Secure two-party Computation

Oblivious Transfer and Secure Function Evaluation

Panagiotis Grontas

Network Algorithms and Complexity

22.07.2014

Secure Multi Party Computation

* m parties want to jointly compute the function $f(x_1, x_2, \dots, x_m)$
* Each P_i contributes x_i
* Can it be done?
  * Without releasing no other information (x_i) except the result
  * What is the computational complexity
  * What is the communication complexity
* Generalization
  * Each party has its own function
  * But requires input from all other
* Using a trusted third party is not acceptable

The millionaire problem

- Yao 1982
- Two millionaires want to find out who is richer
  - Without revealing their fortunes
- A case of SMP:
  - *m=2 (Alice and Bob)*
  - $f(a, b,) = if\ \ a < b\ then\ 1\ else\ 0$
  - a, b are bounded in range 1 to n

Yao's First Solution

* **Bob**
  * 'creates' n identical boxes
  * selects a number and puts it in box number b
  * Fills the rest of the boxes randomly
* **Alice**
  * Receives the boxes and opens all of them
  * Leaves the first a boxes unchanged
  * Increments the rest $n - a$
  * Sends them to Bob
* **Bob** reviews the boxes
  * **If his number is unchanged, Alice is richer**
  * **If his number is incremented, Bob is richer**

