# MPC – Lecture 3:

## One Time Truth Table (with trusted dealer)

DR. ADI AKAVIA

UNIVERSITY OF HAIFA

# Today’s Agenda

Intro – Secure-SUM example

The Preprocessing Model

One-Time Truth Table (OTTT) 2PC, passive, preprocessing model

Secret sharing

*As time permits* – BeDoZa 2PC, passive, preprocessing model

# Intro: Secure-SUM Example

# Secure-SUM Protocol

(in Plain Model)

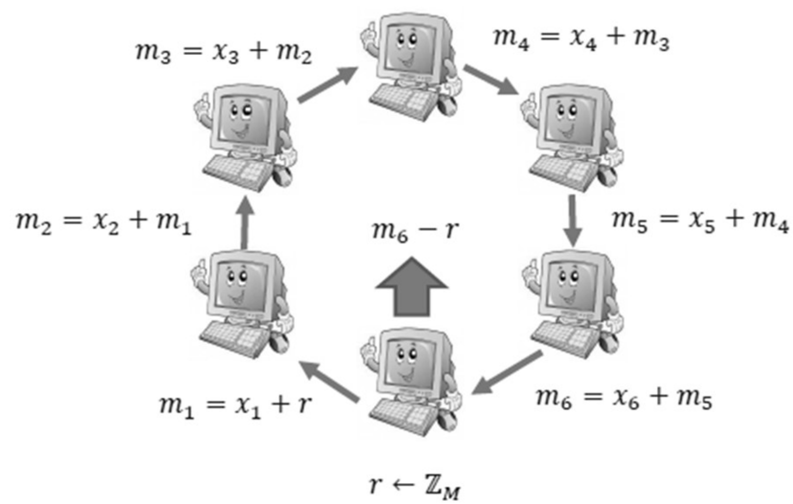
SECURE AGAINST A PASSIVE ADVERSARY CORRUPTING AT MOST 1 PARTY

# Secure-SUM

Protocol (pictorially):

## Example – Computing Sum

- Each $P_i$ has input $x_i < M$ (work modulo $M$)
- Want to compute $\sum x_i$
- Is the protocol is secure facing one corruption (semi-honest)?



# Secure-SUM Protocol

**Parties:** $P_1,..,P_n$, where party $P_i$ has input $x_i$

**The protocol**:

$P_1$ draws a random $r \leftarrow_R [0..M-1]$ and sends to $P_2$ $\quad a_1 = x_1 + r \pmod M$

For i=2,…,n,

Upon receiving $a_{i-1}$ from party $P_{i-1}$, Party $P_i$ sends to Party $P_{i+1 \bmod n}$ $\quad a_i = a_{i-1} + x_i \pmod M$

Upon receiving $a_n$ Party 1 computes and broadcasts $\quad out = a_n - r \pmod M$

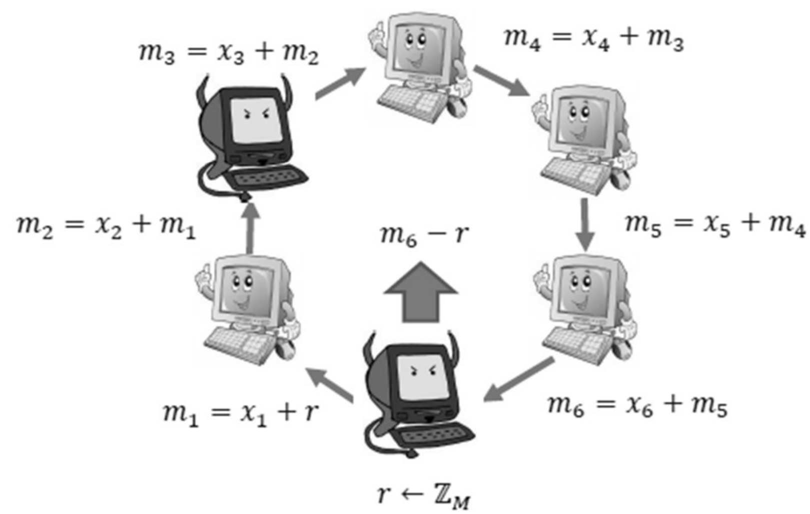
# Secure-SUM

Attack (pictorially):

<u>fix</u>

## Example – Computing Sum

- Each $P_i$ has input $x_i < M$ (work modulo $M$)
- Want to compute $\sum x_i$
- Is the protocol is secure facing one corruption (semi-honest)?
- What about two corruptions?



