MIT 6.875

Foundations of Cryptography

Lecture 17

An Application of NIZK:

Non-malleable and Chosen Ciphertext Secure Encryption Schemes

Non-Malleability

$m \leftarrow \mathrm{Dec}(\mathbf{sk}, c)$

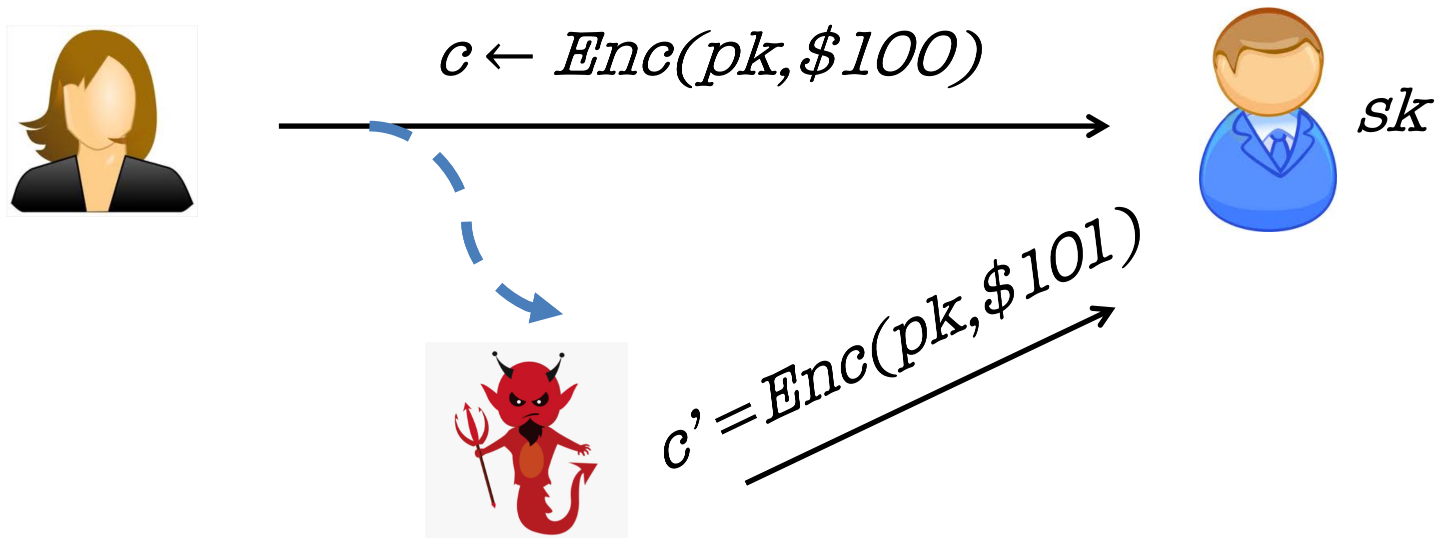
$c \leftarrow \mathrm{Enc}(\mathbf{pk}, m)$

