The Zero-Knowledge Proof Revolution
interesting insights about

Alan Szepieniec
alan@asdm.gmbh
Revolution
Revolution

SNARK
Revolution

- ZCash
- STARK
- SNARK
- Monero

Graph showing the growth of ZCash, STARK, and Monero from 2008 to 2020.
Revolution
Questions

Why is there a ZKP Revolution?

What does it mean for me / Bitcoin / cryptocurrencies?

What can ZKPs achieve?

What does the future hold?
Questions

1. Philosophy

Why is there a ZKP Revolution?

What does it mean for me / Bitcoin / cryptocurrencies?

How do ZKPs work?

What can ZKPs achieve?

What does the future hold?
Questions

1. Philosophy

Why is there a ZKP Revolution?

What does it mean for me / Bitcoin / cryptocurrencies?

2. Technology

How do ZKPs work?

What can ZKPs achieve?

What does the future hold?
Questions

1. Philosophy
   - Why is there a ZKP Revolution?
   - What does it mean for me / Bitcoin / cryptocurrencies?
   - How do ZKPs work?

2. Technology
   - What can ZKPs achieve?
   - How do ZKPs work? (Crossed out)

3. Applications
   - What does the future hold?
1. PHILOSOPHY
Motivation
Motivation

what is money?
Motivation

what is money?

...
Motivation

what is money?

... ... ...

cryptography

natural medium is the for money
Motivation

what is money?

... cryptography

is the natural medium

for money
Motivation

what is money?

... cryptography is the natural medium for money
<table>
<thead>
<tr>
<th>Function</th>
<th>Properties</th>
<th>Subjective</th>
<th>Context-dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium of exchange</td>
<td>practical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store of value</td>
<td>recursive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit of account</td>
<td>not subjective</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Most marketable good</td>
<td>subjective</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Money?

what you buy stuff with
What is Money?

✓ practical
what you buy stuff with
× recursive

✓ medium of exchange
✓ store of value
✓ unit of account

∼ context-dependent
x subjective

the most marketable good
What is Money?

- practical
- what you buy stuff with
- recursive

{ medium of exchange
  store of value
  unit of account }
What is Money?

- what you buy stuff with
- practical

✓ medium of exchange
✓ store of value
✓ unit of account

- functions
- not properties
What is Money?

✓ practical
what you buy stuff with
✓ recursive

medium of exchange
store of value
unit of account
✓ functions
× not properties
× subjective
What is Money?

- practical
  - what you buy stuff with
  - not recursive

- functions
  - not properties
  - subjective

- medium of exchange
- store of value
- unit of account

the most marketable good
What is Money?

What you buy stuff with
✓ practical

✓ functions
× not properties
× subjective

✓ medium of exchange
✓ store of value
✓ unit of account

the most marketable good
What is Money?

✓ practical

what you buy stuff with

× recursive

✓ functions

× not properties

× subjective

the most marketable good

~ context-dependent
What is Money?

✓ practical
what you buy stuff with
× recursive

✓ functions
× not properties
× subjective

medium of exchange
store of value
unit of account

✓ subjective
the most marketable good
∼ context-dependent
× subjective
Robinson Crusoe

![Graph showing happiness over time](image)
Robinson Crusoe

- ✓ economize
- ✓ capital investment
  - × money
Robinson and Friday

- breads
- fish
- division of labor
- trust
- money
Robinson and Friday

- breads
- fish

- division of labor
- trust
- money
Robinson and Friday

✓ division of labor
✓ ~ trust
× money
Larger Economies Need Money
Larger Economies Need Money

trust

![Diagram](attachment://diagram.png)
Larger Economies Need Money

trust

coincidence of wants
Larger Economies Need Money

trust

coincidence of wants

coordination
Larger Economies Need Money

trust

coincidence of wants

coordination

money fixes this
Larger Economies Need Money

trust

coincidence of wants

coordination

money fixes this
Larger Economies Need Money

trust

coincidence of wants

coordination

money fixes this

trust

coincidence of wants

coordination

money fixes this
Larger Economies Need Money

trust

coincidence of wants

coordination

↑

money fixes this
Larger Economies Need Money

trust \rightarrow adversarial context ✓

coincidence of wants

coordination

\uparrow

money fixes this
Larger Economies Need Money

trust $\rightarrow$ adversarial context ✓

coincidence of wants $\rightarrow$ medium of exchange ✓

coordination

$\uparrow$

money fixes this
Larger Economies Need Money

- trust → adversarial context ✓
- coincidence of wants → medium of exchange ✓
- coordination
  - money fixes this
Larger Economies Need Money

