From Secure MPC to Efficient Zero-Knowledge

David Wu

March, 2017

The Complexity Class **NP**

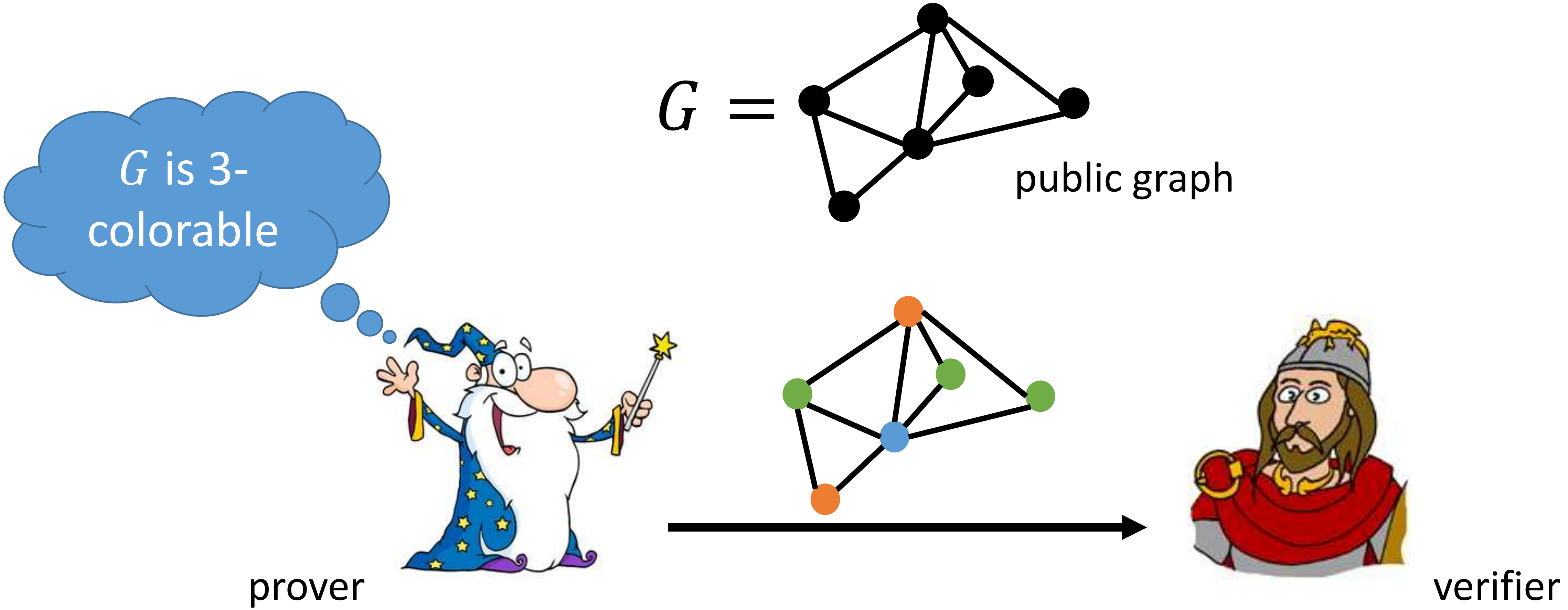
NP – the class of problems that are *efficiently verifiable*

a language $\mathcal{L}$ is in **NP** if there exists a polynomial-time verifier R such that

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathrm{poly}(|x|)}\ R(x, w) = 1$$

Interactive Proof Systems [GMR85]

NP admits efficient non-interactive proofs



Interactive Proof Systems [GMR85]

