Zero Knowledge Succinct Arguments: an Introduction

Alessandro Chiesa
UC Berkeley
Motivation
cryptography is a powerful tool for building secure systems
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for data
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for **data confidentiality**
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for data confidentiality

$\text{Alice} \xrightarrow{\text{Enc}(m)} \text{Bob}$
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for **data**

**confidentiality**

\[ \text{Alice} \xrightarrow{\text{Enc}(m)} \text{Bob} \]

**authenticity**

\[ \text{Alice} \xrightarrow{m} \text{Bob} \]
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for **data**

**confidentiality**

Alice \[\text{Enc(m)}\] Bob

**authenticity**

Alice \[m\]  \[\text{Sig(m)}\]  Bob
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for data

confidentiality

\[ \text{Alice} \xrightarrow{\text{Enc}(m)} \text{Bob} \]

authenticity

\[ \text{Alice} \xrightarrow{\text{Sig}(m)} \text{Bob} \]

what about security properties for computation?
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for **data**

confidentiality

\[
\begin{array}{c}
\text{Alice} \\
\text{Enc(m)} \\
\text{Bob}
\end{array}
\]

authenticity

\[
\begin{array}{c}
\text{Alice} \\
\text{Sig(m)} \\
\text{Bob}
\end{array}
\]

what about security properties for **computation**?

**cryptographic proofs** offer privacy-preserving integrity for computation
cryptography is a powerful tool for building secure systems

much of the cryptography used today offers security properties for **data**

- confidentiality
  - Alice $\xrightarrow{\text{Enc}(m)}$ Bob
- authenticity
  - Alice $\xrightarrow{\text{Sig}(m)}$ Bob

what about security properties for **computation**?

**cryptographic proofs** offer privacy-preserving integrity for computation

one of the exciting crypto deployment frontiers today
Cryptographic Proofs
Cryptographic Proofs

a powerful defense against malicious behavior especially in distributed protocols
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute $y=F(x_1,\ldots,x_n)$ via a multi-party protocol
Cryptographic Proofs

a powerful defense against malicious behavior especially in distributed protocols

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

[GMW87]
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

Key properties
- zero knowledge
- proof of knowledge
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

Key properties
- zero knowledge
- proof of knowledge

2010s blockchain technology
Cryptographic Proofs

a powerful defense against malicious behavior especially in distributed protocols

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

[GMW87] passive security active security

Key properties
- zero knowledge
- proof of knowledge

2010s blockchain technology
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

**1980s** securely compute \( y = F(x_1, \ldots, x_n) \)

via a multi-party protocol

- passive security

- active security

**2010s** blockchain technology

- zero knowledge
- proof of knowledge

\[ I \text{ know } x \quad \text{s.t. } y = F(x) \]
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

![Diagram of passive and active security](image)

**Key properties**
- zero knowledge
- proof of knowledge

2010s blockchain technology

I know $x$ s.t. $y = F(x)$ proof
Cryptographic Proofs

a powerful defense against malicious behavior
especially in **distributed protocols**

1980s  securely compute \( y = F(x_1, \ldots, x_n) \)
via a multi-party protocol

![Diagram showing passive and active security](GMW87)

2010s  blockchain technology

I know \( x \)
s.t. \( y = F(x) \)
proof

**Key properties**
- zero knowledge
- proof of knowledge

**Additional key properties**
- non-interactive
- publicly verifiable
- succinct
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute \( y = F(x_1, \ldots, x_n) \) via a multi-party protocol

2010s blockchain technology

**zk-SNARK**

**Key properties**
- zero knowledge
- proof of knowledge

**Additional key properties**
- non-interactive
- publicly verifiable
- succinct
Cryptographic Proofs

a powerful defense against malicious behavior especially in **distributed protocols**

1980s securely compute $y = F(x_1, \ldots, x_n)$ via a multi-party protocol

[GMW87]

**Key properties**
- zero knowledge
- proof of knowledge

2010s blockchain technology

**Additional key properties**
- non-interactive
- publicly verifiable
- succinct

zk-SNARK
Origins
Zero Knowledge Proofs

[GMRT85]
Zero Knowledge Proofs

\[ \exists x : y = F(x) \rightarrow \Pr[P(F, y, x) \text{ convinces } V(F, y)] = 1 \]

soundness

\[ \neg \exists x : y = F(x) \rightarrow \forall P', \Pr[P' \text{ convinces } V(F, y)] \approx 0 \]

zero knowledge

\[ \exists x : y = F(x) \rightarrow \forall V', S(V', F, y) \approx \text{view of } V' \text{ with } P(F, y, x) \]

\[ \text{[GMR85]} \]
Zero Knowledge Proofs

[GM85]
Zero Knowledge Proofs

[GM8R5]

\[ \exists x : y = F(x) \rightarrow Pr[P(F, y, x) \text{ convinces } V(F, y)] = 1 \]

\[ \forall x : y = F(x) \rightarrow \forall P', Pr[P' \text{ convinces } V(F, y)] \approx 0 \]

\[ \exists x : y = F(x) \rightarrow \forall V', S(V', F, y) \approx \text{view of } V' \text{ with } P(F, y, x) \]
Zero Knowledge Proofs
[GMW85]

“\[\text{I know } x \text{ s.t. } y = F(x)\]”

\[\text{Prover}\]

\[\text{Verifier}\]

- Function \(F\)
- Claimed output \(y\)
- Private input \(x\)
Zero Knowledge Proofs

[GM85]

“\(I\ know\ x\ s.t.\ y=F(x)\)"

\[\exists x: y=F(x) \to \Pr[P(F,y,x)\ convinces\ V(F,y)]=1\]

completeness
Zero Knowledge Proofs

I know \( x \) s.t. \( y = F(x) \)

