Finding a Nash Equilibrium is No Easier than Breaking Fiat-Shamir

Arka Rai Choudhuri    Pavel Hubáček    Chethan Kamath
Krzysztof Pietrzak    Alon Rosen    Guy Rothblum

JHU Theory Seminar
What is the **Cryptographic Hardness** in PPAD?
The Story Line
The Story Line

Games
The Story Line

Games

Complexity
The Story Line

Games  Complexity  Crypto
The Story Line

Games  Complexity  Crypto
The Story Line

Games  Complexity  Crypto
Game Theory and Nash Equilibrium
Game Theory and Nash Equilibrium
Game Theory and Nash Equilibrium

<table>
<thead>
<tr>
<th></th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>1 \ -1</td>
<td>-1 \ 1</td>
</tr>
<tr>
<td>Right</td>
<td>-1 \ 1</td>
<td>1 \ -1</td>
</tr>
</tbody>
</table>
Game Theory and Nash Equilibrium

[Nash’51]: A (mixed) equilibrium always exists
How hard is finding a Nash Equilibrium?
How hard is finding a Nash Equilibrium?
Not NP-hard unless NP=coNP
[ Megiddo-Papadimitriou’89 ]
The Class PPAD [Papadimitriou’94]
The Class PPAD [Papadimitriou’94]
The Class PPAD [Papadimitriou’94]

Defined through its complete problem:
END OF THE LINE (EOL)
End of the Line (EOL)
End of the Line (EOL)

Directed graph
End of the Line (EOL)

Directed graph

Every node has in and out degree $\leq 1$
End of the Line (EOL)

Directed graph

Every node has in and out degree $\leq 1$
End of the Line (EOL)

Directed graph

Every node has in and out degree $\leq 1$

How easy is it to solve this?
End of the Line (EOL)
End of the Line (EOL)
End of the Line (EOL)

$$0^n \rightarrow \cdots \rightarrow P(v) \rightarrow v \rightarrow S(v) \rightarrow \cdots$$

$v$ \rightarrow \text{Successor} \rightarrow S(v)

$v$ \rightarrow \text{Predecessor} \rightarrow P(v)$
PPAD and NASH [Papadimitriou’94]
PPAD and NASH [Papadimitriou’94]
PPAD and NASH [Papadimitriou’94]

[Daskalakis-Goldberg-Papadimitriou 05],
[Chen-Deng 05]
The Story Line

Games        Complexity        Crypto
Hard-on-Average $L \in \text{NP}$
[HNY'17]
Hard-on-Average $L \in \text{NP}$
[HNY'17]

Factoring
[Bur-Op'06, Jer'12]
Hard-on-Average $L \in \text{NP}$ [HNY'17]

Factoring [Bur-Op’06, Jer’12]

One-way Perm. [Pap’94]
Hard-on-Average $L \in \text{NP}$

[HNY’17]

Factoring

[Bur-Op’06, Jer’12]

One-way Perm.

[Par’94]

Collision-resistant Hash Functions
Hard-on-Average $L \in \text{NP}$
[HNY’17]

Factoring
[Bur-Op’06, Jer’12]

One-way Perm.
[Pap’94]

Collision-resistant Hash Functions

Obfuscation/FE
[AKV’04, BPR’15, GPS’16]
Hard-on-Average $L \in \text{NP}$
[HNY'17]

Factoring
[Bur-Op'06, Jer'12]

Obfuscation/FE
[HY'17, KS'17]

One-way Perm.
[Pap'94]

Collision-resistant Hash Functions

Obfuscation/FE
[AKV'04, BPR'15, GPS'16]
Hard-on-Average $L \in \text{NP}$
[HNY'17]

Factoring
[Bur-Op'06, Jer'12]

One-way Perm.
[Pap'94]

Obfuscation/FE
[HY'17, KS'17]

Collision-resistant Hash Functions

Today
EOL Hardness from Obfuscation

[Bitansky-Paneth Rosen’15]
EOL Hardness from Obfuscation

[Bitansky-Paneth Rosen’15]

PPAD/CLS hardness can be based on indistinguishability obfuscation (iO)
Sink of Verifiable Line (SVL)

[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]
Sink of Verifiable Line (SVL)

[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Sink of Verifiable Line (SVL)
[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Sink of Verifiable Line (SVL)

[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Sink of Verifiable Line (SVL)

[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.

Based on reversible computation [Bennett’84]
Sink of Verifiable Line (SVL)
[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then Predecessor is for free.
Based on reversible computation [Bennett’84]

SVL hard \rightarrow EOL hard

[AKV’04, BPR’15]
Sink of Verifiable Line (SVL) 
[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Based on reversible computation [Bennett’84]

---

iO hard \[\rightarrow\] SVL hard \[\rightarrow\] EOL hard

[BPR’15, GPS’16] \[\rightarrow\] [AKV’04, BPR’15]
Sink of Verifiable Line (SVL)  
[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Based on reversible computation [Bennett’84]

---

\[ i - 1, \sigma_{i-1} \rightarrow i, \sigma_i \rightarrow i + 1, \sigma_{i+1} \rightarrow N, \sigma_N \]

---

iO hard \[ \rightarrow \] SVL hard \[ \rightarrow \] EOML hard

[BPR’15,GPS’16] \[ \rightarrow \] [HY’17]
Sink of Verifiable Line (SVL)

[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.

