Formal security analysis of MPC-in-the-head zero-knowledge protocols

Nikolaj Sidorenco[1] Sabine Oechsner[1] Bas Spitters[12]

{sidorenco, oechsner, spitters}@cs.au.dk

[1]Department of Computer Science
Aarhus University

[2]Concordium Blockchain Research Center
Aarhus University

to appear at CSF'21

What is MPC-in-the-head

A technique to construct zero-knowledge proofs for any NP-relation [IKOS07]

Works by constructing a secure MPC protocol computing such relation as a function.

Recently used to construct post-quantum signature schemes [CDGORR17], which was then improved upon in the NIST post-quantum alternate, Picnic.

What is MPC

N parties want to jointly compute a N-ary function $f(x_1, \ldots, x_N)$.

What is MPC

N parties want to jointly compute a N-ary function $f(x_1, \ldots, x_N)$.

Each party supplies a private input x_i. Everyone learns $f(x_1, \ldots, x_n)$. Parties should not learn any private inputs.

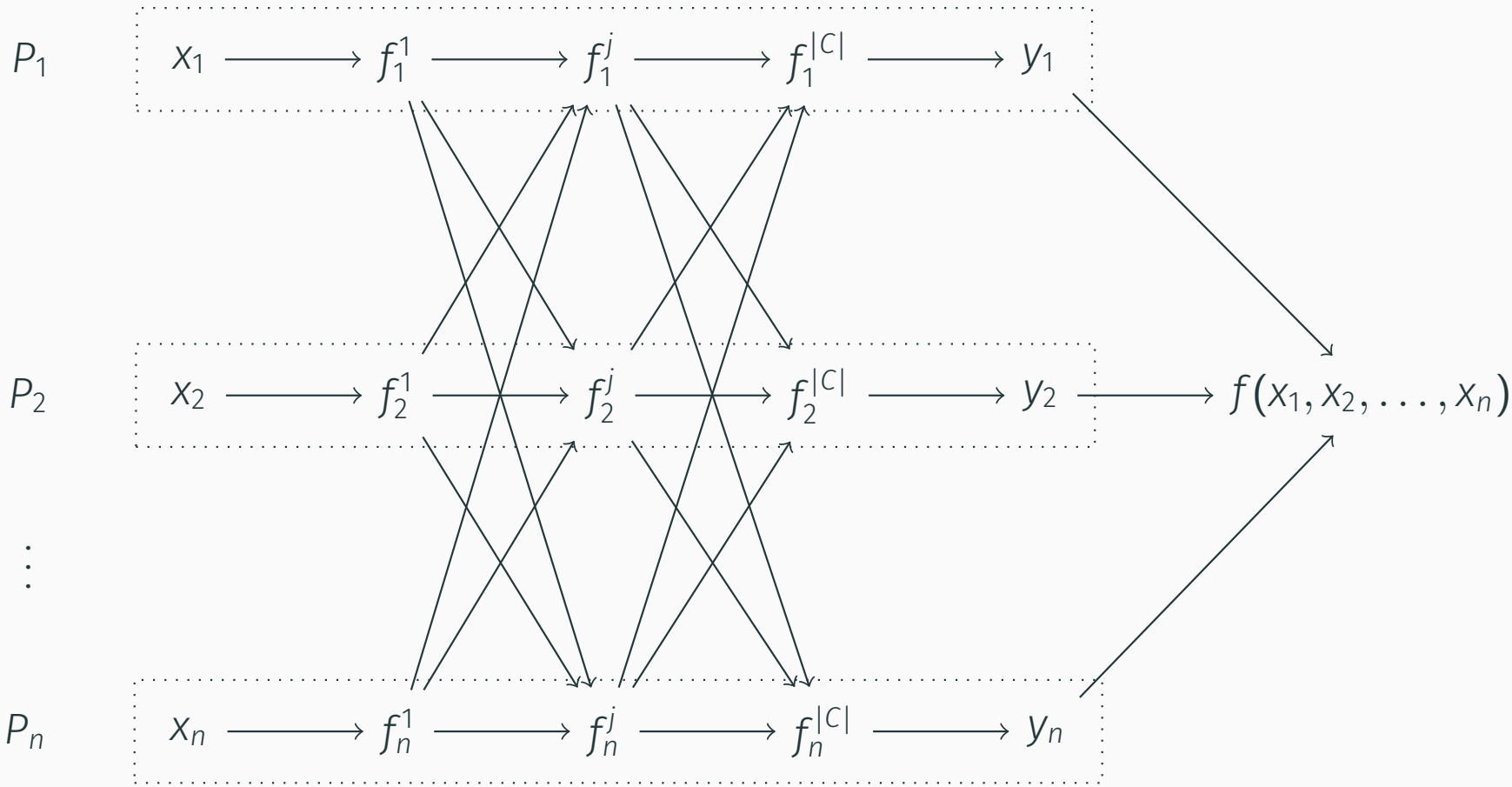
What is MPC

N parties want to jointly compute a N-ary function $f(x_1, \ldots, x_N)$.

Each party supplies a private input x_i. Everyone learns $f(x_1, \ldots, x_n)$. Parties should not learn any private inputs.

f is usually expressed as a circuit. A round in the MPC protocol computes a single gate in the circuit. We denote f_i^j for the function party i uses to compute gate j in the protocol.

What is MPC


P_1
x_1
f_1^1
f_1^j
$f_1^{|C|}$
y_1
P_2
x_2
f_2^1
f_2^j
$f_2^{|C|}$
y_2
$f(x_1, x_2, \ldots, x_n)$
P_n
x_n
f_n^1
f_n^j
$f_n^{|C|}$
y_n

Semi-honest MPC

Definition (Correctness)

An MPC protocol is correct, if $\Pr[f(x) = \mathbf{MPC}(f, x)] = 1$.

Semi-honest MPC

Definition (Correctness)

An MPC protocol is correct, if $\Pr[f(x) = \mathbf{MPC}(f, x)] = 1$.

Definition (d-Private)

An MPC protocol is d private, if there exists a simulator, which given $f(x)$ and f can simulate a subset of views of size d indistinguishable from a size d subset of views from an honest execution.

Zero-knowledge

From MPC we can instantiate a Σ-Protocol, which is a special class of zero-knowledge.

A party computes a shares for each gate based on their own state and incoming messages form other parties.

If we reveal at least two parties, we can verify if a party has partially followed the protocol.