Completeness: \( \exists x: y = F(x) \rightarrow Pr[P(F, y, x) \text{ convinces } V(F, y)] = 1 \)

Soundness: \( \not\exists x: y = F(x) \rightarrow \forall P', Pr[P' \text{ convinces } V(F, y)] \approx 0 \)
Zero Knowledge Proofs

[GMR85]

"I know $x$ s.t. $y = F(x)$"

**Completeness**

$\exists \ x: \ y = F(x) \rightarrow \text{Pr}[P(F,y,x) \text{ convinces } V(F,y)] = 1$

**Soundness**

$\nexists \ x: \ y = F(x) \rightarrow \forall P', \text{Pr}[P' \text{ convinces } V(F,y)] \approx 0$

**Zero Knowledge**

$\exists \ x: \ y = F(x) \rightarrow \forall V', S(V',F,y) \approx \text{view of } V' \text{ with } P(F,y,x)$
Zero Knowledge Proofs

[GM85]

"I know $x$ s.t. $y = F(x)$"

**Completeness**

$\exists \ x \ : \ y = F(x) \ \rightarrow \ \Pr[P(F, y, x) \text{ convinces } V(F, y)] = 1$

**Soundness**

$\nexists \ x \ : \ y = F(x) \ \rightarrow \ \forall P', \ Pr[P' \text{ convinces } V(F, y)] \approx 0$

**Zero Knowledge**

$\exists \ x \ : \ y = F(x) \ \rightarrow \ \forall V', \ S(V', F, y) \approx \text{view of } V' \text{ with } P(F, y, x)$

simulator
Zero Knowledge Proofs

[GMRT85]

"I know \( x \) s.t. \( y = F(x) \)"

\[ P \text{ prover} \]

- \( F \) function
- \( y \) claimed output
- \( x \) private input

\[ V \text{ verifier} \]

- \( F \) function
- \( y \) claimed output
Zero Knowledge Proofs

[GMR85]

“I know $x$ s.t. $y = F(x)$”

[GMR85]: ZKPs for certain number-theoretic problems (QR,QNR)
Zero Knowledge Proofs

[GM8R5]: ZKPs for certain number-theoretic problems (QR,QNR)

"I know $x$ s.t. $y = F(x)$"

If one-way functions exist:
Zero Knowledge Proofs

[GMW86]: ZKPs for all poly-time computable functions $F$

If one-way functions exist:

[GMW86]: ZKPs for all poly-time computable functions $F$
Zero Knowledge Proofs

[GM85]: ZKPs for certain number-theoretic problems (QR, QNR)

If one-way functions exist:

[GMW86]: ZKPs for all poly-\textit{time} computable functions $F$

[BGGHKMR88]: ZKPs for all poly-\textit{space} computable functions $F$
Zero Knowledge Proofs
[GMRT85]

"I know $x$ s.t. $y = F(x)$"

Prover

Verifier

F function
y claimed output
x private input

Powerful cryptographic primitive.
Zero Knowledge Proofs

[GMR85]

“"I know \(x\) s.t. \(y = F(x)\)"

Powerful cryptographic primitive.

BUT
Zero Knowledge Proofs

[GM1R85]

"I know $x$ s.t. $y=F(x)$"

**Prover**

- $F$ function
- $y$ claimed output
- $x$ private input

**Verifier**

- $F$ function
- $y$ claimed output

**Powerful cryptographic primitive.**

BUT

interactive
Zero Knowledge Proofs

[GMR85]

"I know $x$ s.t. $y = F(x)$"

Powerful cryptographic primitive.

BUT

interactive

not succinct
Zero Knowledge Proofs

[GMRS85]

“I know \( x \) s.t. \( y = F(x) \)”

Prover

Verifier

F function

y claimed output

x private input

Powerful cryptographic primitive.

BUT

interactive

not succinct

communication complexity & verification complexity are proportional to \( \text{time}(F) \)
Zero Knowledge Proofs
[GM85]

"I know $x$ s.t. $y = F(x)$"

**Prover**
- $F$ function
- $y$ claimed output
- $x$ private input

**Verifier**
- $F$ function
- $y$ claimed output

**Powerful cryptographic primitive.**

**BUT**

**interactive**

**not succinct**

for typical $F$, $\text{size}(F) \ll \text{time}(F)$

communication complexity & verification complexity are proportional to time$(F)$
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]

"I know $x$ s.t. $y = F(x)$"
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]

“\(I\know\ x\ \text{s.t.}\ y=F(x)\)"

<table>
<thead>
<tr>
<th>completeness</th>
<th>(\exists\ x: y=F(x) \implies \Pr[P(F,y,x)\text{ convinces }V(F,y)]=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>soundness</td>
<td>(\nexists\ x: y=F(x) \implies \forall P' \Pr[P'\text{ convinces }V(F,y)]=0)</td>
</tr>
<tr>
<td>zero knowledge</td>
<td>(\exists x: y=F(x) \implies \forall V', S(V',F,y) \equiv \text{view of } V' \text{ with } P(F,y,x))</td>
</tr>
<tr>
<td>succinctness</td>
<td>(V(F,y)\text{ runs in time proportional to }</td>
</tr>
</tbody>
</table>
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]

“I know \( x \) s.t. \( y = F(x) \)”

| completeness | \( \exists x : y = F(x) \rightarrow \Pr[P(F,y,x) \text{ convinces } V(F,y)] = 1 \) |
| soundness*   | \( \nexists x : y = F(x) \rightarrow \forall P' \Pr[P' \text{ convinces } V(F,y)] \approx 0 \) |
| zero knowledge | \( \exists x : y = F(x) \rightarrow \forall V', S(V',F,y) \approx \text{view of } V' \text{ with } P(F,y,x) \) |
| succinctness | \( V(F,y) \) runs in time proportional to \(|F|+|y|\) (not time(F)+|y|) |
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]