Problems

Exponential Number Of Boxes

Somebody deviates from the protocol

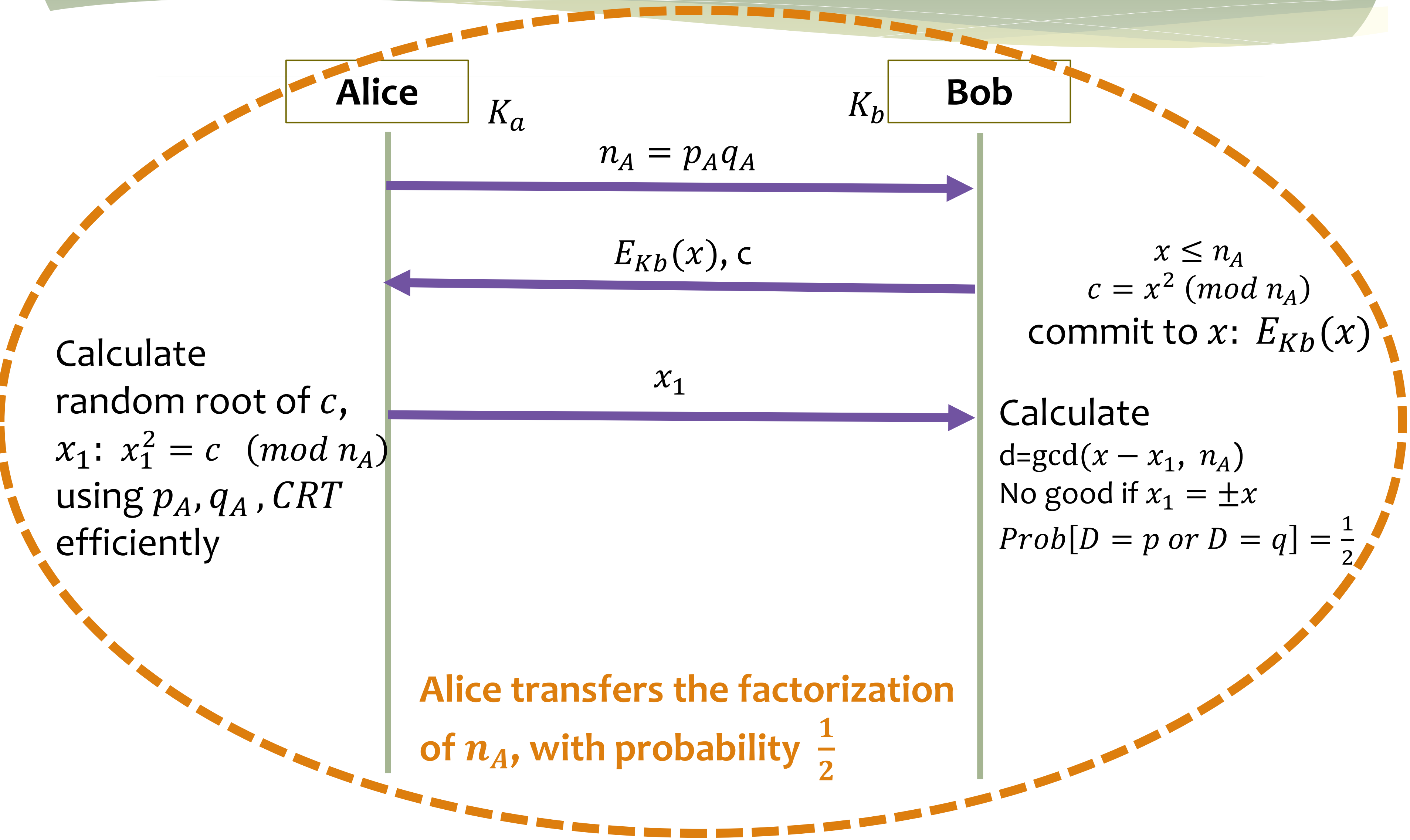
Exchange of secrets

* Alice and Bob want to exchange secrets s_a, s_b (without a TTP)
* Problems
  * Cheating:
    * Receive but not send or send invalid
  * Timing:
    * The exchange must be simultaneous
* Any EOS protocol is problematic
  * $s_a = f(a_1, a_2, \ldots, a_n)$
  * $s_b = g(b_1, b_2, \ldots, b_n)$
  * There is a k such that s_A can be computed from $a_1, a_2, \ldots, a_k$ but s_B cannot be computed from $b_1, b_2, \ldots, b_{k-1}$

Oblivious transfer

* **Solution:**
  * Construct an EOS protocol such that if Bob knows s_a, Alice can construct s_b.
* **(Real world) assumptions:**
  * Alice will find out if Bob learns her secret
  * Use of an invalid secret will make it useless
* **Primitive: Oblivious Transfer**
  * The sender of a message does not know if the recipient received the information or not
* **First implementation:**
  * Quadratic residues (Rabin)

Rabin's Protocol for OT



Rabin's Protocol for EOS

Alice | **Bob**

Oblivious transfer of $\boldsymbol{n_A}$

Oblivious transfer of $\boldsymbol{n_B}$

$v_a = 0\ iff\ factors\ where\ transferred$

$\varepsilon_a = v_a \oplus s_a$

$v_b = 0\ iff\ factors\ where\ transferred$

$\varepsilon_b = v_b \oplus s_b$

$\varepsilon_a, \varepsilon_b$

$embed\ s_a$ in m_a

$c_a = E_{n_A}(m_a)$

$embed\ s_b$ in m_b

$c_b = E_{n_B}(m_b)$

c_a, c_b

$v_a = 0$

$\rightarrow decryption\ of\ c_b$

$v_b = 0$

$\rightarrow decryption\ of\ c_a$

If Bob decrypts c_a then Alice knows that $v_b = 0$ and as a result $\varepsilon_b = s_b$

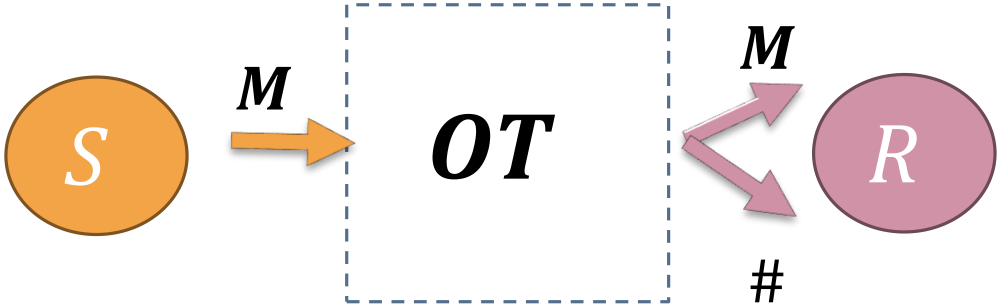
Formalisation

(Even, Goldreich, Lempel)

An oblivious transfer $OT(S, R, M)$ of a message M is a protocol by which a sender S, transfers to a receiver R the message M st:

- R gets M with probability ½
- The a-posteriori probability that R got M for S is 1/2

- If R does not receive the message he gains no helpful partial information
- Any attempt from S to deviate from the protocol is detected by R
- Formalisation of a noisy wire



