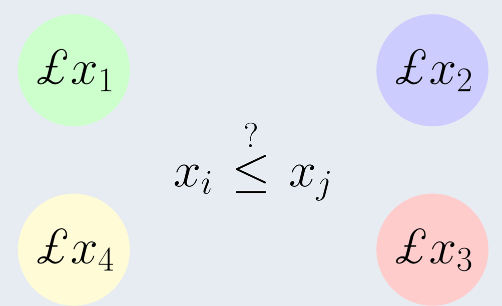
Formalising Secure Multi-Party Computation in Isabelle

David Butler
The Alan Turing Institute and The University of Edinburgh

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.



The four party millionaire's problem, who is the most wealthy?

Formally the goal of SMPC is to compute a map from pairs of inputs to pairs of outputs, with each party receiving it's corresponding output such that their input remains private. This map is called the protocol's *functionality* as it represents the specification of what the protocol should achieve — it may be deterministic or probabilistic. Here we will consider the two party case for simplicity.

$$f : input_1 \times input_2 \longrightarrow output_1 \times output_2$$

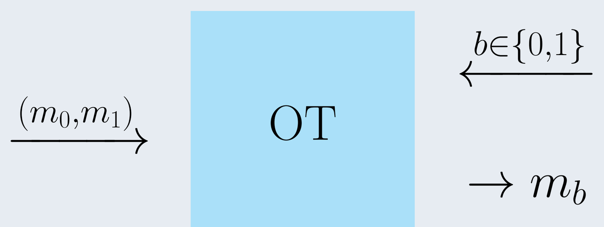
For example, a secure multiplication protocol which allows each party to output an additive share of the multiplication has a functionality of the form

$$f(x, y) = (s_1, x.y - s_1),\ s_1 \overset{\$}{\leftarrow} \mathbb{Z}_q.$$

We uniformly sample one of the outputs so one output alone does not reveal the value of $x.y$.

Oblivious Transfer

Oblivious transfer (OT) is a fundamental primitive used in many SMPC protocols. The illustration demonstrates the functionality of Oblivious Transfer.

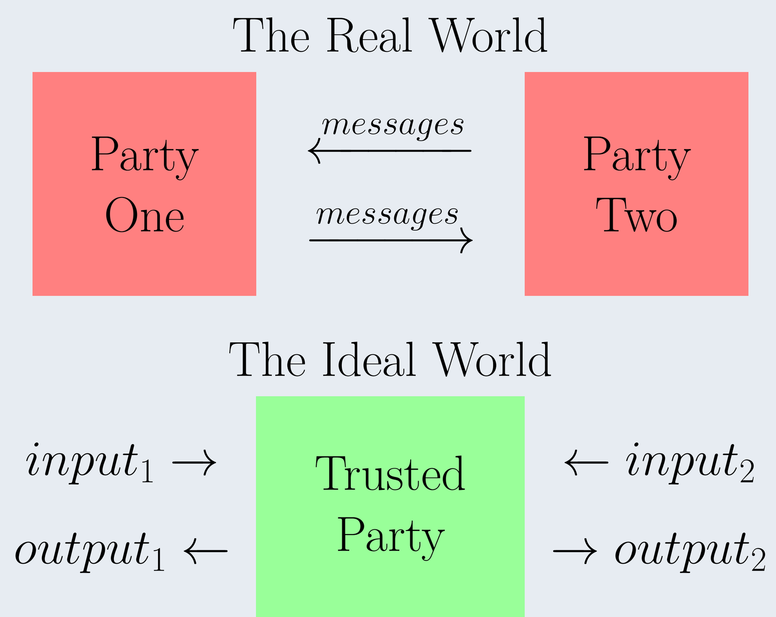


The functionality for Oblivious Transfer.

One party holds two messages, (m_0, m_1), and the other a choice bit $b \in \{0, 1\}$. The second party receives m_b as output while the first party receives nothing. Without any security requirements the problem is trivial, but to compute this functionality without the first party learning b and the second party learning m_{1-b} is difficult.

Defining Simulation Based Security

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

Let π 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

$$v_i^\pi(x, y, n) = (w, r^k, m_1^i, ..., m_t^i)$$

  where w is the input, r^k accumulates random values generated by the party in the execution, and the m_j^i are the messages received by the party.
- Denote the joint output as

$$out^\pi(x, y, n) = (out_1^\pi(x, y, n), out_2^\pi(x, y, n)).$$

Definition of Security

A protocol, π, 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

$$\{S_1(1^n, x, f_1(x, y)), f(x, y)\} \overset{c}{\equiv} \{v_1^\pi(x, y, n), out^\pi(x, y, n)\}$$

$$\{S_2(1^n, y, f_2(x, y)), f(x, y)\} \overset{c}{\equiv} \{v_2^\pi(x, y, n), out^\pi(x, y, n)\}.$$

Here $X \overset{c}{\equiv} Y$ 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 (spmfs) he introduces. For example, the functionality for the secure multiplication protocol is defined in Isabelle as:

$$\begin{aligned} &f\ x\ y = do\ \{ \\ &\quad s_1 \leftarrow sample\_uniform\ q; \\ &\quad return_{spmf}(s_1, x.y - s_1)\} \end{aligned}$$

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

Two lemmas we use in our proofs are:

- $X = Y \Longrightarrow X \overset{c}{\equiv} Y$
- $[X \overset{c}{\equiv} Y;\ Y \overset{c}{\equiv} Z] \Longrightarrow X \overset{c}{\equiv} Z.$

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 (I_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.

$$Dist_1 = I_1 = I_2 = ... = I_n = Dist_2$$

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 a the distributions $(y + b)\ mod\ q$ where $b \overset{\$}{\leftarrow} \mathbb{Z}_q$ and $b' \overset{\$}{\leftarrow} \mathbb{Z}_q$ are equal would take the form

$$\begin{aligned} &map_{spmf}\ (\lambda b.\ (y + b)\ mod\ q)\ (sample\_uniform\ q) \\ &\qquad = sample\_uniform\ q \end{aligned}$$

in Isabelle.

Reductions

The proof method runs as follows:

- Assume $\mathcal{D}$ can distinguish the two distributions.
- Using $\mathcal{D}$, construct an adversary $(A(\mathcal{D}))$ that *breaks* a known hard problem.
- Show computational indistinguishability using the assumption on the known hard problem.

Formally we show the advantage of $\mathcal{D}$ in distinguishing the two original distributions is the same as the advantage $A(\mathcal{D})$ has against the known hard problem. That is we show:

$$adv_{dist}\ (\mathcal{D}) = adv_{hard}\ (A(\mathcal{D}))$$

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

[1] Andreas Lochbihler. Probabilistic Functions and Cryptographic Oracles in Higher Order Logic. In *ESOP*, volume 9632 of *Lecture Notes in Computer Science*, pages 503–531. Springer, 2016.

[2] Yehuda Lindell. How to Simulate It A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017.