Based on reversible computation [Bennett’84]

---

**Equations:**

\[ i - 1, \sigma_{i-1} \quad i, \sigma_i \quad i + 1, \sigma_{i+1} \]

**Diagram:**

- 1, \sigma_1
- \ldots
- i - 1, \sigma_{i-1}
- i, \sigma_i
- i + 1, \sigma_{i+1}
- \ldots
- N, \sigma_N

**Security Levels:**

- quasi-poly private key FE + injective OWF hard
- SVL hard
- EOML hard

**References:**

[KS’17] [HY’17]
Sink of Verifiable Line (SVL)
[Abbott-Kane-Valiant’04, Bitansky-Paneth Rosen’15]

If the path is verifiable, then **Predecessor** is for free.
Based on reversible computation [Bennett’84]

- 1, $\sigma_1$
- $i - 1, \sigma_{i-1}$
- $i, \sigma_i$
- $i + 1, \sigma_{i+1}$
- $N, \sigma_N$

quasi-poly private key FE + injective OWF hard → SVL hard → EOC

[KS’17] [HY’17]

“real cryptographers don’t use iO”
Our Result
Our Result

CLS is as hard as breaking soundness of Fiat-Shamir when applied to the sumcheck protocol.
Our Construction

SVL Is No Easier Than Breaking Fiat-Shamir
Basic Idea

\[ 1, \sigma_1 \rightarrow 2, \sigma_2 \rightarrow \cdots \rightarrow i - 1, \sigma_{i-1} \rightarrow i, \sigma_i \rightarrow i + 1, \sigma_{i+1} \rightarrow \cdots \rightarrow N, \sigma_N \]
Basic Idea

Reduce to SVL from #SAT
Basic Idea

Reduce to SVL from \#SAT

$$\varphi(z_1, \ldots, z_n) \rightarrow (S, V, N)$$
Basic Idea

Reduce to SVL from #SAT
Basic Idea

Reduce to SVL from \#SAT

$$\varphi(z_1, \ldots, z_n)$$

$$(S, V, N)$$

$$2^n$$

$\# \bar{z} \in \{0, 1\}^n$$ such that $$\varphi(\bar{z}) = 1$$
Basic Idea

1, σ₁ → 2, σ₂ → i - 1, σᵢ₋₁ → i, σᵢ → i + 1, σᵢ₊₁ → N, σₙ
Basic Idea

1, \sigma_1 \rightarrow 2, \sigma_2 \rightarrow \cdots \rightarrow i - 1, \sigma_{i-1} \rightarrow i, \sigma_i \rightarrow i + 1, \sigma_{i+1} \rightarrow \cdots \rightarrow N, \sigma_N

i, y_i, \pi_i

N, y_N, \pi_N
Basic Idea

$1, y_1, [\pi_1]$

$\ldots$

$i, y_i, [\pi_i]$

$\ldots$

$N, y_N, [\pi_N]$
Basic Idea

\[ V(i, y_i, \pi_i) = \text{ACCEPT} \iff y_i \text{ is the # of } \tilde{z} \leq i \text{ such that } \varphi(\tilde{z}) = 1 \]
Basic Idea

\[ S(i, y_i, \pi_i) = i + 1, y_{i+1}, \pi_{i+1} \]
Relaxed SVL (rSVL)
Relaxed SVL (rSVL)
Relaxed SVL (rSVL)

Should be hard to find "off-path" \( j, y_j, \tilde{\pi}_j \) such that

\[
V \left( j, y_j, \tilde{\pi}_j \right) = \text{ACCEPT}
\]
Relaxed SVL (rSVL)

Solving an instance of rSVL
Relaxed SVL (rSVL)

Solving an instance of rSVL
solve #SAT instance $\varphi$
Relaxed SVL (rSVL)

Solving an instance of rSVL
solve \#SAT instance \( \varphi \)
break (computational) soundness of \( \Pi \)
Challenges

1, \( y_1, \pi_1 \)

.............

i, \( y_i, \pi_i \)

.............

