Secure Multi-Party Computation

C. Soni[1] A. Satapathy[2]

[1,2]Information and Communication Laboratory
Industrial Technology Research Institute
Taiwan

Under the guidance of Dr. Tzi-cker Chiueh

2017, June 29th

工業技術研究院
Industrial Technology
Research Institute

Outline

1. Introduction
   - What is SMC
   - SMC Models
   - Type of Adversaries
   - SMC Approaches
   - Applications
   - Goals
   - Actions
2. Literature Review
   - SMC Operations
   - Mitigation Mechanisms

Outline

What is SMC

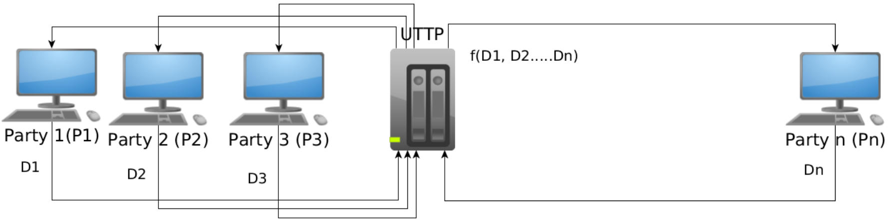
- In Secure Multiparty Computation (SMC), multiple parties carry out computation over their confidential data without any loss of data security/privacy.
- Let multiple parties P_1, P_2.....P_n want to perform computation C_i on their private data. D_1, D_2.....D_n be the data corresponding to P_1, P_2.....P_n.
- D_i should not be accessible to any P_j during computation C_i where $i \neq j$ and $j = 1,2.....n$

Outline

SMC Models

- Generally two model paradigms are popular
  - Ideal model prototype of SMC
  - Real model prototype of SMC
- Ideal model prototype of SMC is also called **Uncorrupted Trusted Third Party** (UTTP). Parties send their data to UTTP to perform computation.
- In real model prototype of SMC, no external party is used. Both parties agree on a protocol to preserve privacy and maintain correctness result.
- Let D_i is private data of P_i, $i = 1,2.....n$. In ideal model, data are send to UTTP directly where as in real model, $f(D_1)$, $f(D_2)$.....$f(D_n)$ exchange between the parties.

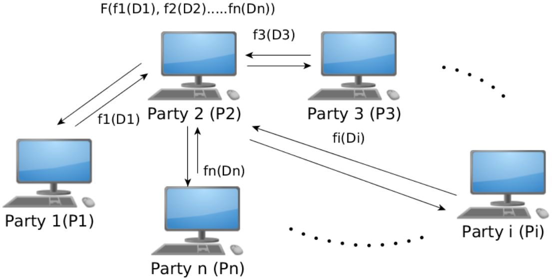
Figure: Ideal model prototype of SMC



Limitation

- UTTP turns corrupt, the privacy will be destroyed.
- It is costly due to the cost of working of the UTTP.

Figure: Real model prototype of SMC



Limitation

- Adversary (a party) can carry out attack in the real model.
- Attack can be passive or active.

Outline

Type of Adversaries

- An adversary can be static or adaptive in nature.
- A **static adversary** is malicious in nature prior to the execution of protocol. An **adaptive adversary** is malicious during the execution of protocol.
- A **semi-honest adversary** follows the protocol but tries to learn other than the output of the computation.
- A **corrupt or malicious adversary** does not follows the protocol and tries to learn other than result.

Outline

1. Introduction
   - What is SMC
   - SMC Models
   - Type of Adversaries
   - SMC Approaches
   - Applications
   - Goals
   - Actions
2. Literature Review
   - SMC Operations
   - Mitigation Mechanisms

SMC Approaches

- Mainly three techniques are used for SMC
  - Randomization methods
  - Cryptographic techniques
  - Anonymization methods
- In **randomization methods**, participants use random numbers for obscuring their input.
- In **cryptographic techniques**, secret input are encrypted at participants side. Computation is performed on encrypted data.
- In **anonymization methods**, the identity of the parties are hide rather than hiding individual parties' data. It is the ideal model where TTP is used.

Outline

Applications

- **Private Information Retrieval:** Client requests the server to provide i^{th} bit/word without the server knowing anything about it. Client is also not aware of bit/ word sequence.
- **Privacy-Preserving Data Mining:** One or multiple parties execute data mining operation on the private database of another party without knowing any details.
- **Privacy-Preserving Database Query:** One party has a string S_i and other party has database D to be searched. Such that other party does not know about S_i and first one does not know about D.
- **Privacy-Preserving Intrusion Detection:** Party B enters the hacker's information and searches A's database, B only gets the comparison results.

