CIS 5560

Cryptography
Lecture 24

Course website:
pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW10** due **Thursday Apr 25** at 11:59PM on Gradescope
- **HW11 will be released tomorrow evening**

Recap of Last Lecture

- Complete proof of ZK for 3COL
- Succinct Arguments
- PCPs
- Kilian construction of succinct arguments from PCPs

Why is 3COL Protocol ZK?

Simulator S works as follows:

1. First pick a random edge (i^*, j^*)

   Color vertices i^* and j^* with random, different colors
   Color all other vertices red.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$

edge (i, j)

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

send openings r_i and r_j

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

Why is this zero-knowledge?

Simulator S works as follows (call this Hybrid 0)

1. First pick a random edge (i^*, j^*)

   Color vertices i^* and j^* with random, different colors
   Color all other vertices red.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$

2. Feed the commitments of the colors to V^* and get edge (i, j)

edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

send openings r_i and r_j

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

Why is this zero-knowledge?

Not-a-Simulator S works as follows (call this Hybrid 1)

1. First pick a random edge (i^*, j^*)

   Permute a legal coloring and color all vertices correctly.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$

edge (i, j)

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

send openings r_i and r_j

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

Why is this zero-knowledge?

Here is the real view of V* (Hybrid 2)

1. ~~First pick a random edge~~ $\cancel{(i^*, j^*)}$

   Permute a legal coloring and color all edges correctly.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$

edge (i, j)

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. ~~If~~ $\cancel{(i, j) \neq (i^*, j^*)}$, ~~go back and repeat.~~

send openings r_i and r_j

4. ~~If~~ $\cancel{(i, j) = (i^*, j^*)}$, output the commitments and openings r_i and r_j as the transcript.

Today’s Lecture

- Secure Multi-party Computation

Secure Computation

Input: x **Input:** y

Alice Bob

Output: $F_A(x, y)$ **Output**: $F_B(x, y)$

Secure Two-Party Computation

Input: x

Input: y

Alice

Bob

Output: $F_A(x, y)$

Output: $F_B(x, y)$

Security:

Semi-honest Security:

- Alice should not learn anything more than x and $F_A(x, y)$.
- Bob should not learn anything more than y and $F_B(x, y)$.

Secure Two-Party Computation

Input: x

Alice

Output: $F_A(x, y)$

Input: y

Bob

Output: $F_B(x, y)$

Malicious Security:

- No (PPT) Alice* can learn anything more than x^* and $F_A(x^*, y)$.
- No (PPT) Bob* can learn anything more than y^* and $F_B(x, y^*)$.

Tool 1: Secret Sharing

Secret Sharing

- ❑ Any **"authorized"** subset of players **can recover** b.
- ❑ No other subset of players **has any info** about b.

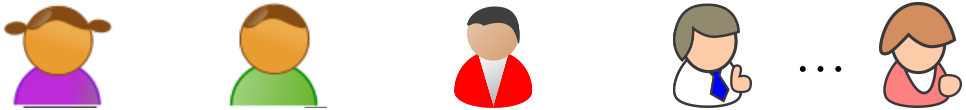
- ○ Threshold (or t-out-of-n) SS [Shamir'79, Blakley'79]:

  "authorized" subset = has size $\geq$ t.

secret b $\in Z_p$

n-out-of-n Secret Sharing

Dealer



P_1 P_2 P_3 P_4 … P_n

share s_1: random

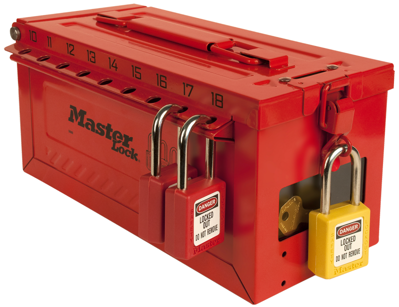
share s_2: random

share s_3: random

share s_4: random

…

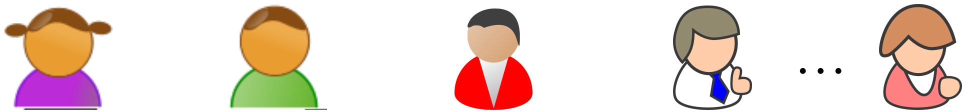
share $s_n = b - (s_1 + s_2 + \ldots + s_{n-1})$ mod p