# Secure-SUM

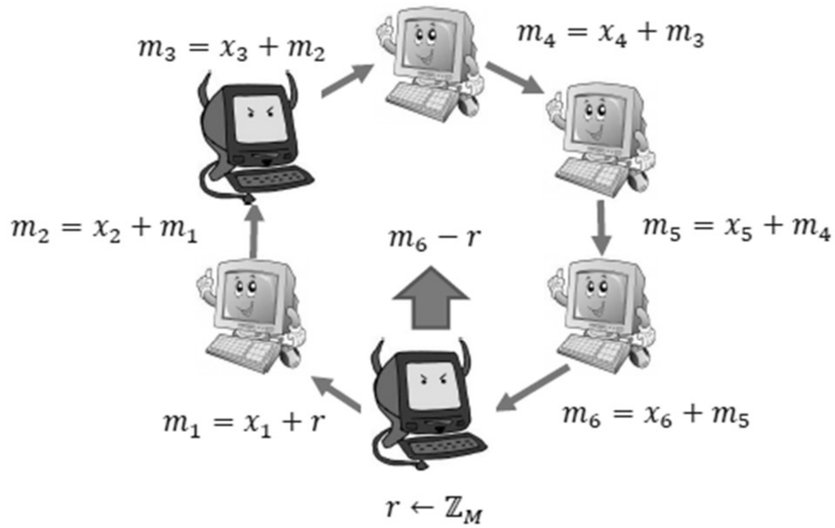
## in Pre-Processing Model (Correlated Randomness)

SECURE AGAINST A PASSIVE ADVERSARY CORRUPTING ANY NUMBER OF PARTIES

# Recall

## Example – Computing Sum

- Each $P_i$ has input $x_i < M$ (work modulo $M$)
- Want to compute $\sum x_i$
- Is the protocol is secure facing one corruption (semi-honest)?
- What about two corruptions?



# How to achieve security against up to *n* corrupt parties?

**Correlated randomness**: Suppose each party $P_i$ (on top of its input $x_i$) has a random value $r_i$, where $r_1, .., r_n$ are correlated in the sense that:

$$\sum_i r_i = 0$$

**Q:** Can you modify the protocol to be secure against collusions ?
**Hint:** Use the above correlated randomness

# **Q:** modify protocol to secure against collusions

**Ans:** ______________________________

# **Solution:** Secure-SUM against collusions using correlated randomness

**Parties:** $P_1,..,P_n$, where party $P_i$ has input $x_i$ and randomness $r_i$ s.t. $\sum_i r_i = 0$

**The protocol**:

$P_1$ ~~draws a random $r \leftarrow_R [0..M-1]$ and~~ sends to $P_2$ $\quad a_1 = x_1 + r_1 \quad$ (mod M)

For i=2,…,n,

Upon receiving $a_{i-1}$ from party $P_{i-1}$, Party $P_i$ sends to Party $P_{i+1 \bmod n}$ $\quad a_i = a_{i-1} + x_i + r_i \quad$ (mod M)

Upon receiving $a_n$ Party 1 ~~computes and~~ broadcasts $\quad out = a_n \cancel{- r} \quad$ (mod M)

# MPC in the Preprocessing Model

CORRELATED RANDOMNESS

TRUSTED DEALER

# What is **correlated randomness** ?

A *correlated randomness* $(r_1, \ldots, r_n)$

is a $n$-tuple of random variables

drawn from a joint distribution D.

Example for $D$:

Sample a uniformly random tuple in $\{ (r_1, \ldots, r_n) \mid \Sigma r_i = 0 \}$

# What is the **Pre-Processing Model** ?

**MPC in the Pre-Processing Model:**

**Offline phase:** // When: before protocol starts

$(r_1, \ldots, r_n) \leftarrow D$ // What: draw correlated randomness

Each $P_i$ receives $r_i$ // Note: independent of the inputs !

*Who draws from D ?*

For now: "Trusted Dealer".

Later this course: MPC.

**Online phase:**

Parties execute a protocol on their inputs $x_i$ using their correlated randomness $r_i$.

# Where correlated rand. comes from?

For now: “Trusted Dealer”.

Later this course: MPC.

# Example: Equality Test
## using correlated randomness

**Functionality:**

- The receiver has input $x \in X$, the sender input $y \in X$;
- The receiver learns 1 if $x = y$ or 0 otherwise. The sender learns nothing;

**Preprocessing:**

1. Sample a random 2-wise independent permutation $P : X \to X$, and a random string $r \in_R X$. Compute $s = P(r)$;
2. The preprocessing outputs $(r, s)$ to the receiver and $P$ to the sender;

**Protocol:**

1. The receiver computes $u = x + r$ and sends to sender;
2. The sender computes $v = P(u - y)$ and sends to the receiver;
3. The receiver outputs 1 if $v = s$, and 0 otherwise;

**Figure 1.** A perfectly secure protocol for equality with preprocessing.

From: Ishai, Kushilevitz, Meldgaard,Orlandi, Paskin-Cherniavsky, "*On the Power of Correlated Randomness in Secure Computation*", TCC 2013

# Protocols with preprocessing

An *n*-party protocol can be formally defined by a *next message function*.