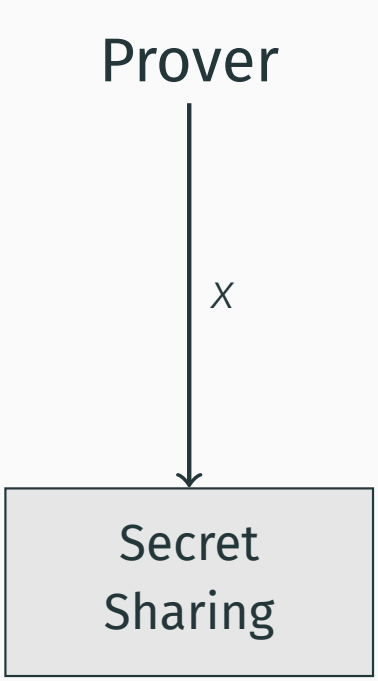
Zero-Knowledge

Prover

$\mathsf{R} = \{(x, y) : f(x) = y\}$

Verifier

Zero-Knowledge



$\mathsf{R} = \{(x, y) : f(x) = y\}$

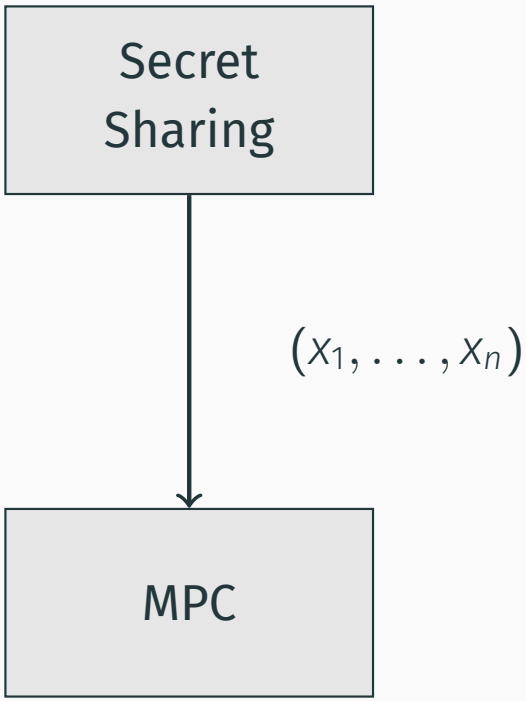
Verifier

Zero-Knowledge

Prover

$\mathsf{R} = \{(x, y) : f(x) = y\}$

Verifier

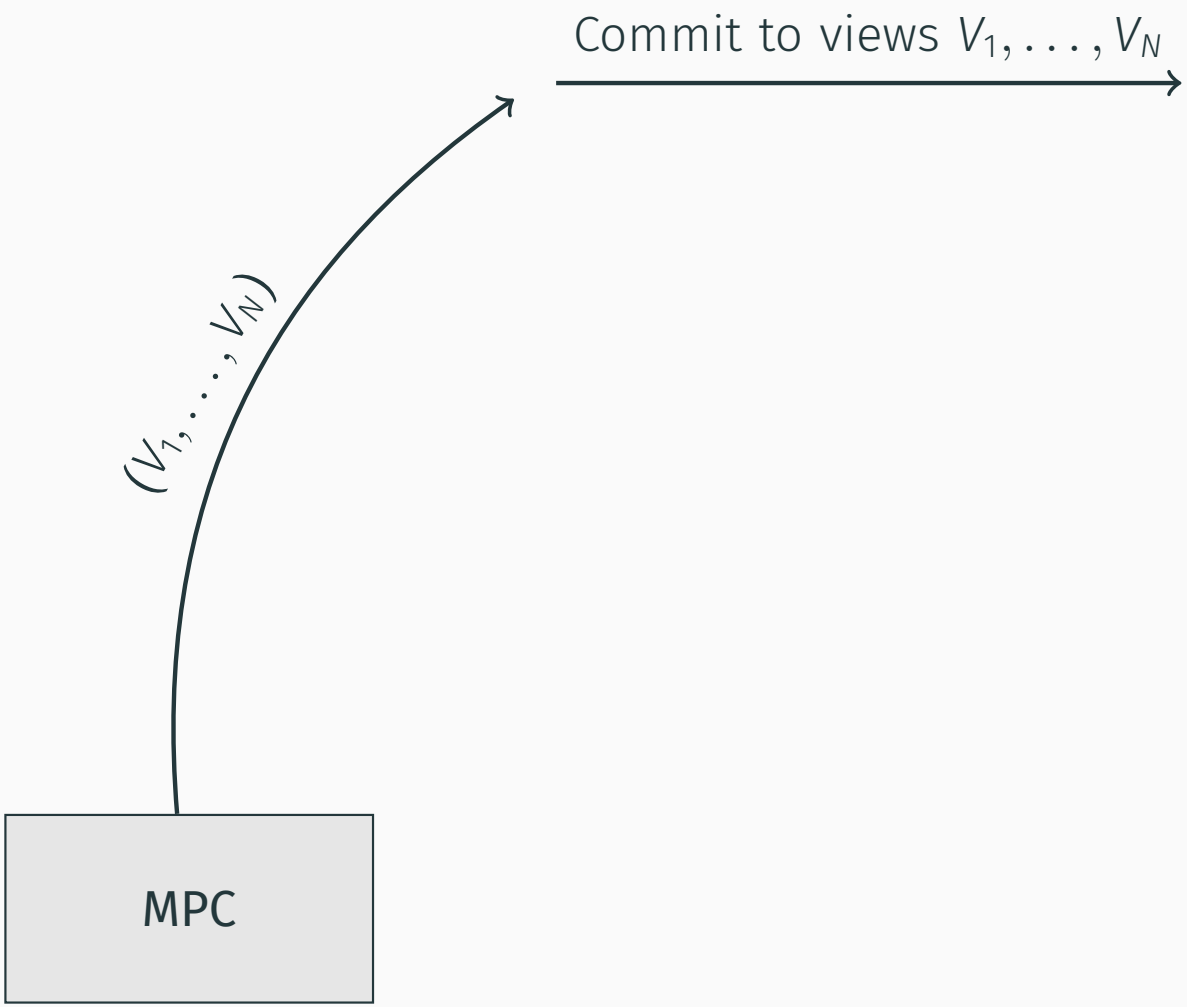


Zero-Knowledge

$\mathsf{R} = \{(x, y) : f(x) = y\}$

Prover

Verifier

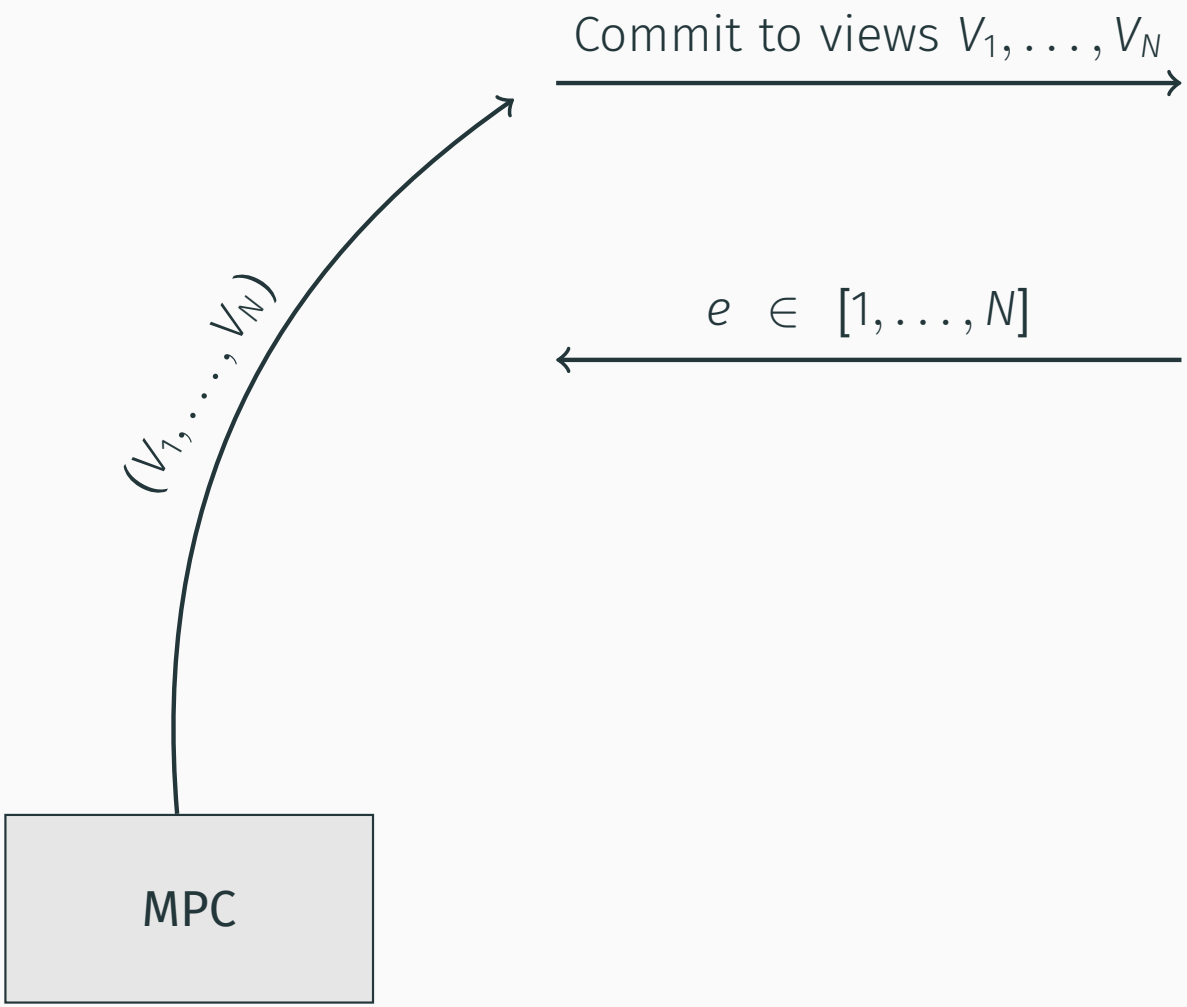


Zero-Knowledge

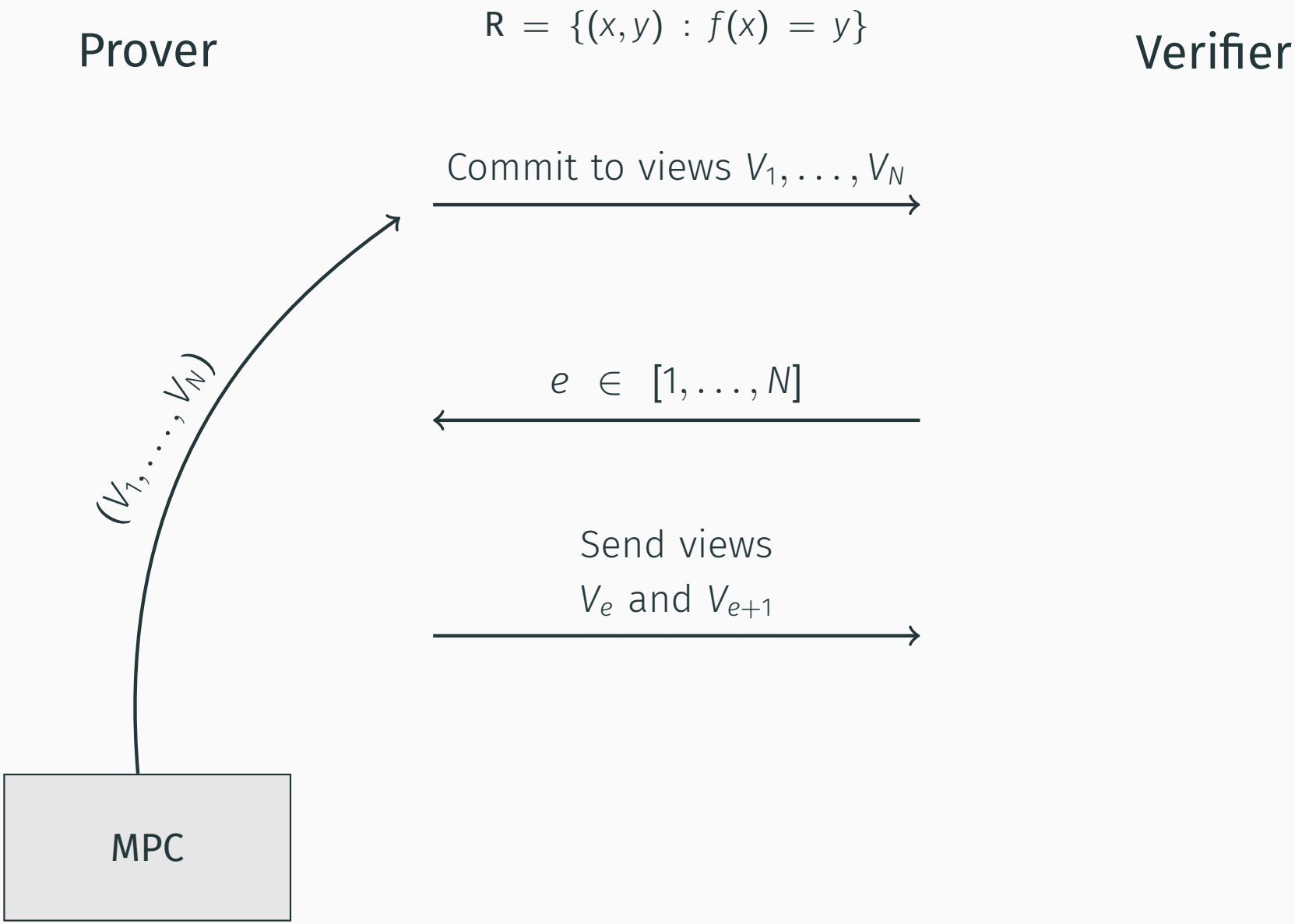
$\mathsf{R} = \{(x, y) : f(x) = y\}$

Prover

Verifier



Zero-Knowledge

