Aarhus University

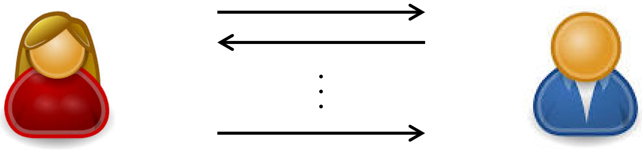
ZKBoo: Faster Zero-Knowledge for Boolean Circuits

Irene Giacomelli, Jesper Madsen and Claudio Orlandi

Usenix Security Symposium 2016

Zero-Knowledge (ZK) Arguments

Alice **Bob**



Private Input : $\mathbf{x}$

"I know $\mathbf{x}$ such that $\mathbf{y} = C(\mathbf{x})$"

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

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*)
  [Gro10, Lip12, GGPR13, Lip 13, DFGK14, GRo 15]
  [PGHR13, BCGTV13, BCTV14, CTV15, CFH$^+$15]
  - 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*)
  [Gro10, Lip12, GGPR13, Lip 13, DFGK14, GRo 15]
  [PGHR13, BCGTV13, BCTV14, CTV15, CFH$^+$15]
  - proofs of small size, fast in verifying :-)
  - large keys needed, slower in proving :-(

- ZKGC (*zero-knowledge from garbled circuits*)
  [Jawurek-Kerschbaum-Orlandi 2013]
  - proving time is decreased :-)
  - interaction is required :-(

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** = SHA-1(**x**)"

	Preproc. (ms)	Prover (ms)	Verifier (ms)	Proof size (B)
ZKBoo	0	**13**	**5**	454840
ZKGC*	0	> 19	> 25	186880
Pinocchio**	9754	12059	8	**288**

* 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

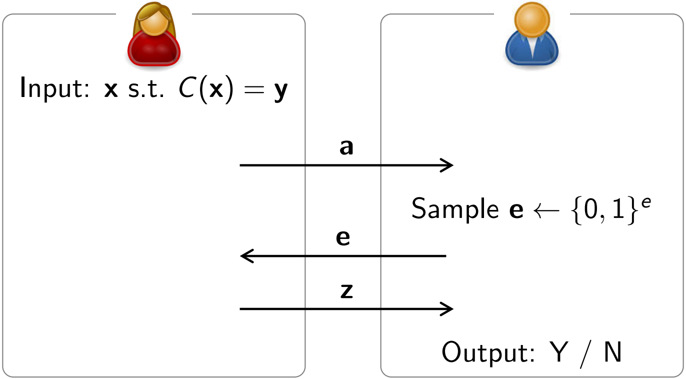
Σ-Protocol

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

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

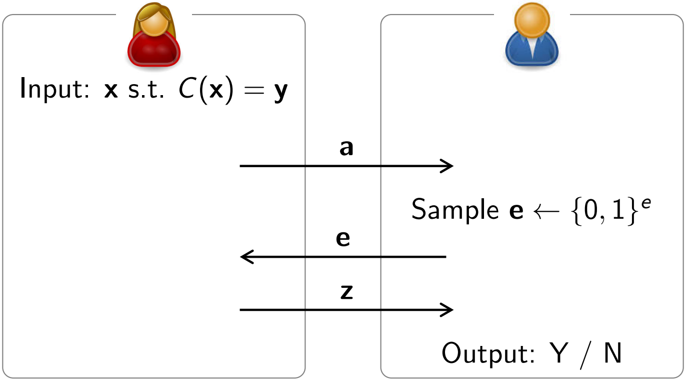
Σ-Protocol

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



Σ-Protocol

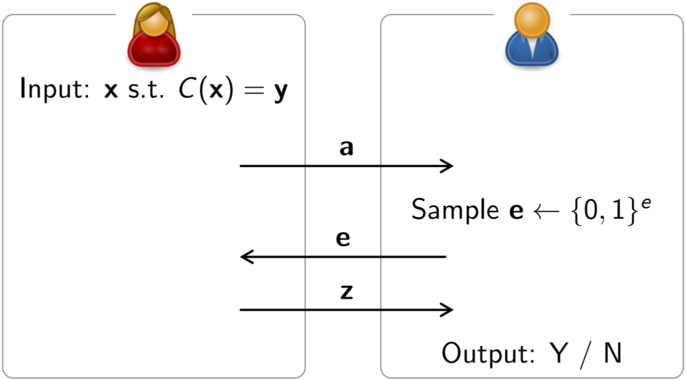
Public data: $C : \{0,1\}^n \to \{0,1\}^m$ (boolean circuit) and $\mathbf{y} \in \{0,1\}^m$