*next message function*: on input $(i, x_i, r_i, j, m)$, specifies an *n-tuple of messages* sent by party $P_i$ in round $j$, when $x_i$ is its input, $r_i$ is its randomness and $m$ describes the messages it received in previous rounds. The next message function may also instruct $P_i$ to *terminate* the protocol, in which case it also specifies the *output* of $P_i$.

In the *preprocessing model*, the specification of a protocol also includes a joint distribution D over $R_1 \times R_2 \ldots \times R_n$, where the $R_i$'s are finite randomness domains. This distribution is used for **sampling correlated random inputs** $(r_1, \ldots, r_n)$ which the parties receive <u>before the beginning of the protocol</u> (in particular, the preprocessing is <u>independent of the inputs</u>). The next message function, in this case, may also depend on the private random input $r_i$ received by $P_i$ from D.

# One-Time Truth Table

AGAINST PASSIVE ADVERSARY

# OTTT: What is it?

| | |
|---|---|
| Key property: | simplest 2PC protocol |
| Security: | against passive unbounded adversary |
| Complexity: | truth table size |
| Usability: | functionalities with small truth tables<br>(e.g., function over small domain such as AND, XOR) |
| Model: | Pre-processing (for now: trusted dealer) |

Later this course: get rid of dealer

# Functionality & Truth Table

**Parties:** Alice A, Bob B, Trusted dealer D

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

$(x,y) \rightarrow (f(x,y), \perp)$

**Truth Table:** $f$ is represented by **truth table** T with $2^n$ rows and $2^n$ columns,
where $T[i,j] = f(i_{bin}, j_{bin})$
for $i_{bin}$ and $j_{bin}$ the binary representation of $i$ and $j$.

# The Protocol: Offline Phase (pre-processing)

Parties: Dealer D with truth table T

Steps:

1. Draw random row/col shifts: $r, c \leftarrow_R \{1,\ldots,2^n\}$ .
2. Draw a random $2^n \times 2^n$ Boolean matrix $M_B$ .
3. Compute: $M_A[i, j] = M_B[i, j] \oplus T[i - r \bmod 2^n, j - c \bmod 2^n]$
4. Output $(r, M_A)$ to Alice, and $(c, M_B)$ to Bob.

# The Protocol: Online Phase

Parties: Alice A with $(r, M_A)$ (from preprocessing) and input $x$,
Bob B with $(c, M_B)$ (from preprocessing) and input $y$ .

Steps:

1. Alice computes $u = x + r \bmod 2^n$, and sends u to Bob
2. Bob computes $v = y + c \bmod 2^n$
   $z_B = M_B[u,v]$, and sends $(v , Z_B)$ to Alice
3. Alice outputs $z = M_A[u,v] \oplus z_B$

# Example: OTTT for Millioner's Problem

# Truth Table: Millioner's Problem (simplified)

Millioner 1 (Alice) has input $x \in \{1,2,3,4\}$ millions

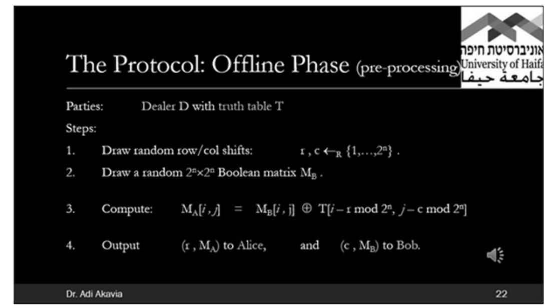
Millioner 2 (Bob) has input $y \in \{1,2,3,4\}$ millions

Truth table of $x > y$ :

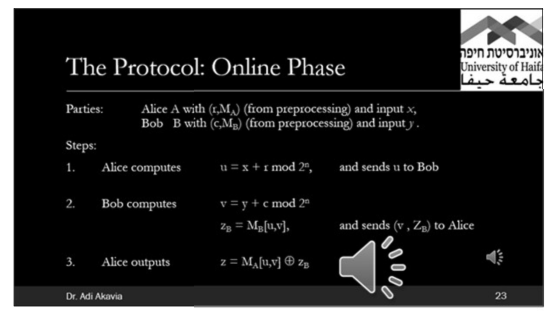
| Alice's input \ Bob's input | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 |
| 4 | 1 | 1 | 1 | 0 |