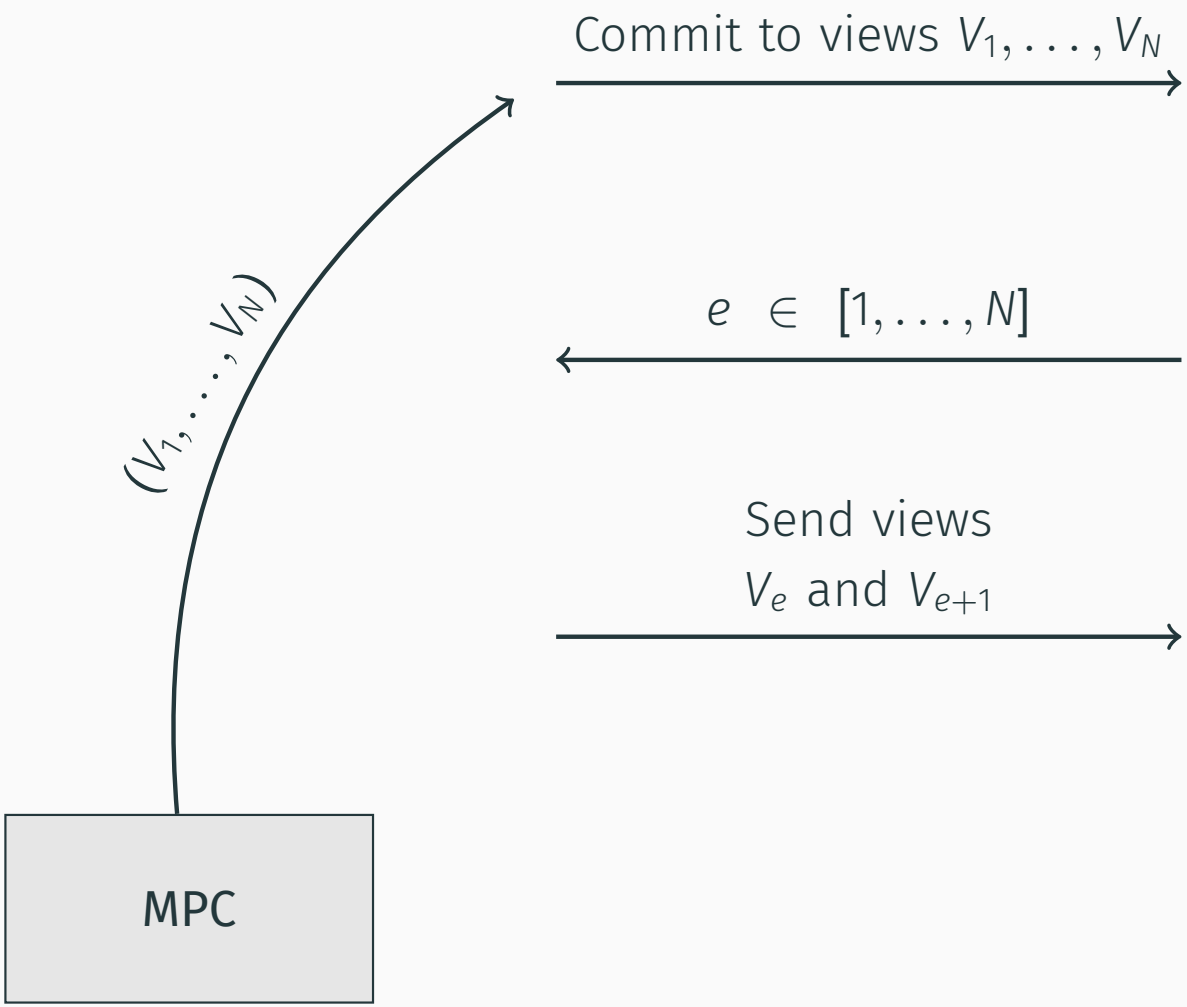


Zero-Knowledge

$\mathsf{R} = \{(x, y) : f(x) = y\}$

Prover

Verifier



Zero-knowledge

What is a secure Σ-Protocol?

Zero-knowledge

What is a secure Σ-Protocol?

Definition (Completeness)

If $(x, y) \in \mathbf{R}$ and the Prover and Verifier are honest, then the Verifier always accepts

Zero-knowledge

What is a secure Σ-Protocol?

Definition (Completeness)

If $(x, y) \in \mathbf{R}$ and the Prover and Verifier are honest, then the Verifier always accepts

Definition (s-Special Soundness)

From any x and s accepting transcripts $\{(a, e_1, z_1), \ldots, (a, e_s, z_s)\}$ with $e_i \neq e_j$ for all $i \neq j$, x such that $(x, y) \in \mathbf{R}$ can be efficiently computed.

Zero-knowledge

What is a secure Σ-Protocol?

Definition (Completeness)

If $(x, y) \in \mathbf{R}$ and the Prover and Verifier are honest, then the Verifier always accepts

Definition (s-Special Soundness)

From any x and s accepting transcripts $\{(a, e_1, z_1), \ldots, (a, e_s, z_s)\}$ with $e_i \neq e_j$ for all $i \neq j$, x such that $(x, y) \in \mathbf{R}$ can be efficiently computed.

Definition (Special Honest-Verifier Zero-Knowledge)

