SNARGs and BARGs from LWE

Arka Rai Choudhuri
Johns Hopkins University

Abhishek Jain
Johns Hopkins University

Zhengzhong Jin
Johns Hopkins University
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

\[ \mathcal{M}, x \]

within \( T \) steps

\[ \mathcal{M}, x \]
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

\[ M, x \]

\[ M, x \]

\[ x \rightarrow M \rightarrow \text{accept} \]

within \( T \) steps

wants to delegate computation to
Succinct Non-Interactive Arguments (SNARGs)
Succinct **Non-Interactive Arguments** (**SNARGs**)

- **Common Reference String (CRS)**
- \( M, x \) input
- \( Π \) verification process
- \( M, x \) output

\[ M \text{ accepts } x \text{ within } T \text{ steps} \]

\( Π \) is publicly verifiable
**Succinct Non-Interactive Arguments (SNARGs)**

A Common Reference String (CRS) is used as input to a non-interactive argument system. The interaction proceeds in a single round, with the verifier running in polylogarithmic time and the proof being publicly verifiable. The system accepts within $T$ steps for a given input $x$.
Succinct Non-Interactive Arguments (SNARGs)

No PPT can produce accepting $\Pi$ if

$\mathcal{M}, x$ is publicly verifiable within $T$ steps

Verifier running time: $\text{polylog}(T)$
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

\[ M, x \]

Verifier running time: \( \text{polylog}(T) \)

\( \Pi \) is publicly verifiable

No PPT can produce accepting \( x, \Pi \) if

\[ x \rightarrow M \rightarrow \text{accept} \]

within \( T \) steps

\[ x \rightarrow M \rightarrow \text{accept} \]

within \( T \) steps

\( M, x \)
Succinct Non-Interactive Arguments (SNARGs)

What kind of computation can we hope to delegate based on standard assumptions?

- Nondeterministic computation (NP)?
  - Unlikely! [Gentry-Wichs'11]

- Deterministic computation?
  - Sub-classes of NP?
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

What kind of computation can we hope to delegate based on standard assumptions?
- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs’11]

Verifier running time: polylog(T)
Π is publicly verifiable
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

Verifier running time: polylog(T)

\( \mathcal{M}, x \) \quad \Pi \quad \mathcal{M}, x

What kind of computation can we hope to delegate based on standard assumptions?
- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs’11]
- Deterministic polynomial-time computation (P)?
Succinct Non-Interactive Arguments (SNARGs)

What kind of computation can we hope to delegate based on standard assumptions?
- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs’11]
- Deterministic polynomial-time computation (P)?
- Sub-classes of NP?
Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

What kind of computation can we hope to delegate based on standard assumptions?

- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs’11]
- Deterministic polynomial-time computation (P)?
- Sub-classes of NP?
Non-Interactive Batch Arguments (BARGs)

\[
\Pi(C, x_1, \ldots, x_k) \in \text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}
\]

\[
\forall i \in [k], (C, x_i) \in \text{SAT}
\]

\(\Pi\) is publicly verifiable
Non-Interactive Batch Arguments (BARGs)

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]
Prior Works
Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali’94, Groth’10, Lipmaa’12, Damgård-Faust-Hazay’12, Gennaro-Gentry-Parno-Raykova’13,
Bitansky-Chiesa-Ishai-Ostrovsky-Paneth’13, Bitansky-Canetti-Chiesa-Tromer’13, Bitansky-Canetti-Chiesa-
Goldwasser-Lin-Rubinstein-Tromer’17]

Some works can delegate NP
Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali’94, Groth’10, Lipmaa’12, Damgård-Faust-Hazay’12, Gennaro-Gentry-Parno-Raykova’13,
Bitansky-Chiesa-Ishai-Ostrovsky-Paneth’13, Bitansky-Canetti-Chiesa-Tromer’13, Bitansky-Canetti-Chiesa-
Goldwasser-Lin-Rubinstein-Tromer’17]