# Example: OTTT for Millioner's Problem

**Offline Phase:**




**Online Phase:**



# Example: OTTT for Millioner’s Problem

**Offline Phase** (Delear)**:**

- Draw (r,c) ← ( 

)
- Rotate T’s rows by r, and cols by c:

$$T \quad \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 \end{array} \xrightarrow{\text{Rows permute}} T_r \quad \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 2 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{array} \xrightarrow{\text{Cols permute}} T_{r,c} \quad \begin{array}{c|cccc} & 3 & 4 & 1 & 2 \\ \hline 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 1 & 0 & 1 & 1 \\ 4 & 0 & 0 & 0 & 0 \end{array}$$

- Draw a random matrix $M$ .
- Compute
  $M_A = M_B \oplus T_{r,c}$
- Output $(r, M_A)$ to Alice
  $(c, M_B)$ to Bob

$$\underset{M_A}{\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}} = \underset{T_{r,c}}{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}} \oplus \underset{M_B}{\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}$$

# Example: OTTT for Millioner's Problem

**Online Phase** (Alice($x$ = 3millions, r = 3) and Bob($y$ = 1million, c = 2)):

Alice computes u $= x + r \bmod 4$
$= 3+3 \bmod 4 = 2$ and sends u to Bob

Bob computes v $= y + c \bmod 4$
$= 1+2 \bmod 4 = 3$ and

$z_B$ $= M_B[u,v]$
$= M_B[2,3] = 0$ and sends $(v, Z_B) = (3,0)$ to Alice

Alice outputs z $= M_A[u,v] \oplus z_B$
$= M_A[2,3] \oplus 0$
$= 1 \oplus 0 = 1$

$M_B$

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |

$M_A$

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |

# Correctness of OTTT

**Exercise**: Prove OTTT is correct

# Correctness of OTTT

| | | |
|---|---|---|
| $z$ | $= M_A[u , v] \oplus z_B$ | //def of $z$ |
| | $= M_A[u , v] \oplus M_B[u , v]$ | //def of $z_B$ |
| | $= T[u - r , v - c]$ | //def of $M_A$ |
| | $= T[x , y]$ | //def of $u , v$ |
| | $= f(x , y)$ | //def of $T$ |

# Privacy of OTTT

**Class Exercise**: Prove OTTT is secure against passive adversary

# Privacy of OTTT

**Proof:** We construct simulators $S_A$ and $S_B$,

so that $S_A$ given Alice's *input* and *output* of the functionality,

produces a *simulated view* that is

indistinguishable from the

*real view*

(analogously $S_B$ for Bob's input, output, view),

# Privacy of OTTT

**Proof cont.:** The real views are:

$$\text{view}_A = (\text{A's input, A's randomness, msgs received})$$
$$= (x, \quad \perp, \quad (r, M_A, v, z_B))$$

$$\text{view}_B = (\text{B's input, B's randomness, msgs received})$$
$$= (y, \quad \perp, \quad (s, M_B, u))$$

# Privacy of OTTT

**Proof cont.:**

**The simulator for Alice $S_A$:**

- Given input $x \in \{0,1\}^n$ and $z \in \{0,1\}$ ,
- Sample $z_B \leftarrow_R \{0,1\}, v , r \leftarrow_R \{0,1\}^n$
- Construct $M_A$ as follows:

  $M_A[\, x+r , v \,] := z \oplus z_B$ , and

  $M_A[\, i , j \,] \leftarrow_R \{0,1\} \quad \forall (i , j) \neq (\, x+r , v)$
- Output "*simulated-view*" := $(x, \quad \perp, \quad (r , M_A , v, z_B)\,)$

# Privacy of OTTT

**Proof cont.:** We show that the simulated view is indistinguishable from the real view.

In both views,

- $x$ is identical (the input)
- r, v and $M_A[i , j]$ for all $(i , j) \neq (u , v)$ are indep. uniformly random;
- $(M_A[u , v], z_B)$ is uniformly random subject to $M_A[u, v] \oplus z_B = z$.