N, \( y_N, \pi_N \)
Several Challenges:

$1, y_1, \pi_1$

$i, y_i, \pi_i$

$N, y_N, \pi_N$
Several Challenges:

Proof size has to be polynomial
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]

Arithmetization
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]

Arithmetization

\[ \varphi(z_1, \ldots, z_n) \quad \rightarrow \quad f: \mathbb{F}^n \rightarrow \mathbb{F} \]
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]

Arithmetization

\[ \varphi(z_1, \ldots, z_n) \quad \rightarrow \quad f: \mathbb{F}^n \rightarrow \mathbb{F} \]

\[ \text{degree}(f) = d = 4 \]
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]

Arithmetization

\[ \varphi(z_1, \ldots, z_n) \xrightarrow{f: \mathbb{F}^n \rightarrow \mathbb{F}} \]

degree\( f \) = \( d = 4 \)

\#SAT \leq \#3SAT^4
Sumcheck Protocol [Lund-Fortnow-Karloff-Nisan’90]

Arithmetization

\[ \varphi(z_1, \ldots, z_n) \rightarrow f : \mathbb{F}^n \rightarrow \mathbb{F} \]

\[ \text{degree}(f) = d = 4 \]

\[ \#\text{SAT} \leq \#\text{3SAT4} \]

Number of \( \vec{z} \in \{0,1\}^n \) such that \( \varphi(\vec{z}) = 1 \) is

\[ y = \sum_{\vec{z} \in \{0,1\}} f(\vec{z}) \]
Sumcheck Protocol
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

Prove it!
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\bar{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!
Sumcheck

\[ \sum_{z \in \{0,1\}^n} f(z) \] is some value \( y \)

Prove it!

\[ \tilde{g}_1(x) = \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n) \]

\[ \tilde{g}_1(0) + \tilde{g}_1(1) = y \]
The sum \( \sum_{z \in \{0,1\}^n} f(z) \) is some value \( y \)

Prove it!

\[
\tilde{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)
\]

\[
\tilde{g}_1(x) \quad \{\tilde{g}_1(0), \ldots, \tilde{g}_1(d)\}
\]

\[
\tilde{g}_1(0) + \tilde{g}_1(1) = y
\]
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\tilde{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)$

$\tilde{g}_1(x)$

$\{\tilde{g}_1(0), \ldots, \tilde{g}_1(d)\}$

$\tilde{g}_1(0) + \tilde{g}_1(1) \stackrel{?}{=} y$

$\beta_1 \leftarrow_R \mathbb{F}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\bar{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)$

$\bar{g}_1(x)$

$\{\bar{g}_1(0), \ldots, \bar{g}_1(d)\}$

$\bar{g}_1(0) + \bar{g}_1(1) \stackrel{?}{=} y$

$\beta_1 \leftarrow_r \mathbb{F}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\tilde{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)$

$y_1 := \tilde{g}_1(\beta_1)$

$\tilde{g}_1(x)$

$\{\tilde{g}_1(0), \ldots, \tilde{g}_1(d)\}$

$\beta_1 \leftarrow_R \mathbb{F}$

$y_1 := \tilde{g}_1(\beta_1)$

$\tilde{g}_1(0) + \tilde{g}_1(1) = y$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

\[ \tilde{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n) \]

\[ y_1 := \tilde{g}_1(\beta_1) \]

\[ \tilde{g}_2(x) := \sum_{z_3, \ldots, z_n \in \{0,1\}} f(\beta_1, x, z_3, \ldots, z_n) \]

\[ \beta_1 \leftarrow \mathbb{F} \]

\[ y_1 := \tilde{g}_1(\beta_1) \]

Prove it!
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$$\tilde{g}_1(x) := \sum_{z_2, \ldots, z_n \in \{0,1\}} f(x, z_2, \ldots, z_n)$$

$$y_1 := \tilde{g}_1(\beta_1)$$

$$\tilde{g}_2(x) := \sum_{z_3, \ldots, z_n \in \{0,1\}} f(\beta_1, x, z_3, \ldots, z_n)$$

$$\tilde{g}_2(x)$$

Prove it!

$$\tilde{g}_1(0) + \tilde{g}_1(1) = y$$

$$\beta_1 \leftarrow_R \mathbb{F}$$

$$y_1 := \tilde{g}_1(\beta_1)$$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

Prove it!
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!

$$\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!

$$\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$$

$$\beta_1$$

$$\vdots$$

$$\tilde{g}_j(x)$$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

\[ \tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n) \]

Prove it!

\[ \tilde{g}_j(0) + \tilde{g}_j(1) = \tilde{g}_{j-1}(\beta_{j-1}) \]
Sumcheck

The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

Prove it!

\[
\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)
\]

\[
\tilde{g}_1(x) \quad \beta_1 \quad \ldots \quad \tilde{g}_j(x) \quad \tilde{g}_j(0) + \tilde{g}_j(1) \overset{?}{=} \tilde{g}_{j-1}(\beta_{j-1})
\]

$\beta_j \leftarrow_R \mathbb{F}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$

$\tilde{g}_j(0) + \tilde{g}_j(1) \equiv \tilde{g}_{j-1}(\beta_{j-1})$

$\beta_j \leftarrow_R \mathbb{F}$
The sum $\Sigma_{z \in \{0,1\}^n} f(z)$ is some value $y$

$\bar{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$

$y_j := \bar{g}_j(\beta_j)$

$j$-th claim

Prove it!

