Secure MPC Lecture Notes

Sahil Muhammed

June 2025

1 Secure MPC (Multi-party company)

- Privacy-preserving Data Mining
- Membership
- Privacy-preserving ML
- MPC is easy if we could trust someone.
- A trusted party could get all the data and compute the value of the function at given inputs and then return it back.
- This is the ideal solution.

1.1 Problem:

- Creates a single point of failure.
- Trust is very rare and volatile.
- If there is trust in the world, then our problem is solved.

1.2 MPC Protocol

- Goal: to emulate the role of trusted party
- P_i should not learn anything beyond what it can form (y, x_i)

2 Secure Multi-Party Computation: Part II - A Toy NPC Protocol

2.1 Sum Function:

- Goal: to securely compute $S = b_1 + \cdots + b_n, b_i \in \{0, 1\}$
- P_i should not learn anything beyond what it can form (S, b_i)
- Parties connected by pair-wise private and authentic channel
- Each P_i has a private bit $b_i \in \{0, 1\}$
- Goal: to securely compute $S = b_1 + \cdots + b_n$
- $S \in \{0, \ldots, n\}$
  - Parties agree upon a public modulus $M > n$
  - All operations are done over $\mathbb{Z}_M$
- k is randomly sampled from $\mathbb{Z}_5$, and c_1 is defined to be the sum of this random value with its input b_1.

- $c_1 = b_1 + k$
- Now, $c_2 = c_1 + b_2 = k + b_1 + b_2$
- $c_3 = c_2 + b_3 = k + b_1 + b_2 + b_3$
- $c_4 = c_3 + b_4 = k + b_1 + b_2 + b_3 + b_4$
- $S = c_1 + c_2 + c_3 + c_4 + k$
- Does P_i learn anything beyond what it can learn from (S, b_i)?
- $View_i$ = 'information' learnt by P_i during the protocol execution
- Security goal: P_i should not learn anything beyond what it can from (S, b_i)

2.2 Toy MPC Protocol: Possible Attacks

- What happens if two parties P_i and P_{i+2} collude together in the MPC protocol?
  - The input b_i will be learnt
  - Not possible in the ideal solution
- Hence, the MPC protocol is not as secure as the ideal solution, if two parties collude.

3 Secure Multi-Party Computation: Part III - Maurer's MPC Protocol

3.1 The Arithmetic Circuit Abstraction

- The protocol is a generic MPC protocol for securely computing any function over a finite Ring $(\mathbb{R}, +, \cdot)$
- Simplifying assumptions (without loss of generality):
  - Each P_i has a single input value from $\mathbb{R}$
  - Single function-output, to be learnt by everyone.
  - The function is a deterministic function
- There is the publicly known circuit for computing the function f.
- The arithmetic-circuit abstraction is without loss of generality.
  - Any (efficient) algorithm/computation can be represented by an efficient Boolean circuit B_{cir} consisting of AND and NOT gates.
  - B_{cir} can be simulated by an equivalent (efficient) arithmetic circuit A_{cir} over $(\mathbb{R}, +, \cdot)$
  - Bit 0 $\leftrightarrow$ additive identity of $\mathbb{R}$
  - Bit 1 $\leftrightarrow$ multiplicative identity of $\mathbb{R}$
  - $\neg b \leftrightarrow$ "1" $-$ "b" over $\mathbb{R}$
  - a AND $b \leftrightarrow$ "a" $\cdot$ "b" over $\mathbb{R}$
- Example of circuit emulation over Boolean ring $\mathbb{R} = \{(0,1), \text{XOR}, \cdot\}$ — all operations are modulo 2.
- Why arithmetic circuits and not boolean circuits?
  - There are instances where we can represent in a compact manner using arithmetic circuits.
- Computing $f(x_1, \ldots, x_n)$ is equivalent to evaluating the corresponding arithmetic circuit over $x_1, \ldots, x_n$.
- If $x_1, \ldots, x_n$ are made public, then the circuit can be evaluated locally (without any interaction) by the parties themselves.
- MPC protocol will also perform circuit-evaluation such that corrupt parties

3.2 Modelling Corruption in an MPC Protocol

- Bad people work together to cause maximum harm.
- Tobe To model this, we assume that there is a single monolithic/centralized entity called adversary, who 'controls' a 'certain' number of parties.
- Threshold model: at most t corrupt parties, where $t < n$. It is not publicly known who the corrupt parties are.
- $t = 2 \Rightarrow$ Possible corruption scenarios is 4 choose 2 = 6 cases.
- Question: What if there was only 1 corrupt party?
- Ans: In that case, the cases will be a subset of the case we are considering.
- Unbounded: Corrupt parties are computationally unbounded.
- Passive/Semi-honest: Adversary is a passive observer, eavesdrops on the corrupted parties.
- Static: Adversary corrupts parties at the onset of a protocol.