So (real) $\text{view}_A \equiv_s$ simulated-$\text{view}_A$

# Privacy of OTTT

**Proof cont.:**

**$S_B$ – simulator for Bob** (easier, **exercise**).

# Pro and Cons of OTTT

- ☑ Perfect security (*i.e.*, unconditional)
- ☑ Optimal online round complexity
- ☑ (Essentially) Optimal communication complexity
- ☒ Trusted dealer required
- ☒ Exponential storage and offline complexity

# Secret Sharing

# Secret Sharing: Informal Def

**What is it good for:** Key tool in secure computation

E.g. masking the truth table T in OTTT.

| | |
|---|---|
| **Notations:** | A – domain of secrets ; B – domain of shares. |
| | $\mathrm{Shr} : A \to B^n$ a sharing algorithm (randomized) |
| | $\mathrm{Rec} : B^k \to A$ a reconstruction algorithm |

**What is it:**

(t,n)-*secret sharing* splits secrets s into *n* shares, such that:

- **Correctness:** Any t shares allow complete reconstruction of the secret s.
- **Privacy:** Any t-1 of the shares reveal no information about s.

# Secret Sharing: Formal Definition

**Definition:** A (t,n)-*secret sharing scheme* (SSS) is a pair of algorithms (Shr, Rec) that satisfies these two properties:

- **Correctness**. For every secret $s \in A$, shares $(s_1, s_2, \ldots, s_n) \leftarrow \mathrm{Shr}(s)$, and $k \geq t$ distinct indices $i_1,..,i_k \in [n]$: $\Pr[\, \mathrm{Rec}(s_{i1}, \ldots, s_{ik}) = s \,] = 1$

Perfect privacy

- **Privacy**. For every secrets $x,y \in A$, $k < t$ indices $i_1,..,i_k \in [n]$ and shares $v = (v_{i1}, \ldots, v_{ik}) \in B^k$: $\Pr[\, \mathrm{Shr}(x)_{|i1,..,ik} = v \,] = \Pr[\, \mathrm{Shr}(y)_{|i1,..,ik} = v \,]$

(where the probability is over the random coins of Shr ).

# Linear Secret Sharing: Definition



**Linearity.** A (t,n)-secret sharing scheme is called *linear* if:

$$\mathrm{Rec}(\ \mathrm{Shr}(x) +_B \mathrm{Shr}(y)\ ) = x +_A y$$

Pictorially:

$$\begin{array}{llcll}
 & & \mathrm{Shr} & & \\
x\,,\,y & \in A & \rightarrow & \mathrm{Shr}(x)\,,\,\mathrm{Shr}(y) & \in B \\
\quad\downarrow +_A & & & \quad\downarrow +_B & \\
x +_A y & \in A & \leftarrow & \mathrm{Shr}(x) +_B \mathrm{Shr}(y) & \in B \\
 & & \mathrm{Rec} & &
\end{array}$$

# Secret Sharing: Example 1

**Construction of (2,2)-secret sharing over GF(2):** $A=B=\{0,1\}$ ,

- Shr(x): Output $(r, x \oplus r)$ for $r \leftarrow_R \{0,1\}$
- Rec(a,b): Output $a \oplus b$

**Properties:**

- **Correctness:** ____________________
- **Privacy:** ____________________
- **Linearity:** ____________________

**Exercise:**
write what these properties say.
prove these properties hold.

**Proof of correctness:**

$$\mathrm{Rec}(\,\mathrm{Shr}(x)\,) = \mathrm{Rec}(\, r, x\oplus r\,) = r \oplus (x\oplus r)\,.$$

**Proof of linearity:**

$$\mathrm{Rec}(\ \ \mathrm{Shr}(x) +_B \mathrm{Shr}(y)\ \ )$$

$$= \mathrm{Rec}(\ \ (r, x\oplus r)\ \ +_B\ \ (r', y\oplus r')\ \ )$$

$$= \mathrm{Rec}(\ \ (r\oplus r',\ \ x\oplus y \oplus r\oplus r')\ \ )$$

$$= (r \oplus r')\ \ \oplus\ \ (x\oplus y \oplus r \oplus r')\,.$$

$$=\ x +_A y$$

**Construction of (2,2)-secret sharin**

- Shr(x): Output $(\, r, x \oplus r\,)$
- Rec(a,b): Output $a \oplus b$

## Properties:

- **Correctness:** $\mathrm{Rec}(\,\mathrm{Shr}(x)\,) = x$
- **Privacy:** For any secrets $x, y \in \{0,1\}$, share $i \in \{1,2\}$, and share value $v \in \{0,1\}$ :

$$\Pr_r[\,\mathrm{Shr}(x)_i = v\,] \ = \ \Pr_r[\,\mathrm{Shr}(y)_i = v\,] \quad (= 1/2\,)$$

- **Linearity:** $\mathrm{Rec}(\ \ \mathrm{Shr}(x) +_B \mathrm{Shr}(y)\ \ ) \ = \ x +_A y$

# Secret Sharing: Example 1, Privacy Proof

**Proof of privacy, Analysis of 1st Share:**

$\text{Shr}(x)_1$ is a uniformly random bit $r \in \{0,1\}$

similarly, $\text{Shr}(y)_1$ is uniformly random $r' \in \{0,1\}$.