sk

Public-key directory

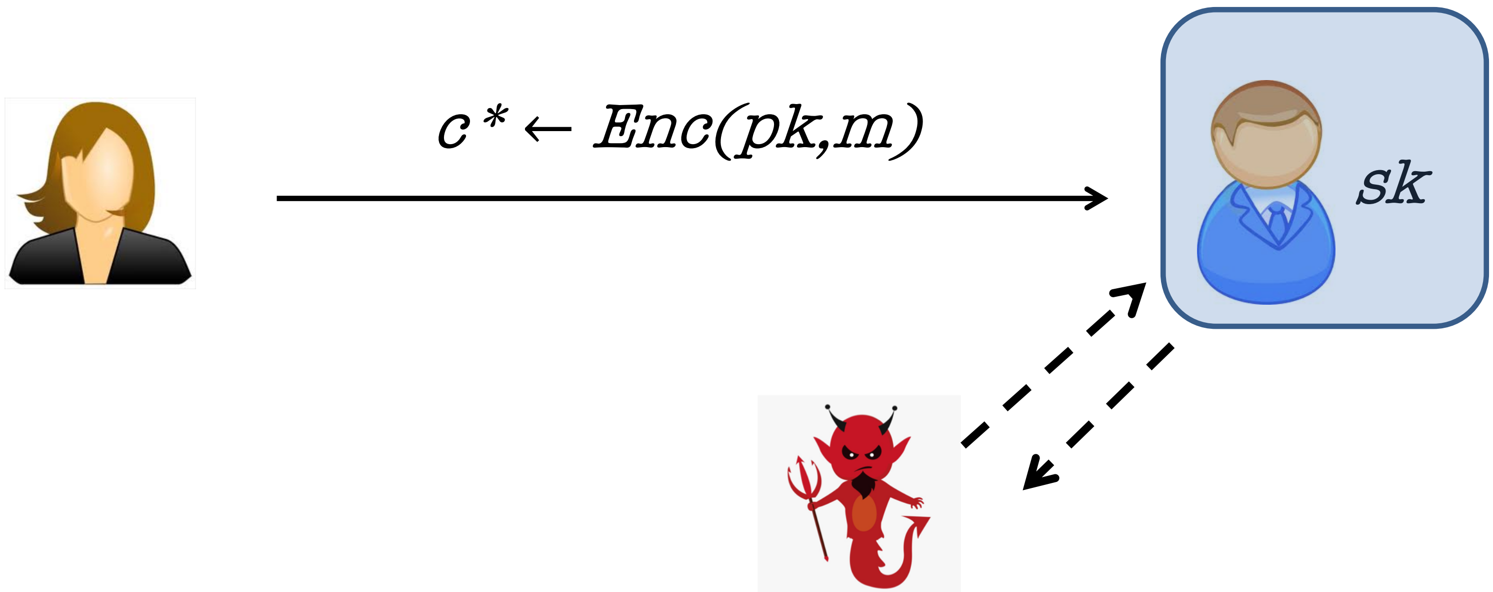
Bob	**pk**

Active Attacks 1: Malleability



ATTACK: Adversary could modify ("maul") an encryption of m into an encryption of a related message m'.

Active Attacks 2: Chosen-Ciphertext Attack



ATTACK: Adversary may have access to a decryption "oracle" and can use it to break security of a "target" ciphertext c^* or even extract the secret key!

In fact, Bleichenbacher showed how to extract the entire secret key given only a "ciphertext verification" oracle.

IND-CCA Security

Challenger

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$ $\longrightarrow$

$\boldsymbol{c_i}$ $\longleftarrow$

$\boldsymbol{Dec(sk, c_i)}$ $\longrightarrow$

$m_0^*, m_1^* \quad s.t.\ |m_0^*| = |m_1^*|$ $\longleftarrow$

$b \leftarrow \{0,1\};\ c^* \leftarrow Enc(pk, m_b^*)$

$\boldsymbol{c^*}$ $\longrightarrow$

$\boldsymbol{c_i \neq c^*}$ $\longleftarrow$

$\boldsymbol{Dec(sk, c_i)}$ $\longrightarrow$

b' $\longleftarrow$

Eve wins if $b' = b$.
IND-CCA secure if no PPT Eve can win with prob. $> \frac{1}{2} + \mathrm{negl}(n)$.

Constructing CCA-Secure Encryption (Intuition)

NIZK Proofs of Knowledge should help!

Idea: The encrypting party attaches an NIZK proof of knowledge of the underlying message to the ciphertext.

C: $(\mathrm{c} = \mathrm{CPAEnc}(m; r), \mathrm{proof}\ \pi\ \textit{that "I know m and r"})$

This idea will turn out to be useful, but NIZK proofs themselves can be malleable!

Constructing CCA-Secure Encryption (Intuition)

Digital Signatures should help!

OUR GOAL: Hard to modify an encryption of m into an encryption of a related message, say m+1.

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$C: (c = \mathrm{CPAEnc}(pk, m; r), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $(sgk, vk) \leftarrow Sign.Gen(1^n)$

Is this CCA-secure/non-malleable?

If the adversary changes vk, all bets are off!

Lesson: NEED to "tie" the ciphertext c **to vk in a "meaningful" way.**

Observation:
IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}(pk, m; r)$, **can an adversary produce** $\mathrm{CPAEnc}(pk', m+1; r)$?

NO! Suppose she could. Then, I can come up with a reduction that breaks the IND-CPA security of $\mathrm{CPAEnc}(pk, m; r)$.

