From Secure MPC to Efficient Zero-Knowledge

David Wu
March, 2017
The Complexity Class \textbf{NP}

\textbf{NP} – the class of problems that are \textit{efficiently verifiable}

A language \( \mathcal{L} \) is in \textbf{NP} if there exists a polynomial-time verifier \( R \) such that

\[
x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)} \; R(x,w) = 1
\]
Interactive Proof Systems [GMR85]

**NP** admits efficient **non-interactive** proofs

$G$ is 3-colorable

$G = \text{public graph}$

prover \hspace{1cm} \rightarrow \hspace{1cm} \text{verifier}
Interactive Proof Systems [GMR85]

**IP:** class of languages that have an interactive proof system

$G$ is not 3-colorable

Prover and verifier exchange a sequence of messages.

Prover

Verifier

Accept
Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms \((P, V)\) with following properties:

- **Completeness:** \(\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = 1] = 1\)
- **Soundness:** \(\forall x \notin \mathcal{L}, \forall P^* : \Pr[\langle P^*, V \rangle(x) = 0] = 1\)
Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms \((P, V)\) with the following properties:

- **Completeness:** \(\forall x \in \mathcal{L} : \Pr[P(x) = 1] = 1\)
- **Soundness:** \(\forall x \notin \mathcal{L}, \forall P^* : \Pr[P^*(x) = 0] = 1\)

Can also allow soundness error \(\epsilon\)
Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms $(P, V)$ with following properties:

- **Completeness:** $\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = 1] = 1$
- **Soundness:** $\forall x \notin \mathcal{L}, \forall P^* : \Pr[\langle P^*, V \rangle(x) = 0] = 1$
- **Efficiency:** $V$ runs in polynomial time (in $|x|$)

computationally unbounded

randomized, efficient
Interactive Proof Systems [GMR85]

**IP**: class of languages that have an interactive proof system

- $d\text{IP} = \text{NP}$
  - ($d\text{IP}$: interactive proofs with deterministic verifier)
- $\text{IP} = \text{PSPACE}$ [LFKN90, Sha90]
Zero-Knowledge Proofs [GMR85]

Can we prove to a verifier that a statement $x$ is in a language $\mathcal{L}$ without revealing anything more about $x$ other than the fact that $x \in \mathcal{L}$?
Zero-Knowledge Proofs [GMR85]

common input: statement \( x \in \mathcal{L} \)

real distribution

\( \langle P, V^* \rangle(x) \)

\( \approx_c \)

ideal distribution

\( S(x) \)

Zero-Knowledge: for all efficient verifiers \( V^* \), there exists an efficient simulator \( S \) such that:

\[ \forall x \in \mathcal{L} : \langle P, V^* \rangle(x) \approx_c S(x) \]
Zero-Knowledge Proofs [GMR85]

Zero-Knowledge: for all efficient verifiers $V^*$, there exists an efficient simulator $S$ such that:

$$\forall x \in \mathcal{L} : \langle P, V^* \rangle(x) \approx_c S(x)$$

- Honest verifier zero-knowledge (HVZK): simulator exists only for honest verifier
- Can also consider statistical zero-knowledge and perfect zero-knowledge

Common input: statement $x \in \mathcal{L}$
Zero-Knowledge Proofs [GMR85]

common input: statement $x \in \mathcal{L}$

\[ \langle P, V^* \rangle (x) \approx_c S(x) \]

real distribution $\mathcal{R}$

ideal distribution $\mathcal{S}$

Assuming the existence of one-way functions (OWFs), every \textbf{NP} (in fact, \textbf{IP}) language has a computational zero-knowledge proof proof system [GMW86]
Two-Party Computation

Zero knowledge is a special case of two-party computation.
Two-Party Computation

Zero knowledge is a special case of two-party computation.

Every message is a deterministic function of the party’s input, its internal randomness, and the set of messages it has received.
Two-Party Computation

Zero knowledge is a special case of two-party computation.

