Building programmable secure computing systems

Dan Bogdanov, PhD
Sharemind team lead
dan@cyber.ee
Primitive operations in secure computing
Secure computing

When a standard computer encrypts data, it must be decrypted before analysis.

Secure computing systems can analyze data without removing the encryption.
Multi-party computing

Parties

\( P_1 \)
\( P_2 \)
... 
\( P_k \)

\( x_1 \)  \( y_1 \)
\( x_k \)  \( y_k \)

secure multi-party computing

Parties

\( P_1 \)
\( P_2 \)
... 
\( P_k \)
Expanded definition

Input parties

Computing parties

Result parties

Step 1: upload and storage of inputs

Step 2: secure computing

Step 3: publishing of results
Property-preserving crypto

- **Analogy**: symmetric crypto that preserves a relation on inputs (e.g., order, equality).

- **Pros**:
  - Low performance overhead.
  - Fits well into existing systems.

- **Cons**:
  - Only allows a few operations (e.g., only equality comparison or ordering).
  - Multi-user systems are a challenge, but can be done with proxy re-encryption.
Fully homomorphic crypto

• **Analogy**: asymmetric crypto that allows addition and multiplication of ciphertexts.

• **Pros**: 
  - Fits well into existing systems.

• **Cons**: 
  - High performance overhead.
  - Multi-user systems are a challenge, but can be done with proxy re-encryption.
Garbled circuits

- **Analogy**: cryptographic versions of electrical circuits.

- **Pros**:
  - Flexible programming model.

- **Cons**:
  - Medium performance overhead.
  - Fixed number of parties (can be solved by combining with other techniques).
Secret-sharing systems

- **Analogy**: give a number of people a random piece of each secret value and let them collaborate to compute results.

- **Pros**: 
  - Low-to-medium performance overhead.
  - Flexible programming model.

- **Cons**: 
  - Distributed deployments do not fit into all existing systems.
Protection domains kinds

- A protection domain kind (PDK) is a set of data representations, algorithms and protocols for storing and computing on protected data.

- Examples:
  - SMC based on secret sharing,
  - SMC based on garbled circuits,
  - (fully) homomorphic encryption,
  - trusted hardware (e.g., Intel SGX).
Protection domains

• A protection domain (PD) is a set of data that is protected with the same resources and for which there is a well-defined set of algorithms and protocols for computing on that data while keeping the protection.

• Examples:
  • data held by a fixed group of servers performing secure multi-party computation,
  • data encrypted under a fixed key of a homomorphomorphism encryption scheme.
1. Pick random number $a_1 = 57$
2. Pick random number $a_2 = 13$
3. Find $a_3 = 25 - 57 - 13 \equiv 55 \mod 100$
4. Send $a_k$ to Server $k$, $(k \in \{1, 2, 3\})$

Student A
Score: 25

1. Pick random number $b_1 = 44$
2. Pick random number $b_2 = 57$
3. Find $b_3 = 33 - 44 - 57 \equiv 32 \mod 100$
4. Send $b_k$ to Server $k$, $(k \in \{1, 2, 3\})$

Student B
Score: 33

---

Server 1
- $a_1 = 57$
- $b_1 = 44$
- $c_1 = a_1 + b_1 = 101 \equiv 1 \mod 100$

Server 2
- $a_2 = 13$
- $b_2 = 57$
- $c_2 = a_2 + b_2 = 70 \equiv 70 \mod 100$

Server 3
- $a_3 = 55$
- $b_2 = 32$
- $c_3 = a_3 + b_3 = 87 \equiv 87 \mod 100$

C calculates $c = 1 + 70 + 87 = 158 \equiv 58 \mod 100$

C learns that the sum of A’s and B’s score is 58 without learning the scores of either student.
Getting more operations

• (continued example)
• Addition derives from the homomorphic property of additive secret sharing.
• Further operations require network communication (resharing, prefix-OR).
• The challenge is finding non-trivial ways to simplify the more complex protocols to make them efficient and keep them composable.
Performance profiles of secure computing
Unintuitive performance

- Floating point multiplication on secret sharing
From primitive operations to algorithms
Two-level coding model

- Application using SMC
  - Secure primitive operations
  - Privacy-preserving algorithms
  - Application logic

- private outputs from private inputs,
- have privacy proofs,
- remain private under sequential or parallel composition,
- optimized to have a low resource footprint.