1-out-of-n Secret Sharing

secret b $\in Z_p$

Dealer



P_1 P_2 P_3 P_4 $\cdots$ P_n

share $s_1 = \mathrm{b}$

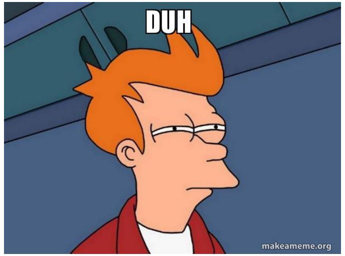
share $s_2 = \mathrm{b}$

share $s_3 = \mathrm{b}$

share $s_4 = \mathrm{b}$

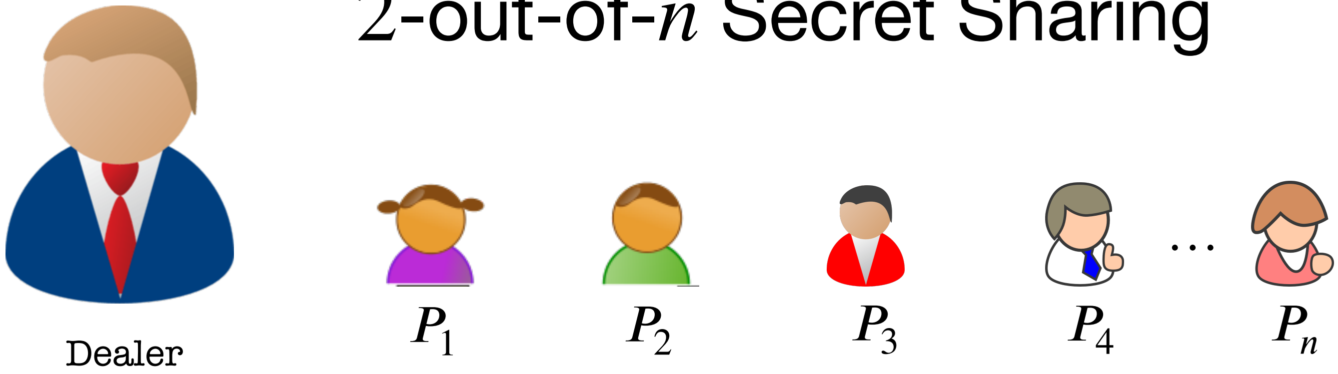
$\cdots$

share $s_n = b$



secret b $\in Z_p$

2-out-of-n Secret Sharing



Here is a solution.

Repeat for every two-person subset $\{P_i, P_j\}$:
Generate a 2-out-of-2 secret sharing (s_i, s_j) of b.
Give s_i to P_i and s_j to P_j

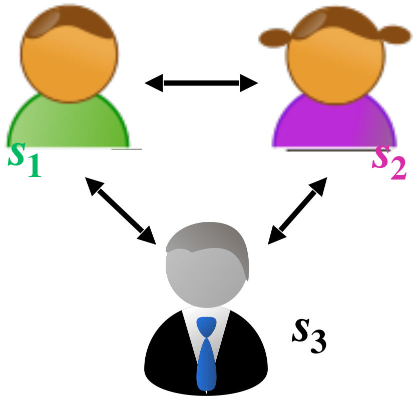
What is the size of shares each party gets?

