ZKBoo: Faster Zero-Knowledge for Boolean Circuits

Irene Giacomelli, Jesper Madsen and Claudio Orlandi

Usenix Security Symposium 2016
Zero-Knowledge (ZK) Arguments

Alice

Private Input: $x$

“I know $x$ such that $y = C(x)$”

$(C \text{ and } y \text{ public})$

Bob

Output:

“yes! / no!”
In theory...

ZK protocols have many applications in designing several crypto primitives!
In theory...

ZK protocols have **many applications** in designing several crypto primitives!

- signature schemes
- user identification protocols
- electronic voting systems
- verifiable delegation of computation
- electronic payment system
- ... ... ...
In practice...

Real-world applications need **practically efficient** solutions for proving **general statement**
In practice...

Real-world applications need **practically efficient** solutions for proving **general statement**

- **SNARGs (Succinct Non-interactive ARGuments)**
  
  [Gro10, Lip12, GGPR13, Lip 13, DFGK14, GRo 15]
  [PGHR13, BCGTV13, BCTV14, CTV15, CFH+15]

- **ZKGC (zero-knowledge from garbled circuits)**
  
  [Jawurek-Kerschbaum-Orlandi 2013]
In practice...

Real-world applications need **practically efficient** solutions for proving **general statement**

- **SNARGs** (*Succinct Non-interactive ARGuments*)
  - proofs of small size, fast in verifying :-)
  - large keys needed, slower in proving :-(

- **ZKGC** (*zero-knowledge from garbled circuits*)
  [Jawurek-Kerschbaum-Orlandi 2013]
In practice...

Real-world applications need **practically efficient** solutions for proving **general statement**

- **SNARGs (Succinct Non-interactive ARGuments)**
  - proofs of small size, fast in verifying :-(
  - large keys needed, slower in proving :-(

- **ZKGC (zero-knowledge from garbled circuits)**
  - proving time is decreased :-(
  - interaction is required :-(

[2] PGHR13, BCGTV13, BCTV14, CTV15, CFH⁺15

In practice…

Real-world applications need **practically efficient** solutions for proving **general statement**

---

New!

- **ZKBoo** (*Zero-Knowledge for Boolean circuits*)
  - can be made non interactive :-)
  - fast in proving and verifying :-)
  - the size of the proof grows linearly with the circuit size :-|
Comparison for $C = \text{SHA-1}$

“I know $x$ such that $y = \text{SHA-1}(x)$”

<table>
<thead>
<tr>
<th></th>
<th>Preproc. (ms)</th>
<th>Prover (ms)</th>
<th>Verifier (ms)</th>
<th>Proof size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZKBoo</td>
<td>0</td>
<td>13</td>
<td>5</td>
<td>454840</td>
</tr>
<tr>
<td>ZKGC*</td>
<td>0</td>
<td>&gt; 19</td>
<td>&gt; 25</td>
<td>186880</td>
</tr>
<tr>
<td>Pinocchio**</td>
<td>9754</td>
<td>12059</td>
<td>8</td>
<td><strong>288</strong></td>
</tr>
</tbody>
</table>

* estimates for the proof size and lower-bounds for the runtime

**[Parno-Howell-Gentry-Raykova 2013]**
In the rest of this talk:

1. Description of the ZKBoo protocol
2. Implementation results
**Σ-Protocol**

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$
Σ-Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Sample $e \leftarrow \{0, 1\}^e$

Output: Y / N

Complete: if Alice and Bob honest and $C(x) = y$, $\Pr[\text{Bob outputs Y}] = 1$

Soundness: from $\geq 2$ accepting conversations $(a_i, e_i, z_i)$ with $e_i \neq e_j$ we can efficiently compute $x'$ s.t. $C(x') = y$

The protocol has soundness error $\epsilon$: if Alice is cheating, then $\Pr[\text{Bob outputs Y}] \leq \epsilon$ (Honest-Verifier)

ZK property: the distribution of $(a, e, z)$ does not reveal info on $x$

It can be made non-interactive! (Fiat-Shamir heuristic)
Σ-Protocol

Public data: \( C : \{0, 1\}^n \rightarrow \{0, 1\}^m \) (boolean circuit) and \( y \in \{0, 1\}^m \)

Input: \( x \) s.t. \( C(x) = y \)

Sample \( e \leftarrow \{0, 1\}^e \)

Output: Y / N

**Complete:** if Alice and Bob honest and \( C(x) = y \),
\[
\Pr[\text{Bob outputs Y}] = 1
\]
Σ-Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Output: Y / N

Soundness: from $\geq 2$ accepting conversations $(a, e_i, z_i)$ with $e_i \neq e_j$ we can efficiently compute $x'$ s.t. $C(x') = y$
**Σ-Protocol**

Public data: \( C : \{0, 1\}^n \rightarrow \{0, 1\}^m \) (boolean circuit) and \( y \in \{0, 1\}^m \)