- publish selected results to make system useful,
- do not leak private inputs or show leakage as acceptable,
- compositions of secure primitive operations,
- optimize for running time.
Public and private values

// Dot product function
secret uint dotProduct (secret uint[[1]] v1,
                       secret uint[[1]] v2) {

    secret uint result = 0;
    // Will not try to hide the size at runtime
    for (uint i = 0; i < size (v1); i++) {
        result[i] = v1[i] * v2[i];
    }

    return result;
}
information and tax secrets in social studies.
sufficient and legally acceptable way of protecting personal
enforced mechanisms to minimize it. We implement
ination about the private inputs.
to achieve, as all practically useful outputs contain informa-
computations or in the outputs. However, this is impossible
information about the private inputs is revealed during the
In our setting, data owners and analysts agree on a study
2.4 Limiting the leakage of private inputs
protection domain kind require more complex protocols, as
to get the shares of
and
y
covering the secret. If each computing party has shares
as an individual share
and calculates the final share
x
multi-party computation on
organizations with a legal responsibility to protect privacy.
processing personal data when the computing parties are
but try to learn private values from its state) is sufficient for
tees often come at a cost of computational power. Security
withstand malicious tampering. Stronger security guaran-
remain private only when there is an honest majority, or
parties collude to reveal private inputs. PDKs also differ in
design flaws. For SMC, the main threat is that computing
threats for a hardware-assisted PDK include backdoors and
SUBMISSION TO IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING 3
Computing parties can process the shares without re-
The threat models for different PDKs vary. For example,
Somepnotation
—our statistical tool will only publish results
—privacy. This makes the computation protocol cryptograph-
secure. We require the PDK to be universally composable
secure and universally composable itself [38]. Hence,
universally composable secure operations is cryptographi-
straight line program. A straight line program consisting of
algorithm for computing
1
2
m
(Algorithms 1, 2, 4, 5, 6, 9) declassify the size of the
filtered subgroup, which is assumed to be public or is a
rithms (Algorithms 1, 2, 4, 5, 6, 9) declassify the size of the
and require separate security proofs. Most of these algo-
paper (Algorithms 3, 7, 8, 10).
First, our algorithms process all private values using
Functionality | Notation
---|---
Protected storage of a private value $x$ (signed integer, floating point, boolean) | $[x]$ 
Conversion to the private form | $[x] \leftarrow x$ 
Support for value vectors and matrices | $[\vec{x}]$ and $[M]$ 
Privacy-preserving binary operations (signed integer, floating point, boolean) | $[z] \leftarrow [x] \circ [y]$ 
Privacy-preserving lookup | $[y] \leftarrow [\vec{x}]_{[i]}$ 
Privacy-preserving functions | $[y] \leftarrow f([x])$ 
Declassify value to computing parties | declassify($[x]$) 
Publish value to result parties | publish($[x]$) 
Private shuffling, linking and sorting | —

Some notation
Goal 1: Cryptographic security. During the evaluation of

\[
([y_1], \ldots, [y_k]) \leftarrow f([x_1], \ldots, [x_m]),
\]