IP: class of languages that have an interactive proof system

G is **not** 3-colorable

accept

prover and verifier exchange a sequence of messages

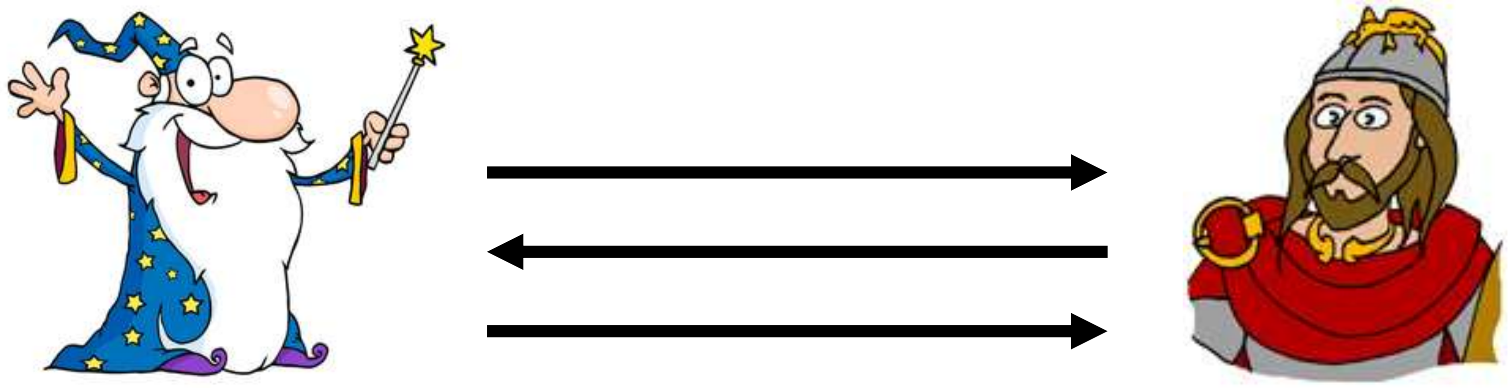
prover

verifier

Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms (P, V) with following properties:

- **Completeness**: $\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = 1] = 1$
- **Soundness**: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = 0] = 1$



prover

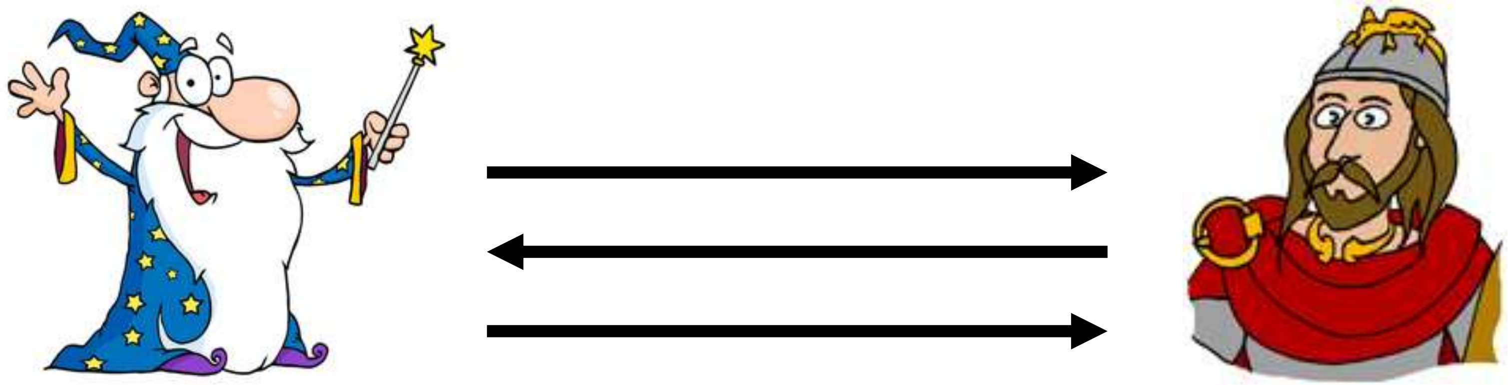
verifier

Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms $(P,$
following properties:

- **Completeness**: $\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = 1] = 1$
- **Soundness**: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = 0] = 1$

can also allow soundness error ϵ



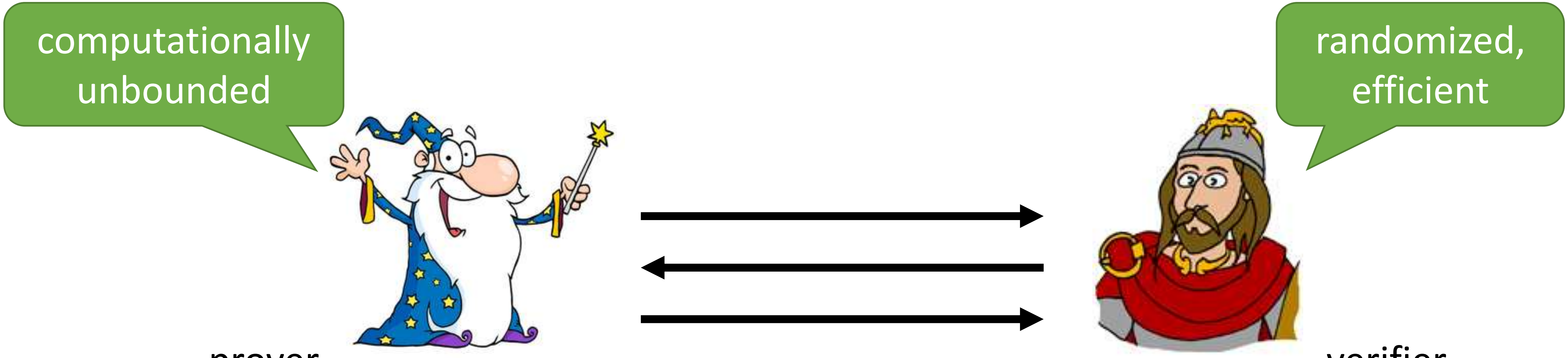
prover

verifier

Interactive Proof Systems [GMR85]

Interactive proof system modeled by two algorithms (P, V) with following properties:

- **Completeness**: $\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = 1] = 1$
- **Soundness**: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = 0] = 1$
- **Efficiency**: V runs in polynomial time (in $|x|$)



prover

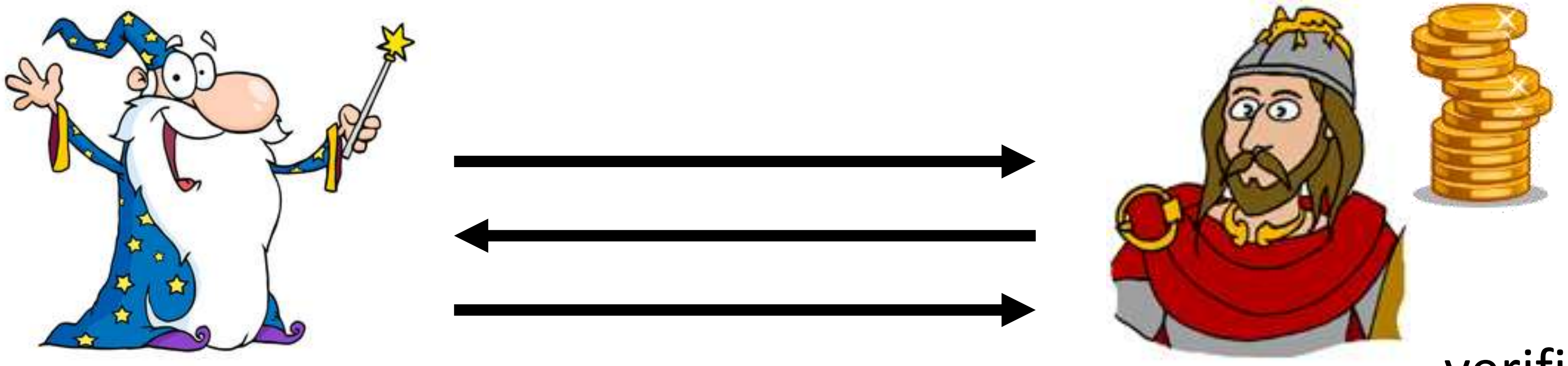
verifier

Interactive Proof Systems [GMR85]