Input: \( x \) s.t. \( C(x) = y \)

Sample \( e \leftarrow \{0, 1\}^e \)

Output: Y / N

The protocol has **soundness error** \( \epsilon \): if Alice is cheating, then \( \Pr[\text{Bob outputs } Y] \leq \epsilon \)
Σ-Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Sample $e \leftarrow \{0, 1\}^e$

Output: Y / N

(Honest-Verifier) ZK property:
the distribution of $(a, e, z)$ does not reveal info on $x$
**Σ-Protocol**

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Sample $e \leftarrow \{0, 1\}^e$

Output: Y / N

It can be made non-interactive!
(Fiat-Shamir heuristic)
$\Sigma$-Protocol Recap

\[ x \text{ s.t. } C(x) = y \]

\[ a \rightarrow e \rightarrow z \rightarrow Y / N \]

- Complete:
  if Alice honest, $\Pr[\text{Bob says } Y] = 1$

- Soundness error:
  if Alice cheats, $\Pr[\text{Bob says } Y] \leq \epsilon$

- ZK property: no info on $x$!

- 3 rounds, public coin $\rightarrow$ non-interactive
Related work:

**IKOS Construction**
(or “MPC-in-the-head”)
[Ishai-Kushilevitz-Ostrovsky-Sahai 2007]

Input: \( x \) s.t. \( C(x) = y \)

Output: Y / N

- a \( \Sigma \)-protocol with error 2/3 (not implemented!)
- ZK protocol with asymptotically good complexity;
Related work:

IKOS Construction
(or “MPC-in-the-head”)
[Ishai-Kushilevitz-Ostrovsky-Sahai 2007]

Input: \( \mathbf{x} \) s.t. \( C(\mathbf{x}) = \mathbf{y} \)

Output: Y / N

- a \( \Sigma \)-protocol with error 2/3 (not implemented!)
- ZK protocol with asymptotically good complexity;
Circuit decomposition:

**Goal:** compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$.
Circuit decomposition:

**Goal:** compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

Share, Rec and

$$\mathcal{F} = \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \}_{j=1,...,N}$$
Circuit decomposition:

Goal: compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

Share, Rec and

$\mathcal{F} = \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \}_{j=1,\ldots,N}$
Circuit decomposition:

Goal: compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

$$\mathcal{F} = \{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}_{j=1,...,N}$$
Circuit decomposition:

**Goal:** compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

$$\mathcal{F} = \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \}_{j=1,\ldots,N}$$
Circuit decomposition:

Goal: compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

Share, Rec and

$$\mathcal{F} = \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \}_{j=1,\ldots,N}$$
Circuit decomposition:

**Goal:** compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

$$\mathcal{F} = \{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}_{j=1,...,N}$$

- correctness: $y = C(x)$
Circuit decomposition:

Goal: compute $C(x)$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $x$

Let $N$ be a fixed integer, consider the following finite set of functions:

$\mathcal{F} = \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \}_{j=1,...,N}$

- correctness: $y = C(x)$
- 2-privacy: $\forall e, \forall j$ $(w_e^j, w_{e+1}^j, y_{e+2})$ doesn’t reveal info on $x$
ZKBoo Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$
ZKBoo Protocol

Public data: $C : \{0,1\}^n \rightarrow \{0,1\}^m$ (boolean circuit) and $y \in \{0,1\}^m$

Input: $x$ s.t. $C(x) = y$
ZKBoo Protocol

Public data: \( C : \{0, 1\}^n \rightarrow \{0, 1\}^m \) (boolean circuit) and \( y \in \{0, 1\}^m \)

Input: \( x \) s.t. \( C(x) = y \)

Check consistency Soundness error: \( \frac{2}{3} \)
ZKBoo Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$
ZKBoo Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$
ZKBoo Protocol

Public data: $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Check consistency

Soundness error: $2/3$
ZKBoo Protocol

Public data: $C : \{0, 1\}^n \to \{0, 1\}^m$ (boolean circuit) and $y \in \{0, 1\}^m$

Input: $x$ s.t. $C(x) = y$

Soundness error: $2/3$
Linear Decomposition:

\[ N = \text{number of gates in the boolean circuit } C \]
Linear Decomposition:

\( N = \text{number of gates in the boolean circuit } C \)

- \( \text{Share}(x) = (w_1^0, w_2^0, w_3^0) \) with \( w_1^0 \oplus w_2^0 \oplus w_3^0 = x \)

- \( \text{Rec}(y_1, y_2, y_3) = y_1 \oplus y_2 \oplus y_3 \)
Linear Decomposition:

\[ N = \text{number of gates in the boolean circuit } C \]