So for every $v \in \{0,1\}$, $\text{Shr}(x)_1 = v$ with probability ½

similarly $\text{Shr}(y)_1 = v$ with probability ½.

Therefore,

$$\Pr_r[\,\text{Shr}(x)_1 = v\,] \quad = \quad \Pr_r[\,\text{Shr}(x')_1 = v\,]$$

# Secret Sharing: Example 1, Privacy Proof

**Proof of privacy, Analysis of 2nd Share:**

$\mathrm{Shr}(x)_2 = r \oplus x$ for a uniformly random $r \in \{0,1\}$,

So $\Pr_r[\,\mathrm{Shr}(x)_2 = 0\,] = \Pr_r[\,r = x\,] = ½$ and

$\Pr_r[\,\mathrm{Shr}(x)_2 = 1\,] = \Pr_r[\,r \neq x\,] = ½$ .

Similarly, $\Pr_r[\,\mathrm{Shr}(y)_2 = 0\,] = \Pr_r[\,\mathrm{Shr}(x')_2 = 1\,] = ½$

Therefore, for every $v \in \{0,1\}$

$$\Pr_r[\,\mathrm{Shr}(x)_2 = v\,] \quad = \quad \Pr_r[\,\mathrm{Shr}(x')_2 = v\,] .$$

# Secret Sharing: Example 2

**Construction of (2,2)-secret sharing over GF(p):** $A=B=\{0,..,p-1\}$ ,

- Shr(x): Output $(r, x - r \bmod p)$ for $r \leftarrow_R \{0,..,p-1\}$
- Rec(a,b): Output $a + b \bmod p$

**Exercise:** prove these properties hold.

**Properties:**

- **Correctness:** $Rec(Shr(x)) = x$
- **Privacy:** For any secrets $x, y \in \{0,..,p-1\}$, share $i \in \{1,2\}$, and share value $v \in \{0,..,p-1\}$ :
  $$Pr_r[Shr(x)_i = v] = Pr_r[Shr(y)_i = v] \qquad (= 1/p)$$
- **Linearity:** $Rec( Shr(x) +_B Shr(y) ) = x +_A y$

# Proof of Correctness and Linearity

**Correctness:** Rec(Shr(x)) = Rec(r, x-r) = r + (x-r) mod p = x

**Linearity:** **Shr**($x$)$+_B$ **Shr**($x$') = (r, x-r) + (r', x'-r)

= (r+r', (x+x')-(r+r'))

**Rec**(r+r', (x+x')-(r+r')) = (r+r') + (x+x')-(r+r')

= $x$+$x$'

# Proof of Privacy, $1^{st}$ share analysis

Recall that $\mathrm{Shr}(x)_1$ is uniformly random in $r \in \{0,1,..,p\text{-}1\}$,

So, for $v \in \{0,1,\ldots,p\text{-}1\}$, $\mathrm{Shr}(x)_1 = v$ w.p. $1/p$

Similarly, $\mathrm{Shr}(x')_1 = v$ w.p. $1/p$

Therefore, $\Pr_r[\,\mathrm{Shr}(x)_1 = v\,] = \Pr_r[\,\mathrm{Shr}(x')_1 = v\,]$

# Proof of Privacy, 2nd share analysis

Recall that $\quad \mathrm{Shr}(x)_2 = \; x - r \bmod p$

and $\quad \mathrm{Shr}(x')_2 = x' - r' \bmod p \quad$ for *iid* r, r' $\leftarrow_R \{0,1,\ldots,p\}$.

So, for every $v \in \{0,1,\ldots,p\text{-}1\}$,

$$\Pr_r[\,\mathrm{Shr}(x)_2 = v\,] = \Pr_r[\,r = x\text{-}v\,] = 1/p$$
$$\Pr_r[\,\mathrm{Shr}(x')_2 = v\,] = \Pr_r[\,r = x'\text{-}v\,] = 1/p$$

Therefore, $\Pr_r[\,\mathrm{Shr}(x)_2 = v\,] = \Pr_r[\,\mathrm{Shr}(x')_2 = v\,]$

# Secret Sharing: Exercises

**Exercise:**

1. Extend Examples 1-2 to (2,2) -secret sharing of d-dimensional tuples (i.e., $A=B=\{0,..,p-1\}^d$) .
2. Extend Examples 1-2 to (n,n) -secret sharing over GF(p)
3. Can you extend Examples 1-2 to (t,n) -secret sharing for $t < n$ ?