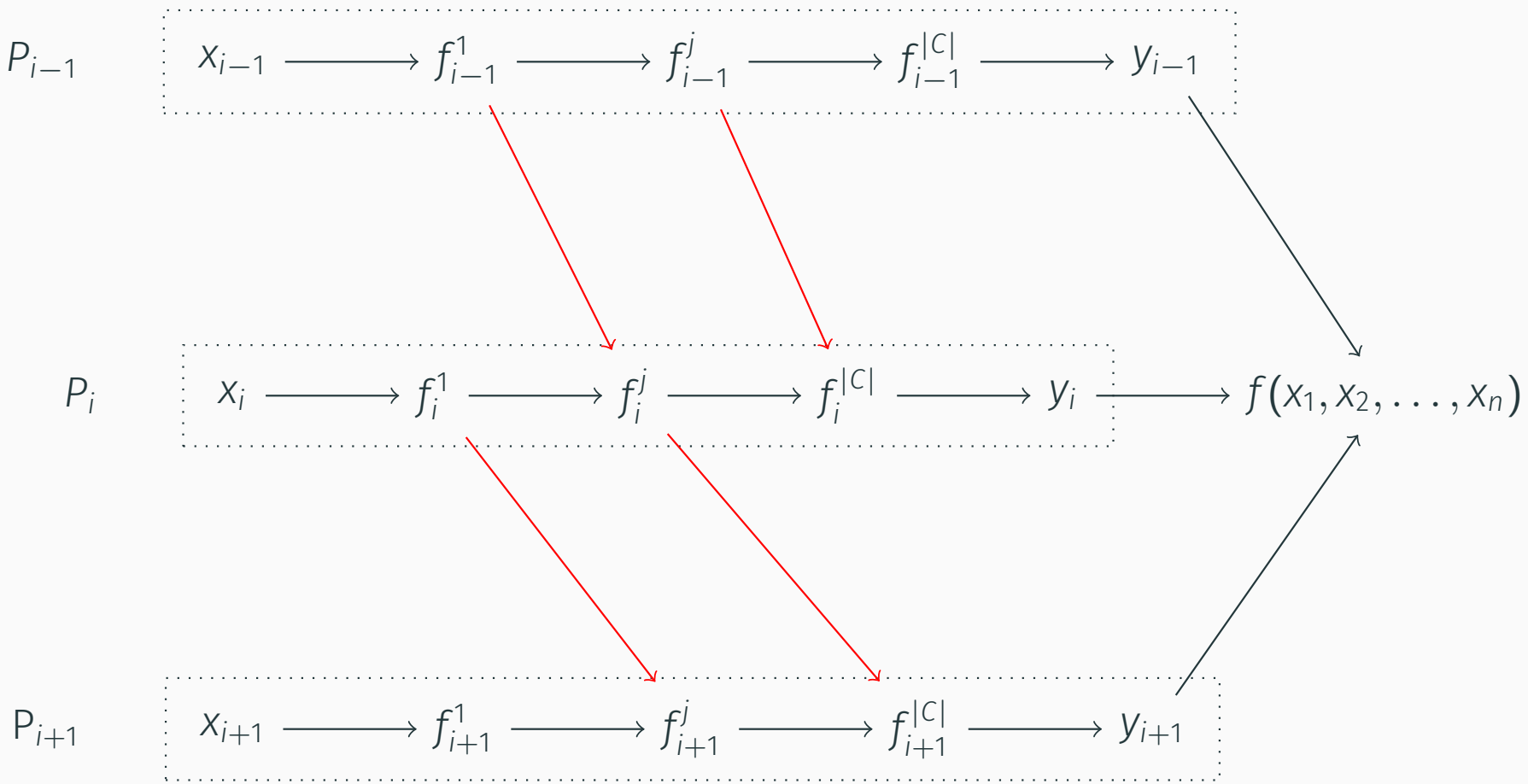
There exists a PPT simulator, which on input x and fixed challenge e can produce an accepting transcript (a, e, z) indistinguishable from a conversation between an honest Prover and Verifier.

ZKBoo

ZKBoo [GMO16]

- First efficient instantiation of the MPC-in-the-head approach.
- Build on a subset of MPC protocols with restricted communication pattern (Called a Decomposition).
- Implemented as a Σ-Protocol.
- Build upon the original MPC-in-the-head approach.
  - Constructs faster Zero-knowledge protocols for proving knowledge of the pre-image of a function.

ZKBoo


P_{i-1}
x_{i-1} f_{i-1}^1 f_{i-1}^j $f_{i-1}^{|C|}$ y_{i-1}
P_i
x_i f_i^1 f_i^j $f_i^{|C|}$ y_i
$f(x_1, x_2, \ldots, x_n)$
P_{i+1}
x_{i+1} f_{i+1}^1 f_{i+1}^j $f_{i+1}^{|C|}$ y_{i+1}

Motivation

From a restricted class of secure MPC we can construct Zero-Knowledge for any relation expressed as a circuit.

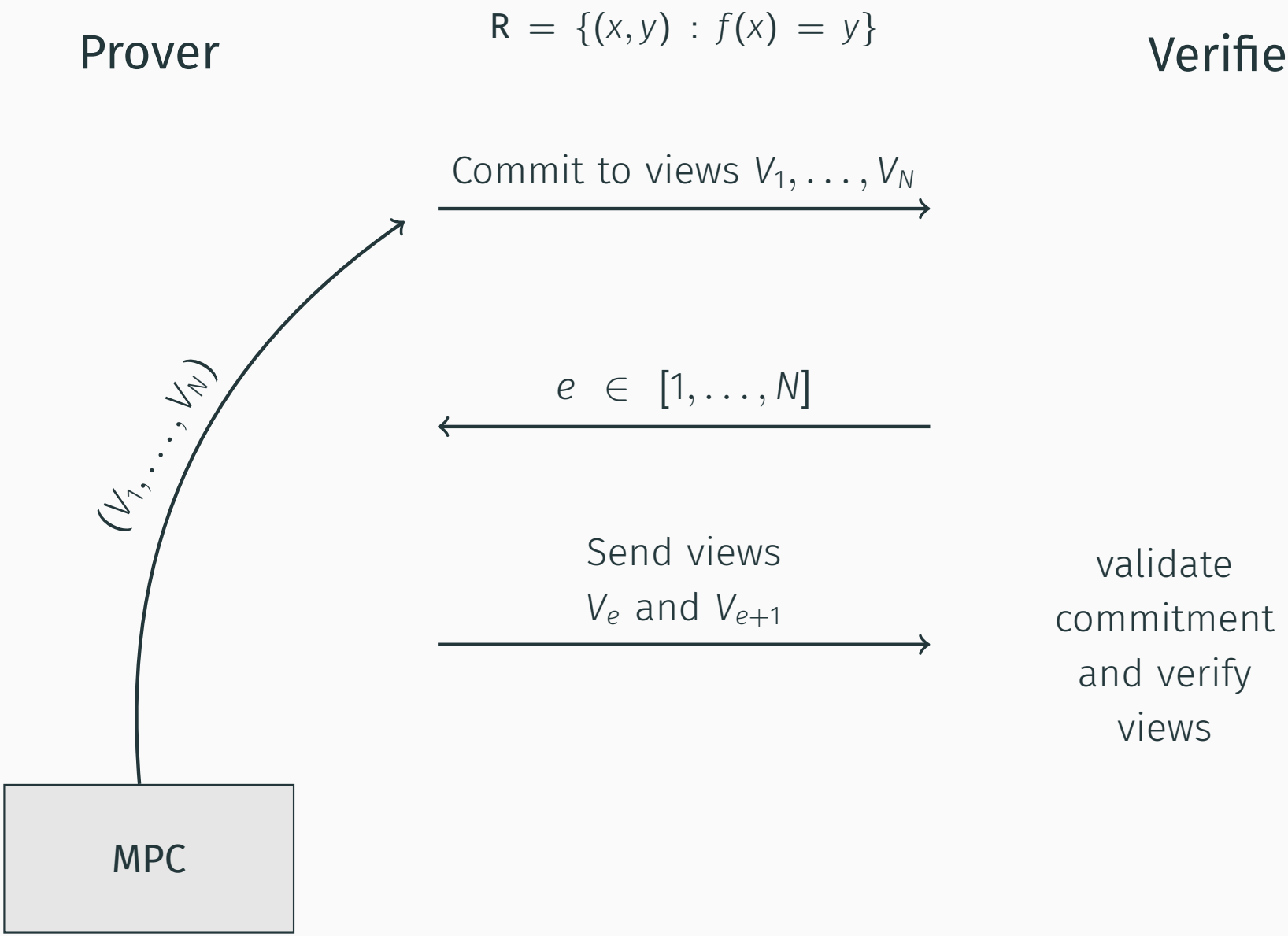
However, the security proofs of recent MPCith Zero-Knowledge [GMO16, CDGORR17, KKW18] depend on implementation-level details of the decomposition.