3.3 (n, t) Secret Sharing Scheme [Shamir 1979, Blakley 1979]

- A set of parties $P = \{P_1, \ldots, P_n\}$ connected by pair-wise private and authentic channels.
- A designated dealer $D \in P$, with a secret input $s \in S$.
- Goal: to distribute a share s_i of s to every party P_i:
  - Should be impossible for any set of t or less number of share-holders to pool their shares and reconstruct back s
    - Even if the t parties are given unlimited time and resources
  - Should be possible for any set of $(t + 1)$ or more share-holders to pool their shares and reconstruct back s.
- Less than $t + 1$ shares: no information about the secret
- $t + 1$ or more shares: complete secret

4 Secure MPC via Secret-Shared Circuit Evaluation

- Secure MPC equivalent to secret-shared circuit evaluation over a ring
- Inputs: (n, t)-secret shared
- Intermediate gates:
  - (n, t) secret-shared gate input(s) $\Rightarrow$ (n, t) secret-shared gate output
- Output gate: publicly reconstruct.
- Any generic MPC protocol follows the above template.
- An instantiation of (n, t) secret-shared.
- Protocols for addition and multiplication over (n, t) secret-shared inputs.

4.1 (n, t) Replicated Secret Sharing (RSS): [ISN87]

- Idea: divide the secret randomly such that every candidate adversarial set of size t misses at least one share
- All operations over a finite Ring $(\mathbb{R}, +, \cdot)$
- $s = s_1 + s_2 + s_3$
- $G_i = P - Z_i$
- A piece s_i can be replicated to multiple entities depending on the group.

4.2 (n, t)-RSS: Sharing Protocol

- $k = |n \text{ choose } t|$
- $Z = \{Z_1, \ldots, Z_k\}$: all subsets of t parties
- $G_i = P - Z_i$
- $Sh_{RSS}(s)$
  - D on having input $s \in \mathbb{R}$ does the following
  - Randomly pick $s_1, \ldots, s_k \in \mathbb{R}$, such that $s = s_1 + \cdots + s_k$
  - For $i = 1, \ldots, k$ send s_i to every party in G_i.
  - For $j = 1, \ldots, n$ party P_j outputs its share S_j where $S_j = \{s_i\}_{P_j \in G_i}$
  - Privacy: if D is honest, then any subset of t shareholders learn no information about s.
  - Adversary will be missing at least one piece s_i which will be random for it.

4.3 (n, t)-RSS: Correctness

- Claim: Any subset A of $t+1$ parties can reconstruct s
- Union of the shares of $A \Rightarrow \{s_1, \ldots, s_k\}$
- If the union of the shares of A misses s_i then it implies that $A = Z_i$. Contradiction: size of A is $t+1$, but the size of these Z_i (adversaries) is t.

4.4 Secure MPC via Secret-Shared Circuit Evaluation

- Secure MPC equivalent to secret-shared circuit evaluation over a ring.

4.5 (n, t)-RSS: Linearity Part I

- $a = a_1 + \cdots + a_i + \cdots + a_k$
- $b = b_1 + \cdots + b_i + \cdots + b_k$
- $a + b = (a_1 + b_1) + \cdots + (a_i + b_i) + \cdots + (a_k + b_k)$

4.6 (n, t)-RSS: Linearity Part II

- $a = a_1 + \cdots + a_i + \cdots + a_k$; public constant c.
- $a + c = (a_1 + c) + a_2 + \cdots + a_k$ (Gives replicated share of $a + c$)
- Non-interactive process
- Similarly, if we tell all G_i to multiply their share with c, then we have the replicated share of $a \cdot c$)
- But, to get the replicated share of $a \cdot b$, we can't ask each of the groups to multiply a_i with b_i. That wouldn't result in $a \cdot b$.
- To get this done, we would require interaction. Non-linear operation $\rightarrow$ multiplication.