Correctness:
\[ y_1 = f(w_1, w_2) = y_2 \]
Two-Party Computation

Zero knowledge is a special case of two-party computation.

\[
\text{View}_{P_i}(w_1, w_2; r_1, r_2) = (w_i, r_i, \{m_i\})
\]
Two-Party Computation

Zero knowledge is a special case of \textit{two-party computation}.

\[
\text{View}_{P_i}(w_1, w_2; r_1, r_2) = (w_i, r_i, \{m_i\})
\]

\[
y_i = \Pi_{f,i} \left( \text{View}_{P_i}(w; r) \right)
\]
Multiparty Computation (MPC)

Correctness: For all inputs $\mathbf{w}$ and all $i \in [n]$

$$\Pr \left[ \Pi_{f,i} \left( \text{View}_{P_i}(\mathbf{w} ; \mathbf{r}) \right) = f(\mathbf{w}) \right] = 1$$
Multiparty Computation (MPC)

**Correctness:** For all inputs $\mathbf{w}$ and all $i \in [n]$

\[
\Pr \left[ \Pi_{f,i} \left( \text{View}_{P_i}(\mathbf{w}; \mathbf{r}) \right) = f(\mathbf{w}) \right] = 1
\]

\[
y_1 = f(\mathbf{w})
\]

\[
y_2 = f(\mathbf{w})
\]

\[
y_3 = f(\mathbf{w})
\]

\[
y_4 = f(\mathbf{w})
\]

\[
y_5 = f(\mathbf{w})
\]
Multiparty Computation (MPC)

**t-Privacy:** For all $T \subseteq [n]$ where $|T| \leq t$, there exists an efficient simulator $S_T$ such that for all inputs $w$:

$$\{\text{View}_{P_i}(w ; r)\}_{i \in T} \equiv S_T(f, \{w_i\}_{i \in A}, f(w))$$
Multiparty Computation (MPC)

**t-Privacy**: For all \( T \subseteq [n] \) where \( |T| \leq t \), there exists an efficient simulator \( S_T \) such that for all inputs \( w \):
\[
\{\text{View}_{P_i}(w ; r)\}_{i \in T} \equiv S_T(f, \{w_i\}_{i \in A}, f(w))
\]

Views of any \( t \)-subset of the parties do not reveal anything more about the private inputs of any other party
Multiparty Computation (MPC)

**t-Robustness:** For all $T \subseteq [n]$ where $|T| \leq t$, and for all $f$ where $f(w) = 0$ for all $w$, then

$$\Pr \left[ \Pi_{f,i} \left( \text{View}_{P_i}(w; r) \right) = 1 \right] = 0$$

for all $i \in [n] \setminus T$ even if the players in $T$ have been *arbitrarily* corrupted.
Multiparty Computation (MPC)

**t-Robustness**: For all \( T \subseteq [n] \) where \(|T| \leq t\), and for all \( f \) where \( f(w) = 0 \) for all \( w \), then

\[
\text{Pr} \left[ \Pi_{f,i} \left( \text{View}_{P_i}(w; r) \right) = 1 \right] = 0
\]

for all \( i \in [n] \setminus T \) even if the players in \( T \) have been arbitrarily corrupted.

If there are no inputs \( w \) to \( f \) where \( f(w) = 1 \), then a malicious adversary corrupting up to \( t \) parties cannot cause an honest party to output 1.
Zero-Knowledge from Two-Party Computation

Zero knowledge is a special case of two-party computation:

Given a statement $x$ for an NP relation $R$, define the function $f_x(w) = R(x, w)$

We require a 1-private, 1-robust two-party computation protocol $\Pi_{f_x}$ for $f_x$

The prover and verifier execute $\Pi_{f_x}$

• Prover’s input: the witness $w$
• Verifier’s input: none

The verifier accepts if the output of $\Pi_{f_x}$ is 1
Zero-Knowledge from Two-Party Computation

Zero knowledge is special case of two-party computation

- General two-party computation with robustness against malicious adversaries requires oblivious transfer (OT) [Yao86, GMW87] and thus, cannot be instantiated from one-way functions

On the other hand, zero knowledge for $\mathbf{NP}$ is known from one-way functions (OWFs) [GMW86]

- Constructions very inefficient – relies on running a Karp reduction to an $\mathbf{NP}$-complete problem (e.g., 3-coloring)

This talk: constructing zero-knowledge for $\mathbf{NP}$ from OWFs + black-box use of any (semi-honest) MPC protocol
“MPC in the Head” [IKOS07]

Let $R(x, w)$ be an \textbf{NP} relation and define the function

$$f_x(w_1, ..., w_n) = R(x, w_1 \oplus \cdots \oplus w_n),$$

where $n \geq 3$

\textbf{Key idea:}

- Prover “simulates” an $n$-party MPC protocol $\Pi_{f_x}$ for the function $f_x$
- Verifier checks that the simulation is correct