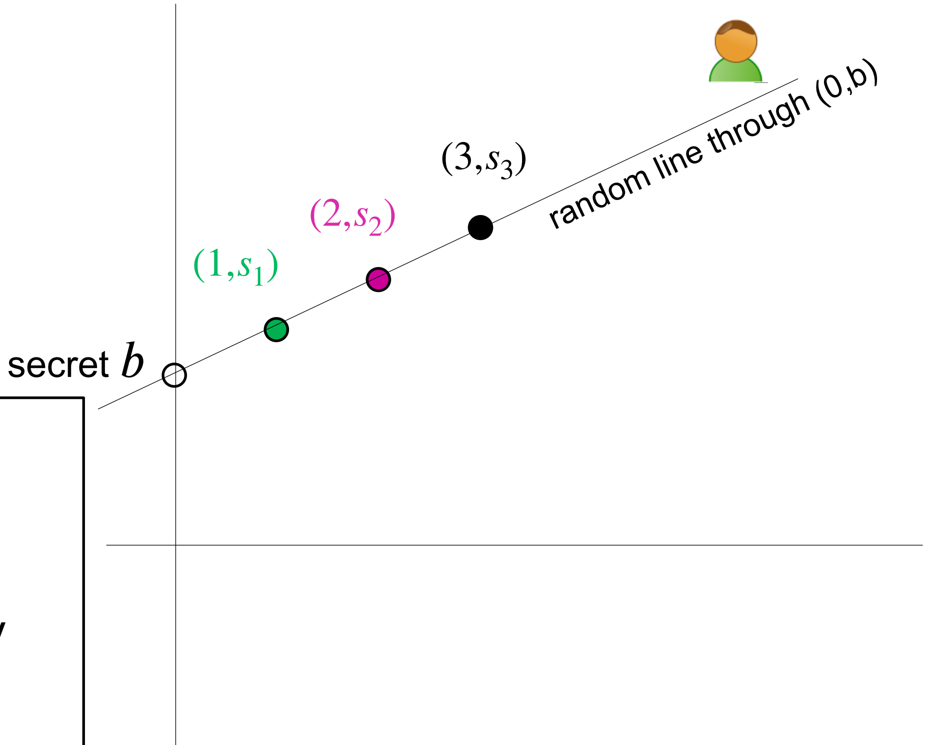
How does this scale to t-out-of-n?

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

Shamir’s 2-out-of-n Secret Sharing





Each share s_i is truly random (independent of secret b)

Any two shares uniquely determine b.

Shamir's 2-out-of-n Secret Sharing

1. The dealer picks a uniformly random line **(mod p)** whose constant term is the secret b.

$f(x) = ax + b$ where a is uniformly random mod p

2. Compute the shares:

$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$

Correctness: can recover secret from any two shares.

Proof: Parties i and j, given shares $s_i = ai + b$ and $s_j = aj + b$ can solve for b $\left(= \dfrac{js_i - is_j}{j - i}\right)$.

Shamir's 2-out-of-n Secret Sharing

1. The dealer picks a uniformly random line **(mod p)** whose constant term is the secret b .

$f(x) = ax + b$ where a is uniformly random mod p

2. Compute the shares:

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Security: any single party has no information about the secret.

Proof: Party i's share $s_i = a * i + b$ is uniformly random, independent of b, as a is random and so is $a * i$.

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

1. The dealer picks a uniformly random degree-(t-1) polynomial **(mod p)** whose constant term is the secret b.

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

2. Compute the shares:

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Correctness: can recover secret from any t shares.

Security: the distribution of any $t-1$ shares is independent of the secret.

Note: need p to be larger than the number of parties n.

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Correctness: *via Vandermonde matrices*.

Let's look at shares of parties $P_1, P_2, \ldots, P_t$.

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \ldots \\ s_t \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 2^2 & \ldots & 2^{t-1} \\ 1 & 3 & 3^2 & \ldots & 3^{t-1} \\ 1 & \ldots & \ldots & \ldots & \ldots \\ 1 & t & t^2 & \ldots & t^{t-1} \end{bmatrix} \begin{bmatrix} b \\ a_1 \\ a_2 \\ \ldots \\ a_{t-1} \end{bmatrix} \pmod{p}$$

*t-by-t Vandermonde matrix which is **invertible***

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Correctness: Alternatively, *Lagrange interpolation* gives an explicit formula that recovers b.

$$b = f(0) = \sum_{i=1}^{t} f(i) \left(\prod_{1 \le j \le t, j \ne i} \frac{-x_j}{x_i - x_j} \right)$$

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Security:

Let's look at shares of parties $P_1, P_2, \ldots, P_{t-1}$.