"I know \( x \) s.t. \( y = F(x) \)"

**completeness**

\[ \exists x : y = F(x) \rightarrow \Pr[P(F,y,x) \text{ convinces } V(F,y)] = 1 \]

**soundness**

\[ \not\exists x : y = F(x) \rightarrow \forall P' \Pr[P' \text{ convinces } V(F,y)] \approx 0 \]

**zero knowledge**

\[ \exists x : y = F(x) \rightarrow \forall V' , S(V',F,y) \approx \text{view of } V' \text{ with } P(F,y,x) \]

**succinctness**

\( V(F,y) \) runs in time proportional to \(|F|+|y|\) (not time(\( F \))+|\( y \)|)

* must relax to *computational* soundness: \( \forall \text{ PPT } P' \ldots \) [GH98]
Zero Knowledge **Succinct** Proofs

[Kilian92][Micali94]

"I know \( x \) s.t. \( y = F(x) \)

| completeness  | \( \exists \ x: y = F(x) \rightarrow \Pr[P(F,y,x) \text{ convinces } V(F,y)] = 1 \) |
| soundness*    | \( \nexists \ x: y = F(x) \rightarrow \forall P' \Pr[P' \text{ convinces } V(F,y)] \approx 0 \) |
| zero knowledge| \( \exists \ x: y = F(x) \rightarrow \forall V', S(V',F,y) \approx \text{view of } V' \text{ with } P(F,y,x) \) |
| succinctness  | \( V(F,y) \) runs in time proportional to \( |F|+|y| \) (not time(\( F \)) + |\( y \)|) |

* must relax to computational soundness: \( \forall \text{ PPT } P' \ldots \) [GH98]
Achieving Succinctness
Achieving Succinctness

Zero Knowledge Succinct Proof
Achieving Succinctness

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

TOFIX

interactive

not succinct
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

TOFIX

interactive

not succinct
Achieving Succinctness

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

TOFIX
interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

TOFIX
- interactive
- not succinct
- bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

TOFIX
interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
(Kilian92)

Zero Knowledge SNARK
(the first)

TOFIX

interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof
[Kilian92]

Zero Knowledge SNARK
(Micali94)

Random Oracle (SHA-256)

TOFIX
interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

(Kilian92)

Zero Knowledge SNARK
(Random Oracle (SHA-256))

TOFIX
interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof

[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

[Kilian92]

Zero Knowledge SNARK

[COM]

Random Oracle (SHA-256)

TOFIX

interactive
not succinct
bad concrete efficiency

bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

Kilian92

Zero Knowledge SNARK
(Random Oracle (SHA-256))

TOFIX

interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

Kilian92

Micali94

Zero Knowledge SNARK

Random Oracle (SHA-256)

TOFIX

interactive
not succinct
bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof

| BFLS91 | FGLSS96 | AS92 | ALMSS92 |

Zero Knowledge Succinct Proof

| Kilian92 |

Zero Knowledge SNARK

| Micali94 |

Random Oracle (SHA-256)

TOFIX

- interactive
- not succinct
- bad concrete efficiency
Achieving Non-Interactivity

Probabilistically Checkable Proof
[BFLS91][FGLSS96][AS92][ALMSS92]

Zero Knowledge Succinct Proof

Zero Knowledge SNARK

Random Oracle (SHA-256)

TOFIX

interactive
not succinct
bad concrete efficiency
Modern Era
The Quest for ZK-SNARKs without Random Oracles
Negative result: constructing them "requires strong assumptions" [GW11]
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

Knowledge of Exponent
[D 92]
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

Knowledge of Exponent [D 92]
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

Knowledge of Exponent

[D 92]

Extractable Hash Functions
The Quest for ZK-SNARKs without Random Oracles

**Negative result**: constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Extractable Hash Functions
  - [BCCT 12]
  - [DFH 12]
  - [GLR 12]
  - [BC 12]
  - [BCCT 13]
  - [BCGLRT 16]
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Extractable Hash Functions
  - [BCCT 12]
  - [DFH 12]
  - [GLR 12]
  - [BC 12]
  - [BCCT 13]
  - [BCGLRT 16]
- based on PCPs
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent
  - [D 92]
- Extractable Hash Functions
- Linear-Only Encryption/Encoding

- [BCCT 12]
- [DFH 12]
- [GLR 12]
- [BC 12]
- [BCCT 13]
- [BC CGLRT 16]

based on **PCPs**
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Extractable Hash Functions
  - [BCCT 12]
  - [DFH 12]
  - [GLR 12]
  - [BC 12]
  - [BCCT 13]
  - [BCGLRT 16]
- Linear-Only Encryption/Encoding
  - [G 10]
  - [L 12]
  - [BCIOP 13]
  - [GGPR 13]
  - [PGHR 13]
  - [BCGTV 13]
  - [BCTV 14a]
  - [DFGK 14]
  - [WSRBW 15]
  - [BBFR 15]
  - [BCTV 14b]
  - [CTV 15]
  - [CFHKKNPZ 15]
  - [BISW 17]
  - ...