Observation:
IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}(pk, m; r)$,
can an adversary produce $\mathrm{CPAEnc}(pk', m+1; r)$?

Reduction = CPA adversary

pk $\longrightarrow$

$CPAEnc(pk, m)$ $\longrightarrow$

$\boldsymbol{m}$ $\longleftarrow$

Pick $(\boldsymbol{pk}', \boldsymbol{sk}')$

pk, pk' $\longrightarrow$

$CPAEnc(pk, m)$ $\longrightarrow$

$\boldsymbol{CPAEnc(pk', m+1)}$ $\longleftarrow$

Decrypt and subtract 1.

Diff-Key NM adversary

Putting it together

CCA Public Key: $2n$ public keys of the CPA scheme

(where $n = |vk|$)

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT, vk, \sigma = Sign(sgk, CT))$.

Putting it together

Non-malleability rationale: Either

- Adversary keeps vk the same (in which case she has to break the signature scheme); or
- She changes the vk in which case she breaks the diff-NM game, and therefore CPA security.

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT, vk, \sigma = Sign(sgk, CT))$.

Call it a day?

We are not done!! Adversary could create ill-formed ciphertexts (e.g. the different cts encrypt different messages) and uses it for a Bleichenbacher-like attack.

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT, vk, \sigma = Sign(sgk, CT))$.

NIZK Proofs to the Rescue…

CCA Public Key: $2n$ public keys of the CPA scheme

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \dots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}, \color{blue}{\boldsymbol{CRS}}$$

NP statement: "there exist $m, r_{i,j}$ such that each $ct_{i,j} = CPAEnc(pk_{i,j}, m; r_{i,j})$"

key pair (sgk, vk)

$$\begin{bmatrix} \quad {}_{2,vk_2} & \dots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = NIZK proof that "CT is well-formed"

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

Are there other attacks?

Did we miss anything else?

Turns out NO. We can prove that this is CCA-secure.

The Encryption Scheme

CCA Keys:

$$\textbf{PK=}\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}, CRS \qquad \textbf{SK=}\begin{bmatrix} sk_{1,0} \\ sk_{1,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

$\pi =$ NIZK proof that "CT is well-formed"

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

The Encryption Scheme

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = NIZK proof that "CT is well-formed"

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

CCA Decryption:

Check the signature.

Check the NIZK proof.

Decrypt with sk_{1,vk_1}.

Proof Sketch

Let's play the CCA game with the adversary.

We will use her to break either the NIZK soundness/ZK, the signature scheme or the CPA-secure scheme.

Proof Sketch

Let's play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Hybrid 1: Observe that $\boldsymbol{vk_i} \neq \boldsymbol{vk}^*$.

(Otherwise break signature)

Observe that this means each query ciphertext-tuple involves a different public-key from the challenge ciphertext. Use the "different private-key" to decrypt.
(If the adv sees a difference, she broke NIZK soundness)

Hybrid 2: Now change the CRS/π into simulated CRS/π!

(OK by ZK)

If the Adv wins in this hybrid, she breaks **IND-CPA!**

New Topic:

Secure Computation

Secure Computation

Input: x

Alice

Output: $F_A(x, y)$

Input: y

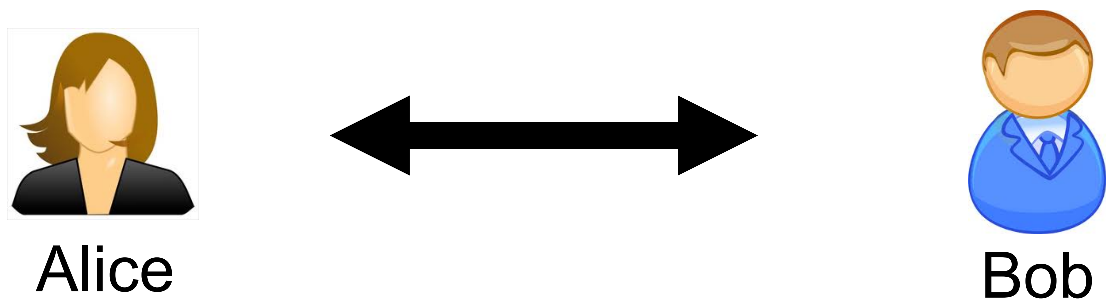
Bob

Output: $F_B(x, y)$

Secure Two-Party Computation

Input: x

Input: y



Output: $F_A(x, y)$

Output: $F_B(x, y)$

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 **Input:** y

Alice Bob

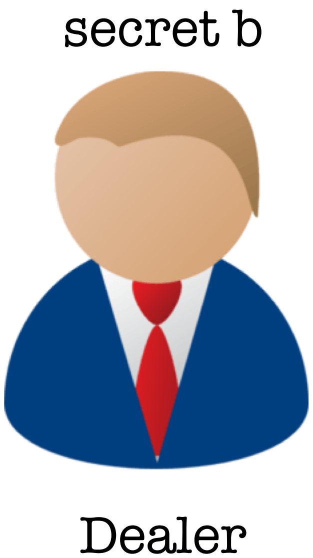
Output: $F_A(x, y)$ **Output**: $F_B(x, y)$

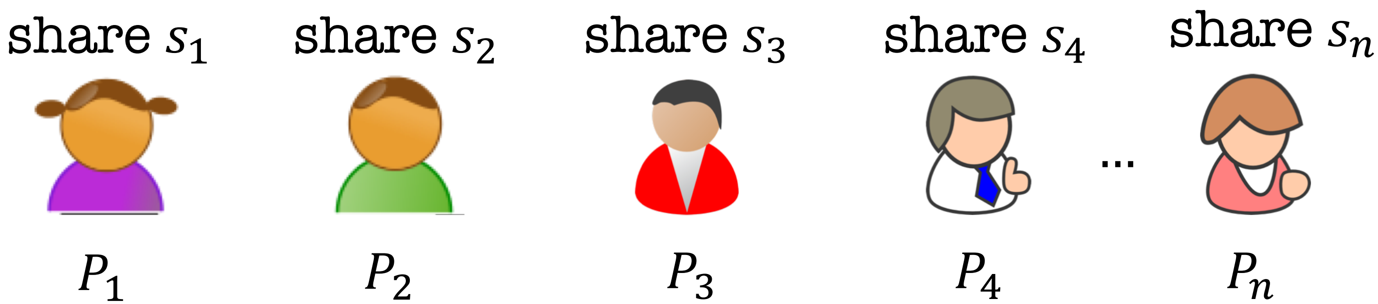
Malicious Security:

- 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

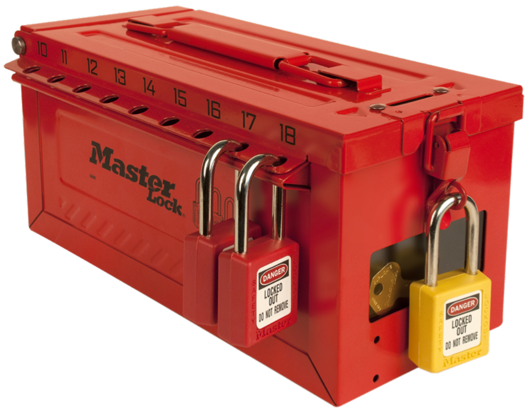
share s_2: random

share s_3: random

share s_4: random

…

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



secret b $\in Z_p$

1-out-of-n Secret Sharing

Dealer

P_1

P_2

P_3

P_4

…

P_n

share $s_1 = \mathrm{b}$

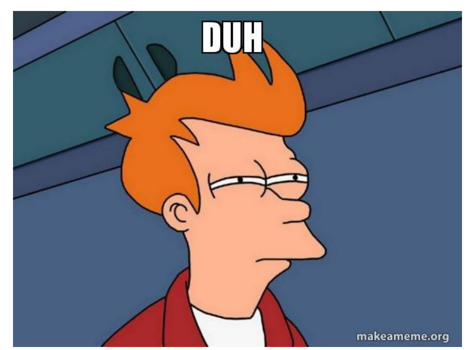
share $s_2 = \mathrm{b}$

share $s_3 = \mathrm{b}$

share $s_4 = \mathrm{b}$

…

share $s_n = b$



secret b $\in Z_p$

2-out-of-n Secret Sharing?