4.7 (n, t)-RSS: Evaluating Multiplication Gate with $t < \frac{n}{2}$

- Idea: for each summand $a_i \cdot b_j$, identify the designated party who will (n,t)-RSS-Share $a_i \cdot b_j$.
- $G_i = P - Z_i \rightarrow$ Parties having the share a_i
- $G_j = P - Z_j \rightarrow$ Parties having the share b_j
- $G_i \cap G_j = P - (Z_i \cup Z_j)$
- $G_i \cap G_j = P - (Z_i \cup Z_j) \neq \emptyset$
- Least indexed party from G_i and G_j ...

4.8 Downside of RSS-Based MPC?

- What will be the share-size of each party?
  - $O(k)$, where k is n choose t.
  - This is exponential in n if $t < \frac{n}{2}$

5 Secure Multi-Party Computation: Part IV - Shamir's Secret Sharing Scheme

5.1 Polynomials Over a Field

- Let $(\mathbb{F}, +, \cdot)$ be a field
- Every non-zero element has a multiplicative inverse, this property is not there in rings.
- Definition: a t-degree polynomial $f(X)$ over $\mathbb{F}$ is of the form $f(X) = a_0 + a_1 \cdot X + \cdots + a_t \cdot X^t$
- Example: consider the field $(\mathbb{Z}_7, +_7, \cdot_7) \rightarrow$ addition/multiplication modulo 7.
- $f(X) = 6 + 2X + 3X^4$
- $f(1) = (6 + 2 + 3) \mod 7 = 4$
- Properties:
  - Theorem (Abstract algebra): a t-degree polynomial over $\mathbb{F}$ has at most t roots.
  - Theorem (Abstract algebra): two distinct t-degree polynomials over $\mathbb{F}$ can have at most t common values
  - Let $f(X) = X^2 + 4X + 2$
  - Let $g(X) = X^2 + 3X + 3$ (over $(\mathbb{Z}_7, +_7, \cdot_7)$)
  - $f(1) = g(1) = 0$
  - Point $(1, 0)$ is common

5.2 Lagrange's Polynomial Interpolation

- Theorem: Let $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ be pairs of elements from $\mathbb{F}$, where $x_1, \ldots, x_{d+1}$ are distinct. Then, there exists a unique d-degree polynomial $f(X)$ over $\mathbb{F}$, such that $f(x_i) = y_i$, for $1 \leq i \leq d+1$.
- Idea: Express the unknown $f(X)$ as a linear combination of $d+1$ d-degree polynomials $\delta_1(X), \ldots, \delta_{d+1}(X)$
- $$f(X) = y_1 \cdot \delta_1(X) + \cdots + y_i \cdot \delta_i(X) + \cdots + y_{d+1} \cdot \delta_{d+1}(X)$$
- $\delta_1(x_1) = 1, \delta_i(x_i) = 1, \delta_{d+1}(x_{d+1}) = 1$
- $\delta_1(x_2) = \cdots = \delta_1(x_{d+1}) = 0, \delta_i(x_1) = \cdots = \delta_i(x_{i-1}) = \delta_i(x_{i+1}) = \ldots, \delta_{d+1}(x_1) = \cdots = \delta_{d+1}(x_d) = 0$
- Cannot do Lagrange's interpolation over a ring.

5.3 Properties of t-degree Polynomial

- Let $P^{s,t}$ = set of all t-degree polynomials over $\mathbb{F}$, with s as the constant term.
- Each $f(X) \in P^{s,t}$ is of the form $f(X) = s + a_1 \cdot X + \cdots + a_t \cdot X^t$, where each $a_1, \ldots, a_t \in \mathbb{F}$
- $$|P^{s,t}| = |\mathbb{F}|^t$$
- Ex: $t = 2$, $\mathbb{F} = (\mathbb{Z}_3, +_3, \cdot_3)$ and $s = 1$