$\bar{g}_j(0) + \bar{g}_j(1) \stackrel{?}{=} \bar{g}_{j-1}(\beta_{j-1})$

$\beta_j \leftarrow_R \mathbb{F}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!

$\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$

$y_j := \tilde{g}_j(\beta_j)$

$j$-th claim
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

Prove it!

$$\tilde{g}_j(x) := \sum_{z_{j+1},\ldots,z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$$

$y_j := \tilde{g}_j(\beta_j)$

$j$-th claim

$$\tilde{g}_j(0) + \tilde{g}_j(1) = \tilde{g}_{j-1}(\beta_{j-1})$$

$\beta_j \leftarrow_R \mathbb{F}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!

$\beta_1 \leftarrow_R \mathbb{F}$

$\beta_j \leftarrow_R \mathbb{F}$

$\beta_n \leftarrow_R \mathbb{F}$

$\tilde{g}_j(0) + \tilde{g}_j(1) \equiv \tilde{g}_{j-1}(\beta_{j-1})$

$j$-th claim

$\gamma_j := \tilde{g}_j(\beta_j)$

$\tilde{g}_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!

\[ \tilde{g}_j(x) = \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n) \]

\[ y_j := \tilde{g}_j(\beta_j) \]

$j$-th claim

\[ \tilde{g}_j(0) + \tilde{g}_j(1) = \tilde{g}_{j-1}(\beta_{j-1}) \]

$\beta_j \leftarrow_R \text{IF}$

\[ \beta_n \leftarrow_R \text{IF} \]

$f(\beta_1, \ldots, \beta_n) = \tilde{g}_n(\beta_n)$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$

Prove it!

- $\beta_j \leftarrow_R \text{IF}$
- $\beta_n \leftarrow_R \text{IF}$
- $f(\beta_1, \ldots, \beta_n) = g_n(\beta)$

$g_j(x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$

$y_j := g_j(\beta_j)$

$g_1(x)$

$\beta_1$

$\cdot$

$\cdot$

$\cdot$

$\beta_{n-1}$

$g_n(x)$

$\beta_n \leftarrow_R \text{IF}$
The sum $\sum_{z \in \{0,1\}^n} f(z)$ is some value $y$.

Prove it!
The $j$-th claim

$$y_j := \tilde{g}_j(\beta_j)$$
The $j$-th claim 

$$y_j := \tilde{g}_j(\beta_j)$$

$$\sum_{z_{j+1}, \ldots, z_n \in \{0, 1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j$$
The $j$-th claim

\[ y_j := \tilde{g}_j(\beta_j) \]

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]
The $j$-th claim

$$y_j := \tilde{g}_j (\beta_j)$$

$$\sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f (\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) \overset{?}{=} y_j$$

Recall

$$\tilde{g}_j (x) := \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f (\beta_1, \ldots, \beta_{j-1}, x, z_{j+1}, \ldots, z_n)$$

size $N/2^j$

claim
Soundness:

if the $j$-th claim is false then $\forall \tilde{g}_{j+1}(x)$ the $(j + 1)$-th claim is also false
Soundness

**Soundness**: if the $j$-th claim is **false** then $\forall \tilde{g}_{j+1}(x)$ the $(j+1)$-th claim is **also false**

Unambiguous Soundness: if $\tilde{g}_{j+1}(x) \neq g_{j+1}(x)$, then the $(j+1)$-th claim is **false** even if $j$-th claim was true
Soundness

**Soundness**: if the $j$-th claim is false then $\forall \tilde{g}_{j+1}(x)$ the $(j + 1)$-th claim is also false

Unambiguous Soundness: if $\tilde{g}_{j+1}(x) \neq g_{j+1}(x)$, then the $(j + 1)$-th claim is false even if $j$-th claim was true

Both with high probability over $\beta_{j+1}$ - Schwartz-Zippel.
Basic Idea

Several Challenges:
Proof size has to be polynomial
Basic Idea

Several Challenges:

- Proof size has to be polynomial

Sumcheck protocol
Basic Idea

Several Challenges:

Proof size has to be polynomial

Sumcheck protocol

Sumcheck Protocol is interactive
[Fiat-Shamir’86] Transformation
[Fiat-Shamir’86] Transformation
[Fiat-Shamir’86] Transformation
[Fiat-Shamir’86] Transformation
[Fiat-Shamir’86] Transformation

Replaced by a hash function $h$
[Fiat-Shamir’86] Transformation

Replaced by a hash function $h$

$$=h\left(\right)$$
Fiat-Shamir for Sumcheck

\[ \beta_j \quad = \quad h(\ldots) \quad \beta_1 \quad \ldots \quad \tilde{g}_j(x) \]
Fiat-Shamir for Sumcheck

Assumption

Resulting non-interactive (deterministic) argument is (adaptively) unambiguously sound
Fiat-Shamir for Sumcheck