# Solution to Exercise 3

**Shamir secret sharing**. Not covered today.

# BeDoZa Protocol

PASSIVE ADVERSARY; TRUSTED DEALER

# BeDoZa Protocol: What is it?

| | | |
|---|---|---|
| Key property: | 2PC protocol for secure **circuit** evaluation | |
| Security: | against passive unbounded adversary | |
| Complexity: | circuit size | |
| Usability: | ppt functionalities | |
| Model: | Pre-processing (for now: trusted dealer) | |
| History: | BeDOZa ~ | (passive) GMW (Goldreich, Micali, Widgerson)<br>+ Beaver triplets (from trusted dealer replacing OT) |

# Functionality & Circuit

**Parties:** Alice A, Bob B, Trusted dealer D

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

$(x,y) \rightarrow (f(x,y), \perp)$

**Circuit:** $f$ is specified by a **Boolean circuit**

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

# Circuit Notations

L wires $x_1,\ldots,x_L$ :  $x_1 ,\ldots, x_n$ – Alice's input

$x_{n+1},\ldots, x_{2n}$ – Bob's input

$x_L$ – output wire

$d$ layers s.t. inputs to gates at layer $i \in \{1,..,d\}$ are from layers $<i$

Gates: XOR with constant or of two wires
AND with constant or of two wires

# BeDoZa Protocol: Overview

Wires' values are secret shared between Alice and Bob:

- secret share input wires
- propagates secret sharing layer by layer,
- once obtained a secret sharing of the output wire, open (=reconstruct).

# BeDoZa Protocol: Notations

Notation: [x] denotes a secret sharing of $x \in \{0,1\}$, where

Alice holds $x_A \in \{0,1\}$

Bob holds $x_B \in \{0,1\}$

and where:

$(x_A, x_B)$ is uniform random in $\in \{0,1\}^2$ subject to:

$$x_A \oplus x_B = x.$$

# BeDoZa Protocol: Offline Phase

Secret share Beaver Triples

**Parties:** Dealer D with input $t \in N$

**Steps:** Repeat $t$ times

1) Sample a "**Beaver triples**": $u, v \leftarrow_R \{0,1\}$ and $w = u \cdot v \mod 2$

2) Secret share
$$[u] := (u_A, u_B) \leftarrow Shr(u)$$
$$[v] := (v_A, v_B) \leftarrow Shr(v)$$
$$[w] := (w_A, w_B) \leftarrow Shr(w)$$

3) Send $(u_A, v_A, w_A)$ to Alice and $(u_B, v_B, w_B)$ to Bob

# BeDoZa Protocol: Online Phase

Securely evaluate a circuit C with #AND $\leq t$

**Parties:** Alice A with input $x$ and the shares $(u_A,v_A,w_A)$
Bob B with input $y$ and the shares $(u_B,v_B,w_B)$

**Steps** (using sub-protocols: Share , XOR , AND, OpenTo specified later. )**:**

1) Alice & Bob **share** their input wires: $[x_i] = (x_{iA},x_{iB}) \leftarrow$ Share( A, $x_i$) for $i=1,..,n$
$[x_i] = (x_{iA},x_{iB}) \leftarrow$ Share( B, $x_i$) for $i=n+1,..,2n$

2) For each circuit layer $i = 1,\ldots,d$, Alice & Bob **securely evaluate** all gates in layer $i$ using XOR and AND subprotocols

3) Alice & Bob reconstruct the output wire value $x^L$ : $(z, \perp) \leftarrow$ OpenTo( A , $[x^L]$ )

# Subprotocol: Sharing input wires

Share(A,$x_i$): Alice computes $(x_{iA}, x_{iB}) \leftarrow Shr(x_i)$ and sends $x_{iB}$ to Bob.

Share(B,$x_i$): Analogous.

# Subprotocol: Opening secret shared values

OpenTo(A, [x]): Bob sends $x_B$ to Alice
Alice outputs $x = x_A \oplus x_B$.

OpenTo(B, [x]): Analogous.

Open([x]): Run both OpenTo(B, [x]) and OpenTo(A, [x]).