trust $\rightarrow$ adversarial context ✓

coincidence of wants $\rightarrow$ medium of exchange ✓

coordination $\rightarrow$ carrier of information ✓

money fixes this

cryptography fixes this
Objectives of Cryptography
Objectives of Cryptography

CONFIDENTIALITY

INTEGRITY

AUTHENTICITY

VERIFIABILITY
Objectives of Cryptography

CONFIDENTIALITY

INTEGRITY

AUTHENTICITY

for money?
Objectives of Cryptography

CONFIDENTIALITY

INTEGRITY

AUTHENTICITY

for money?
Objectives of Cryptography

CONFIDENTIALITY

INTEGRITY

AUTHENTICITY

for money?
Objectives of Cryptography

CONFIDENTIALITY

INTEGRITY

VERIFIABILITY ✓ ✓ ✓

AUTHENTICITY

for money?
Objectives of Cryptography

- **CONFIDENTIALITY**
- **INTEGRITY**
- **AUTHENTICITY**
- **VERIFIABILITY** ✓ ✓ ✓

for money?
2. BASICS of ZERO-KNOWLEDGE PROOFS
Proof Systems in Mathematics
Proof Systems in Mathematics

in theory:

mathematician

mathematician

✓

π

resp

✓/ x
Proof Systems in Mathematics

in theory:

- proposition
  - mathematician

in practice:

- proposition
  - mathematician

π ✓
resp ✓ / x
Proof Systems in Mathematics

in theory:

\begin{center}
\begin{tikzcd}
\text{proposition} \arrow{d} \& \text{proof } \pi \& \text{proposition} \\
\text{mathematician} \& \text{mathematician} \& 
\end{tikzcd}
\end{center}
Proof Systems in Mathematics

in theory:

mathematician → proposition → proof $\pi$ → mathematician

✓
Proof Systems in Mathematics

in theory:

\[
\text{mathematician} \xrightarrow{\text{prop}} \text{mathematician} \xrightarrow{\pi} \text{mathematician} \xrightarrow{\checkmark}
\]

in practice:
Proof Systems in Mathematics

in theory:

proposition \downarrow \quad \text{proof } \pi \quad \downarrow \text{proof } \pi

mathematician \quad \longrightarrow \quad \text{mathematician} \quad \longrightarrow \quad \checkmark

in practice:

proposition \downarrow \quad \pi

mathematician \quad \longrightarrow \quad \text{mathematician} \quad \longrightarrow
Proof Systems in Mathematics

in theory:

```
proposition

mathematician
```

proof $\pi$

```
proposition

mathematician
```


in practice:

```
proposition

mathematician
```

$\pi$

```
proposition

mathematician
```

?

resp

?
Proof Systems in Computer Science
Proof Systems in Computer Science

in theory:

\[
\pi = (p_1, p_2, \ldots)
\]
Proof Systems in Computer Science

- **in theory:**
  - computational proposition
  - prover (program)
  - $p_1$
  - verifier (program)

- **in practice:**
  - computational proposition
  - π = (p₁, p₂, ...)
  - ✓ / x
Proof Systems in Computer Science

in theory:

prover
(program)

|$\Downarrow$

computational proposition

$p_1$

$q_1$

$\leftarrow$

$\rightarrow$

verifier
(program)

computational proposition

$p_2$

$q_2$

$\leftarrow$

$\rightarrow$

✓ / x
Proof Systems in Computer Science

in theory:

computational proposition

prover (program)

\[ p_1 \]

\[ q_1 \]

\[ p_2 \]

\[ q_2 \]

\[ \cdots \]

verifier (program)

\[ \checkmark \] /

\[ \times \]

in practice:
Proof Systems in Computer Science

in theory:

prover (program) → prover (program)

verifier (program) ← verifier (program)

computational proposition

p₁

q₁

p₂

q₂

π = (p₁, p₂, . . .)

