Multiparty Computation (MPC)

Arpita Patra
MPC offers more than Traditional Crypto!

- MPC goes BEYOND traditional Crypto

- Models the distributed computing applications that simultaneously demands usability and privacy of sensitive data
Satellite Collision in Space
Satellite Collision in Space

- NASA tracks 7,000 space crafts and 21,000 objects (space debris) in space
- Approximately 20,000,000,000 pairs
List of High-speed Collisions

• The 1996 collision between the French Cerise military reconnaissance satellite and debris from Ariane rocket

• The 2009 collision between the Iridium 33 communications satellite and the derelict Russian Kosmos 2251 spacecraft over Siberia, which resulted in the destruction of both satellites

• The 22 May 2013 collision between Ecuador's NEE-01 Pegaso and Argentina's CubeBug-1, and the particles of a debris cloud left over from the launch of Kosmos 1666

• On Jan. 22, 2013, debris from the destroyed Chinese satellite Fengyun 1C collided with a small Russian laser-ranging retro-reflector satellite called BLITS ("Ball Lens in The Space").
Preventing Satellite Collision in Space

- NASA tracks 7,000 space crafts and 21,000 objects (space debris) in space
- Approximately 20,000,000 pairs

- High-accuracy positional information is privy to operators

National secret
Secure Multiparty Computation (MPC)

MPC is the holy grail

- \( P_1, \ldots, P_n \)
  - ‘some’ are corrupted
  - \( P_i \) has private input \( x_i \)
  - A common \( n \)-input function \( f \)

**Goals:**
- **Correctness:** Compute \( f(x_1, x_2, \ldots, x_n) \)
- **Privacy:** Nothing beyond function output should be leaked.

**Applications:** (Dual need of data privacy & data usability)
- E-voting
- E-auction
- Biometrics
- Data Mining
- Bioinformatics
The Goal of MPC

\[ y = f(x_1, x_2, x_3, x_4) \]

REAL world
The Goal of MPC

\[ y = f(x_1, x_2, x_3, x_4) \]

REAL world
The Goal of MPC

**IDEAL world**

\[ y = f(x_1, x_2, x_3, x_4) \]

**REAL world**
The Goal of MPC

Any task

IDEAL world

\[ y = f(x_1, x_2, x_3, x_4) \]

REAL world

\[ y \approx \]

Protocol

1. \( y_1, y_2 \) \text{ input from } x_1, x_2
2. \( y_1, y_2 \) \text{ send } y_1, y_2 \text{ to } x_3, x_4
3. \( y_1, y_2 \) \text{ compute } y = y_1 + y_2 \text{ in } x_3, x_4
4. \( y_1, y_2 \) \text{ send } y \text{ to } x_1, x_2

Invisible

\( \Rightarrow \)
Important Parameters of an MPC Protocol

1. **Communication Complexity** $(b)$: Total number of bits communicated by the honest parties.

2. **Round Complexity** $(r)$: Total number of rounds of interaction in the protocol.
Two models of MPC for any function $f$

### Boolean Circuit (AND, OR, NOT, XOR)

$\land$  
$\lor$  
$\oplus$

$$f(x_1, x_2, x_3, x_4);$$

inputs are bits

### Arithmetic Circuit over finite field (Addition and Multiplication)

$$+$$

$$\cdot$$

$$f(x_1, x_2, x_3, x_4);$$

inputs are field elements

Secure Circuit evaluation: Nothing other than the output gate value will be revealed
Threshold Adversary

Assumption: any $t$ out of $n$ parties can be corrupted

- $t$ is the threshold
- which $t$ don’t known
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]

Secret Dealer
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]

Sharing Phase

Secret s Dealer

\( v_1 \) \( v_2 \) \( v_3 \) \( \ldots \) \( v_n \)

Reconstruction Phase

Less than t + 1 parties have no info’ about the secret
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]

