How to Simulate it in Isabelle: Towards Formal Proof for Secure Multi-Party Computation

David Butler, David Aspinall, Adrià Gascón

The Alan Turing Institute
University of Edinburgh

Outline

- Motivation for formal methods in Cryptography.
- Introduce Secure Multi-Party Computation.
- How is security defined in SMPC?
- CryptHOL, the framework we use in Isabelle.
- Basic proof techniques.
- Toy example to demonstrate how formalisation works.

Do we have a problem with cryptographic proof?

Do we have a problem with cryptographic proof?

> *"... Yes we do. The problem is that as a community, we generate more proofs than we carefully verify."*
>
> S. Halevi. 2005

Do we have a problem with cryptographic proof?

"... Yes we do. The problem is that as a community, we generate more proofs than we carefully verify."

S. Halevi. 2005

"In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor".

Bellare and Rogaway. 2004

Do we have a problem with cryptographic proof?

"... Yes we do. The problem is that as a community, we generate more proofs than we carefully verify."

S. Halevi. 2005

"In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor".

Bellare and Rogaway. 2004

"Security proof for even simple cryptographic systems are dangerous and ugly beasts. Luckily, they are only rarely seen: they are usually safely kept in the confines of "future full-versions" of papers, or only appear in cartoon-ish form, generically labelled as ... "proof sketch""

Bristol Crypto Group. 2017

Contributions

Contributions

Starting from Lindell's tutorial 'How to Simulate it: A tutorial on the simulation proof technique.'

- Demonstrated how the simulation-based proof method can be formalised.
- Defined computational indistinguishability - up to polynomial time distinguishers.

Protocols formalised:

- Secure multiplication protocol.
- Noar Pinkas Oblivious Transfer.
- Protocol that securely computes an AND gate.

Secure Multi-Party Computation (SMPC)

Secure Multi-Party Computation (SMPC)

- Parties jointly compute a function on their private inputs.

  Goal **:** Their inputs remain private.

Secure Multi-Party Computation (SMPC)

- Parties jointly compute a function on their private inputs.

  Goal **:** Their inputs remain private.
- First introduced in *The Millionaire's Problem* - Yao, 1982.

$$\pounds x_1 \qquad x_i \overset{?}{<} x_j \qquad \pounds x_2$$

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

Secure Multi-Party Computation (SMPC)

- Parties jointly compute a function on their private inputs.

  Goal **:** Their inputs remain private.

- First introduced in *The Millionaire's Problem* - Yao, 1982.

$$\pounds x_1 \qquad x_i \overset{?}{<} x_j \qquad \pounds x_2$$

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

- Recommended as an *"emerging approach that enhances privacy protections"* in the report of the US commission on Evidence-Based Policymaking, 2017.

SMPC - A brief history

- Initially cryptographers studied MPC, but did not have efficient solutions.

SMPC - A brief history

- Initially cryptographers studied MPC, but did not have efficient solutions.
- Goldreich, Micali and Widgerson (GMW) showed how to compute any boolean circuit securely.

SMPC - A brief history

- Initially cryptographers studied MPC, but did not have efficient solutions.
- Goldreich, Micali and Widgerson (GMW) showed how to compute any boolean circuit securely.
- Efficient implementations with applications in mind were developed in 2000s.
- SMPC began to become more widely implemented.
  - First real life deployment was in 2008 at a Danish sugar beat auction.

SMPC - A brief history

- Initially cryptographers studied MPC, but did not have efficient solutions.
- Goldreich, Micali and Widgerson (GMW) showed how to compute any boolean circuit securely.
- Efficient implementations with applications in mind were developed in 2000s.
- SMPC began to become more widely implemented.
  - First real life deployment was in 2008 at a Danish sugar beat auction.
- Could be considered as a counterpart to Homomorphic encryption.

Our set up and security model

Our set up and security model

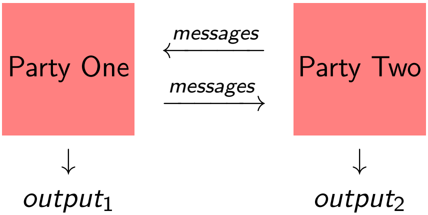
***What are we considering*?**

- Two party setting.
- Semi honest adversary model - honest but curious adversaries.
  - Adversaries follow the protocol specification exactly.
  - They attempt to learn additional information by analysing the transcript of messages received during the execution.