\textbf{Key advantage:} relies only on OWFs and \textbf{semi-honest} secure MPC
“MPC in the Head” [IKOS07]

Key cryptographic primitive: commitment scheme

\[
\text{Commit}(m; r) \rightarrow c
\]

\[
\text{Open}(c, r) \rightarrow m
\]
“MPC in the Head” [IKOS07]

Key cryptographic primitive: commitment scheme

- **Perfectly binding**: each commitment can be opened in exactly one way
  \[ \forall r_0, r_1 : \text{Commit}(m_0 ; r_0) = \text{Commit}(m_1 ; r_1) \Rightarrow m_0 = m_1 \]

- **Computationally hiding**: commitment hides committed value to any bounded adversary:
  \[ \text{Commit}(m_0 ; r) \approx_c \text{Commit}(m_1 ; r) \]

- Non-interactive commitments can be constructed from any injective OWF [Blu81]
- Interactive commitments can be constructed from any OWF
“MPC in the Head” [IKOS07]

\[ f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n) \]

\(R(x, w)\) is an \textbf{NP} relation
“MPC in the Head” [IKOS07]

\[ f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \ldots \oplus w_n) \]

Step 1: Secret share the witness

uniformly random strings
“MPC in the Head” [IKOS07]

\[ f_x(w_1, \ldots, w_n) = R(x, w_1 \ominus \cdots \ominus w_n) \]

\[ w = w_1 \ominus w_2 \ominus w_3 \ominus w_4 \ominus w_5 \]

uniformly random strings

Step 2: Simulate \( \Pi_{f_x} \) using randomness \( r \)
“MPC in the Head” [IKOS07]

\[ f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n) \]

\[ c_1 = \text{Commit}(\text{View}_{P_1}(w; r); r'_1) \]
\[ \vdots \]
\[ c_n = \text{Commit}(\text{View}_{P_n}(w; r); r'_n) \]

Step 3: Commit to the view of each party
“MPC in the Head” [IKOS07]

public input: statement $x$

Step 4: Prover sends the commitments to the verifier
“MPC in the Head” [IKOS07]

public input: statement $x$

Step 5: Verifier challenges prover to open two of the views (at random)
“MPC in the Head” [IKOS07]

public input: statement $x$

Step 6: Prover opens up commitments to requested views

$(x, w)$

$c_1, c_2, \ldots, c_n$

$1 \leq i < j \leq n$

Open$(c_i, r_i'),$ Open$(c_j, r_j')$
"MPC in the Head" [IKOS07]

Verification conditions:
1. Commitments are correctly opened
2. The outputs of both $P_i$ and $P_j$ is 1
3. The views $\text{View}_{P_i}$ and $\text{View}_{P_j}$ are consistent with an honest execution of $\Pi_{fx}$

Open($c_i, r'_i$), Open($c_j, r'_j$)

$1 \leq i < j \leq n$

$c_1, c_2, ..., c_n$

Step 7: Verifier checks the proof
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Completeness:**

- Suppose $R(x, w) = 1$
- Prover is honest so $w = w_1 \oplus \cdots \oplus w_n$
- By construction, $f_x(w_1, \ldots, w_n) = R(x, w) = 1$
- Perfect correctness of $\Pi_{f_x}$ implies that all parties in honest execution output $f_x(w_1, \ldots, w_n) = 1$
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Soundness:**

- Suppose $R(x, w) = 0$ for all $w$
- By perfect correctness of $\Pi_{f_x}$, for all choices of $w_1, \ldots, w_n$, parties in an honest execution of $\Pi_{f_x}$ will output 0
- Either all outputs are 0 or there is at least one pair of views that are inconsistent
- Verifier rejects with probability at least $1/n^2$ (commitments are perfectly binding)
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Soundness:**
- Suppose $R(x, w) = 0$ for all $w$.
- By perfect correctness of $\Pi_{f_x}$, for all choices of $w_1, \ldots, w_n$, parties in an honest execution of $\Pi_{f_x}$ will output 0.
- Either all outputs are 0 or there is at least one pair of views that are inconsistent.
- Verifier rejects with probability at least $1/n^2$ (commitments are perfectly binding).