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \ldots \\ s_{t-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 2^2 & \ldots & 2^{t-1} \\ 1 & 3 & 3^2 & \ldots & 3^{t-1} \\ 1 & \ldots & \ldots & \ldots & \ldots \\ 1 & t-1 & (t-1)^2 & \ldots & (t-1)^{t-1} \end{bmatrix} \begin{bmatrix} b \\ a_1 \\ a_2 \\ \ldots \\ a_{t-1} \end{bmatrix} \pmod{p}$$

$(t-1)$-by-t Vandermonde matrix

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Security: For every value of b there is a unique polynomial with constant term b and agrees with f on $s_1, \ldots, s_{t-1}$.

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \ldots \\ s_{t-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 2^2 & \ldots & 2^{t-1} \\ 1 & 3 & 3^2 & \ldots & 3^{t-1} \\ 1 & \ldots & \ldots & \ldots & \ldots \\ 1 & t-1 & (t-1)^2 & \ldots & (t-1)^{t-1} \end{bmatrix} \begin{bmatrix} b \\ a_1 \\ a_2 \\ \ldots \\ a_{t-1} \end{bmatrix} \pmod{p}$$

$(t-1)$-by-t Vandermonde matrix

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

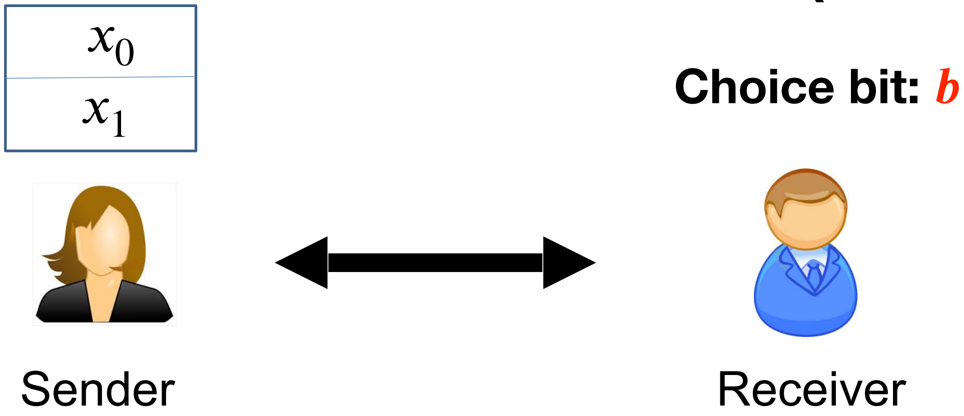
$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Security: For every value of b there is a unique polynomial with constant term b and agrees with f on $s_1, \ldots, s_{t-1}$.

Corollary: for every value of the secret b is equally likely given the shares $s_1, s_2, \ldots, s_{t-1}$. In other words, the secret b is perfectly hidden given $t-1$ shares.

Tool 2: Oblivious Transfer

Oblivious Transfer (OT)



- Sender holds two bits/strings x_0 and x_1.
- Receiver holds a choice bit b.
- Receiver should learn x_b, sender should learn nothing.

(We will consider **honest-but-curious** adversaries; formal definition in a little bit…)

Why OT? Computing ANDs

$\alpha \in \{0,1\}$

Alice and Bob want to compute the AND $\alpha \wedge \beta$.

$\beta \in \{0,1\}$

Why OT? Computing ANDs

$\alpha \in \{0,1\}$

Alice and Bob want to compute the AND $\alpha \wedge \beta$.

$\beta \in \{0,1\}$

$x_0 = 0$
$x_1 = \alpha$

Run an OT protocol

Choice bit $b = \beta$

Bob gets α if $\beta = 1$, and 0 if $\beta = 0$

Here is a way to write the OT selection function: $\boldsymbol{x_1 b + x_0(1 - b)}$

which, in this case is $= \alpha\beta$.

The Billionaires' Problem

Net worth: $X

Net worth: $Y

Who is richer?

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

X

Y

$\cdots$	0	1	0	0	$\cdots$

$\cdots$	0	1	1	1	1	1	1

Unit Vector u_X = 1 in the X^{th} location and 0 elsewhere

Vector v_Y = 1 from the $(Y+1)^{th}$ location onwards

$$f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]$$

~~Compute each AND individually and sum it up~~?

