Better 2-round adaptive MPC

Ran Canetti, Oxana Poburinnaya, Muthuramakrishnan Venkitasubramaniam
Secure Multiparty Computation [Yao’82]

\[ f(x_1, x_2, x_3) \]
Secure Multiparty Computation [Yao’82]

Correctness: every party learns $y = f(x_1, x_2, x_3)$
Security: even if a party is dishonest, it only learns the output $y$, but nothing else
Our results:

Semi-honest case

2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)
Our results:

- Semi-honest case
  - 2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)

- Malicious case

GP’15 2 round fully adaptive MPC becomes RAM-efficient
Our results:

- Semi-honest case
  - 2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)
- Malicious case
  - ZK proofs with RAM efficiency

Plug into GP'15

GP'15 2 round fully adaptive MPC becomes **RAM-efficient**
Static vs Adaptive Security

when do parties become dishonest?

**Static security:**
a set of dishonest parties is fixed before the protocol starts.

\[ f(x_1, x_2, x_3) \]
Static vs Adaptive Security

when do parties become dishonest?

Static security:
a set of dishonest parties is fixed before the protocol starts

Adaptive security:
parties may become dishonest during the execution of the protocol
Static vs Adaptive Security

when do parties become dishonest?

**Static security:**
a set of dishonest parties is fixed before the protocol starts

- $x_1$
- $x_3$
- $f(x_1, x_2, x_3)$

**Adaptive security:**
parties may become dishonest during the execution of the protocol

- $x_1$
- $x_3$
- $f(x_1, x_2, x_3)$
Static vs Adaptive Security

when do parties become dishonest?

Static security:
a set of dishonest parties is fixed before the protocol starts

Adaptive security:
parties may become dishonest during the execution of the protocol
Adaptive Security of MPC

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution
Adaptive Security of MPC

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. simulate communication (without knowing $x_1, \ldots, x_n$)
Adaptive Security of MPC

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. simulate communication (without knowing $x_1, \ldots, x_n$)
Adaptive Security of MPC

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. simulate communication (without knowing $x_1, \ldots, x_n$)
Adaptive Security of MPC

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. simulate communication (without knowing $x_1, \ldots, x_n$)
2. simulate $r_i$ of corrupted parties, consistent with communication and $x_i$
Example: Adaptively Secure Encryption (NCE)

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
Example: Adaptively Secure Encryption (NCE)

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. Sim() → cs, state
Example: Adaptively Secure Encryption (NCE)

Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. Sim() → c^s, state
2. Sim(state, m) → r^s_{Enc}, k^s
Example: Adaptively Secure Encryption (NCE)

Adaptive corruptions:
advocate can decide who to corrupt adaptively during the execution

Adv gets:
- either real \((r, k, c = Enc_k(m; r))\)
- or fake \((r_{Enc}^s, k^s, c^s)\)

Simulator:
1. \(\text{Sim}() \rightarrow c^s, \text{state}\)
2. \(\text{Sim}(\text{state}, m) \rightarrow r_{Enc}^s, k^s\)

possible for a non-committing encryption (NCE)
Full Adaptive Security

Full adaptive security:
● No erasures
Full Adaptive Security

Full adaptive security:
- No erasures
- Security even when all parties are corrupted
Full Adaptive Security

**Full adaptive security:**
- No erasures
- Security even when all parties are corrupted
Full Adaptive Security

**Full adaptive security:**
- No erasures
- Security even when all parties are corrupted

Until 2015: # of rounds $\sim$ depth of circuit (CLOS02)

Constant round protocols: CGP15, DKR15, GP15.
## Full Adaptive Security: state of the art, semi-honest

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
</tr>
</tbody>
</table>

The only 2 round MPC

*need a CRS even for HBC case!
### Full Adaptive Security: state of the art, semi-honest

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
</tr>
</tbody>
</table>

Q1: can we build 2 round MPC with **global (non-programmable)** CRS?
Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

<table>
<thead>
<tr>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP'15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
</tr>
<tr>
<td>DKR'15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
</tr>
</tbody>
</table>