Sharing Phase

Secret $s$

Dealer

 Reconstruction Phase

$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_n$

$\geq t + 1$ parties can reconstruct the secret
(n,t) Secret Sharing

For MPC: Linear (n,t) Secret Sharing

**Linearity:** The parties can do the following

- \( s_1 \oplus s_2 \) from \( s_1 \) and \( s_2 \)
- \( c \otimes s \) from \( c \) and \( s \)

\( c \): public constant
Shamir-sharing: \((n,t)\) - Secret Sharing for Semi-honest Adversaries

Secret \(x\) is Shamir-Shared if

Random polynomial of degree \(t\)
Reconstruction of Shamir-sharing: \((n,t)\) - Secret Sharing for Semi-honest Adversaries

The same is done for all \(P_i\)
Secure Circuit Evaluation
Secure Circuit Evaluation

1. \((n, t)\)-secret share each input
Secure Circuit Evaluation

1. \((n, t)\)-secret share each input

2. Find \((n, t)\)-sharing of each intermediate value
Secure Circuit Evaluation

1. \((n, t)\)-secret share each input

2. Find \((n, t)\)-sharing of each intermediate value
Secure Circuit Evaluation

1. (n, t)- secret share each input
2. Find (n, t)-sharing of each intermediate value

Linear gates: Linearity of Shamir Sharing - Non-Interactive
Secure Circuit Evaluation

1. (n, t)- secret share each input

2. Find (n, t)-sharing of each intermediate value

Linear gates: Linearity of Shamir Sharing - Non-Interactive

Non-linear gate: Require degree-reduction Technique. Interactive
Secure Circuit Evaluation

Privacy follows (intuitively) because:

1. No inputs of the honest parties are leaked.
2. No intermediate value is leaked.
One Significant Contribution

Underlying Network: Synchronous

Asynchronous

NO Efficient MPC protocols
Synchronous Model

- **Global Clock**
- **Channels have fixed delay**

Knows how long to wait

- ... Compute and send \( x \)
- ... Wait to receive \( x \)
Asynchronous Model

- **No Global Clock**
- Channels have *arbitrary yet finite* delay

- ... Compute and send $x$
- ... Wait to receive $x$
Asynchronous Model

- Compute and send $x$
- Wait to receive $x$

Oh! I have to drop the message
Challenges in Asynchronous Model

- $n$ parties and $t$ corrupted

$n$ parties

$X_1\rightarrow X_2\rightarrow X_n$

Cannot wait for all

Else endless waiting

Can afford to wait to listen from $(n-t)$ parties

But leads to ignoring messages of $t$ honest parties
### Synchronous vs. Asynchronous

<table>
<thead>
<tr>
<th>Linear ((O(n))) overhead MPC</th>
<th>(O(n^3)) overhead MPC</th>
</tr>
</thead>
</table>

**Our contribution:**

- **Journal of Cryptology:** \(O(n^2)\) overhead MPC without error
- **DISC'14 (Full version submitted to IEEE Transactions on Information Theory):** \(O(n)\) overhead MPC with negligible error
- **Scalable Solution** (par party communication is independent of no. of parties)
Thank You!
Adaptive Corruption stronger than Static Corruption

- Hackers constantly trying to break into computers running secure protocols but could do so after the protocol has started.

- The attacker first looks at the communication and then decide who to corrupt (not allowed in static model)
<table>
<thead>
<tr>
<th>Static</th>
<th>vs.</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient; Constant Round MPC</td>
<td></td>
<td>In-efficient; NO constant MPC</td>
</tr>
</tbody>
</table>

Our contribution:

- **TCC’14 (Full version awaiting acceptance in Journal of Cryptology):** Proposed slightly restricted models (one adaptive corruption). Proposed solutions $\approx$ static protocols

- **PODC’15 (Full version submitted to Journal of Cryptology):** Proposed yet another practical model (partial erasure). Proposed solutions $\approx$ static protocols partial erasure
Threshold Adversary

Assumption: any $t$ out of $n$ parties can be corrupted

- $t$ is the threshold
- which $t$ don’t known
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]