Problems

If we consider the MPCith construct as a program, how would we design a general interface between MPC and Zero-Knowledge.

Interface needs to be general enough to support the various optimisations done in related works.

Problems


$\mathsf{R} = \{(x, y) : f(x) = y\}$
Prover
Verifier
Commit to views $V_1, \ldots, V_N$
$e \in [1, \ldots, N]$
$(V_1, \ldots, V_N)$
Send views
V_e and V_{e+1}
validate
commitment
and verify
views
MPC

Problems

Consider the method the verifier uses to verify the views. We call this abstract procedure `verify`.

Problems

Consider the method the verifier uses to verify the views. We call this abstract procedure `verify`.

Definition (Completeness)

If $(x, y) \in \mathbf{R}$ and the Prover and Verifier are honest, then the Verifier always accepts

Definition (Correctness)

An MPC protocol is correct, if $\Pr[f(x) = \mathbf{MPC}(f, (x)] = 1$.

No guarantee that the verifier will accept the views. Only possible if we fix the same verification method for all MPC protocols.

Problems

Definition (s-Special Soundness)

From any x and s accepting transcripts $\{(a, e_1, z_1), \ldots, (a, e_s, z_s)\}$ with $e_i \neq e_j$ for all $i \neq j$, x such that $(x, y) \in \mathbf{R}$ can be efficiently computed.

A transcript is accepting if the commitments are valid and the revealed views verify.

Problems

Definition (s-Special Soundness)

From any x and s accepting transcripts $\{(a, e_1, z_1), \ldots, (a, e_s, z_s)\}$ with $e_i \neq e_j$ for all $i \neq j$, x such that $(x, y) \in \mathbf{R}$ can be efficiently computed.

A transcript is accepting if the commitments are valid and the revealed views verify.

Intuitively, each party has a share of the original input. Given enough shares we can reconstruct it.

However, without any assumptions on the structure of the views we do not know where to look.

