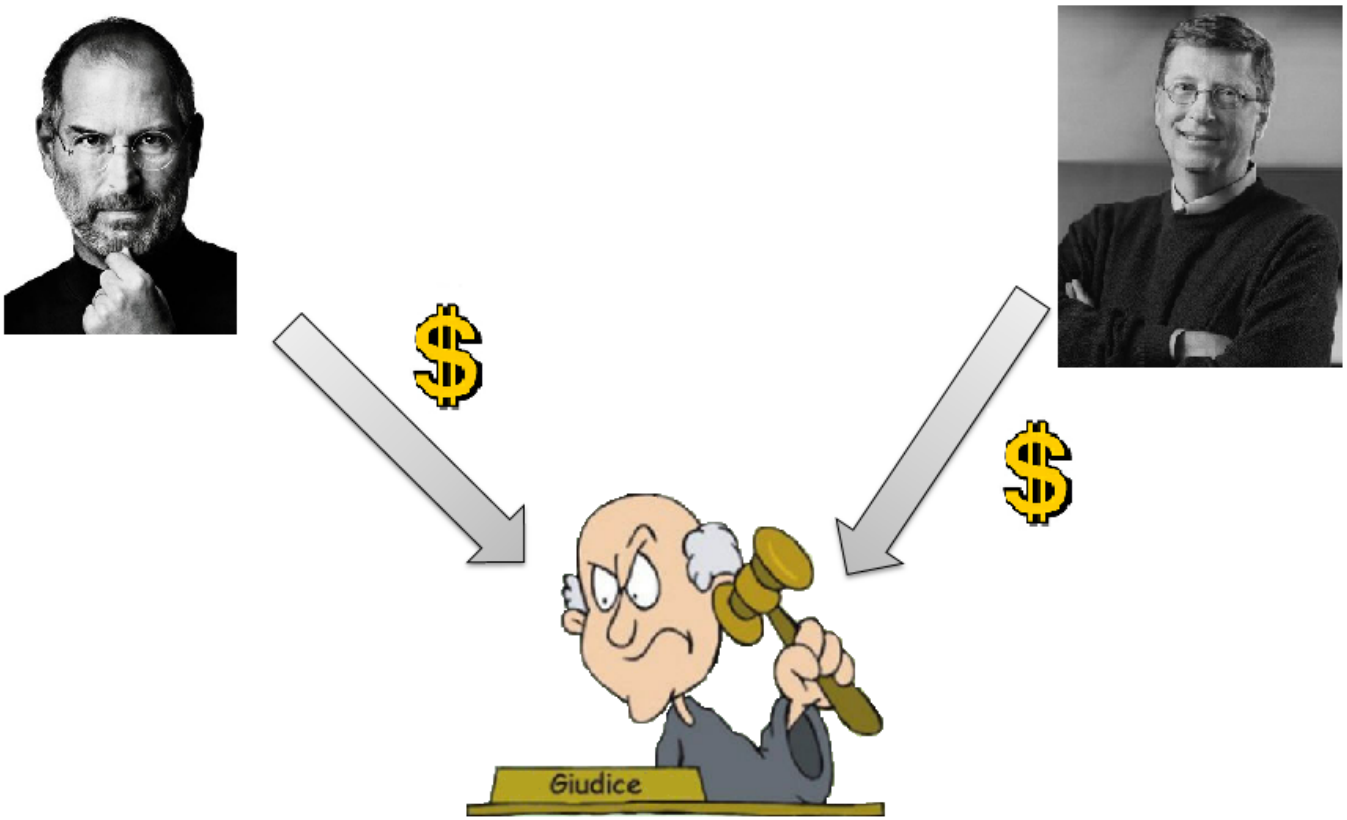
Specifica del Protocollo

Motivazioni



- Chi è più ricco?
- Nessuno dei due intende rivelare la propria ricchezza!