Based on PCPs
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Extractable Hash Functions
- Linear-Only Encryption/Encoding

- Generic Group Model [G 16]
- Based on PCPs

- [BCCT 12]
- [DFH 12]
- [GLR 12]
- [BC 12]
- [BCCT 13]
- [BCGLRT 16]

- [G 10]
- [L 12]
- [BCIOPT 13]
- [GGPR 13]
- [PGHR 13]
- [BCGTV 13]
- [BCTV 14a]
- [DFGK 14]

- [WSRBW 15]
- [BBFR 15]
- [BCCTV 14b]
- [CTV 15]
- [CFHKKNPZ 15]
- [BISW 17]

::
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Generic Group Model [G 16]
- Extractable Hash Functions
  - [BCCT 12]
  - [DFH 12]
  - [GLR 12]
  - [BC 12]
  - [BCCT 13]
  - [BCCGLRT 16]
  - based on **PCPs**
- Linear-Only Encryption/Encoding
  - [G 10]
  - [L 12]
  - [BCIOPT 13]
  - [GGPR 13]
  - [PGHR 13]
  - [BCGT 13]
  - [BCCTV 14a]
  - [DFGK 14]
  - [WSRBW 15]
  - [BBFR 15]
  - [BCCTV 14b]
  - [CTV 15]
  - [CFHKKNPZ 15]
  - [BISW 17]
  - based on **linear PCPs**
The Quest for ZK-SNARKs without Random Oracles

**Negative result:** constructing them "requires strong assumptions" [GW11]

**Positive results** (under strong assumptions):

- Knowledge of Exponent [D 92]
- Generic Group Model [G 16]
- Extractable Hash Functions
  - Based on PCPs
    - [BCCT 12]
    - [DFH 12]
    - [GLR 12]
    - [BC 12]
    - [BCCT 13]
    - [BCCGLRT 16]
- Linear-Only Encryption/Encoding
  - Based on linear PCPs
    - [G 10]
    - [L 12]
    - [BCIO 13]
    - [GGPR 13]
    - [PGHR 13]
    - [BCGTV 13]
    - [BCTV 14a]
    - [DFGK 14]
    - [WSRBW 15]
    - [BBFR 15]
    - [BCTV 14b]
    - [CTV 15]
    - [CFHKKNPZ 15]
    - [BISW 17]
    - \[ \cdots \]
ZK-SNARKs from Linear PCPs
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BcioP13]

Zero Knowledge SNARK

P → ⟨\tilde{\alpha}, \cdot⟩ → Q → D

[BCioP13]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]

Zero Knowledge SNARK

Setup

\( \langle \tilde{a}, \cdot \rangle \)
ZK-SNARKs from Linear PCPs

Linear PCP

\[ \text{Setup} \]

\[ \text{pk} \]

\[ \text{vk} \]

Zero Knowledge SNARK

\[ \text{Setup} \]

\[ \text{pk} \]

\[ \text{vk} \]

\[ \text{P} \]

\[ \text{V} \]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]

Zero Knowledge SNARK

Setup

\[ \langle \bar{\alpha}, \cdot \rangle \]

linear-only encodings

\[ [\text{BCIO13}] \]

\[ \text{Setup} \]

\[ \text{pk} \]

\[ \text{vk} \]

\[ \text{V} \]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]

Zero Knowledge SNARK

Setup

pk

vk

P

Q

D

\langle \vec{a}, \cdot \rangle

linear-only encodings

[BCIOP13]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]

Zero Knowledge SNARK

Setup

pk

vk

P

Q

D

linear-only encodings

\langle \bar{\alpha}, \cdot \rangle

\bar{\alpha}

\cdot

Pk

V

\text{Setup}

\text{pk}

\text{vk}
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]

(P, Q, D)

linear-only encodings

Zero Knowledge SNARK

Setup

pk
vk

P

V

pk

Enc

vk
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]

Zero Knowledge SNARK

Setup

pk
vk

P

Q

D

linear-only encodings

P

Q

V

P

pk

Enc

vk
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]

Zero Knowledge SNARK

Setup

P

pk

vk

V

Q

pk

Enc

Q

vk

V

D

P

\langle \bar{\alpha}, \cdot \rangle
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIOP13]

Zero Knowledge SNARK

Setup

pk

vk

P

V

\langle \bar{\alpha}, \cdot \rangle

linear-only encodings

\langle \bar{\alpha}, \cdot \rangle

Hom Eval

Hom Eval

\text{Enc}

\text{Enc}

\text{Eval}
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]

Zero Knowledge SNARK

Setup

\[pk\]

\[vk\]

\[V\]
ZK-SNARKs from Linear PCPs

Linear PCP
[IKO07][BCIO13]

Zero Knowledge SNARK

Setup

Setup

P

pk

vk

pk

Q

Enc

Eval

 Hom

vk

ZeroTest

D
ZK-SNARKs from Linear PCPs
ZK-SNARKs from Linear PCPs

arithmetic circuit

C
ZK-SNARKs from Linear PCPs

arithmetic circuit

Setup
ZK-SNARKs from Linear PCPs

Setup

Q

arithmetic circuit
ZK-SNARKs from Linear PCPs

proving key
pk_c

arithmetic circuit
C

Setup

Enc

Q

Enc

Enc

vk_c
verification key
ZK-SNARKs from Linear PCPs

**Setup**

- **proving key**: pkC
- **verification key**: vkC

**Arithmetic circuit**

**Prover**

- **input**: x
- **witness**: w

Given public C, x exists a secret w s.t. C(x, w) = 0
ZK-SNARKs from Linear PCPs

Given public $C, x \in \mathbb{F}$, there exists a secret $w$ such that $C(x, w) = 0$.
ZK-SNARKs from Linear PCPs

\[ \begin{align*}
    \text{Setup} & \quad \text{proof key} \quad \text{verification key} \\
    \text{Prover} & \quad \text{given public } C, x \quad \exists \text{ secret } w \text{ s.t. } C(x, w) = 0 \\
    \text{input} x \quad \text{witness} w
\end{align*} \]
ZK-SNARKs from Linear PCPs

Given public $C, x$ 
$\exists$ secret $w$ s.t. $C(x, w) = 0$
ZK-SNARKs from Linear PCPs

Given public $C, x \exists$ secret $w$ s.t. $C(x, w) = 0$

Arithmetic circuit

Prover

Verifier

Setup

Input $x$

Witness $w$

Proving key $pk_c$

Verification key $vk_c$

Proof system:

1. Setup
   - Choose public parameters and generate proving and verification keys.
   - $pk_c, vk_c$

2. Prove
   - Prover receives input $x$ and witness $w$.
   - Computes $\tilde{\alpha}$ using the proving key.
   - Evaluates $C$ using the verification key.
   - Prover sends $\langle \tilde{\alpha}, \cdot \rangle$ to the verifier.