**local CRS**
**Full Adaptive Security: state of the art, semi-honest**

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>subexp iO</td>
<td></td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>iO</td>
<td></td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>subexp. iO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(even in HBC case)</td>
<td></td>
</tr>
</tbody>
</table>

Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

![Diagram showing local CRS](image-url)
Full Adaptive Security: state of the art, semi-honest

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>-</td>
</tr>
</tbody>
</table>

Q1: can we build 2 round MPC with **global (non-programmable) CRS?**

![Diagram showing local CRS 1 and local CRS 2 connected to each other with arrows, indicating communication or interaction.](image)
**Full Adaptive Security: state of the art, semi-honest**

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>- (even in HBC case)</td>
</tr>
</tbody>
</table>

Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

- **local CRS 1**
- **local CRS 2**
- **global CRS**
Full Adaptive Security: state of the art, semi-honest

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP'15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DKR'15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>GP'15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>- (even in HBC case)</td>
<td>-</td>
</tr>
</tbody>
</table>

Q2: can we achieve **randomness hiding**? (Evaluation of $f(x_1, ..., x_n; r)$ hides $r$ even if everyone is corrupted)

choose $N = pq$
nobody knows $p$, $q$
<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
<th>supports RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>- (even in HBC case)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Q3: can we use the fact that $f$ is a succinct RAM program?
<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
<th>supports RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Q4: can we build 2 round MPC from **weaker assumptions**? (e.g. remove the need for subexp. iO)
## Full Adaptive Security

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
<th>supports RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>2</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

This work
### Full Adaptive Security

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
<th>supports RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP’15</td>
<td>2</td>
<td>2</td>
<td>OWF subexp iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>DKR’15</td>
<td>n</td>
<td>4</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>GP’15</td>
<td>n</td>
<td>2</td>
<td>TDP subexp. iO</td>
<td>- (even in HBC case)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>This work</td>
<td>n</td>
<td>2</td>
<td>OWF iO</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Subsequent work**

<table>
<thead>
<tr>
<th></th>
<th># of parties</th>
<th># of rounds</th>
<th>assumptions</th>
<th>global CRS</th>
<th>randomness hiding</th>
<th>supports RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPV’16</td>
<td>2</td>
<td>2</td>
<td>hardware tokens OWF</td>
<td>no CRS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPV’16</td>
<td>2 (n)</td>
<td>2 (const)</td>
<td>NCE*</td>
<td>no CRS</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Part I: HBC protocol with global CRS
First attempt

\[ x_i = \text{Enc}_{PK}(x_i) \]
First attempt

\[ x_i = \text{Enc}_{PK}(x_i) \]

- decrypt each using SK
- output \( f(x_1, \ldots, x_n) \)
First attempt

\[ x_i = \text{Enc}_{PK}(x_i) \]

- decrypt each using SK
- output \( f(x_1, \ldots, x_n) \)

- decrypt each using SK
- output \( f(x_1, \ldots, x_n) \)

\[ y = f(x_1, x_2, \ldots, x_n) \]
First attempt

\[ x_i = \text{Enc}_{PK}(x_i) \]

\[ x_1 \quad x_2' \quad \ldots \quad x_n \]

- decrypt each using SK
- output \( f(x_1, \ldots, x_n) \)

\[ y' = f(x_1, x_2', \ldots, x_n) \]
Our protocol

\[ x_i = \text{Commit}(x_i; r_i) \]

\[ x_i r_i = \text{Enc}_{PK}(x_i||r_i||\quad\quad\quad\quad) \]
Our protocol

\[ x_i = \text{Commit}(x_i; r_i) \]

\[ x_i r_i = \text{Enc}_{PK}(x_i||r_i||... ) \]

- decrypt each using SK
- check that are the same in each
- verify each
- output \( f(x_1, ..., x_n) \)
Our protocol

\[ x_i = \text{Commit}(x_i; r_i) \]

\[ x_i r_i = \text{Enc}_{PK}(x_i \| r_i \| \text{...}) \]

- decrypt each using SK
- check that are the same in each
- verify each
- output \( f(x_1, \ldots, x_n) \)

\[ y = f(x_1, x_2, \ldots, x_n) \]
Our protocol

\[ x_i = \text{Commit}(x_i; r_i) \]

\[ x_i r_i = \text{Enc}_{PK}(x_i || r_i || \ldots ) \]