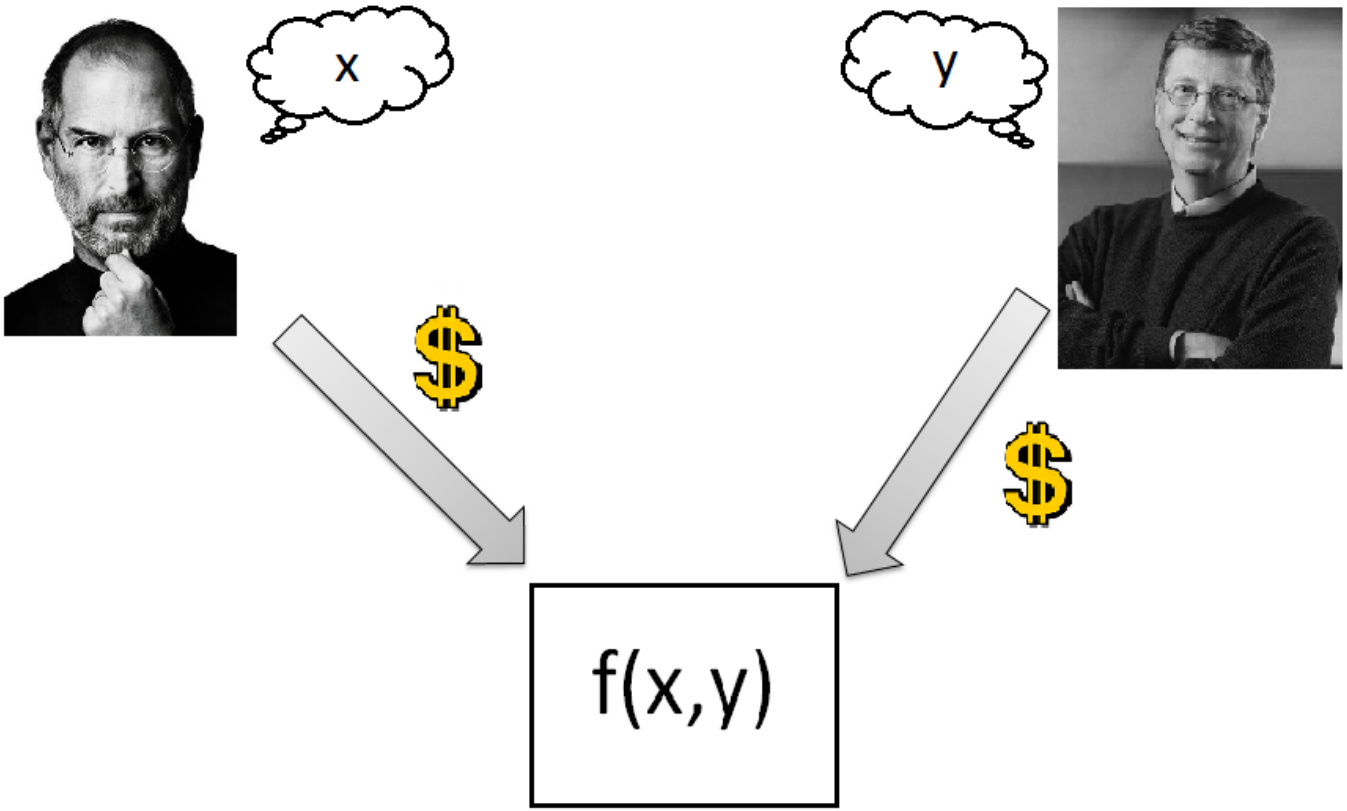
Motivazioni



Obiettivo

- Vogliamo risolvere il problema **senza l'uso di una parte fidata**
- Serve un protocollo che le due parti possano eseguire che *emuli* la presenza di una terza parte fidata

Problema Generale



- Le due parti desiderano calcolare $f(x, y)$ mantenendo i propri input x e y privati

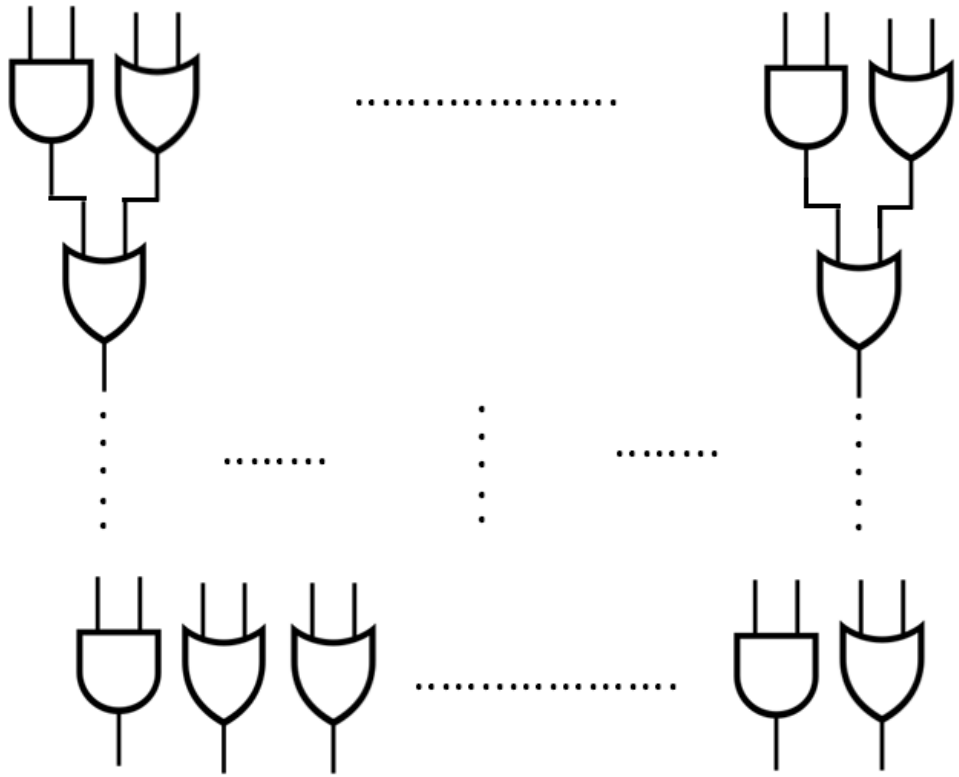
Setting

- P_1 ha input $x \in \{0,1\}^n$
- P_2 ha input $y \in \{0,1\}^n$
- $f : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n$
- P_1 e P_2 vogliono calcolare congiuntamente $f(x,y)$, in modo tale che:

  $\Rightarrow$ la computazione dia $f(x,y)$ e nessuna *informazione* sugli input x e y che non *possa* essere dedotta dalla conoscenza di $f(x,y)$

- Una computazione può essere descritta in diversi modi:
  - $\Rightarrow$ Algoritmo espresso in codice/pseudocodice
  - $\Rightarrow$ Macchina di Turing

  ...
  - $\Rightarrow$ **Circuito Booleano** $\rightarrow$ Scegliamo questa rappresentazione!
- Supponiamo pertanto che la nostra $f(x, y)$ sia rappresentata da un circuito $C(x, y)$ con $2n$ fili di input ed n fili di output.
- Supponiamo inoltre che ogni porta logica (AND, OR) abbia 2 fili di input (fun in) e un filo di output (fun out)



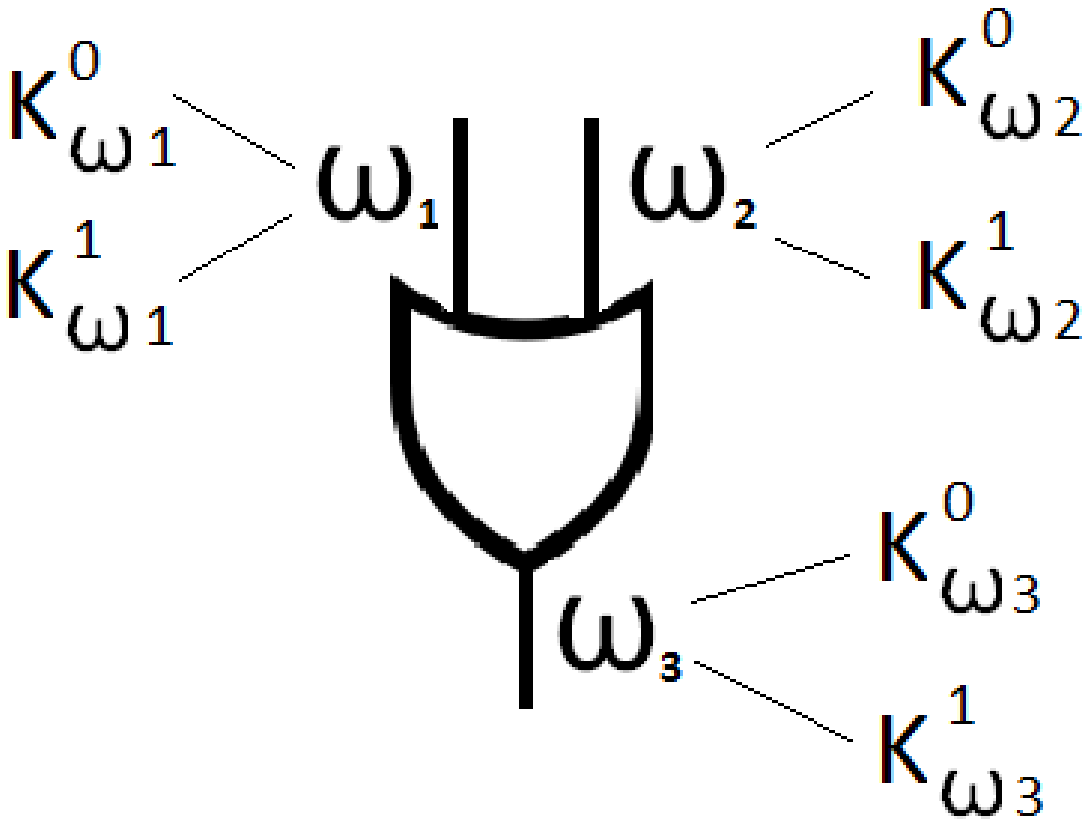
- Dato un circuito $C(x, y)$ che calcola $f(x, y)$, applicando ai fili di input i valori di x e y, e propagando i valori dalle porte del primo livello fino alle porte dell'ultimo livello, si ottiene il valore di output, cioè $f(x, y)$

- Tuttavia, volendo preservare la privacy degli input, nessuna delle due parti può dare il proprio input in *chiaro* per calcolare il circuito