Why black-box security proofs

MPC — Zero-Knowledge

ZKBoo

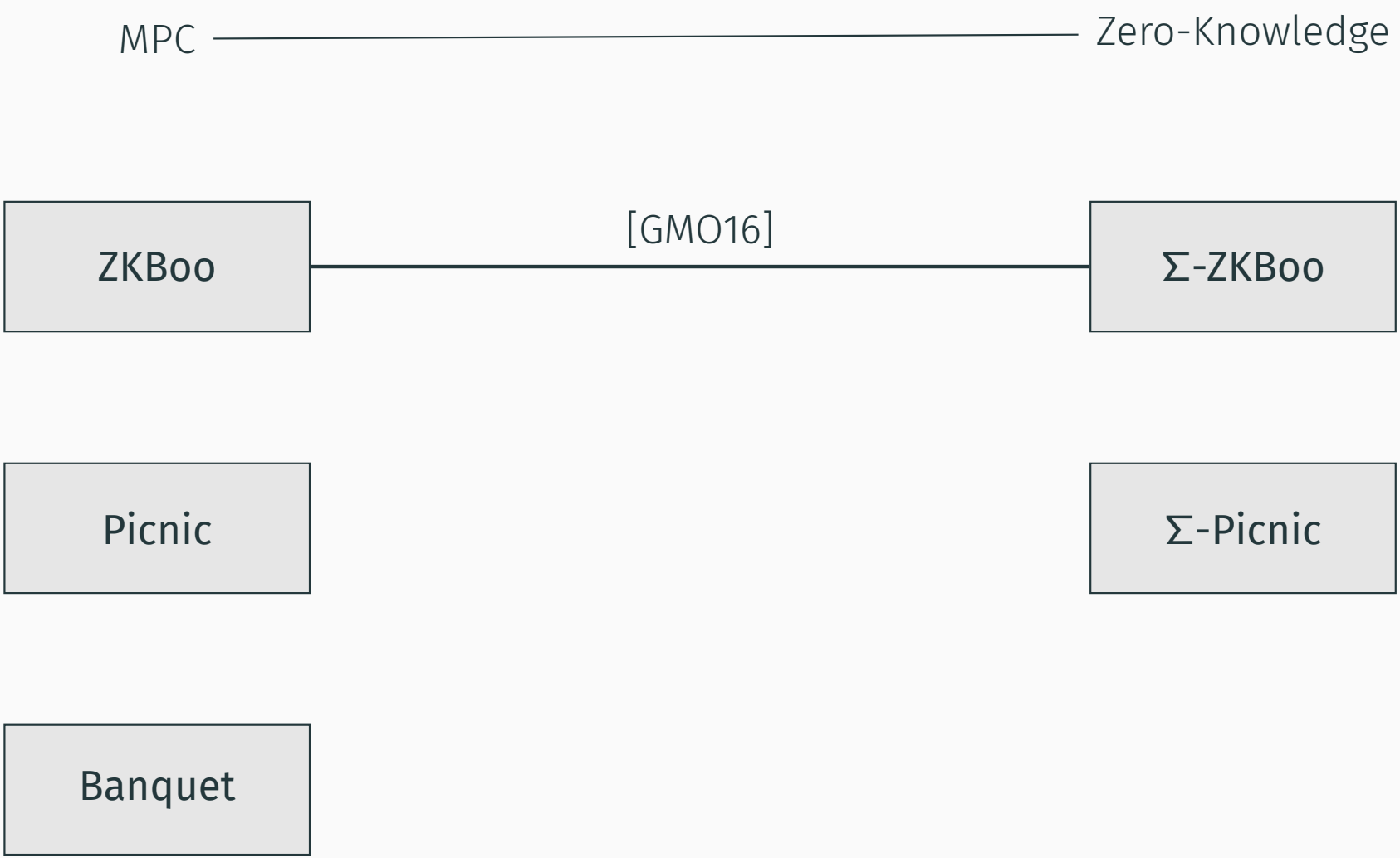
Σ-ZKBoo

Picnic

Σ-Picnic

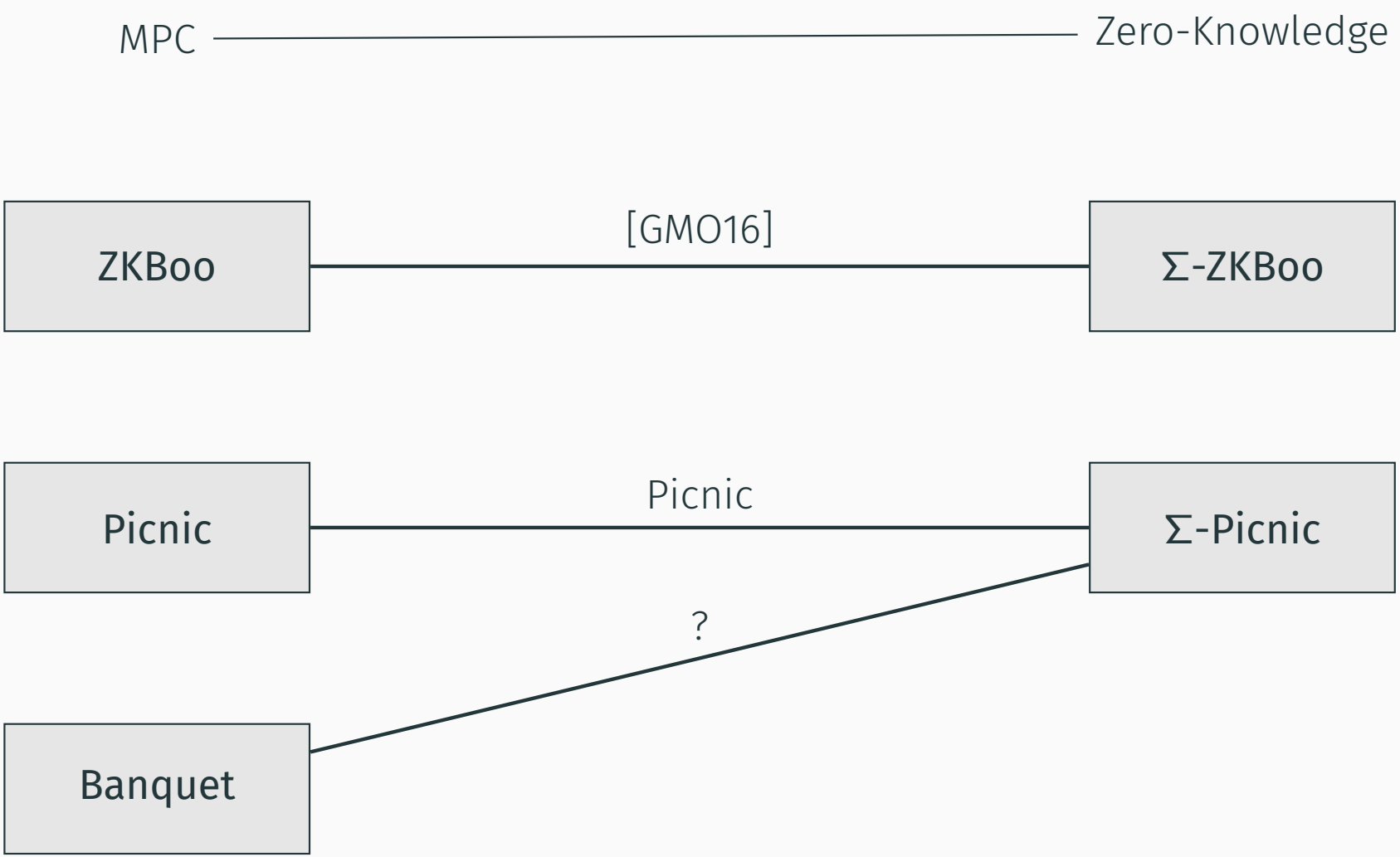
Banquet

Why black-box security proofs



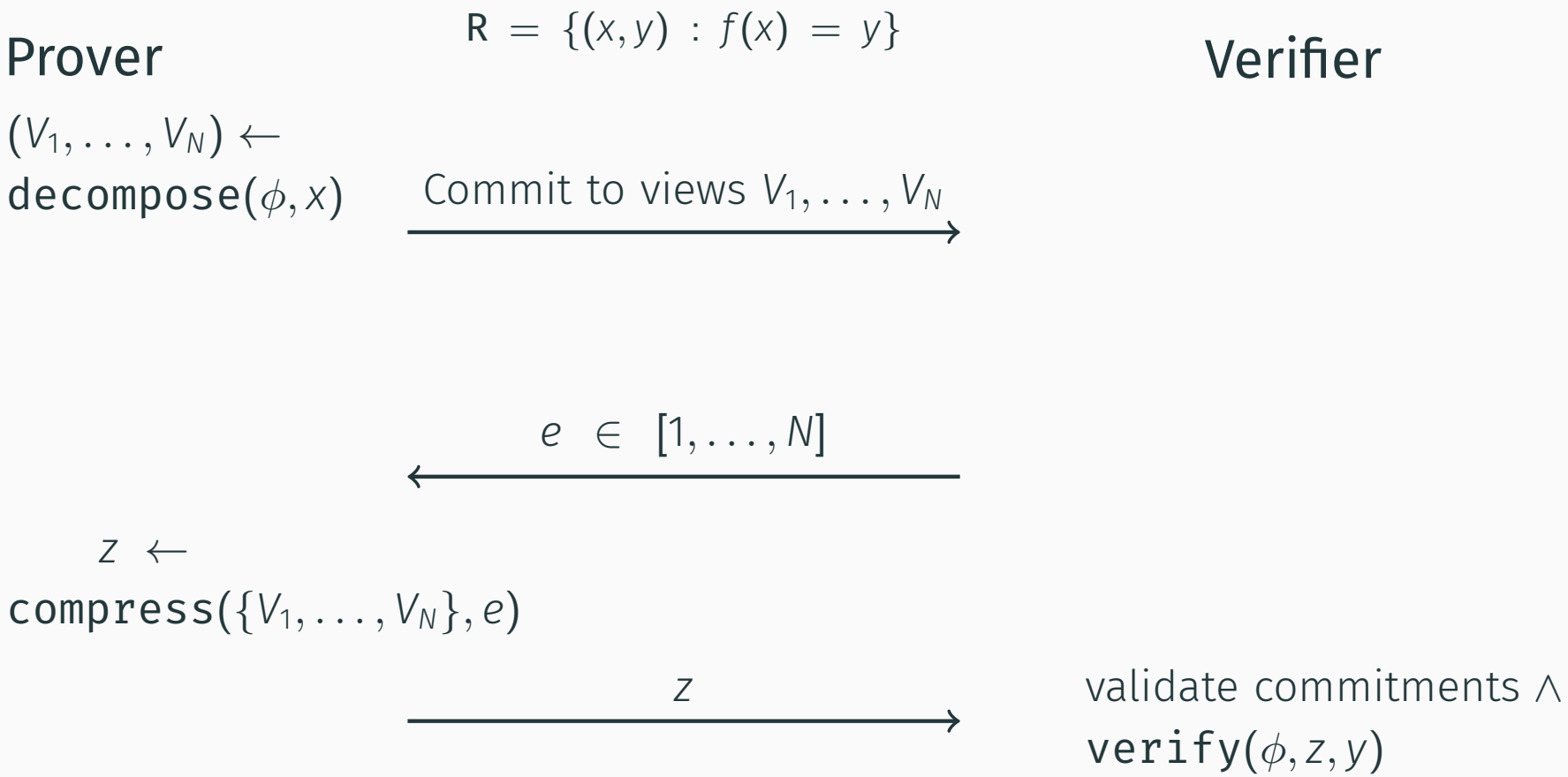
Why black-box security proofs

Why black-box security proofs



New notion of decompositions

Based on the definition by [GMO16] we define the notion of a decomposition.