- decrypt each using SK
- check that are the same in each
- verify each
- output \( f(x_1, \ldots, x_n) \)
Our protocol

\[ x_i \] = Commit(x_i; r_i)

\[ x_{i} r_i \] = Enc_{PK}(x_i || r_i || \ldots )

- decrypt each \( x_i \) using SK
- check that \( x_i \) are the same in each
- verify each
- output \( f(x_1, \ldots, x_n) \)

The adversary cannot mix and match encryptions.
Wanted: Encryption

**Problem:**
cannot use security of encryption since SK is in the program

- decrypt each $x_i$ using SK
- check that $x_i$ are the same in each
- verify each
- output $f(x_1, \ldots, x_n)$
Wanted: Encryption

Problem:
cannot use security of encryption since SK is in the program

- decrypt each $x_i$ using SK
- check that $x_i$ are the same in each
- verify each
- output $f(x_1, \ldots, x_n)$
Wanted: Encryption

Problem:
cannot use security of encryption since SK is in the program

- decrypt each using SK
- check that are the same in each
- verify each
- output $f(x_1, \ldots, x_n)$
Problem:
cannot use security of encryption since SK is in the program

Wanted: Encryption

- decrypt each using SK
- check that are the same in each
- verify each
- output $f(x_1, \ldots, x_n)$
Wanted: Encryption

Problem:
cannot use security of encryption since SK is in the program

- decrypt each using SK
- check that are the same in each
- verify each
- output f(x₁, ..., xₙ)

Problem:
cannot use security of encryption since SK is in the program

PK, SK

Adv

PK

m

GM

PK, SK

c = Enc(m) or simulated c, SK{c}
**Problem:**
cannot use security of encryption since SK is in the program

**Solution:**
Puncturable randomized encryption (PRE) (from iO and OWFs)

**Property:**
simulation-secure even when almost all SK is known
Achieving globality and full adaptive security

Simulation: not global
Achieving globality and full adaptive security

Simulation: not global

Solution: Modify the protocol to sample PK, during the execution.

PK  SK{                  }

PK

x₁ x₂ ... xₙ

x₁ x₂ ... xₙ

KSW’14
CPR’16
Achieving globality and full adaptive security

Simulation: not global

Solution: Modify the protocol to sample PK, SK during the execution.
How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
$$F(x_1, \ldots, x_n; r) = \text{garbled } f, \text{garbled } x_1, \ldots, x_n.$$
How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
\[ F(x_1, \ldots, x_n; r) = \text{garbled } f, \text{garbled } x_1, \ldots, x_n. \]

Any MPC protocol + RAM-efficient garbling (e.g. CH’16) = RAM-efficient protocol
How to make the protocol RAM-efficient

**Ishai-Kushilevitz paradigm:**
- Use MPC to evaluate garbling:
  \[ F(x_1, \ldots, x_n; r) = \text{garbled } f, \text{garbled } x_1, \ldots, x_n. \]

Any MPC protocol + RAM-efficient garbling (e.g. CH’16) = RAM-efficient protocol

Only works for n-1 corruptions!
How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
\[ F(x_1, \ldots, x_n; r) = \text{garbled } f, \text{garbled } x_1, \ldots, x_n. \]

Any MPC protocol + RAM-efficient garbling (e.g. CH’16) = RAM-efficient protocol

Only works for n-1 corruptions!
For full adaptive security:

randomness-hiding MPC protocol + RAM-efficient garbling (e.g. CH’16) = RAM-efficient protocol
Our results:

- **Semi-honest case**
  - 2 round fully adaptive MPC with nice properties (randomness-hiding, RAM-efficient, global CRS...)