Dealer

P_1 P_2 P_3 P_4 P_n

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

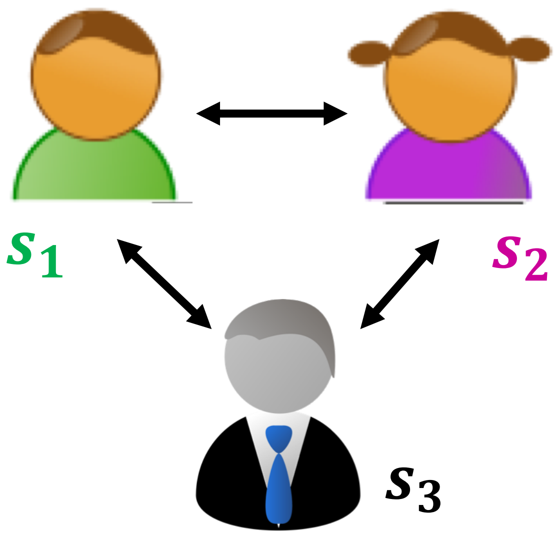
What is the size of shares each party gets?

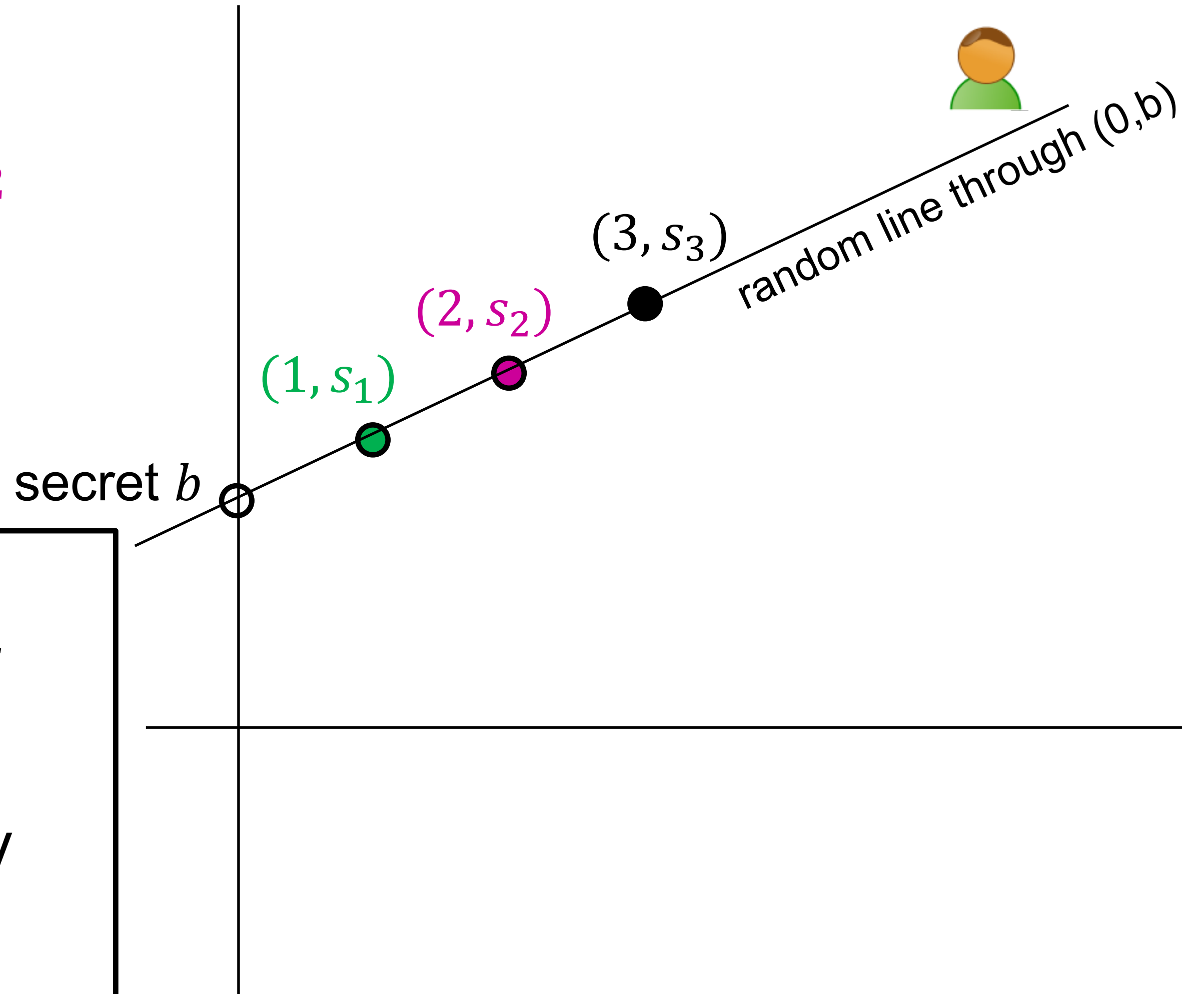
How does this scale to t-out-of-n?