Sharing Phase

Secret Dealer

\(v_1\) \(v_2\) \(v_3\) \(\ldots\) \(v_n\)

Reconstruction Phase

Less than \(t + 1\) parties have no info' about the secret
(n, t) - Secret Sharing
[Shamir 1979, Blackley 1979]

Sharing Phase

Reconstruction Phase

\[ v_1, v_2, v_3, \ldots, v_n \]

\[ \geq t + 1 \text{ parties can reconstruct the secret} \]
(n,t) Secret Sharing

Linear (n,t) Secret Sharing of secret s

For MPC: Linear (n,t) Secret Sharing

**Linearity:** The parties can do the following

\[ s_1 \oplus s_2 \quad \text{from} \quad s_1 + s_2 \]

\[ c \times s \quad \text{from} \quad c \times s \]

\[ s \quad \text{from} \quad s \]

\[ \text{c: public constant} \]
Reconstruction of Linearly Shared Secrets

The efficiency of MPC reduces to the efficiency of reconstructing a secret efficiently.

**Part I: How to reconstruct a secret efficiently?**

- **Semi-honest Model:** $(n,t)$ secret sharing and reconstruction with linear $(O(n))$ overhead;
- **Malicious Model:** $(n \geq 3t+1,t)$ secret sharing and reconstruction with linear $(O(n))$ overhead; $n \geq 3t+1$ is necessary and sufficient for error-free reconstruction;
- **Malicious Model:** $(n \geq 2t+1,t)$ secret sharing and reconstruction with linear $(O(n))$ overhead; $n \geq 2t+1$ is necessary and sufficient for reconstruction.

**Part II: How MPC reduces to reconstruction of secrets?**

- **Semi-honest Model:** Linear Overhead MPC
- **Malicious Model:** Linear Overhead Error-free MPC with $n \geq 3t+1$
- **Malicious Model:** Linear Overhead MPC with $n \geq 2t+1$
Shamir-sharing: \((n, t)\) - Secret Sharing for Semi-honest Adversaries

Secret \(x\) is Shamir-Shared if

Random polynomial of degree \(t\)
Reconstruction of Shamir-sharing: \((n,t)\) - Secret Sharing for Semi-honest Adversaries

\[
P_1, x_1
\]
\[
P_2, x_2
\]
\[
P_3, x_3
\]
\[
P_n, x_n
\]

The same is done for all \(P_i\)

Communication Complexity (CC): \(O(n^2)\)
Efficient Reconstruction of $(n,t)$- Shamir for Semi-honest Adversaries

$\Rightarrow$ Can we do better? $O(n)$ Easy 😊😊

\[ P_1 x_1 P_1 \]
\[ P_2 x_2 P_2 \]
\[ P_3 x_3 P_3 \]
\[ P_n x_n P_n \]

......Because we are assuming semi-honest adversaries.

Challenge: Linear Solution tolerating malicious adversaries
Reconstruction of \( (n \geq 3t+1, t) \) - Shamir Secret Sharing for Malicious Adversaries

\[ P_1 \quad x_1 \]

\[ P_2 \quad x_2 \]

\[ P_i \]

\[ P_3 \quad x'_3 \]

\[ P_n \quad x'_n \]

Error Correction Possible to get back \( x \)!!

\[ \Rightarrow \text{Shamir-sharing} \rightarrow \text{Reed-Solomon Code (linear) with distance } n-t \geq 2t \]

\[ \Rightarrow \text{Fundamental result: distance } \geq 2 \times \text{errors} \]

The same is done for all \( P_i \)  

Communication Complexity (CC): \( O(n^2) \)
Efficient Reconstruction of \((n \geq 3t+1, t)\) - Shamir Secret Sharing for Malicious Adversaries

>> Can we do better? \(O(n)\)

Can we use the same trick as before?