Assumption
Resulting non-interactive (deterministic) argument is (adaptively) unambiguously sound

Given $h$, no poly-time prover can find accepting proof:
Fiat-Shamir for Sumcheck

Assumption
Resulting non-interactive (deterministic) argument is (adaptively) unambiguously sound

Given $h$, no poly-time prover can find accepting proof:

1. $\pi$ for a false statement $y$, or
Fiat-Shamir for Sumcheck

Given $h$, no poly-time prover can find accepting proof:

1. $\pi$ for a false statement $y$, or
2. $\tilde{\pi} \neq \pi$ for true statement $y$

Assumption
Resulting non-interactive (deterministic) argument is (adaptively) unambiguously sound
Fiat-Shamir for Sumcheck

Assumption

Resulting non-interactive (deterministic) argument is (adaptively) unambiguously sound

Given $h$, no poly-time prover can find accepting proof:

1. $\overline{\pi}$ for a false statement $y$, or
2. $\overline{\tilde{\pi}} \neq \overline{\pi}$ for true statement $y$

True if $h$ is a random oracle.
Basic Idea

Several Challenges:

- Proof size has to be polynomial
- Sumcheck protocol
- Sumcheck Protocol is interactive
Basic Idea

Several Challenges:

- Proof size has to be polynomial
- Sumcheck protocol
- Sumcheck Protocol is interactive
- Fiat-Shamir Transform
Basic Idea

Several Challenges:

- Proof size has to be polynomial
- Sumcheck protocol
- Sumcheck Protocol is interactive
- Fiat-Shamir Transform

Computing $S\left( i, y_i, \pi_i \right)$
Basic Idea

Several Challenges:

- Proof size has to be polynomial
- Sumcheck protocol
- Sumcheck Protocol is interactive
- Fiat-Shamir Transform
- Computing $S(i, y_i, \pi_i)$
- Incremental Proof Updates
Incremental Proof Updates

\[ 1, y_1, \pi_1 \rightarrow \ldots \rightarrow i, y_i, \pi_i \rightarrow i', y_{i'}, \pi_{i'} \rightarrow \ldots \rightarrow N, y_N, \pi_N \]
Incremental Proof Updates
Incremental Proof Updates
Incremental Proof Updates

\[1, y_1, \pi_1\]

\[i, y_i, \pi_i\]

\[i', y_{i'}, \pi_{i'}\]

\[j, y_j, \pi_1, \pi_2, \ldots, \pi_{p(n)}\]

\[N, y_N, \pi_N\]
Incremental Proof Updates

\[ T, y_T, \pi_T \]

\[ T = 2^{O(n)} \]

\[ \# \text{ of } \tilde{z} \leq 2^n \text{ such that } \varphi(\tilde{z}) = 1 \]

\[ y_j \in \mathbb{F} \]

\[ j, y_j, \pi_1 \pi_2 \ldots \pi_p(n) \]

\[ i', y_i', \pi_i' \]

\[ i, y_i, \pi_i \]

\[ 1, y_1, \pi_1 \]
Naïve Recursive Construction

Construction for $N/2 \rightarrow$ to construction for $N$

First $N/2$ assignments: Do recursively

Second $N/2$ assignments: Add second proof $y_i$, $\pi_i$
Naïve Recursive Construction

Construction for $N/2 \rightarrow$ to construction for $N$

First $N/2$ assignments: Do recursively

Second $N/2$ assignments: Add second proof $y_i, \pi_i$
Naïve Recursive Construction

Construction for $N/2 \rightarrow$ to construction for $N$

First $N/2$ assignments: Do recursively
Second $N/2$ assignments: Add second proof $y_i, \pi_i$

Proof size: $P(N) = 2 \cdot P(N/2)$

# Steps: $T(N) = 2 \cdot T(N/2)$
Naïve Recursive Construction

Construction for \( N/2 \rightarrow \) to construction for \( N \)

First \( N/2 \) assignments: Do recursively

Second \( N/2 \) assignments: Add second proof \( y_i, \pi_i \)

Proof size: \( P(N) = 2 \, P(N/2) \)

# Steps: \( T(N) = 2 \, T(N/2) \)
Merge Proofs [Valiant’06]

Merge proofs for $y_{N/2}$, $\pi_{N/2}$ and $y'_{N/2}$, $\pi'_{N/2}$
Merge Proofs [Valiant’06]

Merge proofs for $y_{N/2}, \pi_{N/2}$ and $y'_{N/2}, \pi'_{N/2}$
Merge Proofs [Valiant’06]

Proof size: $P(N) = P(N/2)$

# Steps: $T(N) = 2T(N/2) + 1$
Merge Proofs [Valiant’06]

Proof size: \( P(N) = P(N/2) \)

# Steps: \( T(N) = 2T(N/2) + 1 \)

Requires “super-extractable” SNARKs
New Idea: Incremental Merge

Merge via (long) incrementally verifiable computation.
New Idea: Incremental Merge

Merge via (long) incrementally verifiable computation.