# Subprotocol: Evaluating XOR gates

| | | |
|---|---|---|
| XOR([x],c): | Alice outputs | $z_A = x_A \oplus \mathrm{c}$ |
| | Bob outputs | $z_B = x_B$ |
| XOR([x],[y]): | Alice outputs | $z_A = x_A \oplus y_A$ |
| | Bob outputs | $z_B = x_B \oplus y_B$ |

# Subprotocol: Evaluating AND gates

AND([x],c): Alice outputs $z_A = c \cdot x_A$
Bob outputs $z_B = c \cdot x_B$

AND([x],[y]):

[d] ← XOR([x],[u])
d ← **Open**([d])

[e] ← XOR([y],[v])
e ← **Open**([e])

Compute: $[z] = [w] \oplus (e \cdot [x]) \oplus (d \cdot [y]) \oplus (e \cdot d)$
(using subprotocols XOR([·],[·]), AND([·],·))

# Correctness of subprotocols

Share, OpenTo, Open: follows from correctness of secret sharing scheme

XOR([x],c), XOR([x],[y]), AND([x],c): straightforward (check!)

AND([x],[y]):

$$
\begin{aligned}
z &= w \oplus ex \oplus dy \oplus ed \\
&= uv \oplus (yx+vx) \oplus (xy+uy) \oplus (xy + uy + xv + uv) \\
&= xy
\end{aligned}
$$

# Correctness of entire protocol

**Theorem (informal):** For every wire w in the circuit, the parties hold a secret-sharing $[v_w]$ of the value $v_w$ on that wire.

**Proof:** By induction.

Base case – Input wires: ________________________________

Induction step – XOR & AND gates: ________________________

**Corollary:** Opening the output wire returns the correct circuit output.

# Privacy

We show there exists ppt algorithms $S_A$,$S_B$ that generate simulated views for Alice and Bob respectively that are computationally-indistinguishable from their real views.

# Privacy: Simulated View for Alice

$S_A(x, f(x,y))$ output a simulated view consisting of:

1) Alice's input: $x$. 2) Alice's randomness: $S_A$ run Share(A,$x_i$), add used randomness to view.

3) Alice's received messages:

- $S_A$ plays dealer's role:
  Sample t Beaver triples, secret-share and add to view Alice's shares $(u_A, v_A, w_A)$ .
- $S_A$ plays Bob's role in input sharing Share(B,$x_i$):
  Sample and add to view: $n$ random bits $r_1,\ldots,r_n$ in place of Alice's shares for Bob's inputs.
- $S_A$ plays Bob's role in Open(A,[d]) and Open(A,[e]) during AND([x],[y]) gate evaluation:
  Sample and add to view random bits $d_B$ and $e_B$ .
- $S_A$ plays Bob's in output wire opening:
  Add to view the value $x^L{}_B = x^L \oplus x^L{}_A$

# Privacy: Simulated View ≡ Real View

We next argue that simulated view $\equiv_{\text{perfect}}$ real view .

Input and randomness are sampled identically to the real view.

Messages received:

- Simulated messages for the t Beaver triples shares $(u_A, v_A, w_A)$ and Bob's input shares $r_1, \ldots, r_n$ are generated as in real protocol – identically distributed.
- Simulated messages (shares) for $d_B$ and $e_B$ are random bits; the real messages are $d_B = x_B \oplus u_B$ and $e_B = y_B \oplus v_B$ for random bits $u_B, v_B$ – identically distributed.
- Simulated message (share) for Bob's output wire is $x^L{}_B = x^L \oplus x^L{}_A$ ; in the real protocol: $x^L = x^L{}_B \oplus x^L{}_A$ – identically distributed.

# Privacy: Simulated View for Bob

Exercise. (easy)

# Pro and Cons of BeDOZa

- ☺ Perfect security (*i.e.*, unconditional)
- ☺ Dealer only needs to known an upper bound on #AND
- ☺ (Essentially) Optimal computational complexity
- 😐 Communication & Storage complexity ~ |C|
- ☹ Round complexity ~ #layers in circuit
- ☹ Trusted dealer required

# Exercise

**Exercise**:

1. Give a dry-run example of the protocol.
2. Extend protocol to arithmetic circuits over $Z_p$.

   **Hint:** use additive secret sharing for $Z_p$.

# Conclusions

# Summary of Today's Class

| | |
|---|---|
| Plain vs. Preprocessing Model | Secure-SUM example |
| One-Time Truth Table (OTTT) | 2PC, passive adversary, preprocessing model |
| Secret sharing | what it is<br>n-out-of-n linear secret sharing over GF(2) and GF(p)<br>Shamir secret sharing (over GF(p)) |
| BeDoZa (*if time permitted*) | 2PC, passive adversary, preprocessing model<br>Key advantages over OTTT: Complexity ~ \|C\| |

# … Next time:
# BeDoZa & Active Adversary