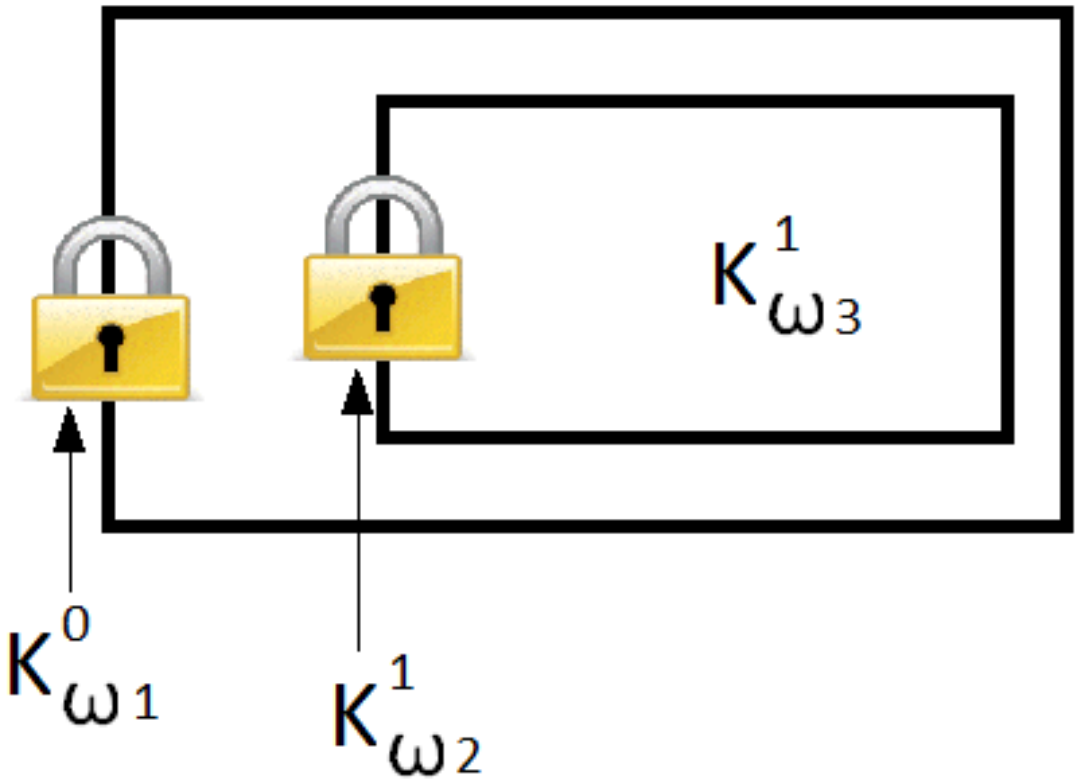
- **Idea**: piuttosto che effettuare la computazione sui bit in chiaro, vogliamo effettuare la computazione usando *valori casuali* che non danno informazioni sui bit cui però sono associati

Ci torneremo più tardi...
Ragioniamo in termini di chiavi e lucchetti fisici!

- Diamo un nome ad ogni filo del circuito: ω_1, ω_2...
- Sia ω un filo generico
- Associamo a ω due chiavi K_ω^0 e K_ω^1, che rappresentano 0 e 1.
- Consideriamo una generica porta logica, per esempio una porta OR:

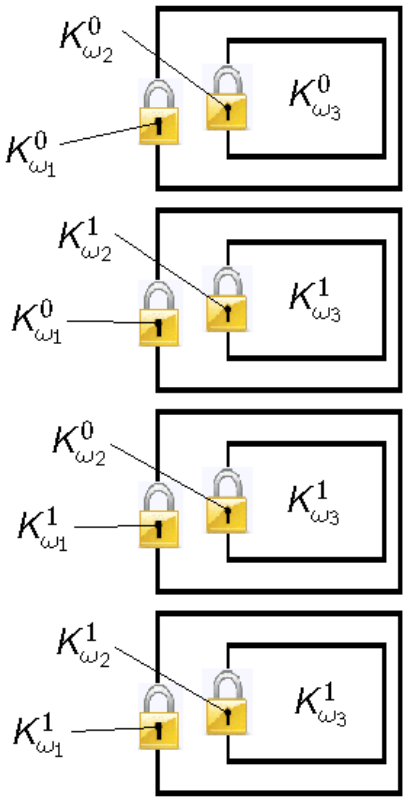


- Associamo alla porta 4 scatole doppie, una interna ed una esterna, chiuse con due lucchetti che necessitano di due chiavi diverse per essere aperti
- La scatola interna contiene una copia di K_ω^0 o di K_ω^1



- In particolare, le 4 scatole per una porta OR sono costruite come segue:

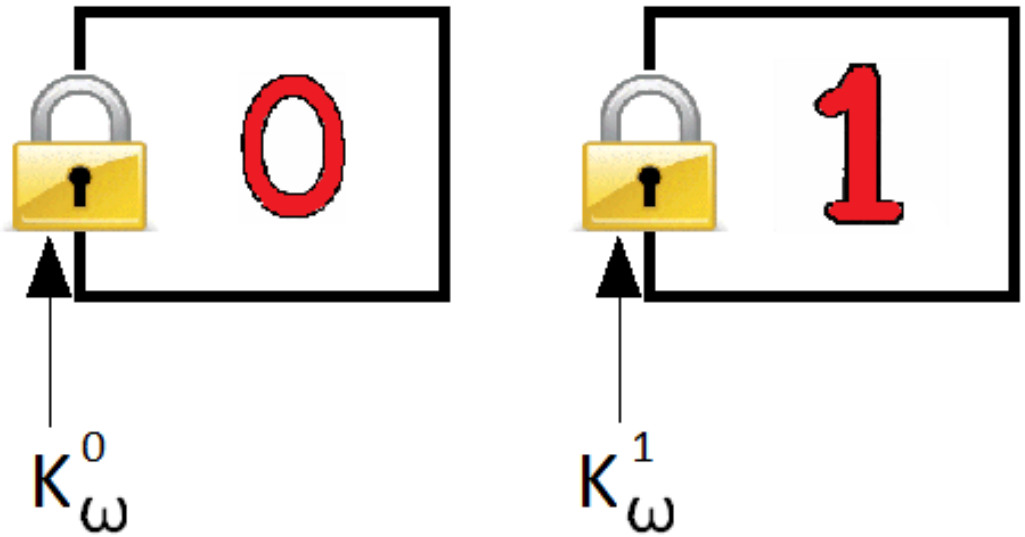
ω_1	ω_2	bit	OR	ω_3
$K^0_{\omega_1}$	$K^0_{\omega_2}$	0,0	0	$K^0_{\omega_3}$
$K^0_{\omega_1}$	$K^1_{\omega_2}$	0,1	1	$K^1_{\omega_3}$
$K^1_{\omega_1}$	$K^0_{\omega_2}$	1,0	1	$K^1_{\omega_3}$
$K^1_{\omega_1}$	$K^1_{\omega_2}$	1,1	1	$K^1_{\omega_3}$



- Le scatole doppie vengono permutate a caso

- Ogni coppia di chiavi apre *una ed una sola* scatola doppia
- Se le due parti, P_1 e P_2, hanno *una sola chiave* per ogni filo di input, possono ottenere *una sola chiave* dall'apertura di ogni scatola doppia associata ad ogni porta di primo livello
- Con le chiavi ottenute aprendo le scatole del primo livello, si possono aprire le scatole associate alle porte del secondo livello (una ed una sola per ogni porta) e, iterando il processo, si può ottenere una ed una sola chiave per ogni filo di output, fili associati alle porte dell'ultimo livello

- Le chiavi dei fili di output del circuito aprono scatole singole contenenti 0 o 1:



- Pertanto, supponendo che *una parte fidata* costruisca il circuito $C(x, y)$ che calcola $f(x, y)$, costruisca le scatole doppie per ogni porta e dia a P_1 e P_2 le chiavi fisiche dei fili di input a seconda dei rispettivi valori di x e di y, allora P_1 e P_2 sono in grado di aprire le scatole doppie in successione e ottenere il valore della funzione, che si trova nelle scatole singole finali

Esempio

- n = 4
- x = 0100
- y = 1101

$$\text{Chiavi di } P_1 = K^0_{\omega_1}, K^1_{\omega_2}, K^0_{\omega_3}, K^0_{\omega_4}$$

$$\text{Chiavi di } P_2 = K^1_{\omega_5}, K^1_{\omega_6}, K^0_{\omega_7}, K^1_{\omega_8}$$

...ma non vogliamo la parte fidata!!!
Se l'avessimo, non avremmo bisogno di mettere in piedi un meccanismo così articolato...
Come facciamo?

Idea

- P_1 costruisce il circuito $C(x, y)$ che calcola $f(x, y)$ ed in accordo ad esso la sequenza di scatole
- P_1 possiede *tutte* le chiavi e vuole far eseguire la computazione a P_2
- P_1 invia le chiavi $K^0_{\omega_1}, K^1_{\omega_2}, K^0_{\omega_3}, K^0_{\omega_4}$ che corrispondono al suo input a P_2. Si noti che le chiavi sono solo pezzi di ferro e non danno informazioni sui bit cui sono associate

Problema

- Come fa P_1 a dare a P_2 le chiavi associate all'input di P_2?

1. Soluzione: P_2 comunica a P_1 il valore y! Privacy persa!!!
2. Soluzione: P_1 le invia tutte: $K^0_{\omega_5} K^1_{\omega_5}, K^0_{\omega_6} K^1_{\omega_6} \ldots K^0_{\omega_8} K^1_{\omega_8}$
   P_2 può aprire diverse scatole e quindi calcolare **non solo** $C(x, y)$ ma $C(x, \bar{y})$ per ogni $\bar{y} \in \{0,1\}^4$!!!

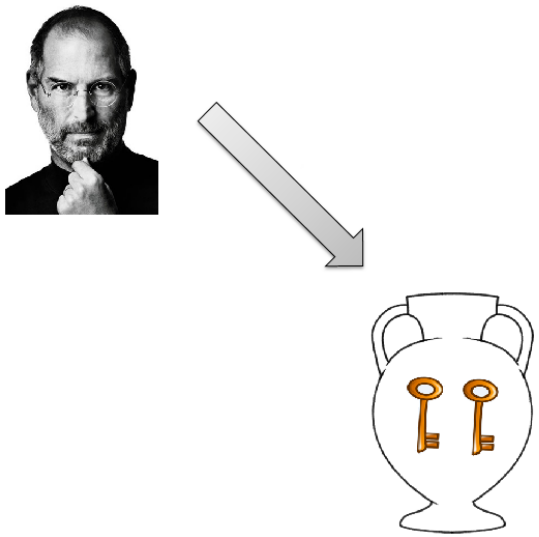
Non va bene...

