Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) aims to allow a set of parties to jointly evaluate a function on their inputs while keeping their inputs private. The origins of this work can be traced back to Yao’s Millionaire’s problem:
• A group of millionaires want to know who has the most money without revealing their individual wealth.

We uniformly sample one of the outputs so one output

\[ f(x, y) = \{(s_i, x - y), s_i \sim Z\} \]

We formally prove security in the computational model using the simulation based technique. To do this we establish a simulation between the real world, where the protocol plays out, and an ideal world, which is taken as the definition of security. This formalises the intuition that a protocol is secure if it can be simulated in an ideal environment in which there is no data leakage by definition.

The Real World

\[ \text{Party One} \rightarrow \text{messages} \rightarrow \text{Party Two} \]

The Ideal World

\[ \text{input} \rightarrow \text{output} \rightarrow \text{Trust} \rightarrow \text{Party} \rightarrow \text{output} \]

The real world is where the protocols runs and the ideal world has no data leakage by definition.

Let \( \pi \) be a two party protocol with inputs \((x, y)\) and with security parameter \( n \).
• The real view of the \( i^{th} \) party is denoted by \( v_i(x, y, n) = (w_i(x, y), m_1, \ldots, m_n) \)
where \( w \) is the input. \( s_i \) accumulates random values generated by the party in the execution, and the \( m_j \) are the messages received by the party.
• Denote the joint output as
\[ \text{out}(x, y, n) = (\text{out}_1(x, y, n), \text{out}_2(x, y, n)) \]

Definition of Security

A protocol, \( \pi \), is said to securely compute \( f \) in the presence of a semi-honest adversary (an honest but curious adversary) if there exist probabilistic polynomial time algorithms (simulators) \( S_1, S_2 \) such that
\[ S_1(1^n, x, f(x, y)) \equiv (v_{S_2}(x, y, n), out_{S_2}(x, y, n)) \]
\[ S_2(1^n, y, f(x, y)) \equiv (v_{S_1}(x, y, n), out_{S_1}(x, y, n)) \]

Here \( X \equiv Y \) means \( X \) and \( Y \) are computationally indistinguishable — that is no polynomial time distinguisher can distinguish the distributions with greater than negligible probability. So security is expressed by showing equivalence between the real and ideal worlds.

Formalising in Isabelle

We are able to define the real and simulated views as probabilistic programs and thus create the required distributions to show security. We do this in Isabelle using theory from Andreas Lochbihler’s CryptHOL[1]. In particular we make use of the sub probability mass functions (spmf) he introduces. For example, the functionality for the secure multiplication protocol is defined in Isabelle as:

\[ f(x, y) = \{(s_i, x - y), s_i \sim Z\} \]

In order to formalise proofs of security we first provide a definition of computational indistinguishability, following the definition of Lindell in [2].

Two lemmas we use in our proofs are:
• \( X = Y \Rightarrow X \equiv Y \)
• \( X \equiv Y, Y \equiv Z \Rightarrow X \equiv Z \)

These two lemmas help us to formalise the proofs of security. We show either:
• information theoretic security if distributions are equal.
• a reduction to a known hard problem, both of which satisfy the definitions of security we require.

Information Theoretic Security

If the two distributions are equal then we are able to show information theoretic security. Our proof method is as follows:
• Define intermediate probabilistic programs \( f_i \) that will take us from one distribution to the other.
• Show equality between successive intermediate \( I_i \).
• Then show equality between the two distributions.
\[ D_{\text{dist}, i} = I_1 = I_2 = \cdots = I_n = D_{\text{dist}} \]

To show equality between \( I_i \) and \( I_{i+1} \), we use our own lemmas and ones from CryptHOL. The main ‘jumps’ are made by using one time pad lemmas that we prove — these are usually left to the cryptographers intuition. For example, a lemma showing that the distributions \((y + b) \mod q\) where \( b \sim Z_q \) and \( b' \sim Z_q \) are equal would take the form
\[ \text{map}_{\text{spmf}}(\lambda b \cdot (y + b) \mod q)(\text{sample}_{\text{uniform}} q) = \text{sample}_{\text{uniform}} q \]
in Isabelle.

Reducions

The proof method runs as follows:
• Assume \( D \) can distinguish the two distributions.
• Using \( D \), construct an adversary \( A(D) \) that breaks a known hard problem.
• Show computational indistinguishability using the assumption on the known hard problem.

Formally we show the advantage of \( D \) in distinguishing the two original distributions is the same as the advantage \( A(D) \) has against the known hard problem. That is we show:
\[ \text{adv}_{\text{dist}}(D) = \text{adv}_{\text{hard}}(A(D)) \]

Formalised Proofs

Examples of the simulation based proofs we have formalised include:
• a Secure Multiplication Protocol — each party outputs a share of the multiplication.
• an information theoretic OT which uses a trusted initialiser.
• the Noar-Pinkas OT.
• a protocol based on GMW to securely compute an AND gate.

Conclusion and Future Work

We have shown a general approach for capturing simulation based proofs and have formalised some proofs.
• This is only a starting point for the development of theory that will hopefully further formalise SMPC.
• Current work includes formalising the proof of security for the GMW protocol — a protocol that allows for the evaluation, between two parties, of any function that can be represented as a boolean circuit.

References