1-out-of-2 Oblivious Transfer

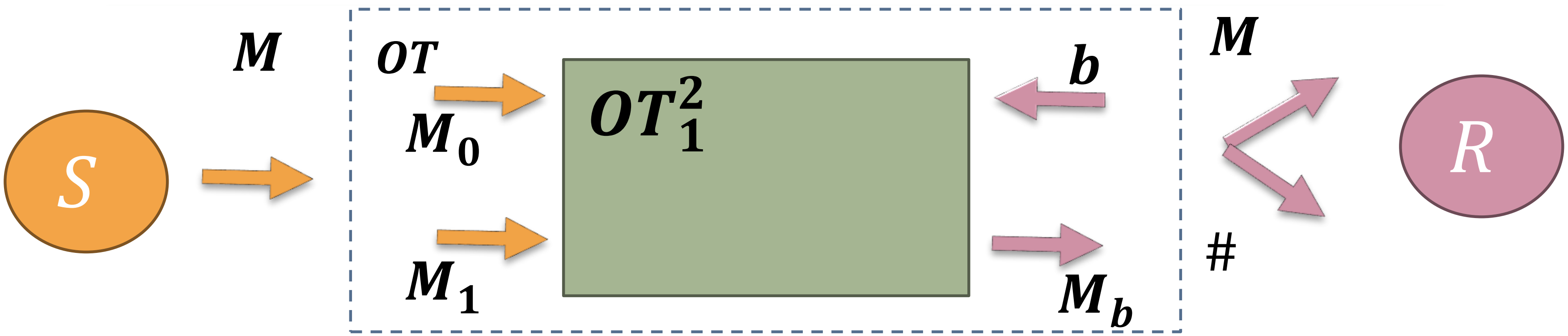
$\boldsymbol{OT_1^2}$ $(\boldsymbol{S, R, M_0, M_1})$: A protocol by which a sender S transfers ignorantly to a receiver R one message out of two.

R selects which message to receive without S learning it

R requests each message with probability 1/2

Result: $OT\ and\ OT_1^2$ are equivalent

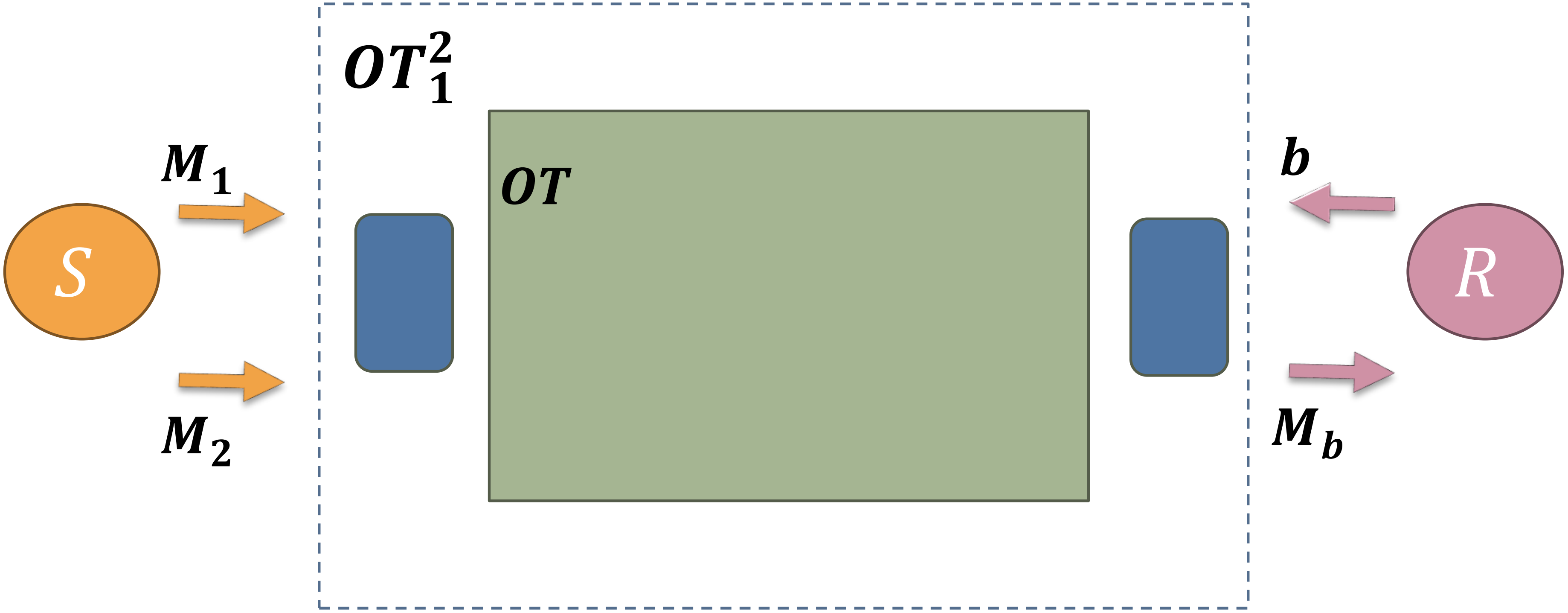
OT from OT_1^2 (EGL)