Can be amplified by running the protocol multiple times ($\kappa n^2$ times to achieve negligible soundness error $2^{-\kappa}$).
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Zero-Knowledge:**

- Suppose that $R(x, w) = 1$
- View of verifier consists of committed views to all parties, and views $\text{View}_{P_i}(w; r)$ and $\text{View}_{P_j}(w; r)$ (which include $w_i$ and $w_j$) for two of the parties
- When $n \geq 3$, $w_i, w_j$ are uniformly random strings
- By 2-privacy of $\Pi_{f_x}$, $\text{View}_{P_i}(w; r)$ and $\text{View}_{P_j}(w; r)$ can be simulated given just $f_x, w_i, w_j, f_x(w) = 1$
"MPC in the Head" [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Zero-Knowledge**
- Suppose that $R(x, w) = 1$
- View of verifier consists of committed views to all parties, and views $\text{View}_{P_i}(w; r)$ and $\text{View}_{P_j}(w; r)$ (which include $w_i$ and $w_j$) for two of the parties
- When $n \geq 3$, $w_i, w_j$ are uniformly random strings
- By 2-privacy of $\Pi_{f_x}$, $\text{View}_{P_i}(w; r)$ and $\text{View}_{P_j}(w; r)$ can be simulated given just $f_x, w_i, w_j, f_x(w) = 1$

Since prover is honest here, the proof only requires privacy against *semi-honest* parties
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose the commitment scheme is perfectly binding and computationally hiding and that $\Pi_{f_x}$ is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation $R$.

**Concrete instantiations:**
- Information-theoretic: 5-party BGW protocol [BGW88]
- Computational (based on OT): 3-party GMW protocol [GMW87]
- ... and many more
“MPC in the Head” [IKOS07]

Using an $n$-party MPC protocol, the soundness error is $1 - 1/n^2$

Consequence: achieving negligible soundness $2^{-\kappa}$ requires $\Omega(\kappa)$ repetitions of the protocol

*Can we obtain negligible soundness error without performing the $\Omega(\kappa)$ repetitions of the protocol?*
“MPC in the Head” [IKOS07]

Using an $n$-party MPC protocol, the soundness error is $1 - 1/n^2$

Soundness error is large because verifier checks only a single view

Can reduce the soundness error by having the prover open up more views (e.g., $t = \Theta(\kappa)$ views)

- Zero-knowledge maintained as long as $\Pi_{f_x}$ is $t$-private
- Soundness amplification will rely on leveraging robustness of $\Pi_{f_x}$
“MPC in the Head” [IKOS07]

Using an $n$-party MPC protocol, the soundness error is $1 - 1/n^2$

Soundness error is large because verifier checks only a single view.

Can reduce the soundness error by having the prover open up more views (e.g., $t = \Theta(\kappa)$ views)
- Zero-knowledge maintained as long as $\Pi_{f_x}$ is $t$-private.
- Soundness amplification will rely on leveraging robustness of $\Pi_{f_x}$.

Without robustness, even if the prover open $n - 1$ views, the soundness error can still be $O\left(\frac{1}{n}\right)$.
“MPC in the Head” [IKOS07]

\[ n = \Theta(\kappa) \quad t = \Theta(n) \]

Suppose \( \Pi_{fx} \) is an \( n \)-party MPC protocol that is \( t \)-private and \( t \)-robust.

Verifier can now ask for \( t \) openings \textit{without} compromising zero-knowledge.

\( x, w \)

\( T \subseteq [n] \) where \( |T| = t \)

\( \{\text{Open}(c_i, r'_i)\}_{i \in [T]} \)
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust.

To analyze soundness, define the inconsistency graph $G$ for the prover’s simulated MPC protocol:

- Nodes correspond to parties.
- An edge between $i$ and $j$ denotes an inconsistency between $\text{View}_{P_i}$ and $\text{View}_{P_j}$.  

![Graph Diagram]
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust.

To analyze soundness, define the inconsistency graph $G$ for the prover’s simulated MPC protocol:

Verifier chooses some subset of nodes and rejects if induced subgraph on those nodes contains an edge.

Verifier rejects.
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust.

To analyze soundness, define the inconsistency graph $G$ for the prover’s simulated MPC protocol:

Verifier chooses some subset of nodes and rejects if induced subgraph on those nodes contains an edge.

Verifier may accept
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust

**Case 1:** Suppose $G$ contains a vertex cover $B$ of size at most $t$

\[ B = \{4, 5\} \]
“MPC in the Head” [IKOS07]

Suppose \( \Pi_{f_x} \) is an \( n \)-party MPC protocol that is \( t \)-private and \( t \)-robust.