- Occorre quindi un metodo attraverso il quale P_1 permette a P_2 di *scegliere* le chiavi che corrispondono al proprio input, senza che però P_1 sappia quali ha scelto!!!

Come si fa?

- P_1 dispone di un *Vaso Magico* che può contenere due chiavi, con due proprietà:

1. Una volta inserite le due chiavi, P_1 non le può più recuperare
2. Il vaso magico ha un foro che permette a P_2 di recuperare *una sola* delle chiavi, *a sua scelta*. Una volta estratta, il vaso si chiude ermeticamente





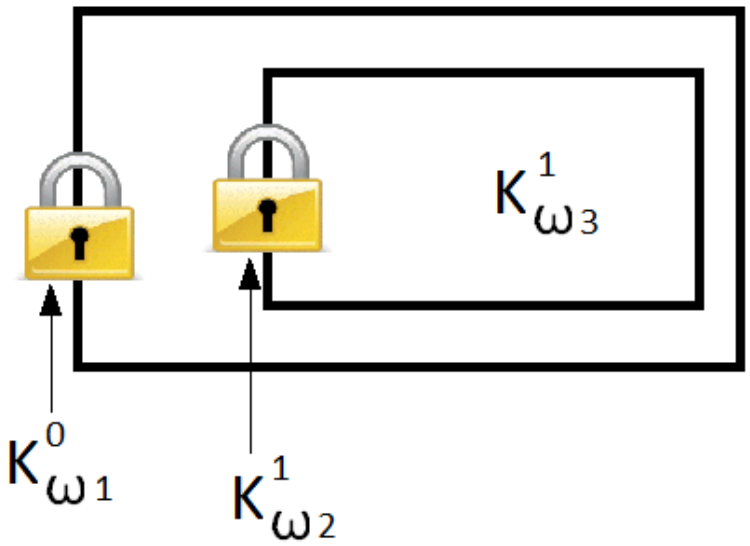
- Per esempio, supponendo che le chiavi introdotte abbiano un'etichetta che al tatto rivela se corrisponde ad un bit 0 o 1, P_2 può estrarre quella che corrisponde al proprio bit!
- In questo modo P_2 ne prende *una ed una soltanto* e P_1 non sapra' quale ha preso

Riassumendo

- P_1 prepara il circuito, le scatole e gli n vasi magici
- P_1 manda a P_2 le chiavi che corrispondono al proprio input
- P_2 recupera dai vasi magici le chiavi associate al proprio input che, insieme a quelle che P_1 gli ha inviato, gli permettono di aprire in successione tutte le scatole associate alle porte del circuito e di ottenere il valore di output $f(x, y)$
- P_2 invia $f(x, y)$ a P_1 o le chiavi dell'ultimo livello per permettere a P_1 stesso di aprire le scatole singole

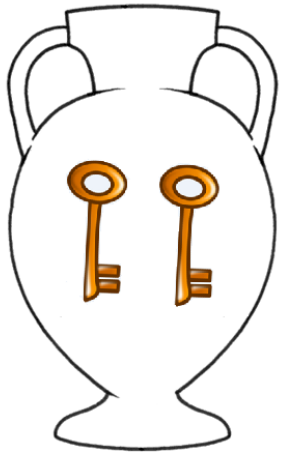
La computazione non rivela informazioni sugli input di P_1 e di P_2. Pertanto la privacy è preservata

Scatole in digitale...



- $E_{K^0_{\omega_1}}(E_{K^1_{\omega_2}}(K^1_{\omega_3}))$
- Cifratura con decifratura *"non ambigua"*
- 4 scatole $\equiv$ 4 cifrature doppie, C_0, C_1, C_2, C_3, permutate a caso

Vasi Magici in digitale...



- Protocollo Oblivious Transfer 1-su-2

	Sender		Receiver
INPUT	m_0		$b \in 0,1$
	m_1		
OUTPUT	$\perp$		m_b

- Sia $F = \{f_K\}_K$ una famiglia di funzioni pseudo-casuali (PRF in breve) tali che