How long?
New Idea: Incremental Merge

Merge via (long) incrementally verifiable computation.

How long? \( O(T(N/2)) \)
New Idea: Incremental Merge

Merge via (long) incrementally verifiable computation.

How long? $O(T(N/2))$

Proof size: $P(N) = P(N/2) + \text{poly}(n)$

# Steps: $T(N) = dT(N/2) + \text{poly}(n)$
New Idea: Incremental Merge

How do you efficiently compute

\[ S(i, y_i, \pi_i) = i + 1, y_{i+1}, \pi_{i+1} \]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0, 1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]

\[ \tilde{g}_{j+1}(x) \]

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]
Proof Merging for (Fiat-Shamir) sumcheck

\[
\sum_{z_{j+1}, \ldots, z_n \in \{0, 1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j
\]

\[
y_{j+1} := \tilde{g}_{j+1}(\beta_{j+1})
\]

\[j + 1\text{-th claim}\]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]

\[ y_{j+1} := \tilde{g}_{j+1}(\beta_{j+1}) \]

\( j + 1 \)-th claim

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]

size \( N/2^{j+1} \) claim
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0, 1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]

\[ y_{j+1} := \tilde{g}_{j+1}(\beta_{j+1}) \]

\( j + 1 \)-th claim

size \( \frac{N}{2^{j+1}} \)

claim

\[ \{\tilde{g}_{j+1}(0), \ldots, \tilde{g}_{j+1}(d)\} \]

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0,1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]

size \( N/2^{j+1} \) claims

\[ \{\tilde{g}_{j+1}(0), \ldots, \tilde{g}_{j+1}(d)\} \]

\[ y_{j+1} := \tilde{g}_{j+1}(\beta_{j+1}) \]

\( j + 1 \)-th claim

size \( N/2^{j+1} \) claim

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]
Proof Merging for (Fiat-Shamir) sumcheck

\[ \sum_{z_{j+1}, \ldots, z_n \in \{0, 1\}} f(\beta_1, \ldots, \beta_{j-1}, \beta_j, z_{j+1}, \ldots, z_n) = y_j \]

\( y_{j+1} := \tilde{g}_{j+1}(\beta_{j+1}) \)

\( j + 1 \)-th claim

size \( N/2^{j+1} \) claim

size \( N/2^{j+1} \) claims

\( \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \)

\( (d + 2) \) size \( N/2^{j+1} \) claims
Proving the $j$-th claim

\[
\begin{align*}
\tilde{g}_{j+1}(0), & \quad \pi_0 \\
\tilde{g}_{j+1}(1), & \quad \pi_1 \\
\vdots & \\
\tilde{g}_{j+1}(d), & \quad \pi_d
\end{align*}
\]
Proving the $j$-th claim

\[ \tilde{g}_{j+1}(0), \pi_0 \]
\[ \tilde{g}_{j+1}(1), \pi_1 \]
\[ \vdots \]
\[ \tilde{g}_{j+1}(d), \pi_d \]

size $N/2^{j+1}$ claims
Proving the $j$-th claim

$\tilde{g}_{j+1}(x)$

\[
\begin{align*}
\tilde{g}_{j+1}(0), & \quad \pi_0 \\
\tilde{g}_{j+1}(1), & \quad \pi_1 \\
& \quad \vdots \\
\tilde{g}_{j+1}(d), & \quad \pi_d \\
\end{align*}
\]

size $N/2^{j+1}$ claims
Proving the $j$-th claim

$\tilde{g}_{j+1}(x)$

$\tilde{g}_{j+1}(0), \pi_0$
$\tilde{g}_{j+1}(1), \pi_1$
$\vdots$
$\tilde{g}_{j+1}(d), \pi_d$

size $N/2^{j+1}$ claims

$\beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1})$
Proving the $j$-th claim

\[ \tilde{g}_{j+1}(x) \]

\[ \tilde{g}_{j+1}(0), \pi_0 \]
\[ \tilde{g}_{j+1}(1), \pi_1 \]
\[ \ddots \]
\[ \tilde{g}_{j+1}(d), \pi_d \]

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]

\[ \tilde{g}_{j+1}(\beta_{j+1}), \pi_{d+1} \]
Proving the $j$-th claim

\[ \tilde{g}_{j+1}(x) \]

\[ \tilde{g}_{j+1}(0), \pi_0 \]
\[ \tilde{g}_{j+1}(1), \pi_1 \]
\[ \vdots \]
\[ \tilde{g}_{j+1}(d), \pi_d \]

size $N/2^{j+1}$ claims

\[ \beta_{j+1} = h(\alpha_1, \beta_1, \ldots, \alpha_{j+1}) \]

\[ \tilde{g}_{j+1}(\beta_{j+1}), \pi_{d+1} \]

size $N/2^{j+1}$ claim
Parameters
Parameters

Proof size: \( P(N) = P(N/2) + \log|F| \)
Parameters

Proof size: $P(N) = P(N/2) + \log|\mathbb{F}|$

$P(0) = \log|\mathbb{F}|$
Proof size: $P(N) = P(N/2) + \log|F|$ 

$P(0) = \log|F|$ 

$P(n) = \text{poly}(n)$
Proof size: \( P(N) = P(N/2) + \log|\mathbb{F}| \)