- $1, 1+X, 1+X^2, 1+X+X^2, 1+2X+X^2$
- $1+2X, 1+2X^2, 1+X+2X^2, 1+2X+2X^2$
- These are in $P^{1,2}$
- $\{(x_{i_1}, y_{i_1}), \ldots, (x_{i_t}, y_{i_t})\} \rightarrow$ arbitrary values from $\mathbb{F}$:
- $$x_{i_1} \neq \ldots \neq x_{i_t} \neq 0$$
- For any given $s \in \mathbb{F}$, there is a unique polynomial from $P^{s,t}$ passing through $\{(0, s), (x_{i_1}, y_{i_1}), \ldots, (x_{i_t}, y_{i_t})\}$
- Input: $s \in \mathbb{F}$
- Pick $f(X) \in_r P^{s,t}$
- Output: $f(x_1), \ldots, f(x_n) \rightarrow y_i = f(x_i)$
- $x_1 \neq \ldots \neq x_n \neq 0$ and publicly known
- This is known as Exp_{Shamir}
- How much information about the input s is learnt through any subset of t output values?
- $\Pr_{f(X) \in_r P^{s,t}}[(f(x_1) = y_1) \text{ AND } \ldots \text{ AND } (f(x_t) = y_t)] = \frac{1}{|P^{s,t}|}$
- $\Pr_{g(X) \in_r P^{s',t}}[(g(x_1) = y_1) \text{ AND } \ldots \text{ AND } (g(x_t) = y_t)] = \frac{1}{|P^{s',t}|}$
- Since both have the same probability, there is no way to distinguish between them.
- In summary, we are hiding in the constant term of the polynomial

5.4 Shamir's (n, t) Secret-Sharing Scheme

- Public set-up: finite field $(\mathbb{F}, +, \cdot)$, with $|\mathbb{F}| > n$ and publicly known, non-zero distinct elements $x_1, \ldots, x_n \in \mathbb{F}$
- $Sh_{Shamir}(s)$
  - Randomly pick $a_1, \ldots, a_t \in \mathbb{F}$
  - Define the polynomial $f(X) = s + a_1 \cdot X + \cdots + a_t \cdot X^t$
  - $$f(X) \in_r P^{s,t}$$
  - For $i = 1, \ldots, n$, compute the share $s_i = f(x_i)$
- Correctness: any set of $(t+1)$ shares suffice to uniquely interpolate back t-degree polynomial $f(X)$
- Privacy: Any set of t shares ...

6 Secure Multi-Party Computation: Part V - BGW MPC Protocol [Michael Ben-Or, Shafi Goldwasser, Avi Wigderson]

6.1 Why do we need to reduce the degree here?

- The degree-of-sharing may blow up so much that we don't have sufficient shares to reconstruct the output
- 100 immediate successive multiplications, would result in the $(n, 102t)$-SSS, and we don't have enough shares to reconstruct it back.
- To reconstruct $y =$ "successive multiplication 100 times", we would need ...

6.2 The Degree Reduction Problem

- No "additional information" about a and b should be leaked in the process.
- Requires interaction among the parties

BGW : Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). STOC 1988: 1-10

GRR : Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. PODC 1998:

6.3 GRR Method of Degree-Reduction with $t < \frac{n}{2}$

- $A(Z) - a_1, a_2, \ldots, a_i, \ldots, a_n$ $(n, t) \to a$
- $B(Z) - b_1, b_2, \ldots, b_i, \ldots, b_n$ $(n, t) \to b$
- $K(Z) - k_1, k_2, \ldots, k_i, \ldots, k_n$ $(n, 2t) \to a \cdot b$
- Idea: ab is a linear function of $k_1, \ldots, k_n$
- $K(Z) = \delta_1(Z) \cdot k_1 + \cdots + \delta_n(Z) \cdot k_n$
- $K(0) = \delta_1(0) \cdot k_1 + \cdots + \delta_n(0) \cdot k_n$
- $ab = d_1 \cdot k_1 + \cdots + d_n \cdot k_n$
- $\Rightarrow a \cdot b = d_1 \cdot k_1$ (n, t)-SSS $+ \cdots + d_n \cdot k_n$ (n, t)-SSS

6.4 Shared Circuit-Evaluation in the Pre-Processing Model

- Also known as shared circuit-evaluation with correlated randomness
- Inputs: Randomly (n, t) - secret-shared
- Shared gate-evaluation: If gate-input(s) is/are randomly (n, t) secret-shared, then gate output is randomly (n, t) secret-shared.
- Linear gates: non-interactive
- Multiplication gates: ...

6.5 Evaluation of Multiplication Gates via Beaver Multiplication Triples

- Multiplication is the main operation which results in a 'bottle-neck'.
- $d = u - a$; $e = v - b$;
- $w = (u - a + a)(v - b + b)$
- $= de + db + ea + c \to$ linear function of (a, b, c)
- Steps of the online phase protocol:
  - 1. Parties compute $d = u - a$; $e = v - b$.
  - 2. Parties reconstruct d and e (Just need 2 public reconstructions)
- Privacy of gate-inputs:
  - (a, b, c): independent of gate-inputs
  - d, e: one-time pad encryption of gate-inputs
- Need to have M multiplication-triples if there are M multiplication gates in the circuit
- Assume that pre-processing is done without knowing how this is done.