- **Malicious case**
  - NIZK with RAM efficiency

Plug into GP’15

GP’15 2 round fully adaptive MPC becomes **RAM-efficient**
Part II: Making GP’15 RAM-efficient
Part II: Making GP’15 RAM-efficient

Any randomness-hiding MPC protocol + RAM-efficient garbling (e.g. CH’16) = RAM-efficient protocol

GP’15 doesn’t hide randomness
Malicious case: achieving RAM-efficiency

**Theorem** (Garg-Polychroniadou’15):
subexponential IO+TDPs $\rightarrow$ malicious MPC (2 round, fully adaptive)
Malicious case: achieving RAM-efficiency

**Theorem** (Garg-Polychroniadou’15):
subexponential IO for RAM +TDPs → malicious MPC for RAM? (2 round, fully adaptive)
Malicious case: achieving RAM-efficiency

**Theorem** (Garg-Polychroniadou’15):
subexponential IO for RAM + TDPs + statistically-sound NIZK for RAM → malicious MPC for RAM (2 round, fully adaptive)
RAM-efficient NIZK

f(x):
For i = 1... 100000000 do {
}

RAM-efficient NIZK

\[
f(x) : \\
\text{For } i = 1 \ldots 100000000 \text{ do } \{ \\
\}
\]

- \(|\text{proof}| \sim |f|_{\text{RAM}}\)
Prior work on RAM-efficient NIZK

\[ f(x): \]
\[ \text{For } i = 1 \ldots 100000000 \text{ do } \{ \}
\]

- \(|\text{proof}| \sim |f|_{\text{RAM}}\) - done

[Gen09, Gro11]:
- \(|\text{proof}| \sim |w|\)
Prior work on RAM-efficient NIZK

\[ f(x) : \]
\[ \text{For } i = 1 \ldots 100000000 \text{ do } \{ \} \]

- \( |\text{proof}| \sim |f|_{\text{RAM}} \) - done

[Gen09, Gro11]:
- \( |\text{proof}| \sim |w| \)
- Verify ~ circuit complexity of \( f \)

Obfuscated program in GP’15:

Verify proof for “\( f(x_1 \ldots x_n) = y, \ldots \)”
...
...
Prior work on RAM-efficient NIZK

\[ f(x) : \]
\[ \text{For } i = 1 \ldots 100000000 \text{ do } \{} \]
\[ \} \]

- $|\text{proof}| \sim |f|_{\text{RAM}}$ - done
- Verification complexity $\sim$ RAM complexity of $f$ - ?

[Gen09, Gro11]:
- $|\text{proof}| \sim |w|
- Verify $\sim$ circuit complexity of $f$

Obfuscated program in GP'15:

Verify proof for “$f(x_1 \ldots x_n) = y, \ldots$”

...
Malicious case

**Theorem** (Garg-Polychroniadou’15):
subexponential IO for RAM + TDPs+ statistically-sound NIZK for RAM
→ malicious MPC for RAM (2 round, fully adaptive)

**Theorem** (Our work):
Garbled RAM + NIZK proofs for circuits → statistically-sound NIZK for RAM.
Malicious case

**Theorem** (Garg-Polychroniadou’15): subexponential IO for RAM + TDPs+ statistically-sound NIZK for RAM \rightarrow malicious MPC for RAM (2 round, fully adaptive)

**Theorem** (Our work): Garbled RAM + NIZK proofs for circuits \rightarrow statistically-sound NIZK for RAM.

**Corollary:** Subexp. iO+TDPs \rightarrow malicious MPC for RAM (2 round, fully adaptive)
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 1

Convince that $\exists w$ such that $R(x; w) = 1$
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 1

Convince that $\exists w$ such that $R(x; w) = 1$

Prover

$x \in L$

$w$

Verifier

$x \in L$

garbled RAM:
- allows to compute $R(x; w)$
- hides $R, x, w$
- RAM-efficient
NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that \( \exists w \) such that \( R(x; w) = 1 \)

Proof \( \pi = R(*,*) x, w \)

Prover
\[ x \in L \]
\[ w \]

Verifier
\[ x \in L \]

Accept if \( \text{Eval}(R(*,*) x, w) = 1 \)

garbled RAM:
- allows to compute \( R(x; w) \)
- hides \( R, x, w \)
- RAM-efficient
NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that \( \exists w \) such that \( R(x; w) = 1 \)

Proof \( \pi = R(*,*) \)

\[ x, w \]