$$f_{K \in \{0,1\}^n} : \{0,1\}^n \rightarrow \{0,1\}^{2n}$$

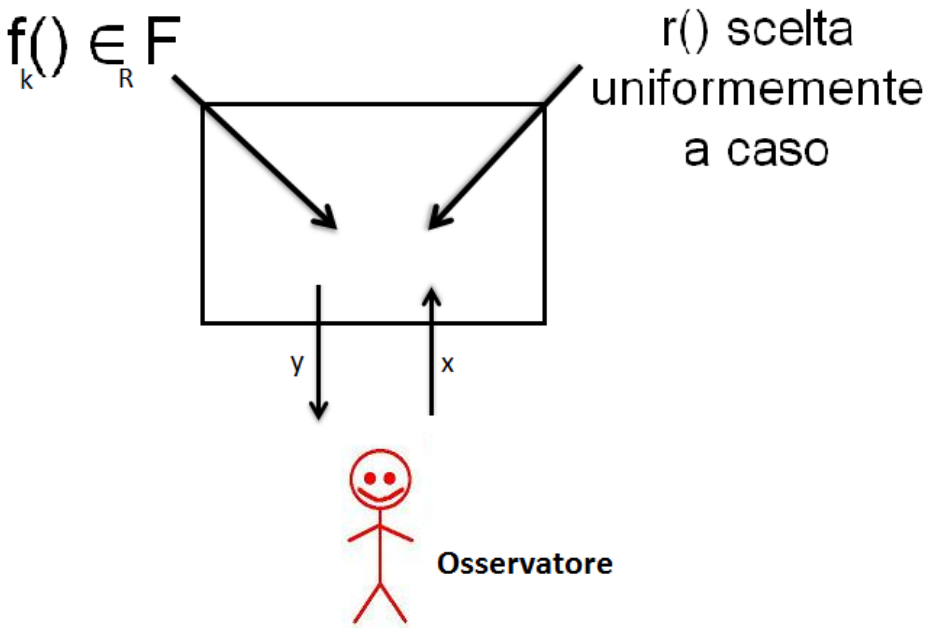
- La cifratura può essere realizzata come segue

$$E_K(x) = (r, f_K(r) \oplus x0^n)$$

dove $x \in \{0,1\}^n$ e $r \in \{0,1\}^n$ scelto uniformemente a caso

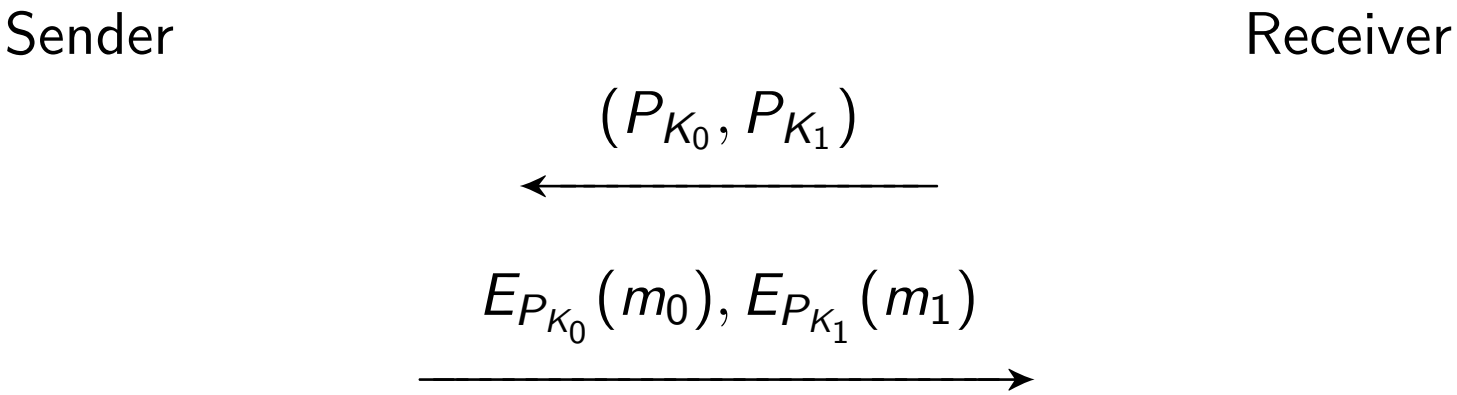
- Chi decifra (r, c), calcola $f_K(r) \oplus c = f_K(r) \oplus (f_K(r) \oplus x0_n)$ e controlla la correttezza dell'operazione verificando che il messaggio ottenuto abbia n bit uguali a 0 alla fine

Che cos'è una famiglia di funzioni Pseudo-casuali?



- Un osservatore che riesce a fare solo "computazioni efficienti" (randomizzate e che richiedono tempo polinominale) non riesce a capire, con probabilità significativamente superiore ad $\frac{1}{2}$, se nella scatola c'è $f_k()$ o $r()$