Outline

Goals

Let D_i is private data of P_i, $i = 1,2.....n$. Wish to perform a computation $f(D_1, D_2.....D_n) = (Y_1, Y_2.....Y_n)$. Y_i is private output value for P_i.

- **Correct:** Parties correctly compute f.
- **Privacy:** For P_1, P_2.....P_n, each player's input remains private.
- **Output Delivery:** Protocol never end until everyone receives an output.
- **Fairness:** If one party gets the answer, so does every one else.

Outline

1. Introduction
   - What is SMC
   - SMC Models
   - Type of Adversaries
   - SMC Approaches
   - Applications
   - Goals
   - Actions
2. Literature Review
   - SMC Operations
   - Mitigation Mechanisms

Actions

1. Data stored at remote site must be obscured.
2. Data must be obscured during transition.
3. Prevent information access pattern of data at remote site from adversaries.
4. Perform operation on obscured data at remote site.

Note: All the above cases need not to be satisfied for all the SMC operations.

Outline

SMC Operations

Type of operations

- **Private Information Retrieval:** Party i retrieve information from party j without its knowledge, i, j $\in$ N
- **Privacy Preserving Computation:** Parties P_1, P_2.....P_n perform computation C_i over their private data D_1, D_2.....D_n without revealing information to eachother.
- **Privacy Preserving Database Query:** Party i queries s_i to party j having Database D_t s.t. party j doesn't know about s_i.

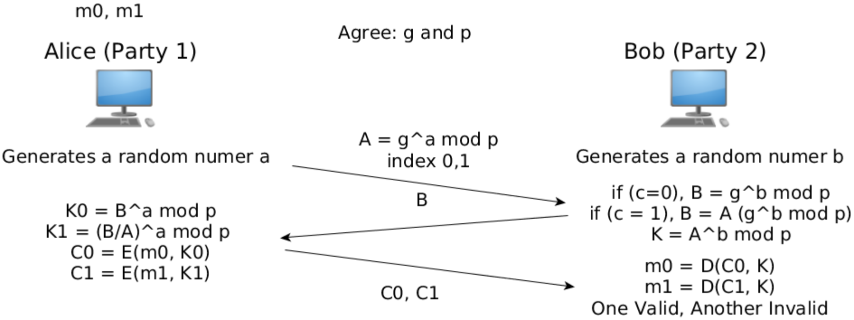
Outline

Mitigation Mechanisms

Private Information Retrieval

Oblivious Transfer (OT): It is a protocol where party A transfer many pieces of information to party B but remain oblivious about which piece of information retrieved by party B.

Figure: OT for private information retrieval



Algorithm

- **step 1:** Alice (Party 1) and Bob (Party 2) agree upon shared input 'g' and 'p'.
- **step 2:** Party 1 generates a random number 'a' and computes $A = g^a \bmod p$. Sends index number of its messages $m_0 = 0$, $m_1 = 1$ with A to Party 2.
- **step 3:** Party 2 generates a random number 'b' and computes $B = g^b \bmod p$ / $A(g^b \bmod p)$ based on its choice 0/1 and sends it to Party 1. Generate $K_s = A^b \bmod p$.
- **step 4:** Party 1 generates $K_0 = B^a \bmod p$ and $K_1 = (B/A)^a \bmod p$. Sends $E_{K_0}(m_0)$ and $E_{k_1}(m_1)$ to Party 2.
- **step 5:** Party 2 decrypts both messages using K_s. One gives valid output but another not.

Example

- **Given:** Alice's $m_0 = 10$, $m_1 = 12$.
- **step 1:** Alice (Party 1) and Bob (Party 2) agree upon shared input g = 3 and p = 77.
- **step 2:** Party 1 generates a = 5 and computes A = 12. Sends index number of its messages $m_0 = 0$, $m_1 = 1$ with A to Party 2.
- **step 3:** Party 2 generates b = 4 and computes B = 4 / 48 based on its choice 0/1 and sends it to Party 1. Generate $K_s = 23$.
- **step 4:** If c = 0 at party 2, party 1 generates $K_0 = 23$ and $K_1 = 0.0041$. Sends $E_{K_0}(10)$ and $E_{k_1}(12)$ to Party 2.
- **step 5:** Party 2 decrypts both messages using K_s. $D_{K_s}(E_{K_0}(10)) = 10$, $D_{K_s}(E_{K_1}(12)) =$ garbage.

Private summation protocol: Parties use random numbers for obscuring their inputs. Perform computation over obscured inputs.