- S wants to transmit $\boldsymbol{M}$ with probability ½ to R
- $\boldsymbol{OT}$ machine flips bits $\boldsymbol{M_0}$, $\boldsymbol{M_1}$ and $\boldsymbol{b}$.
- If b = 0 then send ($\boldsymbol{M}$, $\boldsymbol{M_1}$) to $\boldsymbol{OT_1^2}$ machine
- If b = 1 then send ($\boldsymbol{M_1}$, $\boldsymbol{M}$) to $\boldsymbol{OT_1^2}$ machine

OT_1^2 from OT (Crepeau)

* ***Random OT (R-OT)***: OT with transfer probability p
* $\boldsymbol{OT_1^2}$ can be implemented using ***R -OT (OT)***



OT_1^2 from OT (Crepeau)

- The OT protocol is applied on bit vector $\vec{s}$.
  - Objective: Transfer $\approx n$
  - $\overrightarrow{|s|}$ = $3n$
- R inputs selector bit b
  - It is replaced with 2 sets of indices of length n
    - I_b: The positions in $\vec{s}$ where the transfer succeeded
    - I_{1-b}: Random Positions in $\vec{s}$
- S sends (M_0, M_1)
  - $M_b(\bigoplus_{\boldsymbol{i}\in I_b}\vec{s}_{\boldsymbol{i}})$ is actually sent for $b = 0,1$

OT_1^2 from OT (Crepeau, 1998)

- **Analysis**
  - I_b can be found wvhp
  - I_{1-b} contains at least one position where OT failed wvhp
  - OT Failure => XOR calculation Failure
  - **Exactly one of them can be calculated**
- Exactly one of $\boldsymbol{M_0}, \boldsymbol{M_1}$ can be transferred
- We have $\boldsymbol{OT_1^2}$

Other flavors

* 1-out-of-N oblivious transfer $\boldsymbol{OT_1^n}$
  * S has $\boldsymbol{M_1, M_2, \dots, M_N}$
  * R selects i and receives $\boldsymbol{M_i}$
  * S does not learn i
  * R does not learn $\boldsymbol{M_{j,j \neq i}}$
* k-out-of-N oblivious transfer $\boldsymbol{OT_k^n}$
  * Simultaneously receive k messages
* k-out-of-N adaptive oblivious transfer $\boldsymbol{OT_k^n}$
  * Successive oblivious transfers
  * Selection at each stage depends on messages previously received
* Constructed using $\boldsymbol{OT_1^2}$

Generic Implementation of OT_1^2

- S, R agree on a $PKCS$ (K, E, D) where $M = C$ (eg. RSA)
- S, R are semi-honest
- Objective: Obliviously transmit $\boldsymbol{m_0}, \boldsymbol{m_1}$
- R generates 2 random strings $\boldsymbol{x_0}, \boldsymbol{x_1}$
- To obtain $\boldsymbol{m_0}$:
  - R sends $(\boldsymbol{E}(\boldsymbol{x_0}), \boldsymbol{x_1})$
  - S decrypts $(\boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x_0})), \boldsymbol{D}(\boldsymbol{x_1}))$ = $(\boldsymbol{x_0}, \boldsymbol{D}(\boldsymbol{x_1}))$
  - S applies XOR to tuple $(\boldsymbol{m_0} \oplus \boldsymbol{x_0}, \boldsymbol{m_1} \oplus \boldsymbol{D}(\boldsymbol{x_1}))$
  - R retrieves $\boldsymbol{m_0}$ by XORing again

$$(\boldsymbol{m_0} \oplus \boldsymbol{x_0} \oplus \boldsymbol{x_0}, \boldsymbol{m_1} \oplus \boldsymbol{D}(\boldsymbol{x_1}) \oplus \boldsymbol{x_1})$$

OT and SFE: Yao's construction

- Oblivious transfer implies secure function evaluation
- Use oblivious transfer to compute any function f
  - Express f as a circuit C
  - Construct a protocol that computes C
    - Parties provide inputs
    - They only learn the output
    - All intermediate values are never revealed
      - Random inputs
      - Random outputs
    - **Garbled** truth tables
- Security against semi – honest (passive) players