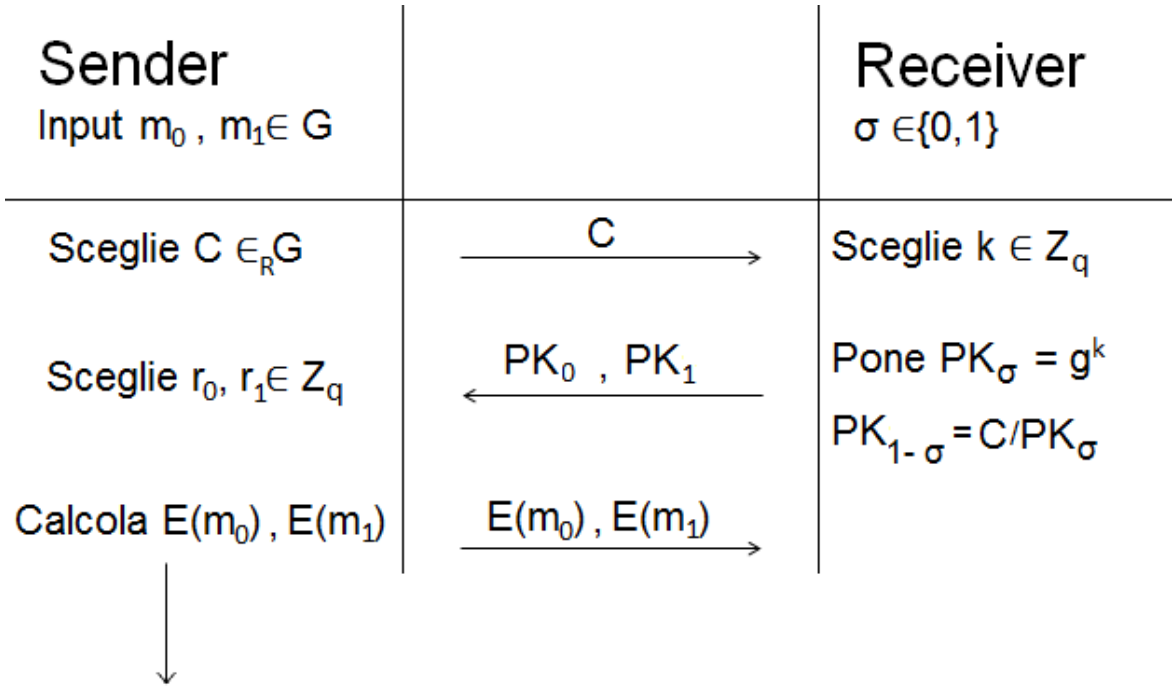
Oblivious Transfer: idea



- Il receiver possiede due chiavi pubbliche P_{K_0} e P_{K_1}, ma *una sola* chiave privata, per esempio, S_{K_0} associata a P_{K_0}. Pertanto, a seconda del messaggio che vuole ottenere, le invia nell'ordine opportuno. Così riesce a decifrare il messaggio che desidera e solo quello

OT: istanza usando il crittosistema di ElGamal

- Sia (G, q, g) un gruppo di ordine primo q con generatore g



$$E(m_0) = (g^{r_0}, (PK_0)^{r_0} \oplus m_0) = (g^{r_0}, c_0)$$
$$E(m_1) = (g^{r_1}, (PK_1)^{r_1} \oplus m_1) = (g^{r_1}, c_1)$$

Come il Receiver calcola m_σ?

- Calcola $(g^{r_\sigma})^K = (g^K)^{r_\sigma} = (PK_\sigma)^{r_\sigma}$ e quindi:
  $\Rightarrow \frac{C_\sigma}{(g^{r_\sigma})^K} = \frac{(PK_\sigma)^{r_\sigma} \oplus m_\sigma}{(PK_\sigma)^{r_\sigma}} = m_\sigma$
- Perchè non riesce a calcolare $m_{1-\sigma}$? Intuizione...
  $\Rightarrow$ Nota che $C = g^S$ per qualche $S \in Z_q$
  $\Rightarrow$ Se il DLP è difficile, il Receiver non riesce a calcolarlo
  $\Rightarrow$ Pertanto non può calcolare $(PK_{1-\sigma})^{r_{1-\sigma}} = (g^{r_{1-\sigma}})^{S-K}$ per recuperare $m_{1-\sigma}$

Protocollo di Yao: passi

- Input: P_1 ha $x \in \{0,1\}^n$, P_2 ha $y \in \{0,1\}^n$
- Input ausiliario: Circuito booleano C tale che $\forall x, y \in \{0,1\}^n$, risulta $C(x,y) = f(x,y)$ dove

$$f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$$

Protocollo:

1. P_1 costruisce il circuito cifrato $G(C(x,y))$
2. Siano $\omega_1, \omega_2 \ldots \omega_n$ i fili di input associati a $x_1, x_2 \ldots x_n$ e $\omega_{n+1} \ldots \omega_{2n}$ i fili di input associati a $y_1, y_2 \ldots y_n$:
   2.1 P_1 invia a P_2 le chiavi $K_1^{x_1} \ldots K_n^{x_n}$
   2.2 Per $i = 1, \ldots, n$, P_1 e P_2 eseguono OT 1-su-2 in cui i segreti di P_1 sono (K_{n+i}^0, K_{n+i}^1) e il bit di scelta di P_2 è y_i
3. P_2, ottenuto il circuito $G(C(x,y))$ e le chiavi associate ai fili di input, calcola il circuito e ottiene $f(x,y)$. Infine, invia $f(x,y)$ a P_1

Riferimenti

Y. Lindell and B. Pinkas,
A Proof of Security of Yao's Protocol for Two-Party Computation, Journal of Cryptology, n. 22, pp. 161-188, 2009

M. Naor, **Foundations of Cryptography, Lecture 15: Oblivious Transfer and Secure Function Evaluation**, reperibile al sito `http://www.wisdom.weizmann.ac.il/~naor/`