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 $(= \frac{js_i - is_j}{j-i})$.

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.

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

2. Compute the shares:

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

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.

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

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!

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

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!

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

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!

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

where a_i are uniformly random mod p

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

Security:

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

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

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

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

Key Idea: Polynomials are Amazing!

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

where a_i are uniformly random mod p

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

Security: For every value of b there is a unique polynomial with constant term b and shares $s_1, s_2, \dots, s_{t-1}$.

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

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

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

Key Idea: Polynomials are Amazing!

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

where a_i are uniformly random mod p

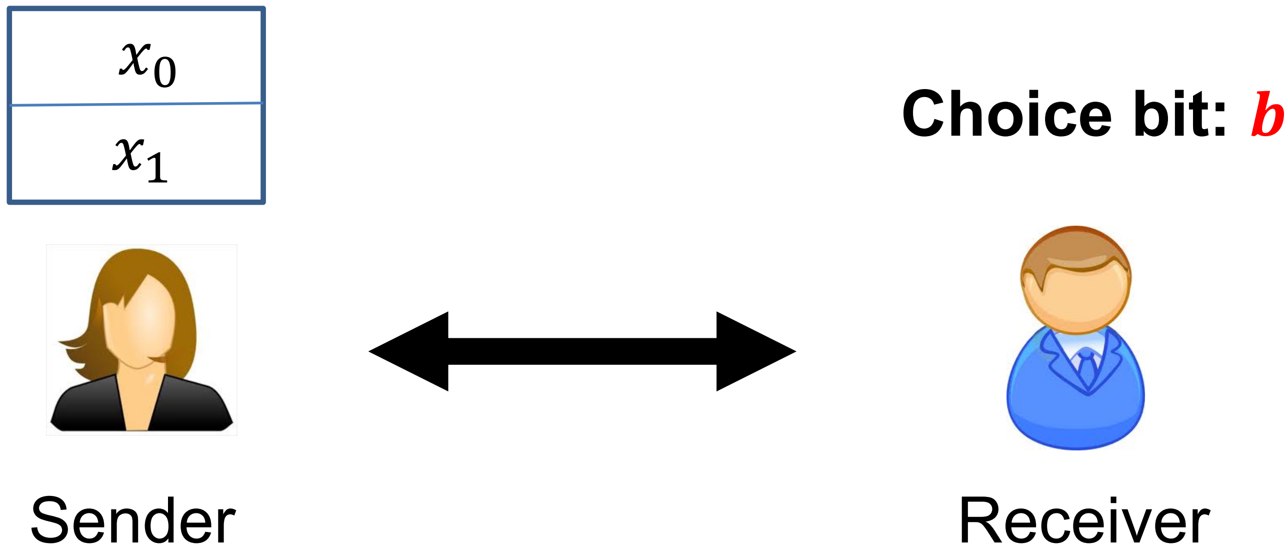
$$s_1 = f(1), s_2 = f(2), \dots, s_i = f(i), \dots, s_n = f(n)$$

Security: For every value of b there is a unique polynomial with constant term b and shares $s_1, s_2, \dots, s_{t-1}$.

Corollary: for every value of the secret b is equally likely given the shares $s_1, s_2, \dots, s_{t-1}$. 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? The Dating Problem

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

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

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

Why OT? The Dating Problem

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

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 β=1, and 0 if β=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

Who is richer?

The Billionaires' Problem

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

X

⋯	0	1	0	0	⋯

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

Y

⋯	0	1	1	1	1	1	1

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

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

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

Detour: OT ⇒ 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

$x_0 = \gamma$
$x_1 = \alpha \oplus \gamma$

Run an OT protocol

Choice bit $b = \beta$

Alice outputs γ.