Trick for semi-honest adversaries does not work 😞
Reconstruction of \((t+1)\) secrets with \(O(n^2)\) Cost for \(n \geq 3t+1\)

\[ f(x) = a_0 + a_1 x + \ldots + a_t x^t \]

\[ b_i = f(i) = a_0 + a_1 i + \ldots + a_t i^t \]

Communication Complexity: \(O(n^2)\)
Reconstruction of \((t+1)\) secrets with \(O(n^2)\) Cost for \(n \geq 3t+1\)

\[
f(x) = a_0 + a_1 x + \ldots + a_t x^t
\]

\[
b_i = f(i) = a_0 + a_1 i + \ldots + a_t i^t
\]

The same is done for all \(P_i\)

Communication Complexity (CC): \(O(n^2)\)
Reconstruction of $(n \geq 2t+1, t)$ - Shamir Secret Sharing for Malicious Adversaries

Error Correction NOT Possible!

- Shamir-sharing $\rightarrow$ Reed-Solomon Code (linear) with distance $n-t \geq t$
- Fundamental result: distance $\geq 2 \times$ errors

- Enhance the secret sharing in such a way that a party cannot lie about his share
- If they lie, they will be detected as corrupted

- Use information-theoretic MACs
Information Theoretic MACs

Message: \(a\)

MAC: \(MAC_K(a)\)

\(a, MAC_K(a)\) → Accept

\(a', MAC_K(a)\) → Reject
Information Theoretic MAC

\[ K = (\alpha, \beta) \] from \( F \)

\[ MAC_K(a) = \alpha a + \beta \]

- For \( a, MAC_K(a) \), Accept
- For \( a', MAC_K(a) \), Reject
Homomorphic Information Theoretic MAC

\[ a \quad MAC_{K_1}(a) = \alpha a + \beta_1 \]

\[ b \quad MAC_{K_2}(b) = \alpha b + \beta_2 \]

\[ a + b \quad MAC_{K}(a + b) = MAC_{K_1}(a) + MAC_{K_2}(b) = \alpha (a + b) + (\beta_1 + \beta_2) \]

\[ K_1 = (\alpha, \beta_1) \]

\[ K_2 = (\alpha, \beta_2) \]

\[ K = (\alpha, \beta_1 + \beta_2) \]

No interaction needed.

We can use similar trick for multiplication by publicly known constant and again no need of interaction.
Enhanced Shamir-sharing: \((n \geq 2t+1, t)\) - Secret Sharing for Malicious Adversaries

Secret \(s\) is enhanced-Shamir-Shared if \(s\) is Shamir-shared + Pairwise MACs

The same holds for all \(P_i\) Enhanced-Shamir-Sharing is also linear 😊
Reconstruction of $(n \geq 2t+1, t)$ - Shamir Secret Sharing for Malicious Adversaries

\[ P_1 \quad x_1 \quad MAC_i(x_1) \]

\[ P_2 \quad x_2 \quad MAC_i(x_2) \]

\[ P_i \]

Verify with $K_1$. If accept, keep $x_1$

Verify with $K_2$. If accept, keep $x_2$

Verify with $K_3$. If reject, ignore $x_3$

\[ P_3 \quad x'_3 \quad MAC_i(x'_3) \]

\[ P_n \quad x'_n \quad MAC_i(x'_n) \]

Will have at least $t+1$ accepted shares (from the honest parties)

The same is done for all $P_i$

Communication Complexity (CC): $O(n^2)$
Efficient Reconstruction of \((n \geq 2t+1,t)\) - Enhanced Shamir Secret Sharing for Malicious Adversaries

\begin{align*}
f(x) &= a_0 + a_1x + \ldots + a_t x^t \\
b_i &= f(i) = a_0 + a_1 i + \ldots + a_t i^t
\end{align*}

Challenge: Linear Overhead tolerating malicious adversaries for \(n \geq 2t+1\)