3. Verify
   - Verifier checks $\langle \tilde{\alpha}, \cdot \rangle$ against $C$.
   - If $C(x, w) = 0$, accepts; otherwise, rejects.
ZK-SNARKs from Linear PCPs

Given public \( C, x \) \( \exists \) secret \( w \) s.t. \( C(x, w) = 0 \)

\( \text{Prover} \)

\( \text{Verifier} \)

\( \text{Setup} \)

\( \text{pkc} \rightarrow \text{Setup} \rightarrow \text{Prover} \rightarrow \text{Verifier} \rightarrow \text{vkC} \)

\( \text{Prover} \leftarrow \text{Setup} \rightarrow \text{Verifier} \)

\text{libsark's implementation of [Groth EUROCRYPT '16]}
**ZK-SNARKs from Linear PCPs**

Given public $C, x$, exists secret $w$ s.t. $C(x, w) = 0$

- **Prover**
  - Input $x$
  - Witness $w$
  - Computes $\langle \vec{\alpha}, \cdot \rangle$
  - Hom Eval

- **Setup**
  - Proving key $pk_c$
  - Arithmetic circuit $C$
  - Verification key $vk_c$

- **Verifier**
  - 3 group elts (128 bytes)
  - ZeroTest $D$

**libsnark's implementation** of [Groth EUROCRYPT '16]
ZK-SNARKs from Linear PCPs

Given public $C, x \in \mathbb{Z}$, secret $w$ s.t. $C(x, w) = 0$.

3 group elts (128 bytes)
3 pairings (3 ms)

libsnowk's implementation of [Groth EUROCRYPT '16]
ZK-SNARKs from Linear PCPs

Given public $C,x$ such that $C(x,w)=0$

Given public $C,x$

Verification key $v_k$ of $C(x,w)=0$

3 group elts (128 bytes)

3 pairings (3 ms)

Prover's key $pk$ of $C(x,w)=0$

0.1ms / gate

1KB / gate

Proof system

$P$:

$\langle \vec{\alpha}, \cdot \rangle$

$\text{Hom Eval}$

Verifier $D$

$\text{ZeroTest}$

Input $x$

Witness $w$

Proving key $pk_c$ of $C(x,w)=0$

Verification key $v_k_c$ of $C(x,w)=0$

ZK-SNARKs from Linear PCPs

$\text{Setup}$

$\text{Eval}$

$\text{Q}$

$\text{Enc}$

$\text{C}$

Arithmetic circuit

libsnark's implementation of [Groth EUROCRYPT '16]
ZK-SNARKs from Linear PCPs

Given public $C,x \in \mathbb{Z}$ secret $w$ s.t. $C(x,w) = 0$

3 group elts (128 bytes)
3 pairings (3 ms)

Prover
- $x$ input
- $w$ witness
- $\mathcal{P}$ Prover
- $\langle \tilde{\alpha}, \cdot \rangle$ Hom Eval
- FFT
- MULTIEXP

Verifier
- ZeroTest
- $D$
- 3 pairings (3 ms)

Setup
- $\text{Enc}$
- $Q$
- $\text{Enc}$
- $\text{pk}_C$
- $\text{vk}_C$

Proving key
- $\text{pk}_C$

Verification key
- $\text{vk}_C$

arithmetic circuit

0.1ms / gate
1KB / gate

libsnark's implementation of [Groth EUROCRYPT '16]
Which approach is better?
Which approach is better?

PCP

Linear PCP

Setup

pk

vk

V
Which approach is better?

PCP

Setup

PK

VK

P

Q

D

memory intensive

Linear PCP

V

Which approach is better?
Which approach is better?

- **Linear PCP**
  - Setup
  - pk
  - vk
  - V

- **PCP**
  - FFT
  - pk
  - FFT
  - MEXP
  - V

The diagram shows a comparison between Linear PCP and PCP approaches, highlighting the memory-intensive aspect of PCP.
Which approach is better?

**PCP**
- memory intensive
- slower verifier
- bigger proofs

**Linear PCP**
- faster verifier
- smaller proofs
  - 3 pairings (3ms)
  - 3 group elts (128 bytes)
Which approach is better?

**Linear PCP**
- Memory intensive
- no trapdoor

**PCP**
- Faster verifier
- Smaller proofs

- Slower verifier
- Bigger proofs

Setup
- 3 pairings (3ms)
- 3 group elts (128 bytes)
Which approach is better?

- **Linear PCP**
  - Faster verifier
  - Smaller proofs
  - 3 pairings (3ms)
  - 3 group elts (128 bytes)

- **PCP**
  - Slower verifier
  - Bigger proofs
  - 3 pairings (3ms)
  - No trapdoor

---

**FFT**

**MEXP**
Which approach is better?

**PCP**
- Memory intensive
- No trapdoor
- Slower verifier
- Bigger proofs

**Linear PCP**
- Faster verifier
- Smaller proofs
  - 3 pairings (3ms)
  - 3 group elts (128 bytes)

---

**Setup**

**FFT**

**MEXP**

**RO**

[BCGTV15]
Which approach is better?