Prover

Verifier

\[ x \in L \]

\[ w \]

\[ x \in L \]

\[ R \rightarrow \]

\[ x, w \rightarrow \]

\[ R(*,*) \]

\[ x, w \]

garbled RAM:
- allows to compute \( R(x; w) \)
- hides \( R, x, w \)
- RAM-efficient

Accept if \( \text{Eval}(R(*,*) \ x, w) = 1 \)

- Verifier doesn't learn anything about \( w \)
NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that $\exists w$ such that $R(x; w) = 1$

Proof $\pi = R(\*,\*)$  
x, w

Verifier doesn’t learn anything about $w$

- Malicious prover can garble $R \equiv 1$

Prover

$x \in L$

w

Garbled RAM:
- allows to compute $R(x; w)$
- hides $R, x, w$
- RAM-efficient

Verifier

$x \in L$

Accept if $\text{Eval}(R(\*,\*) x, w) = 1$

- Verifier doesn’t learn anything about $w$
- Malicious prover can garble $R \equiv 1$
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 2

Convince that $\exists w$ such that $R(x; w) = 1$

NIZK proof: “garbling done correctly, for correct $R$ and $x$”

Accept if $\text{Eval}(R(*,*)_{x, w}) = 1$

and if NIZK verifies.

$x \in L$

$w$

$R \rightarrow R(*,*)$

$x, w \rightarrow x, w$
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 2

Convince that $\exists w$ such that $R(x; w) = 1$

NIZK proof: “garbling done correctly, for correct $R$ and $x$”

Accept if $\text{Eval}(R(*,*), x, w) = 1$

and if NIZK verifies.
NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that \( \exists w \) such that \( R(x; w) = 1 \)

NIZK proof: “garbling done correctly, for correct \( R \) and \( x \)”

Accept if \( \text{Eval}(\text{R}(\ast, \ast), x, w) = 1 \)

and if NIZK verifies.

- Verifier doesn’t learn anything about \( w \)
NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

NIZK proof: “garbling done correctly, for correct R and x”

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn’t learn anything about w
- garbling: for most random coins of garbling, correctness holds.
NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that \( R(x; w) = 1 \)

NIZK proof: “garbling done correctly, for correct R and x”

- Verifier doesn’t learn anything about w
- Garbling: for most random coins of garbling, correctness holds.
- What if for some r garbling always garbles \( R \equiv 1 \)

Accept if \(? R(*,*) x, w \) = 1 and if NIZK verifies.
NIZK + Garbled RAM \rightarrow \text{NIZK for RAM}

Attempt 2

Convince that \( \exists w \) such that \( R(x; w) = 1 \)

NIZK proof: “garbling done correctly, for correct \( R \) and \( x \)”

- \( x \in L \)
- \( w \)
- \( R \rightarrow R(*,*) \)
- \( x, w \rightarrow x, w \)
- \( R(*,*) \)
- \( x, w \)
- \( x \in L \)

Accept if \( \text{Eval}(R(*,*) x, w) = 1 \)
and if NIZK verifies.

- Verifier doesn’t learn anything about \( w \)
- No coin tossing - need \text{perfectly correct} garbled RAM
- Currently do not have garbled RAM with perfect correctness
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 2

Convince that $\exists w$ such that $R(x; w) = 1$

NIZK proof: “garbling done correctly, for correct $R$ and $x$”

- Verifier doesn’t learn anything about $w$
- CH15 garbled RAM satisfies \textbf{perfect correctness with abort} - enough
NIZK + Garbled RAM $\rightarrow$ NIZK for RAM

Attempt 2

Convince that $\exists w$ such that $R(x; w) = 1$

NIZK proof: “garbling done correctly, for correct $R$ and $x$”

- Verifier doesn’t learn anything about $w$
- CH15 garbled RAM satisfies **perfect correctness with abort** - enough
- evaluator either gets **correct** output, or **rejects**

Accept if $\text{Eval}(R(*,*), x, w) = 1$ and if NIZK verifies.
Summary: two round adaptively secure protocols

Semi-honest case:
- global CRS
- supports RAM
- randomness-hiding (e.g. $N = pq$)

Malicious case (GP15 + our RAM efficient NIZK):
- RAM-efficient
Questions?