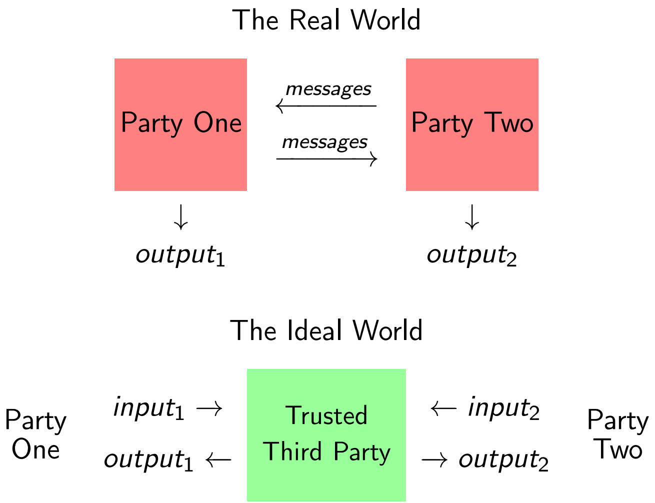
Simulation based security: intuition

Simulation based security: intuition

The Real World

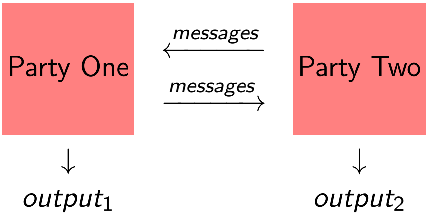


Simulation based security: intuition



Simulation based security: intuition

The Real World



The Ideal World

- Show that the two worlds are *equivalent* or *indistinguishable*.

Simulation based security

Simulation based security

- Assume party one is corrupt.

Simulation based security

- Assume party one is corrupt.

The Real World

- Create the *real view* of the party one - the transcript it sees.

Simulation based security

- Assume party one is corrupt.

The Real World

- Create the *real view* of the party one - the transcript it sees.

The Ideal World

- Construct *simulator*, S_1, which only takes as input the input and output of party one. Outputs the *simulated view of the party*.

Simulation based security

- Assume party one is corrupt.

The Real World

- Create the *real view* of the party one - the transcript it sees.

The Ideal World

- Construct *simulator*, S_1, which only takes as input the input and output of party one. Outputs the *simulated view of the party*.

Show the two output distributions are *computationally indistinguishable*.

$$\{Real_{View1}(input_1, input_2)\} \stackrel{c}{\equiv} \{S_1(input_1, out_1)\}$$

But what about Game-based proof?

But what about Game-based proof?

- A game defines the goal of an attacker explicitly.
  - Prove that no adversary can achieve the goal.

But what about Game-based proof?

- A game defines the goal of an attacker explicitly.
  - Prove that no adversary can achieve the goal.
- Frameworks have been developed to formalise game-based proofs - EasyCrypt, FCF.

But what about Game-based proof?

- A game defines the goal of an attacker explicitly.
  - Prove that no adversary can achieve the goal.
- Frameworks have been developed to formalise game-based proofs - EasyCrypt, FCF.
- Cryptographers view game-based and simulation-based proofs as distinct.

But what about Game-based proof?

- A game defines the goal of an attacker explicitly.
  - Prove that no adversary can achieve the goal.
- Frameworks have been developed to formalise game-based proofs - EasyCrypt, FCF.
- Cryptographers view game-based and simulation-based proofs as distinct.
- We use a game-based framework, CryptHOL, to do simulation-based proofs.

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

- Provides probabilistic programming framework.

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

- Provides probabilistic programming framework.
  - Real and simulated views modelled as probabilistic programs.

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

- Provides probabilistic programming framework.
  - Real and simulated views modelled as probabilistic programs.
- Defines theory on sub probability mass functions (spmfs).