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

Σ-Protocol

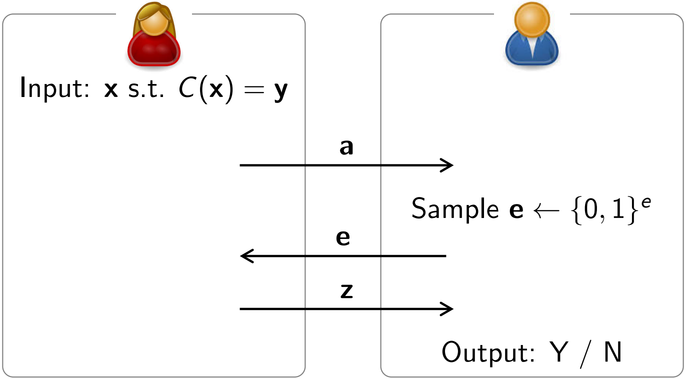
Public data: $C : \{0,1\}^n \to \{0,1\}^m$ (boolean circuit) and $\mathbf{y} \in \{0,1\}^m$



Soundness: from ≥ 2 accepting conversations $(\mathbf{a}, \mathbf{e}_i, \mathbf{z}_i)$ with $\mathbf{e}_i \neq \mathbf{e}_j$ we can efficiently compute $\mathbf{x}'$ s.t. $C(\mathbf{x}') = \mathbf{y}$

Σ-Protocol

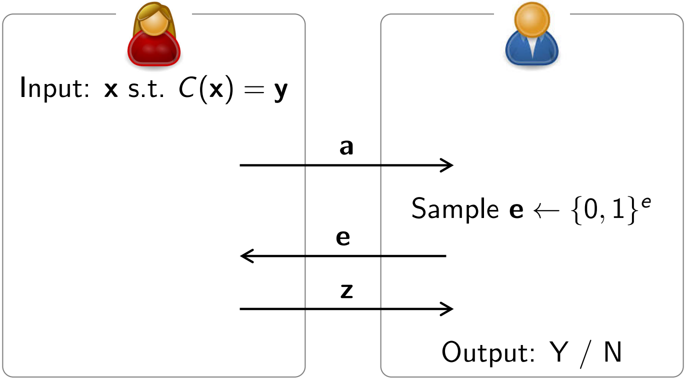
Public data: $C : \{0,1\}^n \to \{0,1\}^m$ (boolean circuit) and $\mathbf{y} \in \{0,1\}^m$



The protocol has **soundness error** ϵ:
if Alice is cheating, then Pr[Bob outputs Y] $\leq \epsilon$

Σ-Protocol

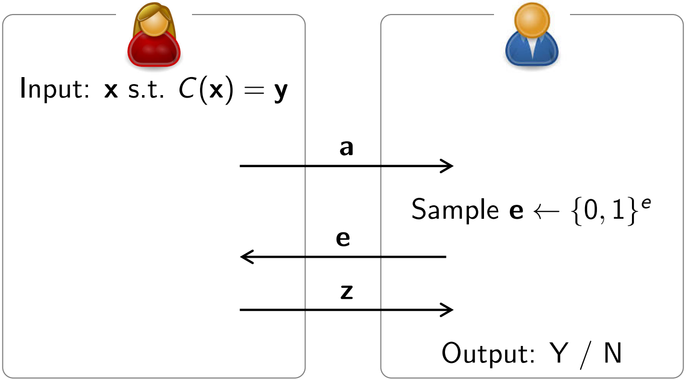
Public data: $C : \{0,1\}^n \to \{0,1\}^m$ (boolean circuit) and $\mathbf{y} \in \{0,1\}^m$



(Honest-Verifier) **ZK property**:
the distribution of $(\mathbf{a}, \mathbf{e}, \mathbf{z})$ does not reveal info on $\mathbf{x}$