**Linear PCP**
- Slower verifier
- Bigger proofs
- Faster verifier
- Smaller proofs

**PCP**
- Memory intensive
- No trapdoor

[CBCGTV15] ceremony deployed today
Frontiers
Authenticated Inputs
Authenticated Inputs
Authenticated Inputs

\[ x \rightarrow \text{Hash} \rightarrow h \]
Authenticated Inputs

\( \mathcal{X} \rightarrow \text{Hash} \rightarrow h \)

\( C \)

\( \text{Setup} \)

\( \text{pk}_C \leftrightarrow \text{vk}_C \)
Authenticated Inputs

\[ \text{Setup} \]

Given public \( C, h \) exists secret \( x, w \) s.t. \( C(x, w) = 0 \) & \( \text{Hash}(x) = h \)
Authenticated Inputs

\[ \text{Setup} \]

- Given public \( C, h \) there exists a secret \( x, w \) such that:
  - \( C(x, w) = 0 \)
  - \( \text{Hash}(x) = h \)

\[ \text{P} \]

\[ \text{V} \]
Authenticated Inputs

Given public $C, h$ exists a secret $x, w$ s.t.
$C(x, w) = 0$ & $\text{Hash}(x) = h$

Given public $D, h$ exists a secret $x, w$ s.t.
$D(x, w) = 0$ & $\text{Hash}(x) = h$
Authenticated Inputs

\[ \text{given public } C, h \quad \exists \text{ secret } x, w \text{ s.t. } C(x, w) = 0 \text{ & } \text{Hash}(x) = h \]

\[ \text{given public } D, h \quad \exists \text{ secret } x, w \text{ s.t. } D(x, w) = 0 \text{ & } \text{Hash}(x) = h \]

\text{generic uses of ZK-SNARKs are expensive}
Authenticated Inputs

\[
\text{Hash} \xrightarrow{} h
\]

\[
\begin{align*}
\text{Setup} & \quad \text{pk}_C \\
\text{Setup} & \quad \text{pk}_D \\
\text{P} & \quad \text{Setup} \\
\end{align*}
\]

\[
\begin{align*}
\exists \text{secret } x, w \text{ s.t. } C(x, w) = 0 & \quad \text{Given public } \text{C}, h \\
\text{D}(x, w) = 0 & \quad \text{Given public } \text{D}, h
\end{align*}
\]

- generic uses of ZK-SNARKs are expensive
- **better**: co-design Hash and SNARK [FFGKOP CCS '16]
Authenticated Inputs

Given public $C, h \exists$ secret $x, w$ s.t. $C(x, w) = 0$ & $\text{Hash}(x) = h$

$\text{Setup}\xrightarrow{C} \text{Setup}\xrightarrow{D}$

• generic uses of ZK-SNARKs are expensive
• better: co-design Hash and SNARK [FFGKOP CCS '16]
• open: preserve ZK?
Post-Quantum Security
Post-Quantum Security
Post-Quantum Security

Looks solid.
Post-Quantum Security

Linear PCP

Looks solid.
Post-Quantum Security

Linear PCP

Looks solid.

Based on hardness of DLOG in EC groups.
Post-Quantum Security

Looks solid.

Based on hardness of DLOG in EC groups.

[BIWW EUROCRYPT '17]
Post-Quantum Security

Based on hardness of DLOG in EC groups.

Lattice-based privately-verifiable ZK-SNARK

[BISW EUROCRYPT '17]
Post-Quantum Security

Linear PCP

Post-Quantum Security

Looks solid.

Based on hardness of DLOG in EC groups.

[BISW EUROCRYPT '17]
Lattice-based privately-verifiable ZK-SNARK

OPEN: public?
PCP-Based ZK-SNARKs
PCP-Based ZK-SNARKs
PCP-Based ZK-SNARKs

✓ succinct
PCP-Based ZK-SNARKs

✓ succinct
✓ non-interactive
PCP-Based ZK-SNARKs

✓ succinct
✓ non-interactive
✓ no setup
PCP-Based ZK-SNARKs

✓ succinct
✓ non-interactive
✓ no setup
✓ post-quantum secure
PCP-Based ZK-SNARKs

- **✓** succinct
- **✓** non-interactive
- **✓** no setup
- **✓** post-quantum secure

😭 terrible concrete efficiency
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]
Interactive Oracle Proofs

[BCS16][RRR16]

The verifier can simultaneously leverage randomness, interaction, and probabilistic checking.
ZK-SNARKs From IOPs
ZK-SNARKs From IOPs

Probabilistically Checkable Proof

Zero Knowledge SNARK

[Micali94]
ZK-SNARKs From IOPs

Probabilistically Checkable Proof

Interactive Oracle Proof

Zero Knowledge SNARK

[Micali94]

[BCS16]
ZK-SNARKs From IOPs

Probabilistically Checkable Proof

Interactive Oracle Proof

Zero Knowledge SNARK

Q: any efficiency gains?
IOPs are more efficient than PCPs
IOPs are more efficient than PCPs
IOPs are more efficient than PCPs

best proof length
without ZK
IOPs are more efficient than PCPs

- quasilinear
  - [BS08][Din07]