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

The Billionaires' Problem

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

⋯	0	1	0	0	⋯

Unit Vector u_X

⋯	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.

$$\gamma_i \oplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$$

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): $\gamma \oplus \delta = \boldsymbol{\langle u_X, v_Y \rangle = f(X,Y)}$.

The Billionaires' Problem

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

...	0	1	0	0	...

Unit Vector u_X

...	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.

$$\gamma_i \oplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$$

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

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

“OT is Complete”

Theorem *(lec18-19):* OT can solve not just love and money, but ***any*** two-party (and multi-party) problem efficiently.

Defining Security:
The Ideal/Real Paradigm

OT Definition

x_0

x_1

Choice bit: $\boldsymbol{b}$

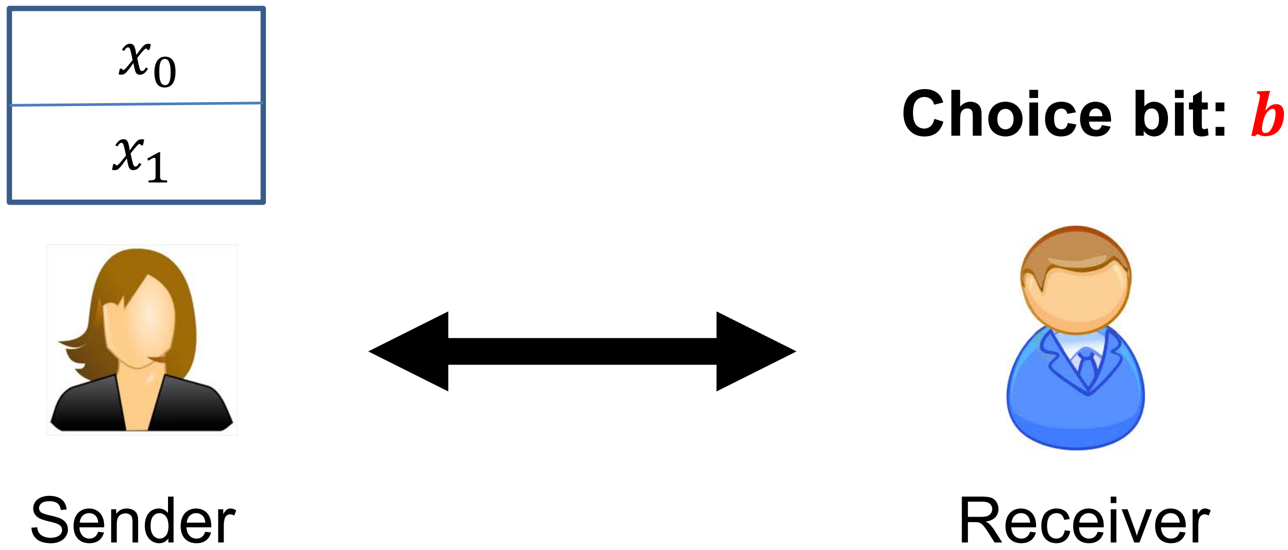
Sender

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

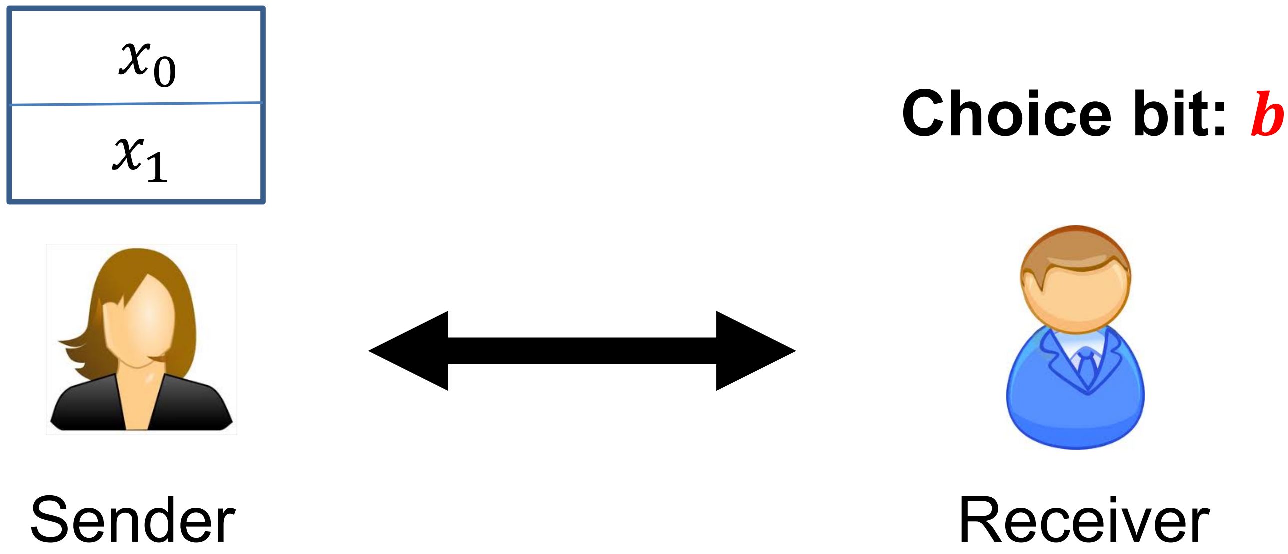


Receiver Security: Sender should not learn b.

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

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