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

- Provides probabilistic programming framework.
  - Real and simulated views modelled as probabilistic programs.
- Defines theory on sub probability mass functions (spmfs).
- Can reason about probabilistic programs.

Formalising security in Isabelle: CryptHOL

CryptHOL (Lochbilher, 2016)

- Provides probabilistic programming framework.
  - Real and simulated views modelled as probabilistic programs.
- Defines theory on sub probability mass functions (spmfs).
- Can reason about probabilistic programs.
- Designed with game-based proofs in mind.

CryptHOL: some key features

CryptHOL: some key features

- Many protocols require uniform sampling from sets.
  - $\mathit{uniform} : \alpha\ \mathit{set} \Rightarrow\ \alpha\ \mathit{spmf}$
  - $\mathit{sample}_{\mathit{uniform}}\ n \equiv \mathit{uniform}\ \{..< n\}$
  - $\mathit{coin}_{\mathit{spmf}} \equiv \mathit{uniform}\ \{\mathit{True}, \mathit{False}\}$

CryptHOL: some key features

- Many protocols require uniform sampling from sets.
  - $\mathit{uniform} : \alpha\ \mathit{set} \Rightarrow\ \alpha\ \mathit{spmf}$
  - $\mathit{sample}_{\mathit{uniform}}\ n \equiv \mathit{uniform}\ \{..< n\}$
  - $\mathit{coin}_{\mathit{spmf}} \equiv \mathit{uniform}\ \{\mathit{True}, \mathit{False}\}$

- Much of our reasoning comes from the functorial structure $\mathit{map}_{\mathit{spmf}}$.
  - $\mathit{map}_{\mathit{spmf}} : (\alpha \Rightarrow \beta) \Rightarrow \alpha\ \mathit{spmf}\ \Rightarrow \beta\ \mathit{spmf}$
  - $\mathit{map}_{\mathit{spmf}}\ f\ p = \mathit{bind}_{\mathit{spmf}}\ p\ (\lambda x.\ \mathit{return}_{\mathit{spmf}}\ (f\ x))$

Formalising security in Isabelle: Proof Method I.

Formalising security in Isabelle: Proof Method I.

Information theoretic security:

Formalising security in Isabelle: Proof Method I.

Information theoretic security:

- Show real and simulated views are equal.

Formalising security in Isabelle: Proof Method I.

Information theoretic security:

- Show real and simulated views are equal.
- Use CryptHOL to manipulate probabilistic programs so we can reason about distributions.

Formalising security in Isabelle: Proof Method I.

Information theoretic security:

- Show real and simulated views are equal.
- Use CryptHOL to manipulate probabilistic programs so we can reason about distributions.
- Then we prove lemmas like:

$$map_{spmf}\ (\lambda b.\ (y + b)\ mod\ q)\ (sample_{uniform}\ q) \\ = sample_{uniform}\ q$$

Formalising security in Isabelle: Proof Method I.

Information theoretic security:

- Show real and simulated views are equal.
- Use CryptHOL to manipulate probabilistic programs so we can reason about distributions.
- Then we prove lemmas like:

$$map_{spmf}\ (\lambda b.\ (y + b)\ mod\ q)\ (sample_{uniform}\ q) = sample_{uniform}\ q$$

To show equality between the views.

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

- Assume the problem is hard.
    - In the Noar Pinkas OT this is the Diffie-Hellman assumption.

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

- Assume the problem is hard.
  - In the Noar Pinkas OT this is the Diffie-Hellman assumption.
- Construct the real and simulated views.

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

- Assume the problem is hard.
  - In the Noar Pinkas OT this is the Diffie-Hellman assumption.
- Construct the real and simulated views.
- Assume there exists a distinguisher, D, that can distinguish the two views.

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

- Assume the problem is hard.
  - In the Noar Pinkas OT this is the Diffie-Hellman assumption.
- Construct the real and simulated views.
- Assume there exists a distinguisher, D, that can distinguish the two views.
- Use D to construct an adversary, $\mathcal{A}(D)$, that *breaks* the known hard problem.

Formalising security in Isabelle: Proof Method II.