\# Steps: \( T(N) = (d + 2) T(N/2) + poly(n) \)

\( P(0) = \log|\mathbb{F}| \)

\( P(n) = poly(n) \)
Parameters

\[
P(N) = P(N/2) + \log|\mathbb{F}|
\]

\[
T(N) = (d + 2)T(N/2) + \text{poly}(n)
\]

\[
P(0) = \log|\mathbb{F}|
\]

\[
P(n) = \text{poly}(n)
\]

\[
T(0) = \text{poly}(\log|\mathbb{F}|)
\]
Parameters

Proof size: $P(N) = P(N/2) + \log|\mathbb{F}|$

$P(0) = \log|\mathbb{F}|$
$P(n) = \text{poly}(n)$

$P(0) = \log|\mathbb{F}|$
$P(n) = \text{poly}(n)$

# Steps: $T(N) = (d + 2)T(N/2) + \text{poly}(n)$

$T(0) = \text{poly}(\log|\mathbb{F}|)$
$T(n) = 2^0(n)$
Open Problems

Instantiating Fiat-Shamir for sumcheck

Sampling small(ish) hard instances of NASH
Thank you. Questions?
The Five Worlds of Impagliazzo

Algorithmica: P = NP
Heuristica: P ≠ NP, NP is easy-on-average
Pessiland: NP is hard-on-average, ∀ one-way functions, ∀ public-key crypto
Minicrypt: ∃ one-way functions, ∄ public-key crypto
Cryptomania: ∃ public-key crypto

The Five Worlds of Impagliazzo
The Five Worlds of Impagliazzo

- **Algorithmica:** $P = NP$
- **Heuristica:** $P \neq NP$
  - NP is easy-on-average
- **Pessiland:**
  - NP is hard-on-average
  - $\exists$ one-way functions
  - $\not\exists$ public-key crypto
- **Minicrypt:**
  - $\exists$ one-way functions
  - $\not\exists$ public-key crypto
- **Cryptomania:**
  - $\exists$ public-key crypto
  - $\not\exists$ one-way functions
The Five Worlds of Impagliazzo

Algorithmica: $P = NP$

Heuristica: $P \neq NP$

NP is easy-on-average

Pessiland: NP is hard-on-average

Minicrypt: $\exists$ one-way functions $\nexists$ public-key crypto

Cryptomania: $\exists$ public-key crypto
The Five Worlds of Impagliazzo

**Algorithmica:**
- $P = NP$

**Heuristica:**
- $P \neq NP$
- NP is easy-on-average

**Pessiland:**
- NP is hard-on-average
- $\nexists$ one-way functions
- $\nexists$ public-key crypto

**Minicrypt:**
- $\nexists$ one-way functions
- $\nexists$ public-key crypto

**Cryptomania:**
- $\exists$ public-key crypto
The Five Worlds of Impagliazzo

- **Algorithmica:**
  - $P = NP$
  - NP is easy-on-average

- **Heuristica:**
  - $P \neq NP$
  - NP is hard-on-average

- **Pessiland:**
  - $P \neq NP$
  - NP is hard-on-average
  - $\not\exists$ one-way functions
  - $\not\exists$ public-key crypto

- **Minicrypt:**
  - $\exists$ one-way functions
  - $\not\exists$ public-key crypto

- **Cryptomania:**
  - $\exists$ public-key crypto
  - $\exists$ one-way functions
The Five Worlds of Impagliazzo

Algorithmica:
$P = NP$

Heuristica:
$P \neq NP$
NP is easy-on-average

Pessiland:
NP is hard-on-average
\nexists one-way functions
\nexists public-key crypto

Minicrypt:
\nexists one-way functions
\nexists public-key crypto

Cryptomania:
\nexists public-key crypto
The Five Worlds of Impagliazzo

- **Algorithmica**: $P = NP$
- **Heuristica**: $P \neq NP$
- **Pessiland**: NP is hard-on-average
- **Minicrypt**: $\exists$ one-way functions $\not\exists$ public-key crypto
- **Cryptomania**: $\exists$ public-key crypto

Factoring, DL, LWE
The Five Worlds of Impagliazzo

- **Algorithmica:**
  - $P = NP$

- **Heuristica:**
  - $P \neq NP$
  - NP is easy-on-average

- **Pessiland:**
  - NP is hard-on-average
  - No one-way functions

- **Minicrypt:**
  - Existence of one-way functions
  - No public-key crypto

- **Cryptomania:**
  - Existence of public-key crypto

- **Factoring, DL, LWE**
  - Cryptographic hardness
  - CRH, AES, SHA

The Five Worlds illustrate various possible scenarios for the relationship between P and NP.
The Five Worlds of Impagliazzo