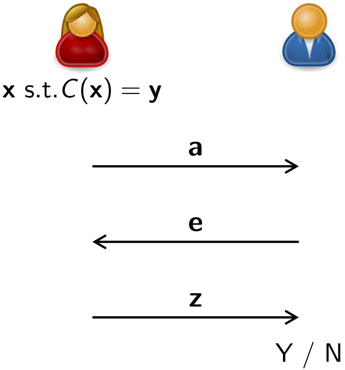
Σ-Protocol

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



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

Σ-Protocol Recap



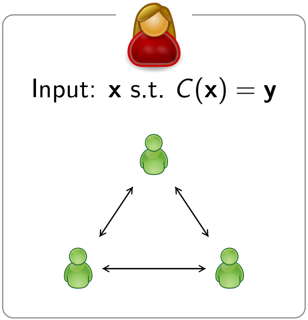
- Complete:
  if Alice honest, $\Pr[\text{Bob says Y}] = 1$
- Soundness error:
  if Alice cheats, $\Pr[\text{Bob says Y}] \leq \epsilon$
- ZK property: no info on **x**!
- 3 rounds, public coin $\rightarrow$ non-interactive

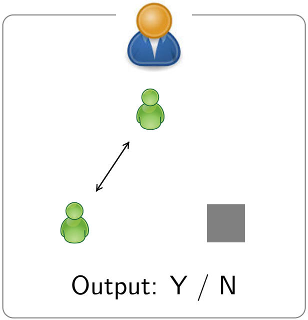
Related work:

IKOS Construction

(or “MPC-in-the-head”)

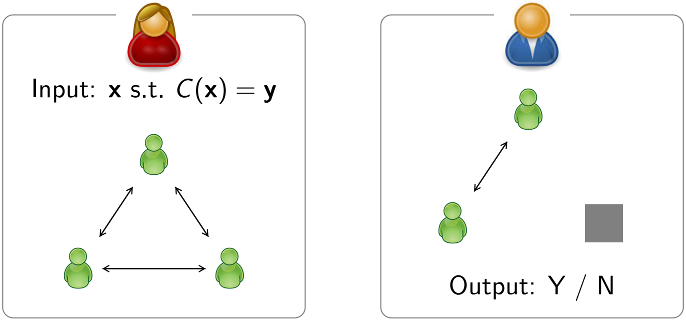
[Ishai-Kushilevitz-Ostrovsky-Sahai 2007]





Related work:

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



- a Σ-protocol with error $2/3$ (not implemented!)
- ZK protocol with *asymptotically* good complexity;

Circuit decomposition:

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

Circuit decomposition:

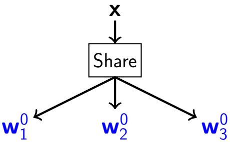
Goal: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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:



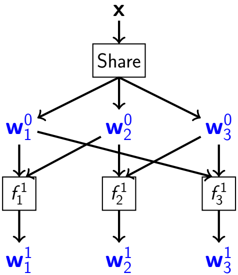
Goal: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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

Share, Rec and

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

Circuit decomposition:



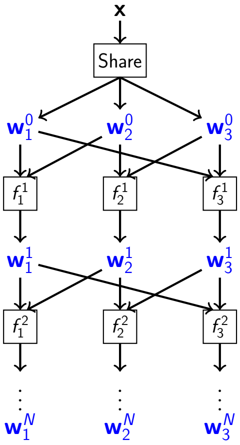
Goal: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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

Share, Rec and

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

Circuit decomposition:



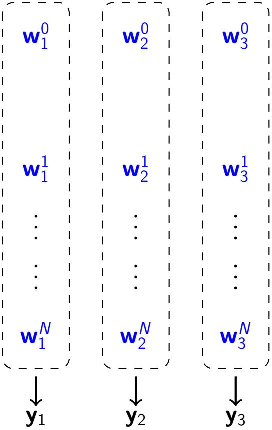
Goal: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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

Share, Rec and

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

Circuit decomposition:



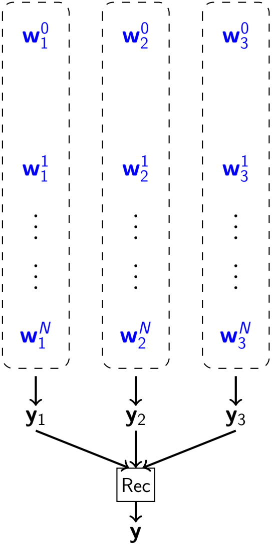
Goal: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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

Share, Rec and

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

Circuit decomposition:



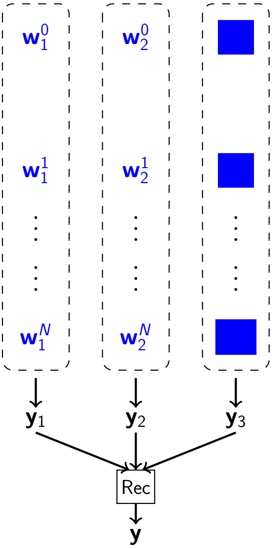
<u>Goal</u>: compute $C(\mathbf{x})$ splitting the computation in 3 branches s.t. looking at any 2 consecutive branches gives no info on $\mathbf{x}$

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

Share, Rec and $\mathcal{F} = \{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}_{j=1,\dots,N}$

- correctness: $\mathbf{y} = C(\mathbf{x})$

Circuit decomposition:



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

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

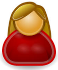
Share, Rec and

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

- correctness: $\mathbf{y} = C(\mathbf{x})$
- 2-privacy: $\forall\, e, \forall\, j$ $(\mathbf{w}_e^j, \mathbf{w}j_{e+1}, \mathbf{y}_{e+2})$ doesn't reveal info on $\mathbf{x}$

ZKBoo Protocol

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

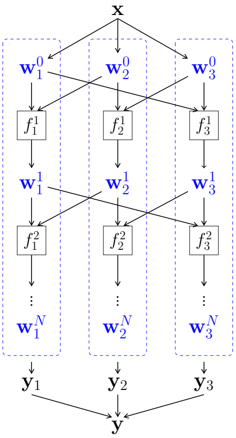


Input: $\mathbf{x}$ s.t. $C(\mathbf{x}) = \mathbf{y}$

ZKBoo Protocol

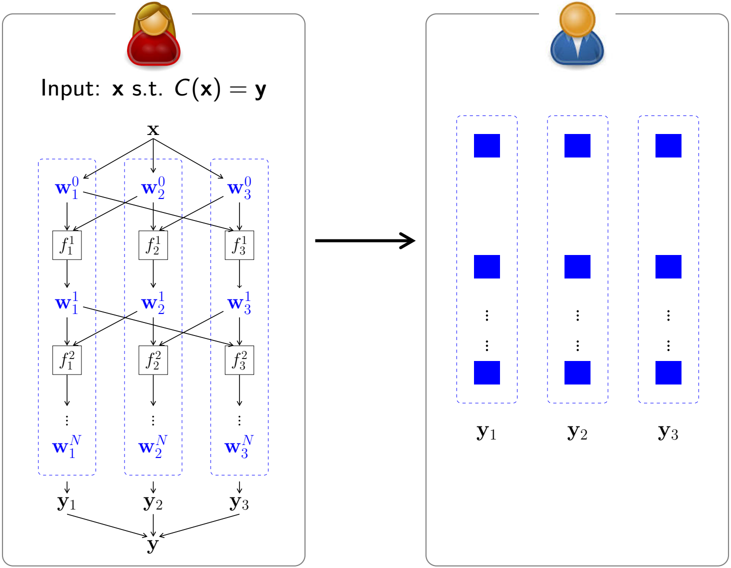
Public data: $C : \{0,1\}^n \to \{0,1\}^m$ (boolean circuit) and $\mathbf{y} \in \{0,1\}^m$