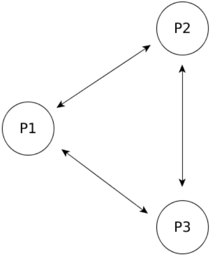
Algorithm

- **Given:** Each party $\mathrm{P_i}$ with input $\mathrm{D_i}$
- **step 1:** Generate random number $\mathrm{r_{i,j}}$ to its neighbour $\mathrm{P_j}$.
- **step 2:** Wait for $\mathrm{r_{j,\,i}}$ from each neighbour $\mathrm{P_j}$.
- **step 3:** Compute $\mathrm{D_i}' = \mathrm{D_i} + \sum_j r_{\mathrm{j,i}} - \sum_j r_{\mathrm{i,j}}$.
- **step 4:** Publish $\mathrm{D_i}'$ to each other.
- **step 5:** Output $= \sum_i D_{\mathrm{i}}'$

Mitigation Mechanisms

Privacy Preserving Computation (Randomization Technique)

Figure: Private summation protocol



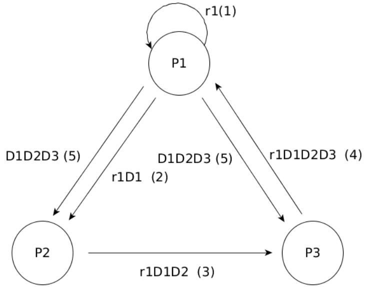
$$D_1' = D_1 - r_{12} - r_{13} + r_{21} + r_{31}$$
$$D_2' = D_2 - r_{21} - r_{23} + r_{12} + r_{32}$$
$$D_3' = D_3 - r_{31} - r_{32} + r_{13} + r_{23}$$
$$\sum_i D_i' = \sum_i D_i$$

Three Party Protocol: Source party uses a random number for obscuring the whole operation where $f(D_1, D_2, D_3) = D_1D_2D_3$.

Algorithm

- **Given:** Parties P_1, P_2 and P_3 have D_1, D_2, D_3 respectively.
- **step 1:** P_1 chooses a random number r_1.
- **step 2:** Computes r_1D_1 and sends it to P_2.
- **step 3:** P_2 computes $r_1D_1D_2$, sends to P_3.
- **step 4:** P_3 computes $r_1D_1D_2D_3$. sends to P_1.
- **step 5:** P1 computes $r_1^{-1}(r_1D_1D_2D_3)$. Sends $D_1D_2D_3$ to P_2 and P_3.

Figure: Three party protocol

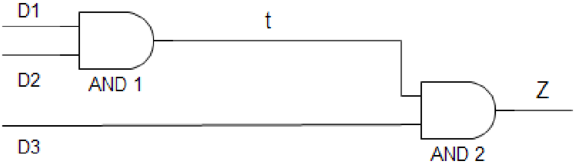


Limitation

- No standardize algorithm for a single operation.

Yao Garbled Circuit: One of the protocol for secure m-party computation. Used to evaluate boolean function.

Figure: Circuit diagram of $D_1 \wedge D_2 \wedge D_3$



Yao Garbled Circuit: It is a 2-party computation protocol. It can be extended to m-party.

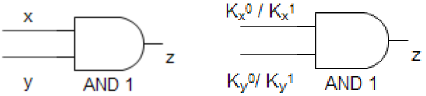
Algorithm (2-party)

- **Given:** Digital Circuit. P_1 is generator and P_2 is evaluator.
- **step 1:** P_1 generates GCT. Encrypts each row of GCT.
- **step 2:** P_1 sends GCT and key associate with its input.
- **step 3:** P_1 and P_2 do oblivious transfer. P2 obtains the key associated with its input.
- **step 4:** P_2 computes circuit output and sends to P_1

Mitigation Mechanisms

Privacy Preserving Computation (Cryptographic Technique)

Figure: Circuit diagram of $x \wedge y$



x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

x'	y'	z'	GCT
${K_x}^0$	${K_y}^0$	0	$E_{{K_x}^0}(E_{{K_y}^0}(0))$
${K_x}^0$	${K_y}^1$	0	$E_{{K_x}^0}(E_{{K_y}^1}(0))$
${K_x}^1$	${K_y}^0$	0	$E_{{K_x}^1}(E_{{K_y}^0}(0))$
${K_x}^1$	${K_y}^1$	1	$E_{{K_x}^1}(E_{{K_y}^1}(1))$

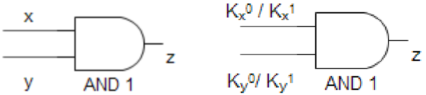
Where ${K_x}^0$, ${K_x}^1$, ${K_y}^0$ and ${K_y}^1$ are random numbers generated by P_1.