Reduction to a hard problem:

- Assume the problem is hard.
  - In the Noar Pinkas OT this is the Diffie-Hellman assumption.
- Construct the real and simulated views.
- Assume there exists a distinguisher, D, that can distinguish the two views.
- Use D to construct an adversary, $\mathcal{A}(D)$, that *breaks* the known hard problem.
- This is a contradiction.

Refresher

What have we seen so far?

- Why formal methods are useful to cryptography.
- What SMPC is and how security is defined.
- The basics of CryptHOL.
- The proof methods we use.

Refresher

What have we seen so far?

- Why formal methods are useful to cryptography.
- What SMPC is and how security is defined.
- The basics of CryptHOL.
- The proof methods we use.

Now we will see a toy example of how we actually formally prove security for an Oblivious Transfer protocol.

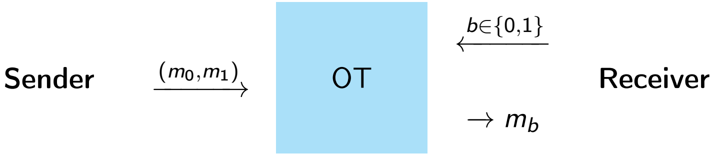
Oblivious Transfer: A fundamental primitive.

Oblivious Transfer: A fundamental primitive.

- Fundamental two party primitive.

Oblivious Transfer: A fundamental primitive.

- Fundamental two party primitive.

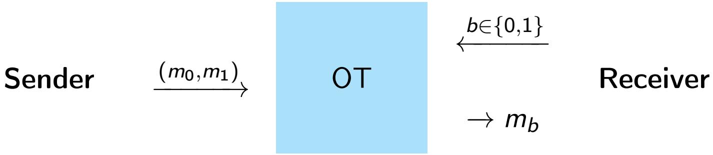


Oblivious Transfer

- The *Sender* holds two messages (m_0, m_1). The *Receiver* has a choice bit, b.

Oblivious Transfer: A fundamental primitive.

- Fundamental two party primitive.

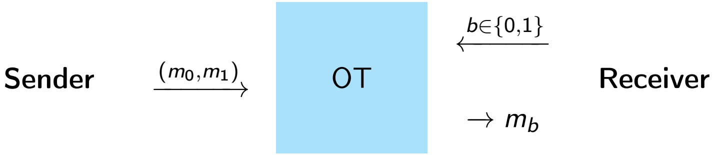


Oblivious Transfer

- The *Sender* holds two messages (m_0, m_1). The *Receiver* has a choice bit, b.
- The *Receiver* learns m_b the *Sender* learns nothing.

Oblivious Transfer: A fundamental primitive.

- Fundamental two party primitive.



Oblivious Transfer

- The *Sender* holds two messages (m_0, m_1). The *Receiver* has a choice bit, b.
- The *Receiver* learns m_b the *Sender* learns nothing.
- The *Receiver* learns nothing of m_{b-1} and the *Sender* does not learn b.

Oblivious Transfer: A toy example.

Oblivious Transfer: A toy example.

Trusted Initialiser

$$r_0, r_1, d \xleftarrow{\$} \{0,1\}$$

$$\begin{array}{ccc}
\mathbf{P_1} & & \mathbf{P_2} \\
m_0, m_1 \in \{0,1\} & & b \in \{0,1\} \\
r_0, r_1 & & d, r_d \\
e & \xleftarrow{\quad\quad \mathrm{e} \quad\quad} & e = b \oplus d \\
f_0 = m_0 \oplus r_e & & \\
f_1 = m_1 \oplus r_{1-e} & \xrightarrow{\quad\quad f_0, f_1 \quad\quad} & f_0, f_1 \\
 & & m_b = f_b \oplus r_d
\end{array}$$

Oblivious Transfer: A toy example - real view for party 2.