Detour: OT $\Longrightarrow$ Secret-Shared-AND

$\alpha \in \{0,1\}$

Alice gets random γ, Bob gets random δ s.t. $\gamma \oplus \delta = \alpha\beta$.

$\beta \in \{0,1\}$

Output: γ

Output: δ

$x_0 = \gamma$
$x_1 = a \oplus \gamma$

Run an OT protocol $\longleftrightarrow$

Choice bit $b = \beta$

Alice outputs γ.

Bob gets $x_1 b + x_0(1 \oplus b) = (x_1 \oplus x_0)b + x_0 = \alpha\beta \oplus \gamma := \delta$

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

$\cdots$	0	1	0	0	$\cdots$

Unit Vector u_X

$\cdots$	0	1	1	1	1	1	1

Vector v_Y

$$\boldsymbol{f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]}$$

1. Alice and Bob run many OTs to get (γ_i, δ_i) s.t. $\gamma_i \bigoplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$

2. Alice computes $\gamma = \oplus_i \gamma_i$ and Bob computes $\delta = \oplus_i \delta_i$

3. Alice reveals γ and Bob reveals δ.

Check (correctness): $\gamma \oplus \delta = \langle u_X, v_Y \rangle = f(X, Y)$.

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

$\cdots$	0	1	0	0	$\cdots$

Unit Vector u_X

$\cdots$	0	1	1	1	1	1	1

Vector v_Y

$$f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]$$

1. Alice and Bob run many OTs to get (γ_i, δ_i) s.t. $\gamma_i \bigoplus \delta_i = u_X[i] \wedge v_Y[i]$

2. Alice computes $\gamma = \oplus_i \gamma_i$ and Bob computes $\delta = \oplus_i \delta_i$

3. Alice reveals γ and Bob reveals δ.

Check (privacy): Alice & Bob get a bunch of random bits.

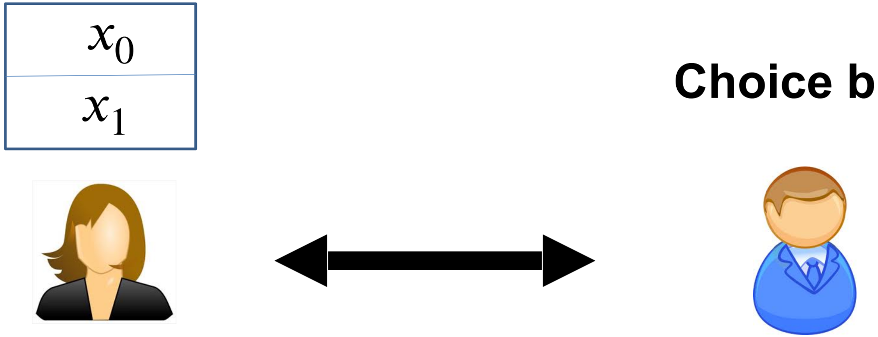
"OT is Complete"

Theorem: OT can solve not just ANDs and money, but ***any*** two-party (and multi-party) problem efficiently.

Defining Security:
The Ideal/Real Paradigm

OT Definition

Choice bit: b



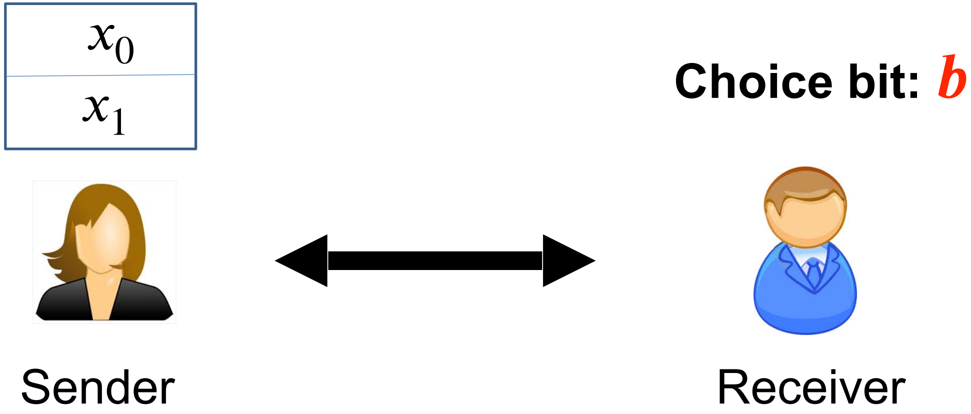
Sender

Receiver

Receiver Security: Sender should not learn b.