“Less standard” assumptions

[Canetti-Holmgren-Jain-Vaikuntanathan’15, Koppula-Lewko-Waters’15, Bitansky-Garg-Lin-Pass-Telang’15,
Canetti-Holmgren’16, Ananth-Chen-Chung-Lin-Lin’16, Chen-Chow-Chung-Lai-Lin-Zhou’16, Paneth-
Rothblum’17, Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs’19, Kalai-Paneth-Yang’19]
Prior Works

Non-falsifiable assumptions/ Random oracle model

“Less standard” assumptions

Designated Verifier (standard assumptions)
[Kalai-Raz-Rothblum’13, Kalai-Raz-Rothblum’14, Kalai-Paneth’16, Brakerski-Holgren-Kalai’17, Badrinarayanan-Kalai-Khurana-Sahai-Wichs’18, Holmgren-Rothblum’18, Brakerski-Kalai’20]
Our Results

BARGs

<table>
<thead>
<tr>
<th>Proof size</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{O}(</td>
<td>C</td>
</tr>
</tbody>
</table>

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman

SAT = \{(C, x) | \exists w \text{ s.t. } C(x, w) = 1\}

\forall i \in [k], (C, x_i) \in SAT
## Our Results

### BARGs

| [C-Jain-Jin’21a] | $\tilde{O}(|C| + \sqrt{k|C|})$ | QR + (LWE/sub-exp DDH) |
|------------------|-------------------------------|------------------------|
| [C-Jain-Jin’21b] | $\text{poly}(\log k, \log|C^*|, |w|)$ | LWE                    |

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman

| SAT = \{((C,x) | \exists w s.t. C(x,w) = 1) | ∀i ∈ [k], (C,x_i) ∈ SAT |
Our Results

**SNARGs**

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>LWE</td>
</tr>
</tbody>
</table>

[C-Jain-Jin’21b]
Our Results

### SNARGs

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[C-Jain-Jin’21b]</td>
<td>RAM</td>
<td>LWE</td>
</tr>
</tbody>
</table>

Previously best known: [Jawale-Kalai-Khurana-Zhang’21] for depth bounded computation based on sub-exponential hardness of LWE.
Key Insights
Fiat-Shamir (FS) Methodology
Fiat-Shamir (FS) Methodology

$\alpha$ $\beta$ $\gamma$

$\beta$ is a random string
Fiat-Shamir (FS) Methodology

Prover($x$) \rightarrow Verifier($x$)

$\alpha$  $\beta$  $\gamma$

$\beta$ is a random string

Prover($x$)  $\beta = h(x, \alpha)$  Verifier($x$)

[Fiat-Shamir’86]
**Fiat-Shamir (FS) Methodology**

\[ \beta \text{ is a random string} \]

**FS methodology is secure for certain protocols under a variety of assumptions (via correlation intractable hash functions)**

Fiat-Shamir (FS) Methodology

$\alpha \beta \gamma$

$\beta$ is a random string

$\beta = h(x, \alpha)$

FS methodology is secure for certain protocols under a variety of assumptions (via correlation intractable hash functions)
Fiat-Shamir (FS) Methodology

FS methodology is secure for certain protocols under a variety of assumptions (via correlation intractable hash functions)

Proven secure if starting with statistically secure interactive protocols (interactive proofs).
Fiat-Shamir (FS) Methodology

FS methodology is secure for certain protocols under a variety of assumptions (via correlation intractable hash functions).

Proven secure if starting with statistically secure interactive protocols (interactive proofs).

No known interactive proofs for batch NP or delegating deterministic polynomial-time computation.
Dual-Mode Interactive Batch Arguments
Dual-Mode Interactive Batch Arguments

Computational security
Dual-Mode Interactive Batch Arguments

CRS generation

Normal mode

CRS

Trapdoor mode at index $i$

$C, x_1, \ldots, x_i, \ldots, x_k$

computational security
Dual-Mode Interactive Batch Arguments

CRS generation

Normal mode

Trapdoor mode at index $i$

$\approx$ computationally indistinguishable

Computational security
Dual-Mode Interactive Batch Arguments

\[ C, x_1, \ldots, x_i, \ldots, x_k \]

**Normal mode**

**CRS generation**

**Trapdoor mode** at index \( i \)

**CRS**

**CRS_i**

\[ C, x_1, \ldots, x_i, \ldots, x_k \]

\[ C, x_1, \ldots, x_i, \ldots, x_k \]