Does the trick for \(3t+1\) work? But Error Correction does not work 😞
Reconstruction of $n(t+1)$ secrets with $O(n^3)$ Cost for $n \geq 2t+1$

\[ f_0(x) = a_{00} + a_{01}x + \ldots + a_{0t}x^t \]

\[ b_{0j} = f_0(j) = a_{00} + a_{01}j + \ldots + a_{0t}j^t \quad j = 1, \ldots, n \]
Reconstruction of $n(t+1)$ secrets with $O(n^3)$ Cost for $n \geq 2t+1$

\[ f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t \]
\[ b_{ij} = f_i(j) = a_{i0} + a_{i1}j + \ldots + a_{it}j^t \quad j = 1, \ldots, n \]

$t+1$ columns of $B$ matrix are required to reconstruct $A$
Reconstruction of $n(t+1)$ secrets with $O(n^3)$ Cost for $n \geq 2t+1$

Communication Complexity: $O(n^3)$
Reconstruction of $n(t+1)$ secrets with $O(n^3)$ Cost for $n \geq 2t+1$

If the parties honestly communicate their column to every other party, we are done.

But there is no guarantee that a corrupted $P_i$ will do that.

Prevent a party from sending a wrong column (he should be caught if he does so)

Communication Complexity: $O(n^3)$
Reconstruction of \( n(t+1) \) secrets with \( O(n^3) \) Cost for \( n \geq 2^{t+1} \)

\[
\begin{align*}
&b_{1i} + r_{1i} \\
&b_{2i} + r_{2i} \\
&b_{ni} + r_{ni} \\
&P_i + P_j \quad r_{1i}b_{1i} + r_{2i}b_{2i} \ldots + r_{ni}b_{ni}
\end{align*}
\]

\( r_{1i}, r_{2i}, \ldots, r_{ni} \text{ are remains hidden from everyone} \)
Reconstruction of $n(t+1)$ secrets with $O(n^3)$
Cost for $n \geq 2t+1$

$b'_1i$    $r_{1i}$    $r_{2i}$    $r_{ni}$

$\rightarrow$ A corrupted $P_i$ will fail very high probability.
$\rightarrow$ $P_j$ will ignore $P_i$'s column

Verify if the sum is same as the below

$p_j$  $(r_{1i}b_{1i} + r_{2i}b_{2i} + \ldots + r_{ni}b_{ni})$

How $P_j$ can compute $(r_{1i}b_{1i} + r_{2i}b_{2i} + \ldots + r_{ni}b_{ni})$ without revealing his random choice.
Reconstruction of $n(t+1)$ secrets with $O(n^3)$ Cost for $n \geq 2t+1$

For each pair of parties Communication Complexity: $O(n)$
Part II: Reduction from MPC to Reconstruction of Secrets

Reduction holds for any linear secret sharing 😊
Secure Circuit Evaluation

\[
\begin{align*}
X_1 & \rightarrow + \rightarrow + \\
X_2 & \rightarrow + \rightarrow + \\
X_3 & \rightarrow \times \\
X_4 & \rightarrow \times \\
\rightarrow Y & = C
\end{align*}
\]
Secure Circuit Evaluation
Secure Circuit Evaluation

1. (n, t)- secret share each input
Secure Circuit Evaluation

1. \((n, t)\)-secret share each input
2. Find \((n, t)\)-sharing of each intermediate value
1. $(n, t)$- secret share each input

2. Find $(n, t)$-sharing of each intermediate value
Secure Circuit Evaluation

1. (n, t)-secret share each input

2. Find (n, t)-sharing of each intermediate value

**Linear gates:** Linearity of Shamir Sharing - Non-Interactive
1. (n, t)- secret share each input

2. Find (n, t)-sharing of each intermediate value

**Linear gates:** Linearity of Shamir Sharing - Non-Interactive

**Non-linear gate:** Require degree-reduction Technique. Interactive
Secure Circuit Evaluation

1. (n, t)-secret share each input
   Reduction to one reconstruction

2. Find (n, t)-sharing of each intermediate value
   Linear gates: Linearity of Shamir Sharing - Non-Interactive
   Non-linear gate: Require degree-reduction Technique. Interactive
   Reduction to two reconstructions

3. Open output by Reconstruction algorithm
Secure Circuit Evaluation

Privacy follows (intuitively) because:

1. **No inputs** of the **honest** parties are leaked.