Oblivious Transfer: A toy example - real view for party 2.

$$
\begin{array}{ccc}
\mathbf{P_1} & & \mathbf{P_2} \\
m_0, m_1 \in \{0,1\}, r_0, r_1 & & b \in \{0,1\}, d, r_d \\
e & \xleftarrow{\quad\quad e \quad\quad} & e = b \oplus d \\
f_0 = m_0 \oplus r_e & & \\
f_1 = m_1 \oplus r_{1-e} & \xrightarrow{\quad\quad f_0, f_1 \quad\quad} & f_0, f_1 \\
 & & m_b = f_b \oplus r_d
\end{array}
$$

$$
\begin{aligned}
&R_2\ (m_0, m_1)\ b = do\ \{ \\
&\qquad r_0, r_1, d \leftarrow coin_{spmf}; \\
&\qquad let\ e = b \oplus d; \\
&\qquad let\ r_e = (if\ \ e\ then\ r_1\ else\ r_0); \\
&\qquad let\ r_{1-e} = (if\ \ e\ then\ r_0\ else\ r_1); \\
&\qquad return_{spmf}(m_0 \oplus r_e, m_1 \oplus r_{1-e})\}
\end{aligned}
$$

Oblivious Transfer: A toy example - security for party 2.

Oblivious Transfer: A toy example - security for party 2.

The real and simulated views.

$$
\begin{aligned}
&R_2\ (m_0, m_1)\ b = do\ \{\\
&\quad r_0 \leftarrow coin_{spmf};\\
&\quad r_1 \leftarrow coin_{spmf};\\
&\quad d \leftarrow coin_{spmf};\\
&\quad let\ e = b \oplus d;\\
&\quad let\ r_e = (if\ e\ then\ r_1\ else\ r_0);\\
&\quad let\ r_{1-e} = (if\ e\ then\ r_0\ else\ r_1);\\
&\quad return_{spmf}(m_0 \oplus r_e, m_1 \oplus r_{1-e})\}
\end{aligned}
$$

$$
\begin{aligned}
&S_2\ b\ m_b = do\ \{\\
&\quad r_0 \leftarrow coin_{spmf};\\
&\quad r_1 \leftarrow coin_{spmf};\\
&\quad return_{spmf}(r_0, r_1)\}
\end{aligned}
$$

Oblivious Transfer: A toy example - security for party 2.

The real and simulated views.

$$
\begin{array}{l}
R_2\ (m_0, m_1)\ b = do\ \{ \\
\quad r_0 \leftarrow coin_{spmf}; \\
\quad r_1 \leftarrow coin_{spmf}; \\
\quad d \leftarrow coin_{spmf}; \\
\quad let\ e = b \oplus d; \\
\quad let\ r_e = (if\ \ e\ then\ r_1\ else\ r_0); \\
\quad let\ r_{1-e} = (if\ \ e\ then\ r_0\ else\ r_1); \\
\quad return_{spmf}(m_0 \oplus r_e, m_1 \oplus r_{1-e})\}
\end{array}
$$

$$
\begin{array}{l}
S_2\ b\ m_b = do\ \{ \\
\quad r_0 \leftarrow coin_{spmf}; \\
\quad r_1 \leftarrow coin_{spmf}; \\
\quad return_{spmf}(r_0, r_1)\}
\end{array}
$$

We can show these two probabilistic programs are equal.

Oblivious Transfer: A toy example - security for party 2.

Oblivious Transfer: A toy example - security for party 2.

We manipulate the real view using lemmas from CryptHOL.

Oblivious Transfer: A toy example - security for party 2.

We manipulate the real view using lemmas from CryptHOL.

$$
\begin{array}{l}
R_2\ (m_0, m_1)\ b = do\ \{ \\
\quad f_0 \leftarrow map_{spmf}(\lambda r_e.\ m_0 \oplus r_e)\ coin_{spmf}; \\
\quad f_1 \leftarrow map_{spmf}(\lambda r_{1-e}.\ m_1 \oplus r_{1-e})\ coin_{spmf}; \\
\quad return_{spmf}(f_0, f_1)\}
\end{array}
\qquad
\begin{array}{l}
S_2\ b\ m_b = do\ \{ \\
\quad r_0 \leftarrow coin_{spmf}; \\
\quad r_1 \leftarrow coin_{spmf}; \\
\quad return_{spmf}(r_0, r_1)\}
\end{array}
$$

Oblivious Transfer: A toy example - security for party 2.