in practice:

prover (program) → prover (program)

verifier (program) ← verifier (program)

computational proposition

✓ / ✗
Computational Propositions

Program $A$:
- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$

Naïve Verifier ($(x, y, z)$):
- run $A(x, y)$ for $T$ steps
- test output against $z$

Prover ($(x, y)$):
- generate $\pi$
- while running $z \leftarrow A(x, y)$

Resource-Constrained Verifier ($(x, z, \pi)$):
- no access to secrets
- running time $t \ll T$
- zero-knowledge proof
- succinct proof (SNARK)
Computational Propositions

Program $A$

Naïve Verifier ($x, y, z$):
- run $A(x, y)$ for $T$ steps
- test output against $z$

Prover ($x, y$):
- generate $\pi$
- while running $z \leftarrow A(x, y)$

Resource-Constrained Verifier ($x, z, \pi$):
- no access to secrets
- running time $t \ll T$
- zero-knowledge proof
  succinct proof (SNARK)
Computational Propositions

Program $A$

- takes public input $x$
Computational Propositions

Program $A$

- takes public input $x$
- takes secret input $y$
Computational Propositions

Program $A$

- takes public input $x$
- takes secret input $y$
- outputs public value $z$
Computational Propositions

Program $A$

- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$
Computational Propositions

Program $A$

- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$

Naïve Verifier $(x, y, z)$:

- run $A(x, y)$ for $T$ steps
- test output against $z$
Computational Propositions

Program $A$

- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$

Naïve Verifier $(x, y, z)$:

- run $A(x, y)$ for $T$ steps
- test output against $z$

Prover $(x, y)$

- generate $\pi$
- while running $z \leftarrow A(x, y)$
Computational Propositions

Program \( A \)
- takes public input \( x \)
- takes secret input \( y \)
- outputs public value \( z \)
- runs in time \( T \)

Naïve Verifier \((x, y, z)\):
- run \( A(x, y) \) for \( T \) steps
- test output against \( z \)

Prover \((x, y)\)
- generate \( \pi \)
- while running \( z \leftarrow A(x, y) \)

Resource-Constrained Verifier \((x, z, \pi)\)
- no access to secrets
- running time \( t \ll T \)
Computational Propositions

Program $A$
- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$

Naïve Verifier $(x, y, z)$:
- run $A(x, y)$ for $T$ steps
- test output against $z$

Prover $(x, y)$
- generate $\pi$
- while running $z \leftarrow A(x, y)$

Resource-Constrained Verifier $(x, z, \pi)$
- no access to secrets
- running time $t \ll T$
Computational Propositions

Program $A$
- takes public input $x$
- takes secret input $y$
- outputs public value $z$
- runs in time $T$

Naïve Verifier $(x, y, z)$:
- run $A(x, y)$ for $T$ steps
- test output against $z$

Prover $(x, y)$
- generate $\pi$
- while running $z \leftarrow A(x, y)$

Resource-Constrained Verifier $(x, z, \pi)$
- no access to secrets → zero-knowledge proof
- running time $t \ll T$ → succinct proof (SNARK)
Succinct and Zero-Knowledge — Intuition
Succinct and Zero-Knowledge — Intuition

1. select row, column, or block
2. shuffle
3. flip and check
Succinct and Zero-Knowledge — Intuition

1. select row, column, or block
2. shuffle
3. flip and check
Succinct and Zero-Knowledge — Intuition

1. select row, column, or block

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
Succinct and Zero-Knowledge — Intuition

1. select row, column, or block
2. shuffle
### Succinct and Zero-Knowledge — Intuition

1. Select row, column, or block
2. Shuffle
3. Flip and check
Succinct and Zero-Knowledge — Intuition

1. select row, column, or block
2. shuffle
3. flip and check

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>4</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
III Applications
III Applications A) to cryptocurrencies
Schnorr zero-knowledge proof

secret key: \( x \in \mathbb{Z}_p \)

public key: \( P = x \times G \)

protocol:

\( V_r \leftarrow \mathbb{Z}_p \)
\( R \leftarrow r \times G \)
\( e_s \leftarrow \text{ex} + r \times s \times G \)?
\( e \times P + R \)

signature scheme

\( e \leftarrow H(R \parallel P \parallel m) \)

signature: \((R, s)\)
Schnorr zero-knowledge proof

secret key: $x \in \mathbb{Z}_p$

public key: $P = x \times G$
Schnorr zero-knowledge proof

secret key: $x \in \mathbb{Z}_p$

public key: $P = x \times G$

protocol:

$\mathcal{P}$

$\mathcal{V}$
Schnorr

Schnorr zero-knowledge proof

secret key: \( x \in \mathbb{Z}_p \)

public key: \( P = x \times G \)

protocol:

\[ P \]
\[ \forall \]
\[ r \leftarrow \mathbb{Z}_p \]
\[ R \leftarrow r \times G \]

\[ R \]
Schnorr

secret key: \( x \in \mathbb{Z}_p \)

public key: \( P = x \times G \)

protocol:

\[
\begin{align*}
\mathcal{P} & \quad \mathcal{V} \\
R & \leftarrow r \times G \\
e & \leftarrow e^{\mathsf{H}(R \parallel P \parallel m)} \\
e & \leftarrow e^{\mathbb{Z}_p} \\
\end{align*}
\]
Schnorr zero-knowledge proof

- **Secret key**: $x \in \mathbb{Z}_p$

- **Public key**: $P = x \times G$

**Protocol**:

\[ P \quad \forall \]
\[ r \leftarrow \mathbb{Z}_p \]
\[ R \leftarrow r \times G \]
\[ e \leftarrow H(R \parallel P \parallel m) \]
\[ s \leftarrow ex + r \]
\[ e \leftarrow \mathbb{Z}_p \]
Schnorr zero-knowledge proof

secret key: $x \in \mathbb{Z}_p$

public key: $P = x \times G$

protocol:

\[ P \quad \rightarrow \quad V \]

\[ r \leftarrow \mathbb{Z}_p \]

\[ R \leftarrow r \times G \]

\[ R \]

\[ e \leftarrow \mathbb{Z}_p \]

\[ e \]

\[ s \leftarrow ex + r \]

\[ s \]

\[ s \times G \overset{?}{=} e \times P + R \]
Schnorr zero-knowledge proof
signature scheme

secret key: \( x \in \mathbb{Z}_p \)

public key: \( P = x \times G \)

protocol:

\[ P \quad \forall \]

\[ r \leftarrow \mathbb{Z}_p \]

\[ R \leftarrow r \times G \]

\[ e \leftarrow H(R \parallel P \parallel m) \]

\[ s \leftarrow ex + r \]

\[ s \times G \overset{?}{=} e \times P + R \]
Schnorr zero-knowledge proof

signature scheme

secret key: \( x \in \mathbb{Z}_p \)

public key: \( P = x \times G \)

protocol:

\( \mathcal{P} \) \( \nabla \)

\( r \leftarrow \mathbb{Z}_p \)
\( R \leftarrow r \times G \)

\( R \leftarrow \)

\( e \leftarrow \mathbb{Z}_p \)
\( e \leftarrow H(R\|P\|m) \)

\( s \leftarrow ex + r \)

\( s \times G = e \times P + R \)
Schnorr zero-knowledge proof

signature scheme

signature: \((R, s)\)

secret key: \(x \in \mathbb{Z}_p\)

public key: \(P = x \times G\)

protocol:

\[ P \quad V \]

\[ r \leftarrow \mathbb{Z}_p \]
\[ R \leftarrow r \times G \]

\[ e \leftarrow H(R || P || m) \]

\[ s \leftarrow ex + r \]

\[ s \times G = e \times P + R \]
Multisig Protocol

Alice

Bob

Joint

\[ \text{Alice} \times \text{Bob} \times \text{Honest-but-curious security} \]

\[ \text{Malicious security} \]

\[ \text{ZKPs} \]
Multisig Protocol

Alice

<table>
<thead>
<tr>
<th>sk</th>
<th>pk</th>
<th>sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_a$</td>
<td>$x_a \times G$</td>
<td>$(r_a \times G, ex_a + r_a)$</td>
</tr>
</tbody>
</table>
# Multisig Protocol