*computational security*
Dual-Mode Interactive Batch Arguments

CRS generation

Normal mode

Trapdoor mode at index $i$

$C, x_1, \ldots, x_i, \ldots, x_k$

computational security

$C, x_1, \ldots, x_i, \ldots, x_k$

$C, x_1, \ldots, x_i, \ldots, x_k$

statistical security at $i$

$C, x_1, \ldots, x_i, \ldots, x_k$
Dual-Mode Interactive Batch Arguments

CRS generation

Normal mode

Trapdoor mode at index $i$

$C, x_1, \ldots, x_i, \ldots, x_k$

$C, x_1, \ldots, x_i, \ldots, x_k$

computational security

$C, x_1, \ldots, x_i, \ldots, x_k$

$C, x_1, \ldots, x_i, \ldots, x_k$

statistical security at $i$

Even unbounded $\mathcal{A}$ cannot make $\mathcal{B}$ accept if $(C, x_i) \notin \text{SAT}$
Security Intuition

$C, x_1, \ldots, x_i, \ldots, x_K$, $C, x_1, \ldots, x_i, \ldots, x_K$
Security Intuition

Switch to trapdoor mode at $i$
Security Intuition

Switch to trapdoor mode at $i$

Rely on FS transformation
Security Intuition

Non-adaptive security

Switch to trapdoor mode at $i$

Rely on FS transformation
Dual Mode Batch Argument

 Protocol Template

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]

\[ \text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \]
Dual Mode Batch Argument
Protocol Template

\[ \begin{align*}
&\forall i \in [k], (C, x_i) \in \text{SAT} \\
&\text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \\
\end{align*} \]
Dual Mode Batch Argument

Protocol Template

\[ w_k \]
\[ w_i \]
\[ w_1 \]

commitment key $K$

\[ SAT = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \]

\[ \forall i \in [k], (C, x_i) \in SAT \]
Dual Mode Batch Argument

Protocol Template

commitment key $K$

SAT = \{ (C, x) | \exists w \text{ s.t. } C(x, w) = 1 \}

\forall i \in [k], (C, x_i) \in SAT
Dual Mode Batch Argument

Protocol Template

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]
Dual Mode Batch Argument

Protocol Template

SAT = \{(C,x) | \exists w s.t. C(x,w) = 1\}

∀i ∈ [k], (C, x_i) ∈ SAT
Dual Mode Batch Argument

Protocol Template

commitment key $K$

information theoretic component

$\forall i \in [k], (C, x_i) \in SAT$

$SAT = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}$

$\{\text{open } f(w_i)\}_{i \in [k]}$
Dual Mode Batch Argument
Protocol Template

\[
\forall i \in [k], (C, x_i) \in \text{SAT}
\]

\[
\text{SAT} = \{(C, x) | \exists w \text{ s.t. } C(x, w) = 1\}
\]
Dual Mode Batch Argument
Protocol Template

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]

SAT = \{ (C,x) \mid \exists w \text{ s.t. } C(x,w) = 1 \}

Somewhere Statistically Binding (SSB) Commitment Scheme
Dual Mode Batch Argument

Protocol Template

 Trapdoor mode

commitment key $K_i^*$

information theoretic component

SAT = \{(C, x) \mid \exists w . \text{s.t. } C(x, w) = 1\}

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme
Dual Mode Batch Argument

Protocol Template

\[ w_1 \quad \ldots \quad w_k \]

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]

Somewhere Statistically Binding (SSB) Commitment Scheme

\[ \text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \]

Commitment key \( K_i^* \)

\[ \text{information theoretic component} \]

\[ \{\text{open } f(w_i)\}_{i \in [k]} \]
Dual Mode Batch Argument

Protocol Template

\[ w_1, \ldots, w_k \]
\[ c_1, \ldots, c_j, \ldots, c_m \]

**Trapdoor mode**

commitment key \( K_i^* \)

\[ \text{poly}(m, \log k) \]

information theoretic component

\[ \{\text{open } f(w_i)\}_{i \in [k]} \]

\[ SAT = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \]

\[ \forall i \in [k], (C, x_i) \in SAT \]