IP: class of languages that have an interactive proof system

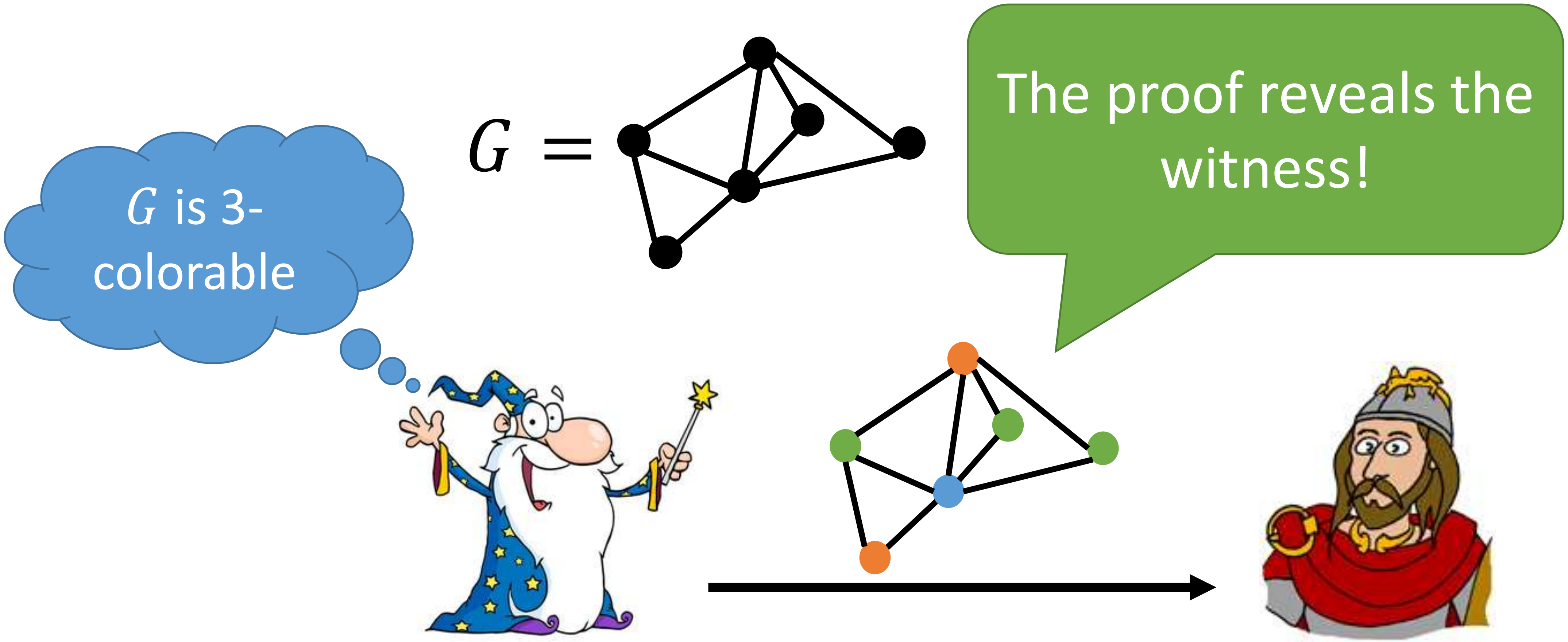
dIP = NP
(dIP: interactive proofs with deterministic verifier)

IP = PSPACE [LFKN90, Sha90]