**Case 1:** Suppose \( G \) contains a vertex cover \( B \) of size at most \( t \).

- By definition, views of all nodes not in \( B \) are consistent (i.e., correspond to an honest protocol execution).
- \( \Pi_{f_x} \) is \( t \)-robust, so all nodes not in \( B \) output 0 on a false statement.
Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust.

Case 1: Suppose $G$ contains a vertex cover $B$ of size at most $t$.

- By definition, views of all nodes not in $B$ are consistent (i.e., correspond to an honest protocol execution).
- $\Pi_{f_x}$ is $t$-robust, so all nodes not in $B$ output 0 on a false statement.
- Verifier can only accept if $T \subseteq B$, so soundness error is bounded by $(t/n)^t = 2^{-\Omega(n)} = 2^{-\Omega(\kappa)}$. 

Failure only if all nodes chosen by verifier fall in $B$.
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust.

**Case 2:** Suppose the minimum vertex cover of $G$ has size greater than $t$.

- Large number of corrupted parties $\Rightarrow$ likely to be detected by verifier.
“MPC in the Head” [IKOS07]

Suppose $\Pi_{f_x}$ is an $n$-party MPC protocol that is $t$-private and $t$-robust

**Case 2:** Suppose the minimum vertex cover of $G$ has size greater than $t$

- Then $G$ has a matching of size greater than $t/2$
- Verifier accepts only if no edges in $G$ between any of the nodes in $T$, and in particular, no edges in the matching
- Since $t = \Theta(n)$, the verifier misses all edges in the matching with probability $2^{-\Omega(n)} = 2^{-\Omega(\kappa)}$
“MPC in the Head” [IKOS07]

**Theorem [IKOS07].** Suppose that the following holds:

- the commitment scheme is perfectly binding and computationally hiding,
- $\Pi_{f_x}$ is $t$-private (against semi-honest adversaries), and $t$-robust (against malicious adversaries) $n$-party protocol for $f_x$.

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-verifier zero-knowledge proof for the NP-relation $R$ with soundness error $2^{-\kappa}$.

*Relies only on OWFs (for the commitments) and black-box access to $\Pi_{f_x}$.***
Theorem [IKOS07]. Suppose that the following holds:

- the commitment scheme is perfectly binding and computationally hiding,
- $\Pi_{f^x}$ is $t$-private (against semi-honest adversaries) and $t$-robust (against malicious adversaries) $n$-party protocol for $f^x$.

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-verifier zero-knowledge proof for the NP-relation $R$ with soundness error $2^{-\kappa}$.

Relies only on OWFs (for the commitments) and black-box access to $\Pi_{f^x}$.
"MPC in the Head" [IKOS07]

Theorem [IKOS07]. Suppose that the following holds:

- the commitment scheme is perfectly binding and computationally hiding,
- $\Pi_{f_x}$ is $t$-private (against semi-honest adversaries) and $t$-robust (against malicious adversaries) $n$-party protocol for $f_x$.

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-verifier zero-knowledge proof for the NP-relation $R$ with soundness error $2^{-\kappa}$.

Relies only on OWFs (for the commitments) and black-box access to $\Pi_{f_x}$.

Concrete parameters: for $2^{-80}$ soundness error, can use $(n, t, r) = (92, 64, 64)$
ZKBoo [GMO16]

\[ S \subseteq [n] \text{ where } |S| = t_1 \]

\( c_1, c_2, \ldots, c_n \)

\{\text{Open}(c_i, r'_i)\}_{i \in [S]}

For concrete soundness targets (e.g., \(2^{-80}\)), most efficient instantiation of IKOS is to use *simple, non-robust* multiparty computation protocol and amplify soundness by repeating the protocol.
ZKBoo [GMO16]

\[ S \subseteq [n] \text{ where } |S| = t_1 \]

\[ c_1, c_2, \ldots, c_n \]

For concrete soundness targets (e.g., \(2^{-80}\)), most efficient instantiation of IKOS is to use simple, non-robust multiparty computation protocol and amplify soundness by repeating the protocol.