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



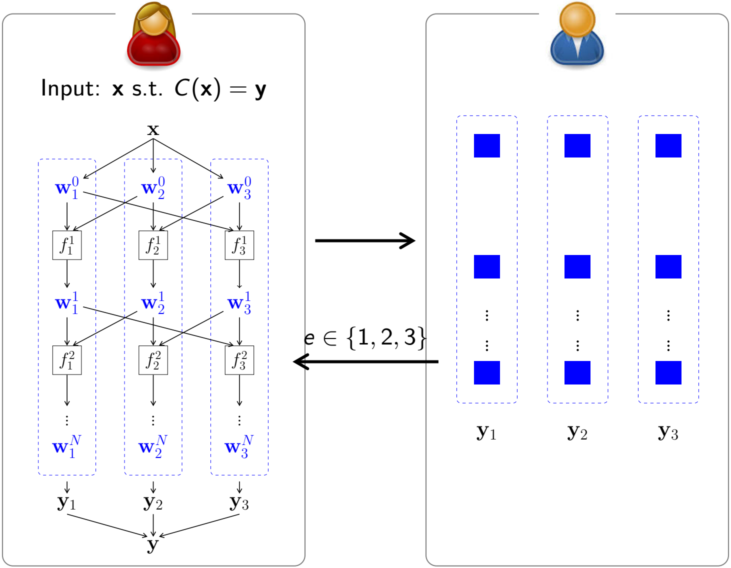
ZKBoo Protocol

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



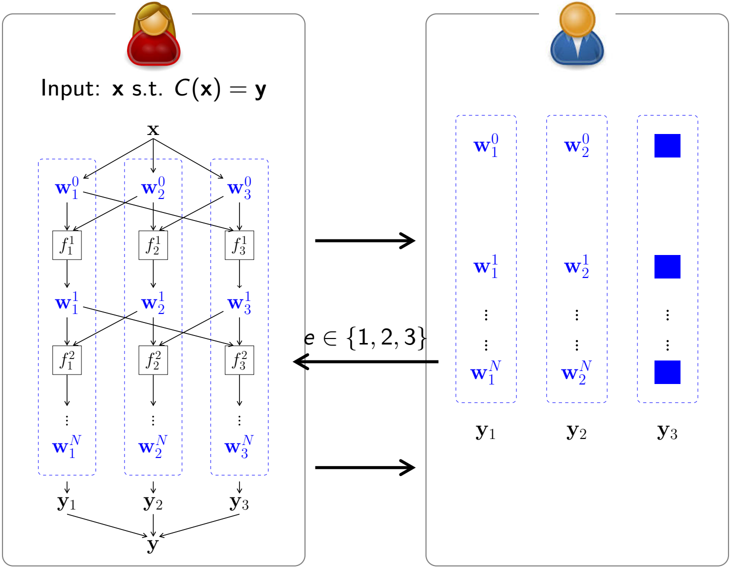
ZKBoo Protocol

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



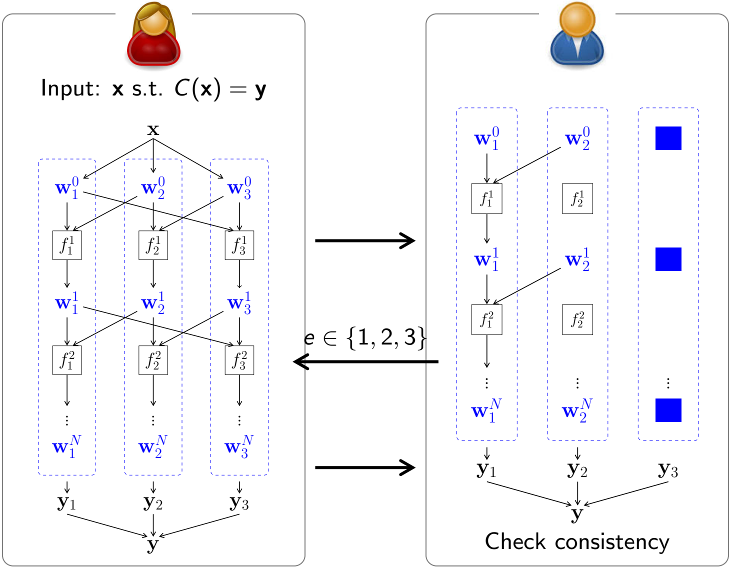
ZKBoo Protocol

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



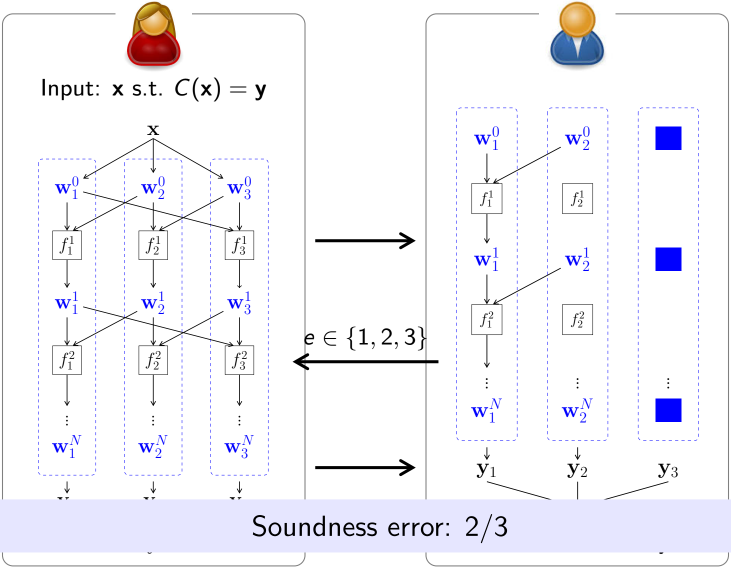
ZKBoo Protocol

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



ZKBoo Protocol

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



Soundness error: $2/3$

Linear Decomposition:

N= number of gates in the boolean circuit C

Linear Decomposition:

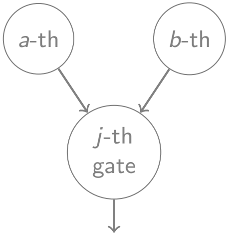
N= number of gates in the boolean circuit C

- $\mathsf{Share}(\mathbf{x}) = (\mathbf{w}_1^0, \mathbf{w}_2^0, \mathbf{w}_3^0)$ with $\mathbf{w}_1^0 \oplus \mathbf{w}_2^0 \oplus \mathbf{w}_3^0 = \mathbf{x}$
- $\mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = \mathbf{y}_1 \oplus \mathbf{y}_2 \oplus \mathbf{y}_3$

Linear Decomposition:

N= number of gates in the boolean circuit C

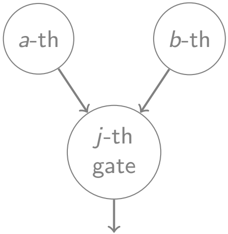
- $\text{Share}(\mathbf{x}) = (\mathbf{w}_1^0, \mathbf{w}_2^0, \mathbf{w}_3^0)$ with $\mathbf{w}_1^0 \oplus \mathbf{w}_2^0 \oplus \mathbf{w}_3^0 = \mathbf{x}$
- $\text{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = \mathbf{y}_1 \oplus \mathbf{y}_2 \oplus \mathbf{y}_3$
- $\{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}$



Linear Decomposition:

$N=$ number of gates in the boolean circuit C

- $\mathsf{Share}(\mathbf{x}) = (\mathbf{w}_1^0, \mathbf{w}_2^0, \mathbf{w}_3^0)$ with $\mathbf{w}_1^0 \oplus \mathbf{w}_2^0 \oplus \mathbf{w}_3^0 = \mathbf{x}$
- $\mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = \mathbf{y}_1 \oplus \mathbf{y}_2 \oplus \mathbf{y}_3$
- $\{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}$



XOR gate

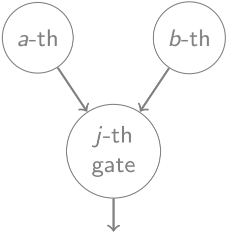
$f_e^{(j)}(\mathbf{w}_e^a, \mathbf{w}_e^b) = \mathbf{w}_e^a \oplus \mathbf{w}_e^b$

$e = 1, 2, 3$

Linear Decomposition:

$N=$ number of gates in the boolean circuit C

- $\text{Share}(\mathbf{x}) = (\mathbf{w}_1^0, \mathbf{w}_2^0, \mathbf{w}_3^0)$ with $\mathbf{w}_1^0 \oplus \mathbf{w}_2^0 \oplus \mathbf{w}_3^0 = \mathbf{x}$
- $\text{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = \mathbf{y}_1 \oplus \mathbf{y}_2 \oplus \mathbf{y}_3$
- $\{f_1^{(j)}, f_2^{(j)}, f_3^{(j)}\}$



XOR gate
$f_e^{(j)}(\mathbf{w}_e^a, \mathbf{w}_e^b) = \mathbf{w}_e^a \oplus \mathbf{w}_e^b$

AND gate
$f_e^{(j)}(\mathbf{w}_e^a, \mathbf{w}_e^b, \mathbf{w}_{e+1}^a, \mathbf{w}_{e+1}^b) = \mathbf{w}_e^a \mathbf{w}_e^b \oplus \mathbf{w}_{e+1}^a \mathbf{w}_e^b \oplus \mathbf{w}_e^a \mathbf{w}_{e+1}^b \oplus \mathbf{r}_j$

$e = 1, 2, 3$

Experiments for ZKBoo

	SHA-1		SHA-256	
	Serial	Paral.	Serial	Paral.
Prover (ms)	31.73	12.73	54.63	15.95
Verifier (ms)	22.85	4.39	67.74	13.20
Proof size (KB)	444.18		835.91	

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*

Experiments for ZKBoo

	SHA-1		SHA-256	
	Serial	Paral.	Serial	Paral.
Prover (ms)	31.73	12.73	54.63	15.95
Verifier (ms)	22.85	4.39	67.74	13.20
Proof size (KB)	444.18		835.91	

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*

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 *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 <u>*any*</u> 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?