Define Sender's view $View_S(x_0, x_1, b)$ = her random coins and the protocol messages.

OT Definition



Receiver Security: Sender should not learn b.

There exists a PPT simulator SIM_S such that for any x_0, x_1 and b:

$$SIM_S(x_0, x_1) \cong View_S(x_0, x_1, b)$$

OT Definition

x_0
x_1

Choice bit: b

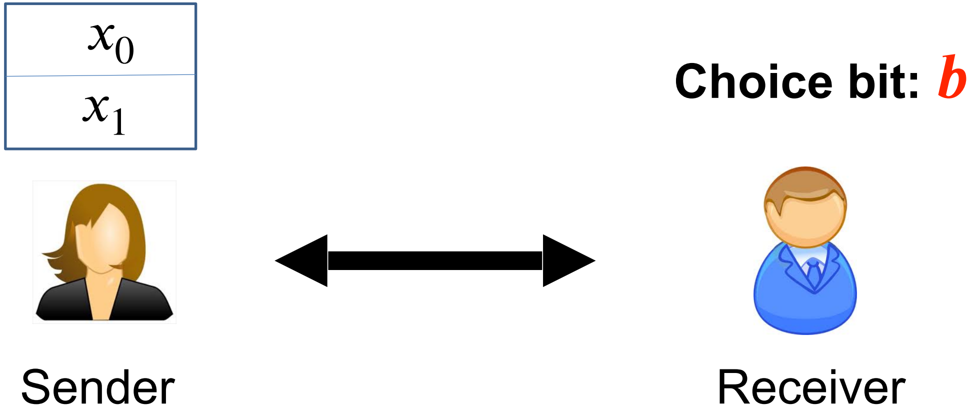
Sender

Receiver

Sender Security: Receiver should not learn x_{1-b}.

Define Receiver's view $View_R(x_0, x_1, b)$ = his random coins and the protocol messages.

OT Definition



Sender Security: Receiver should not learn x_{1-b}.

There exists a PPT simulator SIM_R such that for any x_0, x_1 and b:

$$SIM_R(b, x_b) \cong View_R(x_0, x_1, b)$$

OT Protocols

OT Protocol 1: Trapdoor Permutations

For concreteness, let’s use the RSA trapdoor permutation.

Input bits: (x_0, x_1)

Choice bit: b

Pick $N = PQ$ and RSA exponent e.