\( n\)-party BGW protocol obtaining soundness error \(2^{-80}\) requires \(n = 1122, t = 374\)

Iterating a 3-party protocol with 2-privacy yields proofs which contain 274 bits per multiplication gate for \(2^{-80}\) soundness.
ZKBoo [GMO16]

Emulating an MPC protocol cheaper than running the MPC protocol in the standard model
ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model
ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model
ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In MPC setting, channels are implemented using secure two-party computation
- In “MPC-in-the-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function $f$)
ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In MPC setting, channels are implemented using secure two-party computation.
- In “MPC-in-the-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function $f$).

$$\text{arbitrary channel}$$

$$f(x, y)$$
ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In MPC setting, channels are implemented using secure two-party computation
- In “MPC-in-the-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function $f$)
- New design space for MPC protocols

\[ f(x, y) \]
ZKBoo [GMO16]

\[ f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n) \]

How to construct \( \Pi_{f_x} \)?
(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)

\[ w = w_1 + w_2 + w_3 \]

Function evaluation on secret shared inputs
(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)

\[
\begin{align*}
\forall i: & \quad w_i = w_i^{(0)} + w_i^{(1)} + w_i^{(2)} \\
\forall i: & \quad y_i = y_i^{(1)} + y_i^{(2)}
\end{align*}
\]

\[w = w_1 + w_2 + w_3\]

\[y = y_1 + y_2 + y_3\]

protocol execution proceeds in a series of rounds

gate-by-gate evaluation of \(f\)

Each party communicates with one other party
(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)

\[ \begin{align*}
  w_1^{(0)} & \quad f_1^{(1)} \quad w_2^{(0)} \quad f_2^{(1)} \quad w_3^{(0)} \quad f_3^{(1)} \\
  w_1^{(1)} & \quad w_2^{(1)} \quad w_3^{(1)} \\
  \vdots & \quad \vdots & \quad \vdots \\
  y_1 & \quad y_2 & \quad y_3
\end{align*} \]

\[ \text{View}_{P_i}(w; r) = \{w_i^{(0)}, ..., w_i^{(N)}\} \]

protocol is 2-private
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$
Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

**no interaction needed**

$x = x_1 + x_2 + x_3$

$$f_1$$
$$f_2$$
$$f_3$$

$(x_1 + \alpha) + x_2 + x_3 = x + \alpha$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

$x = x_1 + x_2 + x_3$

$\alpha x_1 + \alpha x_2 + \alpha x_3 = \alpha x$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

**no interaction needed**

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$

$$x_1, y_1 \quad x_2, y_2 \quad x_3, y_3$$

$$x_1 + y_1 \quad x_2 + y_2 \quad x_3 + y_3$$

$$(x_1 + y_1) + (x_2 + y_2) + (x_3 + y_3) = x + y$$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$:

\[
\begin{align*}
 x &= x_1 + x_2 + x_3 \\
 y &= y_1 + y_2 + y_3 \\
 x_1, y_1 & \quad x_2, y_2 \quad x_3, y_3
\end{align*}
\]

\[
\frac{x_1 y_1 + x_2 y_1 + x_1 y_2}{x_1 y_1 + x_2 y_1 + x_1 y_2} + \frac{x_2 y_2 + x_3 y_2 + x_2 y_3}{x_2 y_2 + x_3 y_2 + x_2 y_3} + \frac{x_3 y_3 + x_1 y_3 + x_3 y_1}{x_3 y_3 + x_1 y_3 + x_3 y_1} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy
\]

Diagram:

- **Multiply**
  - Inputs: $x, y$
  - Output: $xy$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

\[
\begin{align*}
    x &= x_1 + x_2 + x_3 \\
    y &= y_1 + y_2 + y_3 \\

    x_1y_1 + x_2y_1 + x_1y_2 &= f_1 \\
    x_2y_2 + x_3y_2 + x_2y_3 &= f_2 \\
    x_3y_3 + x_1y_3 + x_3y_1 &= f_3
\end{align*}
\]

only depends on $x_1, y_1, x_2, y_2$ only depends on $x_2, y_2, x_3, y_3$ only depends on $x_1, y_1, x_3, y_3$

\[
\left(\frac{x_1y_1 + x_2y_1 + x_1y_2}{x_1y_1 + x_2y_1 + x_1y_2}\right) + \left(\frac{x_2y_2 + x_3y_2 + x_2y_3}{x_2y_2 + x_3y_2 + x_2y_3}\right) + \left(\frac{x_3y_3 + x_1y_3 + x_3y_1}{x_3y_3 + x_1y_3 + x_3y_1}\right) = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy
\]
\( (2, 3) \)-Function Decompositions [GMO16]