2. **No intermediate value** is leaked.
Beaver's Circuit-randomization Technique for Multiplication

Don Beaver
CRYPTO 91
Beaver's Circuit-randomization Technique for Multiplication

Multiplication Triple
Beaver’s Circuit-randomization Technique for Multiplication

Ex:

Multiplication Triple
Beaver’s Circuit-randomization Technique for Multiplication

Ex:

 multiplication triple

\[ 2 + 1 + 5 + 9 + 3 + 3 = 21 \]

\[ 1 \times 6 \times 6 = 36 \]
Beaver’s Circuit-randomization Technique for Multiplication

Ex:

Beaver’s Circuit

Multiplication Triple
Beaver's Circuit-randomization Technique for Multiplication
Beaver’s Circuit-randomization Technique for Multiplication

- Random and Private $a, b$

Multiplication Triple
Beaver’s Circuit-randomization Technique for Multiplication

- Random and Private $a, b$
- Independent of the multiplication gate

- Two reconstructions
- Linear operations
Beaver’s Circuit-randomization Technique for Multiplication

- Random and Private $a, b$
- Independent of the multiplication gate

- Two reconstructions
- Linear operations
Let $M$ be the number of multiplication gates in the circuit.
Secure Circuit Evaluation Using Beaver Circuit Randomization

- Let $M$ be the number of multiplication gates in the circuit
- Ask triple-oracle for $M$ multiplication triples
Secure Circuit Evaluation Using Beaver Circuit Randomization

- Let $M$ be the number of multiplication gates in the circuit
- Ask triple-oracle for $M$ multiplication triples
Secure Circuit Evaluation Using Beaver Circuit Randomization
Secure Circuit Evaluation Using Beaver Circuit Randomization
Secure Circuit Evaluation Using Beaver Circuit Randomization
Secure Circuit Evaluation Using Beaver Circuit Randomization
Secure Circuit Evaluation Using Beaver
Circuit Randomization
Secure Circuit Evaluation Using Beaver Circuit Randomization
Secure Circuit Evaluation Using Beaver Circuit Randomization
Why Beaver’s Trick is Useful

Offline Phase - Online Phase Paradigm

**Offline Phase**: Sitting Idle, Generate as many shared triples as possible---raw data

**Online Phase**: Use the raw data for circuit evaluation.

Triple generation **parallelizable** $\rightarrow$ efficiency

On the contrary, multiplications gates can not be evaluated in parallel
Online Complexity

How efficiently can we reconstruct a shared secret?

Reconstruction cost of one shared secret = Cost Per Multiplication (asymptotically)

1. Generate shared data
2. Implement the oracle
Thank You!
Beaver’s Circuit Randomization Technique

\[ xy = ((x-a) +a)((y-b)+b) = (a + a)(\beta + b) = ab + \alpha b + \beta a + \alpha \beta \]

\[ \alpha = x-a \quad \beta = y-b \]
Beaver's Circuit Randomization Technique
Beaver's Circuit Randomization Technique

\[ \text{Open } \alpha = x-a \]

\[ \text{Open } \beta = y-b \]
Linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)
Linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)

\[ \begin{align*}
\alpha_1 & \quad a_1 \\
\alpha_2 & \quad a_2 \\
\alpha_3 & \quad a_3 \\
\alpha_4 & \quad a_4 \\
\end{align*} \]

each party does locally

\[ \begin{align*}
b_1 & \quad b_1 \\
b_2 & \quad b_2 \\
b_3 & \quad b_3 \\
b_4 & \quad b_4 \\
\end{align*} \]

\[ \begin{align*}
c_1 & \quad c_1 \\
c_2 & \quad c_2 \\
c_3 & \quad c_3 \\
c_4 & \quad c_4 \\
\end{align*} \]
Linearity of (n, t) Shamir Secret Sharing

say \( t = 1 \)