We manipulate the real view using lemmas from CryptHOL.

$$
\begin{aligned}
&R_2\ (m_0, m_1)\ b = do\ \{\\
&\quad f_0 \leftarrow map_{spmf}(\lambda r_e.\ m_0 \oplus r_e)\ coin_{spmf};\\
&\quad f_1 \leftarrow map_{spmf}(\lambda r_{1-e}.\ m_1 \oplus r_{1-e})\ coin_{spmf};\\
&\quad return_{spmf}(f_0, f_1)\}
\end{aligned}
$$

$$
\begin{aligned}
&S_2\ b\ m_b = do\ \{\\
&\quad r_0 \leftarrow coin_{spmf};\\
&\quad r_1 \leftarrow coin_{spmf};\\
&\quad return_{spmf}(r_0, r_1)\}
\end{aligned}
$$

Prove the lemma:

$$map_{spmf}(\lambda r.\ m \oplus r)\ coin_{spmf} = coin_{spmf}$$

Oblivious Transfer: A toy example - security for party 2.

We manipulate the real view using lemmas from CryptHOL.

$$R_2\ (m_0, m_1)\ b = do\ \{ \\ \quad f_0 \leftarrow map_{spmf}(\lambda r_e.\ m_0 \oplus r_e)\ coin_{spmf}; \\ \quad f_1 \leftarrow map_{spmf}(\lambda r_{1-e}.\ m_1 \oplus r_{1-e})\ coin_{spmf}; \\ \quad return_{spmf}(f_0, f_1)\}$$

$$S_2\ b\ m_b = do\ \{ \\ \quad r_0 \leftarrow coin_{spmf}; \\ \quad r_1 \leftarrow coin_{spmf}; \\ \quad return_{spmf}(r_0, r_1)\}$$

Prove the lemma:

$$map_{spmf}(\lambda r.\ m \oplus r)\ coin_{spmf} = coin_{spmf}$$

and apply it twice to show:

$$R_2\ (m_0, m_1)\ b = do\ \{ \\ \quad f_0 \leftarrow coin_{spmf}; \\ \quad f_1 \leftarrow coin_{spmf}; \\ \quad return_{spmf}(f_0, f_1)\}$$

Oblivious Transfer: A toy example - security.

Lemma

$$R_2\ (m_0, m_1)\ b = S_2\ b\ m_b$$

Oblivious Transfer: A toy example - security.

Lemma

$$R_2\ (m_0, m_1)\ b = S_2\ b\ m_b$$

This implies security for party two, namely:

$$R_2\ (m_0, m_1)\ b \stackrel{c}{\equiv} S_2\ b\ m_b$$

Together with similar analysis of party one we have the security result.

Oblivious Transfer: A toy example - security.

Lemma

$$R_2\ (m_0, m_1)\ b = S_2\ b\ m_b$$

This implies security for party two, namely:

$$R_2\ (m_0, m_1)\ b \overset{c}{\equiv} S_2\ b\ m_b$$

Together with similar analysis of party one we have the security result.

Theorem The Bit Oblivious Transfer protocol is information theoretic secure in the semi honest adversary model.

The rest of the paper and future work

The rest of the paper and future work

Formalised security in this model for:

- Secure multiplicaton protocol.
- Noar Pinkas OT.
- A protocol to securely compute an AND gate.

The rest of the paper and future work

Formalised security in this model for:

- Secure multiplicaton protocol.
- Noar Pinkas OT.
- A protocol to securely compute an AND gate.

Future work:

- GMW protocol - allows for the secure computation of any boolean circuit.

The rest of the paper and future work

Formalised security in this model for:

- Secure multiplicaton protocol.
- Noar Pinkas OT.
- A protocol to securely compute an AND gate.

Future work:

- GMW protocol - allows for the secure computation of any boolean circuit.
- Garbled circuits - originally how the Millionaire's problem was solved by Yao.

The rest of the paper and future work

Formalised security in this model for:

- Secure multiplicaton protocol.
- Noar Pinkas OT.
- A protocol to securely compute an AND gate.

Future work:

- GMW protocol - allows for the secure computation of any boolean circuit.
- Garbled circuits - originally how the Millionaire's problem was solved by Yao.
- These methods are the main ways in which SMPC is realised.

Thank you