An OR Gate with OT

- S contributes s and R contributes r
- **Step 1: S transforms truth table**
  - selects random permutations $v: \{0,1\} \rightarrow \{0,1\}$
  - Applies permutations to truth table
  - Selects 4 encryption - decryption functions (E_0^S, D_0^S), (E_1^S, D_1^S), (E_0^R, D_0^R), (E_1^R, D_1^R)
  - Applies encryption functions to the result according to the position
  - Send the table and v_r to the R

s	r	s OR r
0	0	0
0	1	1
1	0	1
1	1	1

s	r	s OR r
$v_s(0)$	$v_r(0)$	$E_{v_s(0)}^S(E_{v_R(0)}^R(0))$
$v_s(0)$	$v_r(1)$	$E_{v_s(0)}^S(E_{v_R(1)}^R(1))$
$v_s(1)$	$v_r(0)$	$E_{v_s(1)}^S(E_{v_R(0)}^R(1))$
$v_s(1)$	$v_r(1)$	$E_{v_s(1)}^S(E_{v_R(1)}^R(1))$

An OR Gate with OT (2)

* **Step 2: S computes its part**
  * $v_S(\mathrm{s})$
  * Sends $(v_S(\mathrm{s}), D^S_{v_S(\mathrm{s})})$
* **Step 3: R computes its part**
  * $v_R(r)$
  * In order to decrypt $D^R_{\mathrm{v}_R(\mathrm{r})}$ is required
  * How to get it without revealing $\mathrm{v}_R(\mathrm{r})$?
  * $\boldsymbol{OT_1^2}\,(\boldsymbol{S}, \boldsymbol{R}, D_0^R, D_1^R)$

s	**r**	**s** *OR* **r**
$v_s(0)$	$v_r(0)$	$E^S_{v_s(0)}(E^R_{v_R(0)}(0))$
$v_s(0)$	$v_r(1)$	$E^S_{v_s(0)}(E^R_{v_R(1)}(1))$
$v_s(1)$	$v_r(0)$	$E^S_{v_s(1)}(E^R_{v_R(0)}(1))$
$v_s(1)$	$v_r(1)$	$E^S_{v_s(1)}(E^R_{v_R(1)}(1))$

Finally: Peel off the desired row

$$D^R_{v_R(r)}(D^S_{v_s(0)}(E^S_{v_s(0)}(E^R_{v_R(1)}(1))))$$

and informs R

Question: Why not send both D_0^R, D_1^R?

In reality …

- The rows of the table are randomly permuted
- The result is a random permutation as well
- **View everything as keys (6 keys / gate)**

s	r	s OR r	Computation
k_0^S	k_0^R	k_0^{OR}	$E_{k_0^S}(E_{k_0^R}(k_0^{OR}))$
k_0^S	k_1^R	k_1^{OR}	$E_{k_0^S}(E_{k_1^R}(k_1^{OR}))$
k_1^S	k_0^R	k_1^{OR}	$E_{k_1^S}(E_{k_0^R}(k_1^{OR}))$
k_1^S	k_1^R	k_1^{OR}	$E_{k_1^S}(E_{k_1^R}(k_1^{OR}))$

Building up the circuit

- After computing each gate g, both S, R have access to k_x^g
- This is used as input to another gate
- The output gates will contain the circuit's output w_i
- Each digit is decrypted using output tables
- S constructs the circuit
  - In case of multiple inputs, copy the key
  - In case of multiple inputs, same output key
- R uses oblivious transfer for each bit of its input
- And computes the result
- Complexity:
  - Computation: $O(|C|)$ - 6 keys per gate / 8 encryptions per gate / 2 decryptions per gate
  - Communication: $O(|C|)$ Round complexity: constant
- Proof of security and correctness (*Lindell, Pinkas 2006*)

Bibliography

* Yao, A. C. **"Protocols for secure computations“** (FOCS 1982): 160–164
* Rabin M. O. **"How to exchange secrets by oblivious transfer."** ,TR-81, Harvard University, 1981
* S. Even, O. Goldreich, and A. Lempel. 1985. **A randomized protocol for signing contracts.** *Commun. ACM* 28, 6 (June 1985), 637-647
* Claude Crépeau. 1987. **Equivalence Between Two Flavours of Oblivious Transfers.** In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology* (CRYPTO '87, UK, 350-354.
* Yehuda Lindell and Benny Pinkas. 2009. **A Proof of Security of Yao’s Protocol for Two-Party Computation.** *J. Cryptol.* 22, 2 (April 2009), 161-188
* Ostrofski R., CS 282A/MATH 209A: Foundations of Cryptography, **Lecture 10, Oblivious Transfer**
* Gabriel Bender, **Cryptography and Secure Two-Party Computation,** August 21, 2006, http://www.math.uchicago.edu/~may/VIGRE/VIGRE2006/PAPERS/Bender.pdf