6.6 Generating Secret-Shared Multiplication-Triples

- Goal: to generate M number of (n, t)-secret-shared multiplication-triples (a, b, c) such that:
  - $a \in_r \mathbb{F}$
  - $b \in_r \mathbb{F}$
  - $c = a \cdot b$ (product of the first two elements sampled from $\mathbb{F}$)
  - (a, b, c) remains random for the adversary
- Done in two stages:
  - Stage I: Generating (n, t)-secret-sharing of a random a and a random b
  - Stage II: Computing (n, t)-secret-sharing of $a \cdot b$ from (n, t)-secret-sharing of a and b
- All instances of multiplication-protocol can be executed in parallel (whereas earlier multiplication used to happen sequentially and each step was dependent on the previous multiplication result)

6.7 Generating Shared Multiplication-Triples: Stage I

- Goal: Generating (n, t)-secret-sharing of a random a and a random b
- a and b should remain random for the adversary
- Naive method for generating (n, t)-secret-sharing of a random a
- Each P_i picks a random value and (n, t)-secret-shares it.
- $$a^{(1)} + a^{(2)} + \cdots + a^{(i)} + \cdots + a^{(n)} = a$$
- The final a is defined to be the sum of all the n values
- a is random as long as $t < n$.

6.8 More Efficient Methods for Pre-Processing Phase

- Hyperinvertible matrices: Perfectly-Secure MPC with Linear Communication Complexity
- A more efficient (and direct) method for generating shared multiplication-triples: An Efficient Framework for Unconditionally Secure Multiparty Computation.

7 Secure Multi-Party Computation: Part VII - GMW MPC Protocol [Goldreich, Micali, Wigderson: How to Play any Mental Game or ...]

7.1 Perfectly-Secure MPC Against Passive Corruptions: Summary

- Setup required: pair-wise private and authentic channels.
- Optimal resilience:
- $t < \frac{n}{2}$
- Circuits over a field (for efficient protocols)
- Q. Can these "restrictions" be removed assuming a PPT adversary?

7.2 (n, t) Additive Secret-Sharing Scheme

- Threshold $t < n$. Worst case: $t = n - 1$.
- Intuition: secret s divided into n random shares, such that:
  - Sum of the shares is s
  - Any subset of $n - 1$ shares is independent of s
- $Sh_{Add}(s)$:
  - Select $s_1, \ldots, s_{n-1} \in_r \mathbb{R}$
  - Set $s_n = s - (s_1 + \cdots + s_{n-1})$
  - Output $s_1, \ldots, s_n$
- Rec_{Add}:
- ...

7.3 Linearity of Additive Sharing: Summary

- Any (publicly-known) linear function of (n, t)-additively-shared values can be computed by applying the linear function on the shares.

7.4 1-out-of-2 Oblivious Transfer (OT)

- Michael O. Rabin. "How to exchange secrets by oblivious transfer" Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981
- Sender's security:
  - A corrupt receiver should learn no additional information about m_{1-b}
- Receiver's security:
- ...

7.5 Constructing OT Protocols

- OT protocols can be designed based on varieties of assumptions
  - Factoring problem
  - DDH problem
  - Trapdoor one-way permutations (ex, RSA problem)
  - Noisy communication channel
- Can't we design OT protocols unconditionally (without any assumptions)?
  - Unconditionally-secure OT $\rightarrow$ unconditionally-secure

7.6 Multiplication Over Additive-Secret-Shared Values Using OT

- For simplicity, $n = 2$, $t = 1$ and Boolean ring. $\mathbb{R} = (\{0, 1\}, \text{XOR}, \text{AND})$
- a_1 XOR $a_2 = a$
- b_1 XOR $b_2 = b$
- c_1 XOR $c_2 = c$
- $c = a$ AND b
- a AND $b = (a_1$ XOR $a_2)$ AND $(b_1$ XOR $b_2)$