- best proof length
  - without ZK
IOPs are more efficient than PCPs

- Quasilinear: [BS08][Din07]
- Best proof length without ZK: [BS08][Din07]
- Linear: [BCGRS16]
IOPs are more efficient than PCPs

quasilinear [BS08][Din07]

best proof length without ZK

best proof length with ZK

linear [BCGRS16]
IOPs are more efficient than PCPs

quasilinear \([BS08][Din07]\)  

polynomial \([KPT97]\)  

best proof length without ZK  

best proof length with ZK  

linear \([BCGRS16]\)
IOPs are more efficient than PCPs

**quasilinear**
[BS08][Din07]

**polynomial**
[KPT97]

**best proof length**
without ZK

**best proof length**
with ZK

**linear**
[BCGRS16]

**quasilinear**
[BCGV16]
IOPs are more efficient than PCPs

- IOPs are more efficient than PCPs:
  - **quasilinear**
    - [BS08][Din07]
  - **polynomial**
    - [KPT97]
  - **best proof length without ZK**
    - [BCGRS16]
  - **linear**
    - [BCGRS16]
  - **quasilinear**
    - [BCGV16]
  - **cheaper ZK...**
    - [BCFGRS16][BCFS17]
IOPs are more efficient than PCPs

IOPs are more efficient than PCPs

Encouraging progress, and it already improved working prototypes.
IOPs are more efficient than PCPs

- quasilinear: [BS08][Din07]
- polynomial: [KPT97]

Best proof length:
- without ZK: linear: [BCGRS16]
- with ZK: quasilinear: [BCGV16]

Cheaper ZK:
- [BCFGRS16][BCFS17]

Encouraging progress, and it already improved working prototypes.

Still more research is needed for practical deployment.
Cryptographic Proofs
Mathematics

Complexity Theory

- low-degree testing
- additive combinatorics
- coding theory
- Fourier analysis

Cryptography

- interactive proofs
- zero knowledge
- linear-only encryption
- probabilistically checkable proofs
- function commitments
- privacy-preserving payments

Cryptographic Proofs
Thanks!

I know \( x \) s.t. \( y = F(x) \) proof
A Simple Linear PCP

**Def:** The language $\mathcal{L}$ consists of tuples $(p_1, \ldots, p_m)$ where each $p_i$ is a quadratic polynomial over $F$ in $n$ variables such that there is an assignment $w=(w_1, \ldots, w_n)$ such that $p_1(w) = \ldots = p_m(w) = 0$.

**Theorem:**
The language $\mathcal{L}$ has a linear PCP with
- proof length $(n+1)^2$,
- query complexity 3,
- soundness error $2/|F|$. 

A Simple Linear PCP
Bundling. Let $r_1, \ldots, r_m \in \mathbb{F}$ be random and $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n$.
If $p_1(\omega) = \cdots = p_m(\omega) = 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability 1.
If $\exists j \in [m]$ s.t. $p_j(\omega) \neq 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability $\leq 1/|\mathbb{F}|$. 

F. Author, S. Another (Universities of Somewhere and Elsewhere)
A Simple Linear PCP

**Bundling.** Let $r_1, \ldots, r_m \in \mathbb{F}$ be random and $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n$. If $p_1(\omega) = \cdots = p_m(\omega) = 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability 1. If $\exists j \in [m]$ s.t. $p_j(\omega) \neq 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability $\leq 1/|\mathbb{F}|$.

**Prover.** Given an assignment $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n$, the prover writes the linear function:

$$\bar{\alpha} := \begin{pmatrix}
1 & \omega_1 & \omega_2 & \cdots & \omega_n \\
\omega_1 & \omega_1^2 & \omega_1 \omega_2 & \cdots & \omega_1 \omega_n \\
\omega_2 & \omega_2 \omega_1 & \omega_2^2 & \cdots & \omega_2 \omega_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\omega_n & \omega_n \omega_1 & \omega_n \omega_2 & \cdots & \omega_n^2
\end{pmatrix}.$$
A Simple Linear PCP

Bundling. Let \( r_1, \ldots, r_m \in \mathbb{F} \) be random and \( \omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n \).
If \( p_1(\omega) = \cdots = p_m(\omega) = 0 \) then \( \sum_{i=1}^{m} r_i p_i(\omega) = 0 \) with probability 1.
If \( \exists j \in [m] \) s.t. \( p_j(\omega) \neq 0 \) then \( \sum_{i=1}^{m} r_i p_i(\omega) = 0 \) with probability \( \leq 1/|\mathbb{F}| \).

Prover. Given an assignment \( \omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n \), the prover writes the linear function:
\[
\vec{\alpha} := \begin{pmatrix}
1 & \omega_1 & \omega_2 & \cdots & \omega_n \\
\omega_1 & \omega_1^2 & \omega_1\omega_2 & \cdots & \omega_1\omega_n \\
\omega_2 & \omega_2\omega_1 & \omega_2^2 & \cdots & \omega_2\omega_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\omega_n & \omega_n\omega_1 & \omega_n\omega_2 & \cdots & \omega_n^2
\end{pmatrix}
\]

Verifier. The verifier has oracle access to some linear function
\[
\vec{\alpha} = \begin{pmatrix}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,n} \\
\alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\
\alpha_{2,0} & \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_{n,0} & \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,n}
\end{pmatrix}
\]

and thinks of each quadratic polynomial \( p_i \) as
\[
p_i = \begin{pmatrix}
p_{i,0,0} & p_{i,0,1} & p_{i,0,2} & \cdots & p_{i,0,n} \\
0 & p_{i,1,1} & p_{i,1,2} & \cdots & p_{i,1,n} \\
0 & 0 & p_{i,2,2} & \cdots & p_{i,2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & p_{i,n,n}
\end{pmatrix}
\]
**A Simple Linear PCP**

**Bundling.** Let $r_1, \ldots, r_m \in \mathbb{F}$ be random and $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n$. If $p_1(\omega) = \cdots = p_m(\omega) = 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability 1. If $\exists j \in [m]$ s.t. $p_j(\omega) \neq 0$ then $\sum_{i=1}^m r_i p_i(\omega) = 0$ with probability $\leq 1/|\mathbb{F}|$.