Express \( f_x \) as an arithmetic circuit over finite field \( \mathbb{F} \)

\[
\begin{align*}
  x &= x_1 + x_2 + x_3 \\
  y &= y_1 + y_2 + y_3 \\

  x_1, y_1 &\quad x_2, y_2 &\quad x_3, y_3 \\

  x_1 y_1 + x_2 y_1 + x_1 y_2 &\quad x_3 y_3 + x_1 y_3 + x_3 y_1 \\

  x_2 y_2 + x_3 y_2 + x_2 y_3 \\

  \text{only depends on } x_1, y_1, x_2, y_2 &\quad \text{only depends on } x_2, y_2, x_3, y_3 &\quad \text{only depends on } x_1, y_1, x_3, y_3 \\

  \frac{x_1 y_1 + x_2 y_1 + x_1 y_2}{(x_1 y_1 + x_2 y_1 + x_1 y_2)} + \frac{x_2 y_2 + x_3 y_2 + x_2 y_3}{(x_2 y_2 + x_3 y_2 + x_2 y_3)} + \frac{x_3 y_3 + x_1 y_3 + x_3 y_1}{(x_3 y_3 + x_1 y_3 + x_3 y_1)} &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy
\end{align*}
\]
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

$x = x_1 + x_2 + x_3$
$y = y_1 + y_2 + y_3$

$x_1 y_1 + x_2 y_1 + x_1 y_2 + R_1(c) - R_2(c)$

only depends on $x_1,y_1,x_2,y_2$

$x_2 y_2 + x_3 y_2 + x_2 y_3 + R_2(c) - R_3(c)$

only depends on $x_2,y_2,x_3,y_3$

$x_3 y_3 + x_1 y_3 + x_3 y_1 + R_3(c) - R_1(c)$

only depends on $x_1,y_1,x_3,y_3$

$(x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

$x = x_1 + x_2 + x_3$
$y = y_1 + y_2 + y_3$

$x_1, y_1$  $x_2, y_2$  $x_3, y_3$

$f_3$

random blinding factors ($R_i(c)$ is randomness used by $i^{th}$ party on gate $c$)

$x_1 y_1 + x_2 y_1 + x_1 y_2 + R_1(c) - R_2(c)$

$x_2 y_2 + x_3 y_2 + x_2 y_3 + R_2(c) - R_3(c)$

$x_3 y_3 + x_1 y_3 + x_3 y_1 + R_3(c) - R_1(c)$

only depends on $x_1, y_1, x_2, y_2$
only depends on $x_2, y_2, x_3, y_3$
only depends on $x_1, y_1, x_3, y_3$

$(x_1 y_1 + x_2 y_1 + x_1 y_2) + (x_2 y_2 + x_3 y_2 + x_2 y_3) + (x_3 y_3 + x_1 y_3 + x_3 y_1) = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

$x = x_1 + x_2 + x_3$
$y = y_1 + y_2 + y_3$

$x_1, y_1$  $x_2, y_2$  $x_3, y_3$

1 bit per multiplication

$x_1y_1 + x_2y_1 + x_1y_2 + R_1(c) - R_2(c)$

$x_2y_2 + x_3y_2 + x_2y_3 + R_2(c) - R_3(c)$

$x_3y_3 + x_1y_3 + x_3y_1 + R_3(c) - R_1(c)$

only depends on $x_1y_1, x_2y_2$
only depends on $x_2y_2, x_3y_3$
only depends on $x_1y_1, x_3y_3$

$(x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

Computation on local shares

Two-party computation
(2, 3)-Function Decompositions [GMO16]

Express $f_x$ as an arithmetic circuit over finite field $\mathbb{F}$

Computation on local shares

Two-party computation

137 iterations + open 2 views = 274 bits of communication per multiplication gate in ZKBoo
Summary

• “MPC in the head” gives new paradigm for constructing efficient zero-knowledge proof systems
• New directions in designing efficient MPC protocols for zero-knowledge can be quite efficient in practice
• Zero-knowledge protocols can also be used for signature schemes (Fiat-Shamir) – including post-quantum signatures!
Open Directions

• Designing new MPC protocols for more efficient zero-knowledge
  • Many theoretical MPC protocols with better communication complexity – shorter proofs and (post-quantum) signatures
• Alternative viewpoints: “MPC in the head” as a PCP with large alphabet (i.e., each party’s view)