Somewhere Statistically Binding (SSB) Commitment Scheme
Dual Mode Batch Argument

Protocol Template

\[ w_1, \ldots, w_k \]

Commitment key \( K_i^* \)

Trapdoor mode

\[ \forall i \in [k], (C, x_i) \in \text{SAT} \]

Somewhere Statistically Binding (SSB) Commitment Scheme

\[ \text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\} \]

Information theoretic component

\[ \{\text{open } f(w_i)\}_i \in [k] \]
Dual Mode Batch Argument

Protocol Template

\[ \forall i \in [k], (C, x_i) \in SAT \]

Somewhere Statistically Binding (SSB) Commitment Scheme

Needs to be Fiat-Shamir friendly.
Based on LWE/sub-exp DDH
Dual Mode Batch Argument

Protocol Template

SAT = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}

∀i ∈ [k], (C, x_i) ∈ SAT

Somewhere Statistically Binding (SSB) Commitment Scheme

Needs to be Fiat-Shamir friendly.
Based on LWE/sub-exp DDH

SSB with appropriate opening to f

[CJJ’21a]: (with additional properties) based on QR
[CJJ’21b]: based on LWE
### BARGs

<table>
<thead>
<tr>
<th>Proof size</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{O}(</td>
<td>C</td>
</tr>
<tr>
<td>$\text{poly}(\log k, \log</td>
<td>C^*</td>
</tr>
</tbody>
</table>

### SNARGs

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>LWE</td>
</tr>
</tbody>
</table>

---

Thank you. Questions?

Arka Rai Choudhuri
achoud@cs.jhu.edu
SNARGs for Polynomial-time Computation

\[ x \xrightarrow{\mathcal{M}} \text{accept} \]

within \(T\) steps
SNARGs for Polynomial-time Computation

\[ x \xrightarrow{\text{st}_0} \]

\[ x \stackrel{\mathcal{M}}{\xrightarrow{\text{accept}}} \text{within } T \text{ steps} \]
**SNARGs** for Polynomial-time Computation

$x \xrightarrow\text{single deterministic step} st_0 \xrightarrow\text{single deterministic step} st_1$

$x \xrightarrow M \text{ accept within } T \text{ steps}$
SNARGs for Polynomial-time Computation

\[ x \left\{ \begin{array}{l} s_{t_0} \\ s_{t_1} \\ \vdots \\ s_{t_T} \end{array} \right\} \rightarrow \text{accept} \]

within \( T \) steps
**SNARGs** for Polynomial-time Computation

\[
x \overset{\text{single deterministic step}}{\rightarrow} \quad \ldots \quad \overset{\text{1}}{\rightarrow} \quad \text{accept}
\]

\[
\text{Prove for every } i \in [0, \ldots, T - 1] \text{ that } \text{st}_i \rightarrow \text{st}_{i+1} \text{ is the correct transition.}
\]
SNARGs for Polynomial-time Computation

For every $i \in [0, \ldots, T-1]$
1. Commitment contains $st_i$ and $st_{i+1}$
2. Valid transition $st_i \rightarrow st_{i+1}$
**SNARGs** for Polynomial-time Computation

For every $i \in [0, ..., T - 1]$

1. Commitment contains $st_i$ and $st_{i+1}$
2. Valid transition $st_i \rightarrow st_{i+1}$
SNARGs for Polynomial-time Computation

Local Soundness

$i$-th state transition correct
**SNARGs** for Polynomial-time Computation

- Commitment key $K_{i,i+1}^*$
- For every $i \in [0, ..., T-1]$
  1. Commitment contains $st_i$ and $st_{i+1}$
  2. Valid transition $st_i \rightarrow st_{i+1}$

**BARG**

**Local Soundness**
- $i$-th state transition correct

**Global Soundness**
- Local soundness at all $i$
SNARGs for Polynomial-time Computation

Commitment key $K_{i,i+1}^*$

For every $i \in [0, \ldots, T - 1]$
1. Commitment contains $st_i$ and $st_{i+1}$
2. Valid transition $st_i \rightarrow st_{i+1}$

Local Soundness
$i$-th state transition correct

Global Soundness
Local soundness at all $i$