a computing party \(CP\) cannot learn any private input \(x_j\) where \(j \in \{1, \ldots, m\}\), output \(y_\ell\) where \(\ell \in \{1, \ldots, k\}\) or intermediate value computed by \(f\) unless the value is published to the result parties using the publish function. We also want to prevent leaking private values through changes in the running time of the algorithm.
Goal 2: Source privacy. An algorithm for computing $f(x_1, \ldots, x_m)$ is source-private if all outputs and all intermediate values do not depend on the order of inputs. If an algorithm for computing $f(x_1, \ldots, x_m)$ is cryptographically secure, it is sufficient to prove that the output distributions of $f(x_1, \ldots, x_m)$ and $f(x_{\pi(1)}, \ldots, x_{\pi(m)})$ coincide for all permutations of inputs $\pi$. 
Goal 3: Output privacy. An algorithm is output-private, if its published results do not leak the private inputs. Output privacy cannot be absolute, as this way we learn nothing useful from the inputs. In practice, the amount of allowable leakage is strongly application-dependent. In some settings, input parties will perform a privacy impact assessment of the publishable outputs and decide what can be published. If one is looking for more automatic mechanisms, one should consider output randomization techniques such as differential privacy.
Frequent itemset mining

- Private data representations are the key toward designing privacy-preserving algorithms.
FIM - calculating support

- The data representation allows for very efficient calculation of item supports.

<table>
<thead>
<tr>
<th></th>
<th>rendang</th>
<th>nasi lemak</th>
<th>lontong</th>
<th>chicken satay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_5$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_6$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_7$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$$\sum = 5$$
FIM - combined support

- Checking the joint support of a pair of items simply requires an additional multiplication

<table>
<thead>
<tr>
<th></th>
<th>rendang</th>
<th>nasi lemak</th>
<th>lontong</th>
<th>chicken satay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_5$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_6$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_7$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
&\text{nasi lemak} & \text{chicken satay} & \text{nasi lemak & chicken satay} \\
&1 \times 1 = 1 & 1 \times 0 = 0 & 1 \times 0 = 0 \\
&1 \times 0 = 0 & 0 \times 1 = 0 & 0 \times 1 = 0 \\
&0 \times 1 = 0 & 1 \times 1 = 1 & 1 \times 1 = 1 \\
&1 \times 1 = 1 & 0 \times 0 = 0 & 0 \times 0 = 0 \\
&\sum = 3 & & \\
\end{align*}
\]
FIM - depth-first

• Depth-first search would be intuitive for pruning.

\[
\text{rendang, nasi lemak, lontong, chicken satay}
\]

\[
\text{rendang, nasi lemak, lontong}
\]

\[
\text{rendang, nasi lemak, lontong, chicken satay}
\]

\[
\text{nasi lemak, lontong, chicken satay}
\]

\[
\text{nasi lemak, lontong, chicken satay}
\]

\[
\text{rendang, nasi lemak, lontong, chicken satay}
\]

\[
\text{rendang, nasi lemak, lontong, chicken satay}
\]
FIM - breadth-first

• However, breadth-first search can be done in parallel.

{ rendang }  { nasi lemak }  { lontong }  { chicken satay }

{ rendang, nasi lemak } { rendang, lontong } { rendang, chicken satay } { nasi lemak, lontong } { nasi lemak, chicken satay } { lontong, chicken satay }

{ rendang, nasi lemak, lontong } { rendang, nasi lemak, chicken satay } { rendang, lontong, chicken satay } { nasi lemak, lontong, chicken satay }

{ rendang, nasi lemak, lontong, chicken satay }
• **Challenge**: exploring all possible itemsets leads is slow due to combinatorial explosion.

• Pruning the search tree requires us to declassify itemset supports during computation (leak?).

• **Solution**: consider that the algorithm will publish all frequent itemsets, as that is its intended goal.

• We will compare support to the threshold privately, only declassifying the result bit.

• We will prune the search tree based on that bit.

• Not a leak - if the itemset is frequent, we would have learned it from the outputs anyway.
FIM - performance

- Performance is best with a hybrid approach
References

Easy-to-read survey of programmable SMC


(NB! The eprint version has lots of references to more results!)

Definition of protection domains and their use in programming

References (protocols)

From addition to integer division on secret-shared data


Optimizing protocols running on secret-shared data

References (algorithms)

Privacy-preserving optimization algorithms


Privacy-preserving frequent itemset mining algorithms

References (algorithms)

Privacy-preserving statistical algorithms
http://hdl.handle.net/10062/45343

Privacy-preserving database algorithms
http://hdl.handle.net/10062/50510