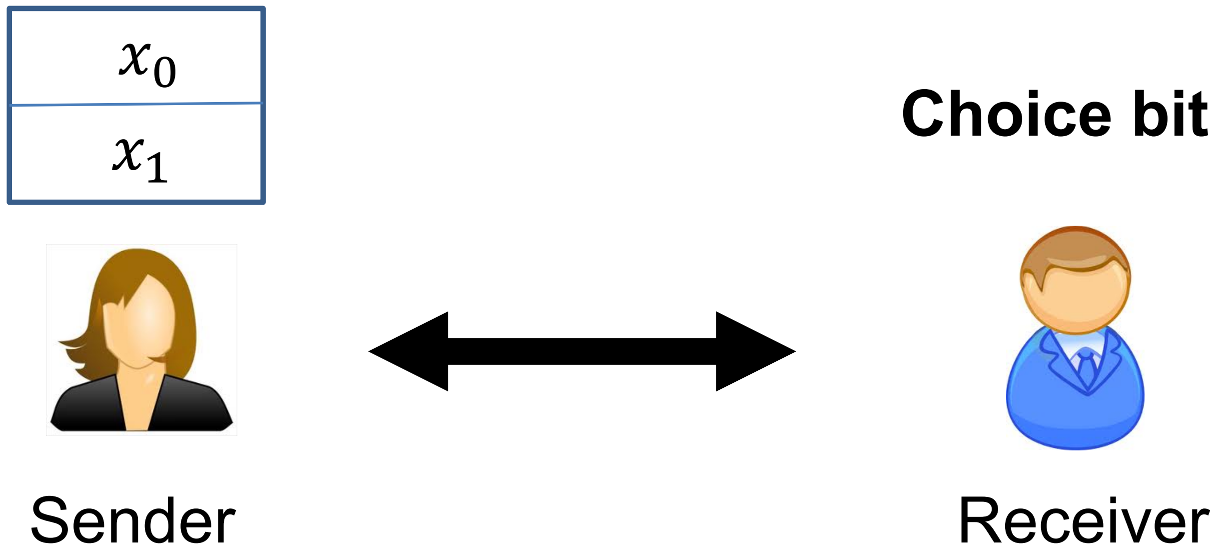
OT Definition



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



Choice bit: $\boldsymbol{b}$

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 one-time pad x_0, x_1 using hardcore bits

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

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

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

N, e $\longrightarrow$

s_0, s_1 $\longleftarrow$

$x_0 \oplus HCB(r_0)$ $\longrightarrow$ $x_1 \oplus HCB(r_1)$

Choice bit: b

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

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

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 →

← s_0, s_1

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

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

Input bits: (x_0, x_1)

N, e $\longrightarrow$

s_0, s_1 $\longleftarrow$

$x_0 \oplus HCB(r_0)$, $x_1 \oplus HCB(r_1)$ $\longrightarrow$

Choice bit: b

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

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

$\longleftarrow$

Homomorphically evaluate the selection function

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

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

$\longrightarrow$

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.