<table>
<thead>
<tr>
<th>Alice $x_a$</th>
<th>Bob $x_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk $sk$</td>
<td>pk $pk$</td>
</tr>
<tr>
<td>$x_a$</td>
<td>$x_a \times G$</td>
</tr>
<tr>
<td>$x_b$</td>
<td>$x_b \times G$</td>
</tr>
<tr>
<td>sig $sig$</td>
<td></td>
</tr>
<tr>
<td>$(r_a \times G, ex_a + r_a)$</td>
<td>$(r_b \times G, ex_b + r_b)$</td>
</tr>
</tbody>
</table>

- **honest-but-curious security**
- **malicious security**
- **ZKPs**
## Multisig Protocol

<table>
<thead>
<tr>
<th></th>
<th>sk</th>
<th>pk</th>
<th>sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$x_a$</td>
<td>$x_a \times G$</td>
<td>$(r_a \times G, ex_a + r_a)$</td>
</tr>
<tr>
<td>Bob</td>
<td>$x_b$</td>
<td>$x_b \times G$</td>
<td>$(r_b \times G, ex_b + r_b)$</td>
</tr>
<tr>
<td>joint</td>
<td>$x_a + x_b$</td>
<td>$(x_a + x_b) \times G$</td>
<td>$((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)$</td>
</tr>
</tbody>
</table>

The protocol involves Alice and Bob who each have a secret key $x_a$ and $x_b$, respectively. They both compute $x_a \times G$ and $x_b \times G$ respectively. The joint computation involves adding their keys and then multiplying the result by $G$. The signature $(r_a \times G, ex_a + r_a)$ and $(r_b \times G, ex_b + r_b)$ are sent to the other party for verification. The final signature is verified using the equation $e(x_a + x_b) + r_a + r_b$. The protocol ensures that both parties must agree to sign a message with their combined secret keys, providing enhanced security compared to single-key signing schemes.
### Multisig Protocol

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
<th>joint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sk</strong></td>
<td>$x_a$</td>
<td>$x_b$</td>
<td>$x_a + x_b$</td>
</tr>
<tr>
<td><strong>pk</strong></td>
<td>$x_a \times G$</td>
<td>$x_b \times G$</td>
<td>$(x_a + x_b) \times G$</td>
</tr>
<tr>
<td><strong>sig</strong></td>
<td>$(r_a \times G, ex_a + r_a)$</td>
<td>$(r_b \times G, ex_b + r_b)$</td>
<td>$((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)$</td>
</tr>
</tbody>
</table>

\[ e = H((R_a + R_b) || P || m) \]
Multisig Protocol

<table>
<thead>
<tr>
<th></th>
<th>sk</th>
<th>pk</th>
<th>sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$x_a$</td>
<td>$x_a \times G$</td>
<td>$(r_a \times G, ex_a + r_a)$</td>
</tr>
<tr>
<td>Bob</td>
<td>$x_b$</td>
<td>$x_b \times G$</td>
<td>$(r_b \times G, ex_b + r_b)$</td>
</tr>
<tr>
<td>joint</td>
<td>$x_a + x_b$</td>
<td>$(x_a + x_b) \times G$</td>
<td>$((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)$</td>
</tr>
</tbody>
</table>

Alice     Bob

$e = H((R_a + R_b) || P || m)$
## Multisig Protocol

<table>
<thead>
<tr>
<th></th>
<th>sk</th>
<th>pk</th>
<th>sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$x_a$</td>
<td>$x_a \times G$</td>
<td>$(r_a \times G, e x_a + r_a)$</td>
</tr>
<tr>
<td>Bob</td>
<td>$x_b$</td>
<td>$x_b \times G$</td>
<td>$(r_b \times G, e x_b + r_b)$</td>
</tr>
<tr>
<td>joint</td>
<td>$x_a + x_b$</td>
<td>$(x_a + x_b) \times G$</td>
<td>$((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)$</td>
</tr>
</tbody>
</table>