- \( \text{Share}(x) = (w_1^0, w_2^0, w_3^0) \) with \( w_1^0 \oplus w_2^0 \oplus w_3^0 = x \)
- \( \text{Rec}(y_1, y_2, y_3) = y_1 \oplus y_2 \oplus y_3 \)
- \( \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \} \)
Linear Decomposition:

\[ N = \text{number of gates in the boolean circuit } C \]

- \( \text{Share}(x) = (w_1^0, w_2^0, w_3^0) \) with \( w_1^0 \oplus w_2^0 \oplus w_3^0 = x \)
- \( \text{Rec}(y_1, y_2, y_3) = y_1 \oplus y_2 \oplus y_3 \)
- \( \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \} \)

\( j \)-th gate

\( f_e^{(j)}(w_e^a, w_e^b) = w_e^a \oplus w_e^b \)

\( e = 1, 2, 3 \)
Linear Decomposition:

\( N = \text{number of gates in the boolean circuit} \ C \)

- \( \text{Share}(x) = (w_1^0, w_2^0, w_3^0) \) with \( w_1^0 \oplus w_2^0 \oplus w_3^0 = x \)
- \( \text{Rec}(y_1, y_2, y_3) = y_1 \oplus y_2 \oplus y_3 \)
- \( \{ f_1^{(j)}, f_2^{(j)}, f_3^{(j)} \} \)

XOR gate
\( f_e^{(j)}(w_e^a, w_e^b) = w_e^a \oplus w_e^b \)

AND gate
\( f_e^{(j)}(w_e^a, w_e^b, w_{e+1}^a, w_{e+1}^b) = w_e^a w_e^b \oplus w_{e+1}^a w_e^b \oplus w_e^a w_{e+1}^b \oplus r_j \)

\( e = 1, 2, 3 \)
Experiments for ZKBoo

<table>
<thead>
<tr>
<th></th>
<th>SHA-1</th>
<th>SHA-256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Serial</td>
<td>Paral.</td>
</tr>
<tr>
<td>Prover (ms)</td>
<td>31.73</td>
<td>12.73</td>
</tr>
<tr>
<td>Verifier (ms)</td>
<td>22.85</td>
<td>4.39</td>
</tr>
<tr>
<td>Proof size (KB)</td>
<td>444.18</td>
<td></td>
</tr>
</tbody>
</table>

Soundness error: $2^{-80}$
(137 repetitions of ZKBoo with soundness 2/3)

SHA-1 → 11680 AND gates
SHA-256 → 25344 AND gates

Implementation available at [https://github.com/Sobuno/ZKBoo](https://github.com/Sobuno/ZKBoo)
## Experiments for ZKBoo

<table>
<thead>
<tr>
<th></th>
<th>SHA-1</th>
<th>SHA-256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Serial</td>
<td>Paral.</td>
</tr>
<tr>
<td><strong>Prover (ms)</strong></td>
<td>31.73</td>
<td>12.73</td>
</tr>
<tr>
<td><strong>Verifier (ms)</strong></td>
<td>22.85</td>
<td>4.39</td>
</tr>
<tr>
<td><strong>Proof size (KB)</strong></td>
<td>444.18</td>
<td></td>
</tr>
</tbody>
</table>

Soundness error: $2^{-80}$
(137 repetitions of ZKBoo with soundness 2/3)

SHA-1 $\rightarrow$ 11680 AND gates
SHA-256 $\rightarrow$ 25344 AND gates

Implementation available at [https://github.com/Sobuno/ZKBoo](https://github.com/Sobuno/ZKBoo)
Recap:

**ZKBoo**: a nearly practical ZK protocol that
Recap:

**ZKBoo**: a nearly practical ZK protocol that

- is non-interactive!
Recap:

ZKBoo: a nearly practical ZK protocol that

- is non-interactive!
- is implemented for SHA-1 and SHA-256!
Recap:

**ZKBoo**: a nearly practical ZK protocol that

- is non-interactive!
- is implemented for SHA-1 and SHA-256!
- has proving time much smaller than SNARGs!
Recap:

**ZKBoo**: a nearly practical ZK protocol that

- is non-interactive!
- is implemented for SHA-1 and SHA-256!
- has proving time much smaller than SNARGs!
- ... has a really cute name!!! :)


What next?

ZKBoo can work for \textit{any} circuit $C$!
(both arithmetic or boolean)
What next?

ZKBoo can work for *any* circuit $C$!
(both arithmetic or boolean)

- implement general-purpose ZKBoo;
What next?

ZKBoo can work for any circuit $C$!
(both arithmetic or boolean)

- implement general-purpose ZKBoo;
- consider another specific circuit (eg $C=$AES) and define new ad-hoc decomposition;
What next?

ZKBoo can work for *any* circuit $C$!
(both arithmetic or boolean)

- implement general-purpose ZKBoo;
- consider another specific circuit (eg $C=$AES) and define new ad-hoc decomposition;

Thanks for the attention! Questions?