Table: Suffled GCT

GCT
$E_{K_x^0}(E_{K_y^0}(0))$
$E_{K_x^1}(E_{K_y^1}(1))$
$E_{K_x^1}(E_{K_y^0}(0))$
$E_{K_x^0}(E_{K_y^1}(0))$

- P_1 suffles the GCT. Send GCT and $K_x{}^a$ to P_2.
- P_2 does oblivious transfer for $K_y{}^b$.
- P_2 decrypts one row successfully. Send the output to P_1.

Figure: Circuit diagram of $x \wedge y$



x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

x'	y'	z'	GCT
3	7	0	$E_3(E_7(0))$
3	9	0	$E_3(E_9(0))$
5	7	0	$E_5(E_7(0))$
5	9	1	$E_5(E_9(1))$

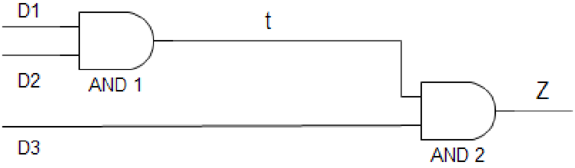
Where 3, 5, 7 and 9 are random numbers generated by P_1.

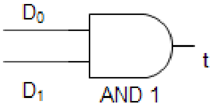
Table: Suffled GCT

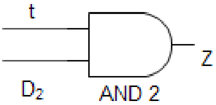
GCT
$E_3(E_7(0))$
$E_5(E_9(1))$
$E_5(E_7(0))$
$E_3(E_9(0))$

- P_1 suffles the GCT. Send GCT and 3 to P_2.
- P_2 does oblivious transfer for ${K_y}^b$. If choice $=0$ then 7 else 9 will be retrieved.
- P_2 decrypts one row successfully. Send the output to P_1.

Figure: Circuit diagram of $D_1 \wedge D_2 \wedge D_3$







For 1st circuit, P_1 is generator and P_2 is evaluator. 2nd circuit, P_2 is generator and P_3 is evaluator.

Algorithm (m-party)

- **Given:** For digital Circuit C_1, P_1 is generator and P_2 is evaluator. Digital Circuit C_2, P_2 is generator and P_3 is evaluator.
- **step 1:** P_1 generates GCT_1. Encrypts each row of GCT_1.
- **step 2:** P_1 sends GCT_1 and key associate with its input.
- **step 3:** P_1 and P_2 do oblivious transfer. P2 obtains the key associated with its input.
- **step 4:** P_2 computes circuit output and one will be valid.
- **step 5:** Repeat step 1 to step 4 for circuit C_2.
- **step 6:** P_3 sends final output to P_1 and P_2.

Database Encryption Scheme: Databases are encrypted to prevent any information leakage.

- To reduce computational time, only sensitive columns of database tables are encrypted.
- Probabilistic encryption is used to encrypt repeated pattern.
- Encryption and decryption keys are known to client.

Mitigation Mechanisms

Privacy Preserving Database Query

Table: Ailment and Patient (original)

Name	Disease
Alice	AIDS
Bob	Flu
Chen	AIDS
Dana	Diabetes

Name	City	Gender
Alice	Seattle	Female
Bob	Madison	Male
Chen	Palo Alto	Male
Dana	New York	Female

Table: Ailment and Patient (encrypted)

Name	Disease
Alice	!@#$xyz
Bob	@%∧abc
Chen	*&#pqr
Dana	(p#z∗94

Name	City	Gender
Alice	Seattle	2xU%b
Bob	Madison	Ry!<4&
Chen	Palo Alto	wl-]%5
Dana	New York	3xt*&i

Privacy preserving database query: P_i performs 'x' operation on database at P_j, where sensitive columns of the databases are encrypted.

Query

```
select Name, Disease, City from Patient join Ailment
on Name where Disease = 'AIDS'
```

Encrypted query

```
select Name, Disease, City from Patient join Ailment
on Name where Disease = 'xxxxxxxxxxxxxxx'
```

Table: Output

Alice	!@#$xyz	Seattle
Chen	*&#pqr	Palo Alto

- Hidding data is not enough, Prevent adversary (server) to understand the query pattern.
- If adversary knows Chen has AIDS externally, then Alice also AIDS.
- Use **Oblivious RAM** to hide query and memory access pattern.
- ORAM executes query and hides read/write memory access by read and write memory access.

To be Continued ...