Addition is Absolutely free
Linearity of (n, t) Shamir Secret Sharing

say \( t = 1 \)

\[ a, a_1, a_2, a_3, a_4 \]

\[ \alpha_1, \alpha_2, \alpha_3, \alpha_4 \]
Linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)

\[ \begin{align*} a_1 & \otimes c \\ a_2 & \otimes c \\ a_3 & \otimes c \\ a_4 & \otimes c \end{align*} \]
Linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)

\[ a \times c \times c \times c \times c \]

c is a publicly known constant
Linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)
Linearity of (n, t) Shamir Secret Sharing

say \( t = 1 \)

\[ a \]

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \]

\[ \otimes c \]

\[ d_1 \quad d_2 \quad d_3 \quad d_4 \]

\[ \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \]
Linearity of (n, t) Shamir Secret Sharing

say $t = 1$

\[ \begin{align*}
\alpha_1 & \quad \alpha_2 & \quad \alpha_3 & \quad \alpha_4 \\
\bullet & \quad \bullet & \quad \bullet & \quad \bullet \\
\alpha & \quad \alpha_1 & \quad \alpha_2 & \quad \alpha_3 & \quad \alpha_4 \\
\downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow \\
d_1 & \quad d_2 & \quad d_3 & \quad d_4 \\
\end{align*} \]
Linearity of $(n, t)$ Shamir Secret Sharing

say $t = 1$

Multiplication by public constants is Absolutely free
Non-linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)
Non-linearity of \((n, t)\) Shamir Secret Sharing

say \(t = 1\)

\[
\begin{align*}
\alpha_1 & \quad \alpha_2 & \quad \alpha_3 & \quad \alpha_4 \\
\times & \quad \times & \quad \times & \quad \times \\
\beta_1 & \quad \beta_2 & \quad \beta_3 & \quad \beta_4
\end{align*}
\]
Non-linearity of (n, t) Shamir Secret Sharing

say \( t = 1 \)
Non-linearity of (n, t) Shamir Secret Sharing

say $t = 1$

\[ a_1 \otimes a_2 \otimes a_3 \otimes a_4 \]

\[ b_1 \otimes b_2 \otimes b_3 \otimes b_4 \]

\[ d_1 \otimes d_2 \otimes d_3 \otimes d_4 \]
Non-linearity of (n, t) Shamir Secret Sharing

say \( t = 1 \)

\[
\begin{align*}
\alpha_1 & \quad \alpha_2 & \quad \alpha_3 & \quad \alpha_4 \\
\times & \quad \times & \quad \times & \quad \times
\end{align*}
\]

\[
\begin{align*}
b_1 & \quad b_2 & \quad b_3 & \quad b_4 \\
\downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow
\end{align*}
\]

\[
\begin{align*}
d_1 & \quad d_2 & \quad d_3 & \quad d_4
\end{align*}
\]
Non-linearity of $(n, t)$ Shamir Secret Sharing

say $t = 1$
Non-linearity of $(n, t)$ Shamir Secret Sharing

Multiplication of shared secrets is not free
Linear (n,t) Secret Sharing

\[ s_1 \oplus s_2 = s_1 + s_2 \]

\[ c \odot s = cs \]

Linear Operation

Non-Linear Operation
An Abstract Tool for the Generic Solution

\[(n, t)\] LOCKED BOX REPRESENTATION

A secret \(s\) is locked in the box

- Any \(t\) parties cannot open the box
An Abstract Tool for the Generic Solution

\((n, t)\) LOCKED BOX REPRESENTATION

A secret \(s\) is locked in the box

- Any \(t\) parties cannot open the box
An Abstract Tool for the Generic Solution

\[(n, t)\] LOCKED BOX REPRESENTATION

A secret \(s\) is locked in the box

- Any \(t\) parties cannot open the box

\[P_1\]
\[P_2\]
\[P_n\]
An Abstract Tool for the Generic Solution

(n, t) LOCKED BOX REPRESENTATION

A secret $s$ is locked in the box

- Any $t$ parties cannot open the box
- Any $(t + 1)$ parties can open the box