Alice

Bob

\[ e = H((R_a + R_b) || P || m) \]


## Multisig Protocol

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk</td>
<td>pk</td>
<td>sig</td>
</tr>
<tr>
<td>$x_a$</td>
<td>$x_a \times G$</td>
<td>$(r_a \times G, ex_a + r_a)$</td>
</tr>
<tr>
<td>$x_b$</td>
<td>$x_b \times G$</td>
<td>$(r_b \times G, ex_b + r_b)$</td>
</tr>
<tr>
<td>$x_a + x_b$</td>
<td>$(x_a + x_b) \times G$</td>
<td>$e((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)$</td>
</tr>
</tbody>
</table>

Alice sends $\pi_a$: $R_a$, $\sigma_a$.
Bob sends $\pi_b$: $R_b$, $\sigma_b$.

$e = H((R_a + R_b) || P || m)$
Multisig Protocol

Alice

\begin{align*}
& x_a \\
& x_a \times G \\
& x_a + x_b \\
& (x_a + x_b) \times G \\
& ((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)
\end{align*}

Bob

\begin{align*}
& x_b \\
& x_b \times G \\
& x_a + x_b \\
& (x_a + x_b) \times G \\
& ((r_b \times G, e) + r_b)
\end{align*}

Joint

\begin{align*}
& x_a + x_b \\
& (x_a + x_b) \times G \\
& ((r_a + r_b) \times G, e(x_a + x_b) + r_a + r_b)
\end{align*}

Alice sends \( R_a, \pi_a \) to Bob, and Bob sends \( R_b, \pi_b \) to Alice. Then Alice computes \( s_a \) and Bob computes \( s_b \). The honest-but-curious security is provided by ZKPs, while malicious security is not.

\[ e = H((R_a + R_b) \| P \| m) \]
Zero-Knowledge Contingent Payments

Alice (information)

Bob ($$\$\$\) 

Blockchain

C = Enc_{K}(information), h = H(K), π

TX needs: 
− valid signature under pk_A 
− preimage K' such that H(K') = h 

with sig_A and K
Zero-Knowledge Contingent Payments

Alice
(information)

Bob
($$$)

Blockchain

\[ C = \text{Enc}_K(\text{information}), \quad h = H(K) \]

TX needs:
- valid signature under \( pk_A \)
- preimage \( K' \) such that \( H(K') = h \)
Zero-Knowledge Contingent Payments

Blockchain

Alice
(information)

Bob
($$$)

TX needs:
- valid signature under pk_A
- preimage K' such that H(K') = h_t with sig_A and K
Zero-Knowledge Contingent Payments

Alice (information)

Blockchain

Bob ($$$)

\[ C = \text{Enc}_K(\text{information}), \ h = \text{H}(K), \ \pi \]
Zero-Knowledge Contingent Payments

$C = \text{Enc}_K(\text{information}), \ h = H(K), \ \pi$

Alice
(information)

Blockchain

Bob
($\$$)

$TX$ needs:
- valid signature under $pk_A$
- preimage $K'$ such that $H(K') = h$
tx with sig $A$ and $K$
Zero-Knowledge Contingent Payments

\[ C = \text{Enc}_K(\text{information}), \ h = \text{H}(K), \ \pi \]

TX needs:
- valid signature under \( \text{pk}_A \)
- preimage \( K' \) such that \( \text{H}(K') = h \)
Zero-Knowledge Contingent Payments

$C = Enc_K(\text{information}), \ h = H(K), \ \pi$

TX needs:
- valid signature under $pk_A$
- preimage $K'$ such that $H(K') = h$

Alice (information)  Bob ( $$$ )

$tx$ with $\text{sig}_A$ and $K$
Zero-Knowledge Contingent Payments

\[ C = \text{Enc}_K(\text{information}), \ h = H(K), \ \pi \]

TX needs:
- valid signature under \( \text{pk}_A \)
- preimage \( K' \) such that \( H(K') = h \)

Alice
(information)

Blockchain

Bob
($$$)

\[ K \]
Privacy Coins

π → "amounts are positive"