Algorithmica: $P = NP$

Heuristica: $P \neq NP$
NP is easy-on-average

Pessiland: NP is hard-on-average
$\exists$ one-way functions
$\nexists$ public-key crypto

Minicrypt: $\exists$ one-way functions
$\nexists$ public-key crypto

Cryptomania: $\exists$ public-key crypto

Factoring, DL, LWE

CRH, AES, SHA

Obfustopia: $\exists$ indistinguishability obfuscation

Factoring, DL, LWE

CRH, AES, SHA

Cryptomania: $\exists$ public-key crypto

Minicrypt: $\exists$ one-way functions
$\nexists$ public-key crypto

Pessiland: NP is hard-on-average
$\exists$ one-way functions

Heuristica: $P \neq NP$
NP is easy-on-average

Algorithmica: $P = NP$
Algorithmica: P = NP

Heuristica: P $\neq$ NP
NP is easy-on-average

Pessiland: NP is hard-on-average
$\exists$ one-way functions
$c$ public-key crypto

Minicrypt: $\exists$ one-way functions
$\neg$ public-key crypto

Cryptomania: $\exists$ public-key crypto

Obfustopia: $\exists$ indistinguishability obfuscation

Factoring, DL, LWE
CRH, AES, SHA
Multilinear Maps

The Five Worlds of Impagliazzo
The Five Worlds of Impagliazzo

**Algorithmica:**
- $P = NP$

**Heuristica:**
- $P \neq NP$
- NP is easy-on-average

**Pessiland:**
- NP is hard-on-average
- $\not\exists$ one-way functions
- $\not\exists$ public-key crypto

**Minicrypt:**
- $\exists$ one-way functions
- $\not\exists$ public-key crypto

**Cryptomania:**
- $\exists$ public-key crypto

**Obfustopia:**
- $\exists$ indistinguishability obfuscation

**Factoring, DL, LWE**

**CRH, AES, SHA**

**Multilinear Maps**
The Five Worlds of Impagliazzo

**Algorithmica:**
- $P = NP$

**Heuristica:**
- $P \neq NP$
- NP is easy-on-average
- $\exists$ one-way functions

**Pessiland:**
- NP is hard-on-average
- $\nexists$ one-way functions
- $\nexists$ public-key crypto

**Minicrypt:**
- $\exists$ one-way functions
- $\nexists$ public-key crypto

**Cryptomania:**
- $\exists$ public-key crypto

**Obfustopia:**
- $\exists$ indistinguishability obfuscation

**TFNP Hardness**

**PWPP Hardness**

**Factoring, DL, LWE**

**CRH, AES, SHA**

**Multilinear Maps**
The Five Worlds of Impagliazzo

**Algorithmica:**
- $P = NP$
- NP is easy-on-average

**Heuristica:**
- $P \neq NP$
- NP is hard-on-average

**Pessiland:**
- $\exists$ one-way functions
- $\nexists$ public-key crypto

**Minicrypt:**
- $\exists$ one-way functions
- $\nexists$ public-key crypto

**Cryptomania:**
- $\exists$ public-key crypto

**Obfustopia:**
- $\exists$ indistinguishability obfuscation

**Multilinear Maps**

**Factoring, DL, LWE**

**TFNP Hardness**

**PWPP Hardness**

**CLS Hardness**
The Five Worlds of Impagliazzo

Algorithmica: $P = \text{NP}$

Heuristica: $P \neq \text{NP}$

Minicrypt: $\exists$ one-way functions $\not\exists$ public-key crypto

Pessiland: $\text{NP}$ is hard-on-average $\not\exists$ one-way functions

TFNP Hardness

PWPP Hardness

Factoring, DL, LWE

CRH, AES, SHA

Multilinear Maps

FS for sumcheck

CLS Hardness

Cryptomania: $\exists$ public-key crypto

Obfustopia: $\exists$ indistinguishability obfuscation

The Five Worlds of Impagliazzo

Factoring, DL, LWE

CRH, AES, SHA

Minicrypt: $\exists$ one-way functions $\not\exists$ public-key crypto

Pessiland: $\text{NP}$ is hard-on-average $\not\exists$ one-way functions

TFNP Hardness

PWPP Hardness

Algorithmica: $P = \text{NP}$

Heuristica: $P \neq \text{NP}$

NP is easy-on-average $\not\exists$ one-way functions

Multilinear Maps

FS for sumcheck

CLS Hardness

Cryptomania: $\exists$ public-key crypto

Obfustopia: $\exists$ indistinguishability obfuscation
Is Crypto hardness Necessary?
[Rosen-Segev-Shachaf’17]

Black box separations

SVL hardness not essential for PPAD hardness

Basing PPAD hardness on OWFs:
  cannot go through SVL, and
  must have exponential sol