- $= (a_1$ AND b_1 XOR a_2 AND $b_1)$ XOR $(a_1$ AND b_2 XOR a_2 AND $b_2) \rightarrow$ first component is c_1 and second is c_2.
- 1-out-of-2 OT $0 \rightarrow [] \leftarrow b_2$
- $a_1 \rightarrow [] \rightarrow a_1$ AND b_2
- 1-out-of-2 OT $b_2 \rightarrow [] \leftarrow 0$
- a_2 AND $b_1 \leftarrow [] \rightarrow \dots$
- Problem with this is that b_2 knows the value of b_2 and so it can figure out what a_1 is from the result of a_1 AND b_2.
- A way around this is to pick two random bits $r_0, r_1 \in \{0, 1\}$
- $r_0 \rightarrow [] \leftarrow b_2$
- r_0 XOR $a_1 \rightarrow [] \rightarrow r_0$ XOR $(a_1$ AND $b_2)$
- $b_1 \rightarrow [] \leftarrow r_1$
- r_1 XOR $(a_2$ AND $b_1) \leftarrow [] \rightarrow r_1$ XOR a_2
- $ab = (a_1$ XOR $a_2)$ AND $(b_1$ XOR $b_2)$
- $= a_1$ AND b_1 XOR a_2 AND $b_1)$ XOR $(a_1$ AND b_2 XOR a_2 AND $b_2)$
- $= (a_1b_1$ XOR r_0 XOR r_1 XOR $a_2b_1)$ XOR $(r_1$ XOR r_0 XOR $a_1b_2 \dots)$

7.7 n-Party Multiplication Over Additive-Secret-Shared Values Using OT

- Each pair of distinct parties (P_i, P_j) execute two OTs to generate additive-sharing of a_ib_j and a_jb_i.
- It's possible to extend this idea to a general ring.

8 Secure Multi-Party Computation: Part VIII - Yao's 2PC Protocol [Protocols for secure computations]

8.1 Yao's Protocol: The Setting

- Semi-honest (passive) corruption
  - Corrupt party does not deviate from protocol instructions.
- Extension for the n-party case possible but non-trivial. (DMR's protocol)
- Function f represented as a Boolean Circuit (AND, OR, NOT, XOR)
- Requires constant number of interactions between the parties. This is what makes Yao's protocol different from the previous ones discussed.
- The price to pay is that these are very expensive operations.

8.2 Yao's Garbled Circuit Construction

- GC Constructor (input a, b); GC Evaluator (input c). Operation $Y = (a$ AND $b)$ OR c
- Constructor assigns two keys for the wire a and labels these keys. Does the same for b, c, and output wire.
- Garbling up the gate: each gate is assigned 4 boxes.

8.3 Replacing keyed-box with Cryptographic Mechanisms

- Each wire is assigned a pair of AES keys.
- Each box for the gate is double-encrypted (?).
- There is a small problem: When we give the two keys for each wire to open one of the boxes of the gate, we cannot know whether the output we get is a valid output or whether it is simply a ciphertext because decryption is just a computation. This was avoided when we had boxes and locks because each of the lock either opens or remains locked.
- Question: How is this issue resolved?
- Ans: "Special Correctness" resolves this issue. It enables the evaluator to directly pinpoint the cipher to the correct box and decrypt.
- (G, E, D) has 'special correctness'
- for two distinct keys (k_1, k_2)

8.4 The Last Thing: Obliviously Transferring The Keys to The Evaluator

- ...

8.5 Yao's 2-Party Protocol

- Evaluator is given the labelling of the output wire since that is safe to reveal.
- GC: (C_1, C_2, C_3, C_4) + decoding info: (k_3^0, k_3^1)
- The keys for x: k_0^1.
- OT is used for
- Evaluator will try to use the keys to get the legitimate output by checking all 4 cases.
- It already knows the output since it has the labelling.
- The second interaction will be asking for the output key to be correctly mapped to the right output.
- The first interaction will be in giving the keys to the evaluator.

8.6 Yao's Garbling - Recent Developments

- Point-and-permute [NPS99]: no 'special correctness' needed
- only one ciphertext needs to be decrypted
- Garbled Row Reduction (only 3 boxes/ciphertexts needed for each gate)

NPS99 : 4-to-3 ciphertexts

GNLP15, ZRE15 : 4-to-2 ciphertexts (optimal)

9 Courses Offered by the Speaker on NPTEL

- Foundations of Cryptography
- Secure Computation: Part I
- Secure Computation: Part II (Byzantine corruption, malicious corruption)