π → "amounts are positive and one of these UTXOs is the true origin"
Privacy Coins

no privacy
confidential transactions

\( \pi \leftrightarrow \text{“amounts are positive”} \)
Privacy Coins

obfuscate origin

\[ \pi \leftrightarrow \text{“amounts are positive and one of these UTXOs is the true origin”} \]
Privacy Coins — Disadvantage
Privacy Coins — Disadvantage

users

attracts scalability

destroys privacy

enable heavy duty cryptography

requires
Scalability – Four Types of Information
Scalability – Four Types of Information

- unverified witness
- state information
- state update
- verified witness
Scalability – Four Types of Information

- unverified witness
- state information
- state update
- verified witness
- utxo
Scalability – Four Types of Information

- Unverified witness
- State information
- Verified witness
- State update

- Utxo
- New tx
Scalability – Four Types of Information

- **unverified witness**
- **state information**
- **state update**
- **verified witness**

\[ \text{utxo} + \text{new tx} = \text{verify} \]
Scalability – Four Types of Information

- unverified witness
- state information
- state update
- verified witness

\[ \text{utxo} + \text{new tx} = \text{verify} \]
Scalability – Four Types of Information

- unverified witness
- state information
- state update
- verified witness

\[ \text{utxo} + \text{new tx} = \text{verified witness} \]
Scalability – Synking

initially synking ...

π initially synked.
Scalability – Synking

full node:

↑

initially

SNARK node:
Scalability – Synking

full node:

synking ...

SNARK node: \( \pi \)

initially \( \pi \) synked.
Scalability – Synking

full node:

\[ \uparrow \text{synking ...} \]
Scalability – Synking

full node:

synking ...

$\pi$

synked.
Scalability – Synking

full node:

synked.
Scalability – Synking

full node:

synked.

SNARK node:

\(\pi\) initially
Scalability – Synking

full node:

SNARK node:

\[ \pi \]

initially

\[ \pi \]

synked.

\[ \pi \]

synked.
Scalability — Aggregation
Scalability — Aggregation
Scalability — Aggregation
Scalability — Aggregation
Scalability — Aggregation
III Applications
III Applications  B) Other
Private Finance
trades

guarantee: integral order matching

hide: algorithm, volume
Private Finance

trades
  guarantee: integral order matching
  hide: algorithm, volume

lending
  guarantee: cash flow positivity,
  qualification
  hide: transaction history
Private Finance

trades

guarantee: integral order matching
hide: algorithm, volume

lending

guarantee: cash flow positivity, qualification
hide: transaction history

investment

guarantee: blacklist non-membership
hide: investor identity, source of funds, allocation
Private Finance

trades
guarantee: integral order matching
hide: algorithm, volume

lending
guarantee: cash flow positivity, qualification
hide: transaction history

investment
guarantee: blacklist non-membership
hide: investor identity, source of funds, allocation

audit
guarantee: regulatory compliance, valid accounting
hide: balance sheet
Private eGovernment
Private eGovernment

identity

guarantee: qualification

hide: personal data
Private eGovernment

identity
guarantee: qualification
hide: personal data

voting
guarantee: election integrity
hide: individual votes

surveillance
guarantee: legality,
blacklist non-membership
hide: movements, interactions,
communications
Private eGovernment

identity

guarantee: qualification
hide: personal data

voting

guarantee: election integrity
hide: individual votes

taxes

guarantee: tax compliance
hide: wealth, assets

surveillance

guarantee: legality, blacklist non-membership
hide: movements, interactions, communications
Private eGovernment

identity

guarantee: qualification
hide: personal data

voting

guarantee: election integrity
hide: individual votes

taxes

guarantee: tax compliance
hide: wealth, assets

surveillance

guarantee: legality,
blacklist non-membership
hide: movements, interactions, communications
Other
gaming

guarantee: unbiased shuffling, legal moves
hide: strategy, position, randomness
gaming

guarantee: unbiased shuffling, legal moves
hide: strategy, position, randomness

exploits and vulnerabilities

guarantee: existence of vulnerability, fair exchange
hide: vulnerability