New notion of decompositions

The typical MPC security properties are then extended:

Definition (Verifiability)

For any honest execution of the decomposition and for all challenges e, `verify` will always output true, if given projected views based on e.

Definition (d-Privacy)

Unaltered from original MPC definition.

Definition (k-Special Soundness)

Given k tuples of projected views, with output y, there exists an efficient program extracting a valid input x', such that $f(x') = y$.

Completeness

$\mathsf{R} = \{(x, y) : f(x) = y\}$

Prover

$(V_1, \ldots, V_N) \leftarrow$
$\texttt{decompose}(\phi, x)$

Commit to views $V_1, \ldots, V_N$ $\longrightarrow$

$e \in [1, \ldots, N]$ $\longleftarrow$

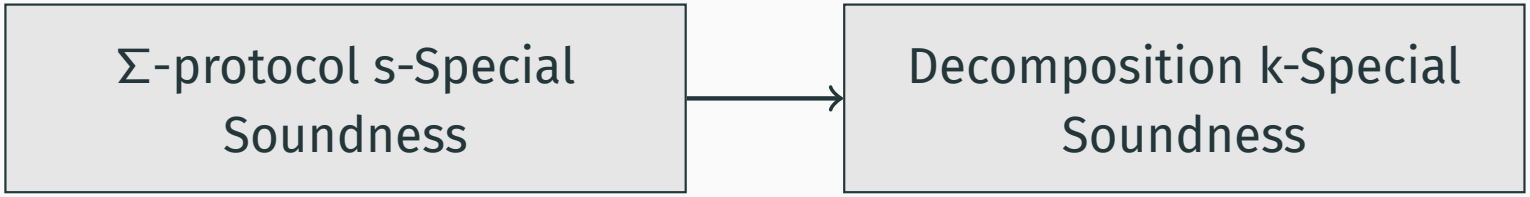
$z \leftarrow$
$\texttt{compress}(\{V_1, \ldots, V_N\}, e)$

z $\longrightarrow$

Verifier

validate commitments $\wedge$
$\texttt{verify}(\phi, z, y)$

Security



The Σ-protocol has d revealed views from each transcript.

If s is large enough such that $k \leq s \cdot d$, then soundness now follows from the decomposition.

This approach requires no additional assumption on the structure of the views!

Modularity of decompositions

Decomposition = {`decompose`, `compress`, `verify`, ...}

With this definition the interface between Σ-Protocol and Decomposition is a few procedures with clearly defined types.

We can now define transformations independently of the underlying decomposition.

Formalisation

EasyCrypt

EasyCrypt is a proof assistant providing us with:

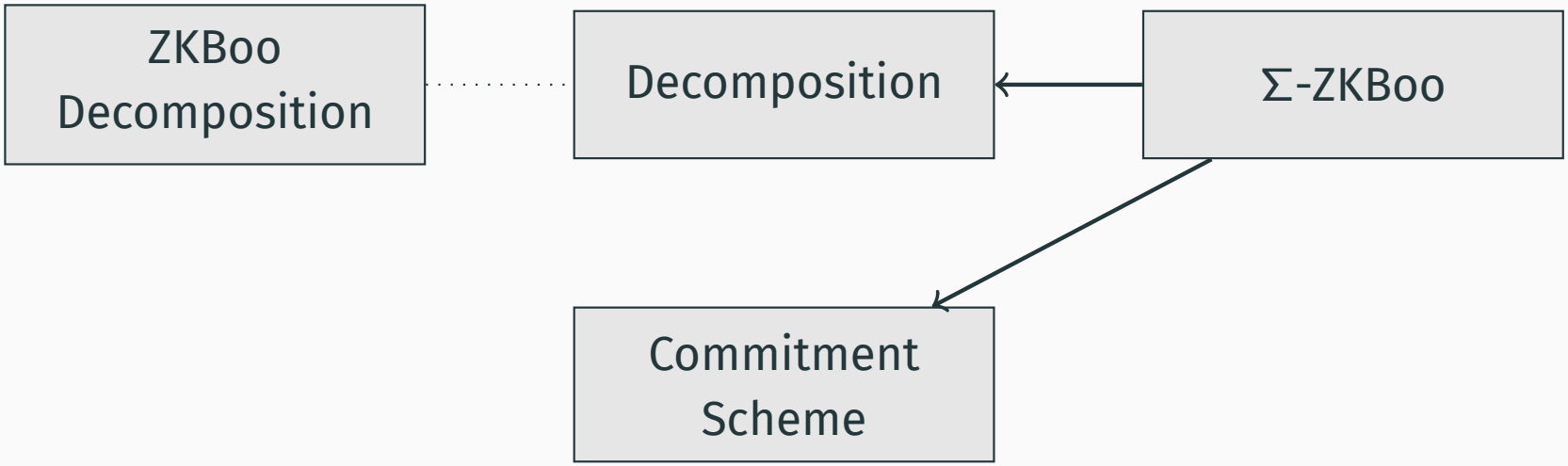
- A probabilistic imperative language (pWhile)
- A number of logics.
  - A probabilistic Hoare logic for reasoning about programs
  - A probabilistic relational logic for comparing programs
  - An ambient logic for mathematical reasoning.
- A rich library of cryptographic primitives.

Formalisation

A decomposition is a generic interface exposing the assumed functions (`decompose`, `verify`, etc.).

The Σ-Protocol instantiates a decomposition with the required types.

The first machine-checked proof of a MPC-in-the-head-style protocol.



Lessons learned

Modularity:

- Objects quickly becomes very large in a formal setting.
- Without modularity the objects quickly becomes hard to manage.

Abstraction:

- Internal representation
- Lazy vs. Eager sampling
  - For the decomposition we need fine-grained control over the individual parties.
  - For the Zero-knowledge part we want to reason about the structure as a whole.

Conclusion

- Modularity is needed to manage complexity.
  - Supports proof effort.
  - Allows us to freely compose decomposition protocols with zero-knowledge transformations.
  - Side effect: defines a common interface between decomposition and zero-knowledge.
  - Achieved without imposing additional assumptions on the structure of the views by specialising the security properties.
- Formal verification:
  - A Zero-knowledge protocol based on new definitions are proven secure with the EasyCrypt proof assistant.
  - first machine-checked Security proof of an MPCith protocol.

`https://eprint.iacr.org/2021/437`