**Prover.** Given an assignment $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n$, the prover writes the linear function:

$\vec{\alpha} := \begin{pmatrix} 1 & \omega_1 & \omega_2 & \cdots & \omega_n \\ \omega_1 & \omega_1^2 & \omega_1 \omega_2 & \cdots & \omega_1 \omega_n \\ \omega_2 & \omega_2 \omega_1 & \omega_2^2 & \cdots & \omega_2 \omega_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n & \omega_n \omega_1 & \omega_n \omega_2 & \cdots & \omega_n^2 \end{pmatrix}.$

**Verifier.** The verifier has oracle access to some linear function

$\vec{\alpha} = \begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,n} \\ \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ \alpha_{2,0} & \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n,0} & \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,n} \end{pmatrix}$,

and thinks of each quadratic polynomial $p_i$ as

$p_i = \begin{pmatrix} p_{i,0,0} & p_{i,0,1} & p_{i,0,2} & \cdots & p_{i,0,n} \\ 0 & p_{i,1,1} & p_{i,1,2} & \cdots & p_{i,1,n} \\ 0 & 0 & p_{i,2,2} & \cdots & p_{i,2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & p_{i,n,n} \end{pmatrix}.$
A Simple Linear PCP

**Bundling.** Let \( r_1, \ldots, r_m \in \mathbb{F} \) be random and \( \omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n \).

If \( p_1(\omega) = \cdots = p_m(\omega) = 0 \) then \( \sum_{i=1}^m r_i p_i(\omega) = 0 \) with probability 1.

If \( \exists j \in [m] \) s.t. \( p_j(\omega) \neq 0 \) then \( \sum_{i=1}^m r_i p_i(\omega) = 0 \) with probability \( \leq 1/|\mathbb{F}| \).

**Prover.** Given an assignment \( \omega = (\omega_1, \ldots, \omega_n) \in \mathbb{F}^n \), the prover writes the linear function:

\[
\tilde{\alpha} := \begin{pmatrix}
1 & \omega_1 & \omega_2 & \cdots & \omega_n \\
\omega_1 & \omega_1^2 & \omega_1 \omega_2 & \cdots & \omega_1 \omega_n \\
\omega_2 & \omega_2 \omega_1 & \omega_2^2 & \cdots & \omega_2 \omega_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\omega_n & \omega_n \omega_1 & \omega_n \omega_2 & \cdots & \omega_n^2 \\
\end{pmatrix}.
\]

**Verifier.** The verifier has oracle access to some linear function

\[
\tilde{\alpha} = \begin{pmatrix}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \cdots & \alpha_{0,n} \\
\alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\
\alpha_{2,0} & \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_{n,0} & \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,n} \\
\end{pmatrix}.
\]

and thinks of each quadratic polynomial \( p_i \) as

\[
p_i = \begin{pmatrix}
p_{i,0,0} & p_{i,0,1} & p_{i,0,2} & \cdots & p_{i,0,n} \\
p_{i,1,0} & p_{i,1,1} & p_{i,1,2} & \cdots & p_{i,1,n} \\
p_{i,2,0} & p_{i,2,1} & p_{i,2,2} & \cdots & p_{i,2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & p_{i,n,n} \\
\end{pmatrix}.
\]

Verifying (cont’d). The verifier samples \( r_1, \ldots, r_m \in \mathbb{F} \) and \( s_0, \ldots, s_n \in \mathbb{F} \) at random and then generates three queries:

- \( \tilde{q}_1 := \sum_{i=1}^m r_i p_i; \)
- \( \tilde{q}_2 := (s_0, s_1, \ldots, s_n) \otimes (1, 0, \ldots, 0); \)
- \( \tilde{q}_3 := (s_0, s_1, \ldots, s_n) \otimes (s_0, s_1, \ldots, s_n). \)

Upon receiving answers \( a_1 := \langle \tilde{\alpha}, \tilde{q}_1 \rangle, a_2 := \langle \tilde{\alpha}, \tilde{q}_2 \rangle, a_3 := \langle \tilde{\alpha}, \tilde{q}_3 \rangle \), check that

\[
a_1 = 0 \quad \text{and} \quad a_2 = a_3.
\]

**Proof.** Observe that:

- \( a_1 = \langle \tilde{\alpha}, \tilde{q}_1 \rangle = \langle \tilde{\alpha}, \sum_{i=1}^m r_i \cdot p_i \rangle = \sum_{i=1}^m r_i \cdot \langle \tilde{\alpha}, p_i \rangle; \)
- \( a_2 = \langle \tilde{\alpha}, \tilde{q}_2 \rangle = \sum_{i=0}^n \alpha_{0,i} \cdot s_i, \) so that \( a_2^2 = \sum_{i,j=0}^n \alpha_{0,i} \alpha_{0,j} \cdot s_i s_j; \)
- \( a_3 = \langle \tilde{\alpha}, \tilde{q}_3 \rangle = \sum_{i,j=0}^n \alpha_{i,j} \cdot s_i s_j. \)

Distinguish between two cases:

1. There are \( i,j \) such that \( \alpha_{i,j} \neq \alpha_{i} \alpha_{j}. \)

   Then \( \Pr_{s_0, \ldots, s_n} [a_2^2 = a_3] \leq 2/|\mathbb{F}|. \)

2. For all \( i,j \) it holds that \( \alpha_{i,j} = \alpha_{i} \alpha_{j}. \)

   Thus there is \( (\omega_0, \ldots, \omega_n) \) such that \( \tilde{\alpha} = (\omega_0, \ldots, \omega_n) \otimes (\omega_0, \ldots, \omega_n). \)

   WLOG can assume that \( \omega_0 = 1. \)

   Then \( \langle \tilde{\alpha}, p_i \rangle = p_i(\omega) \) for every \( i \), so that \( a_1 = \sum_{i=1}^m r_i p_i(\omega). \)