$\xrightarrow{N, e}$

Choose random r_b and set $s_b = r_b^e \bmod N$

Choose random s_{1-b}

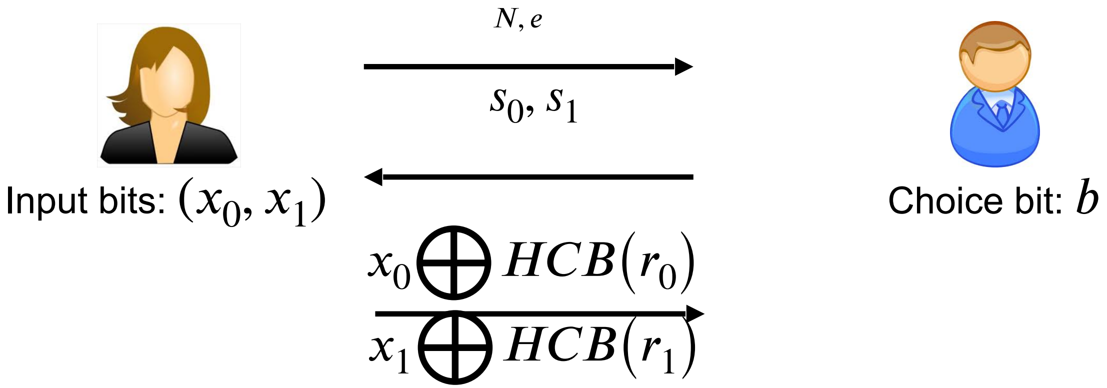
$\xleftarrow{s_0, s_1}$

Compute r_0, r_1 and XOR x_0, x_1 using hardcore bits

$\xrightarrow[x_1 \bigoplus HCB(r_1)]{x_0 \bigoplus HCB(r_0)}$

Bob can recover x_b but not x_{1-b}

OT Protocol 1: Trapdoor Permutations



How about Bob's security
(a.k.a. Why does Alice not learn Bob's choice bit)?

Alice's view is s_0, s_1 one of which is chosen randomly from Z_N^* and the other by raising a random number to the e-th power. They look exactly the same!

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

N, e →

← s_0, s_1

$x_0 \bigoplus HCB(r_0)$ →
$x_1 \bigoplus HCB(r_1)$

Choice bit: b

How about Bob's security
(a.k.a. Why does Alice not learn Bob's choice bit)?

Exercise: Show how to construct the simulator.

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

N, e $\longrightarrow$

$\longleftarrow$ s_0, s_1

$x_0 \oplus HCB(r_0)$

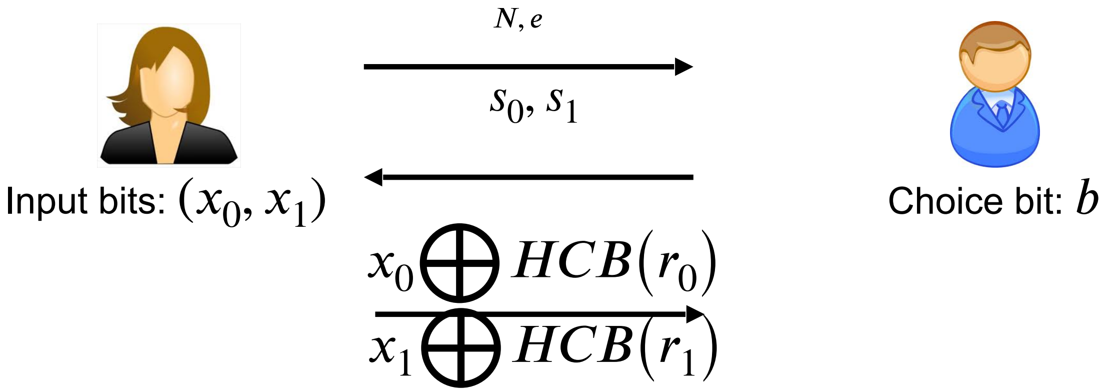
$x_1 \oplus HCB(r_1)$ $\longrightarrow$

Choice bit: b

How about Alice's security
(a.k.a. Why does Bob not learn both of Alice's bits)?

Assuming Bob is semi-honest, he chose s_{1-b} uniformly at random, so the hardcore bit of $s_{1-b} = r_{1-b}^d$ is computationally hidden from him.

OT from Trapdoor Permutations



How about Alice's security
(a.k.a. Why does Bob not learn both of Alice's bits)?

Exercise: Show how to construct the simulator.

OT Protocol 2: Additive HE

Input bits: (x_0, x_1)

Choice bit: b

Encrypt choice bit b

$c \longleftarrow \mathrm{Enc}(sk, b)$

c ←

Homomorphically evaluate the selection function

$\boldsymbol{SEL_{x_0,x_1}(b) = (x_1 \oplus x_0)b + x_0}$

$c' = \mathrm{Eval}(SEL_{x_0,x_1}(b), c)$ →

Decrypt to get x_b

Bob's security: computational, from CPA-security of Enc.

Alice's security: statistical, from function-privacy of Eval.

Many More Constructions of OT

Theorem: OT protocols can be constructed based on the hardness of the Diffie-Hellman problem, factoring, quadratic residuosity, LWE, elliptic curve isogeny problem etc. etc.

Secure 2PC from OT

Theorem **[Goldreich-Micali-Wigderson'87]**:
OT can solve ***any*** two-party computation problem.