prover verifier

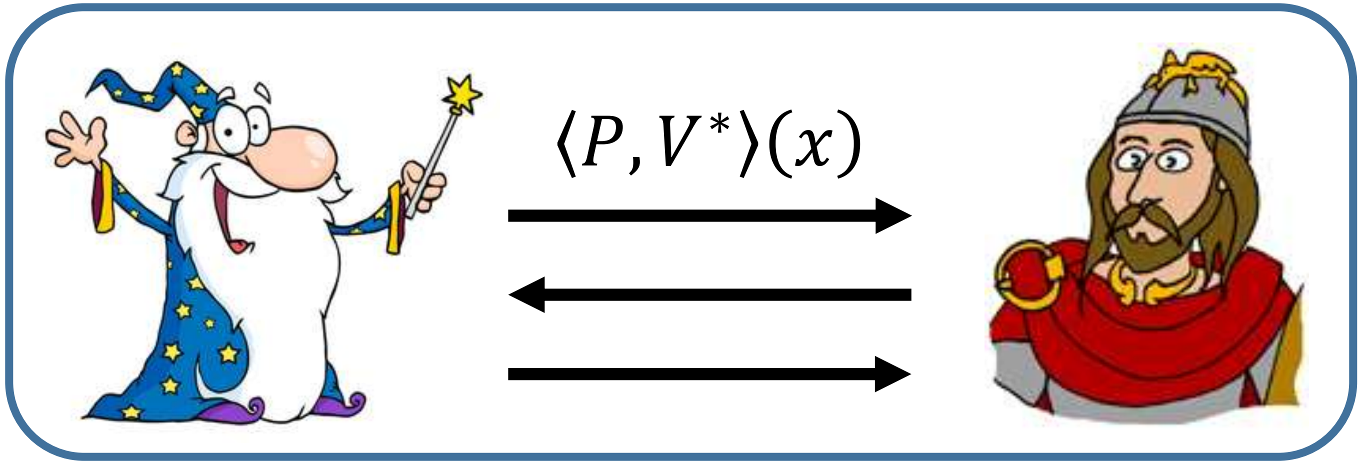
Zero-Knowledge Proofs [GMR85]



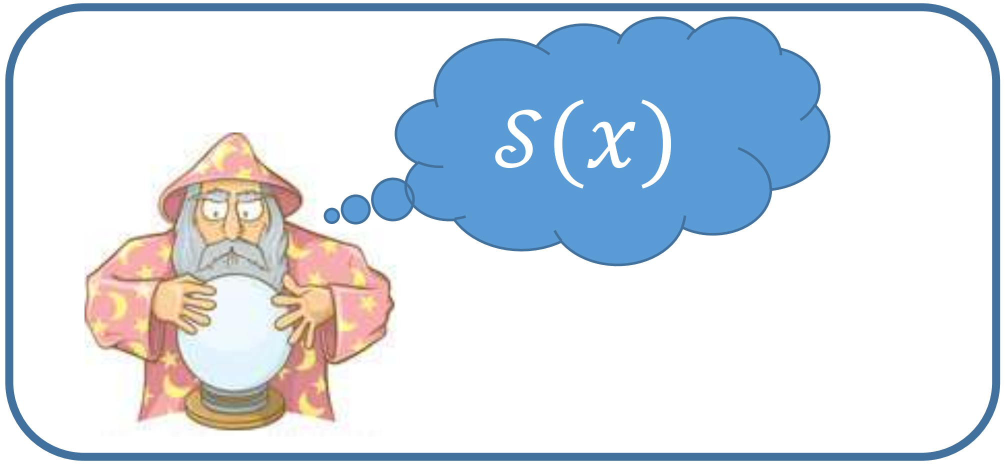
Can we prove to a verifier that a statement x is in a language $\mathcal{L}$ without revealing anything more about x other than the fact that $x \in \mathcal{L}$?

Zero-Knowledge Proofs [GMR85]

common input: statement $x \in \mathcal{L}$



$\approx_c$



real distribution

ideal distribution

Zero-Knowledge: for all efficient verifiers V^*, there exists an efficient simulator $\mathcal{S}$ such that:

$$\forall x \in \mathcal{L} : \langle P, V^* \rangle(x) \approx_c \mathcal{S}(x)$$

Zero-Knowledge Proofs [GMR85]

common input: statement $x \in \mathcal{L}$

$\langle P, V^* \rangle (x)$

$\approx_c$

$\mathcal{S}(x)$

rea

honest verifier zero-knowledge (HVZK): simulator exists only for *honest* verifier

can also consider *statistical* zero-knowledge and *perfect* zero-knowledge

Zero-Knowledge: for all efficient verifiers V^*, th ... or $\mathcal{S}$ such that:

$$\forall x \in \mathcal{L} : \langle P, V^* \rangle (x) \approx_c \mathcal{S}(x)$$

Zero-Knowledge Proofs [GMR85]

common input: statement $x \in \mathcal{L}$

$\langle P, V^* \rangle(x)$

$\approx_c$

$\mathcal{S}(x)$

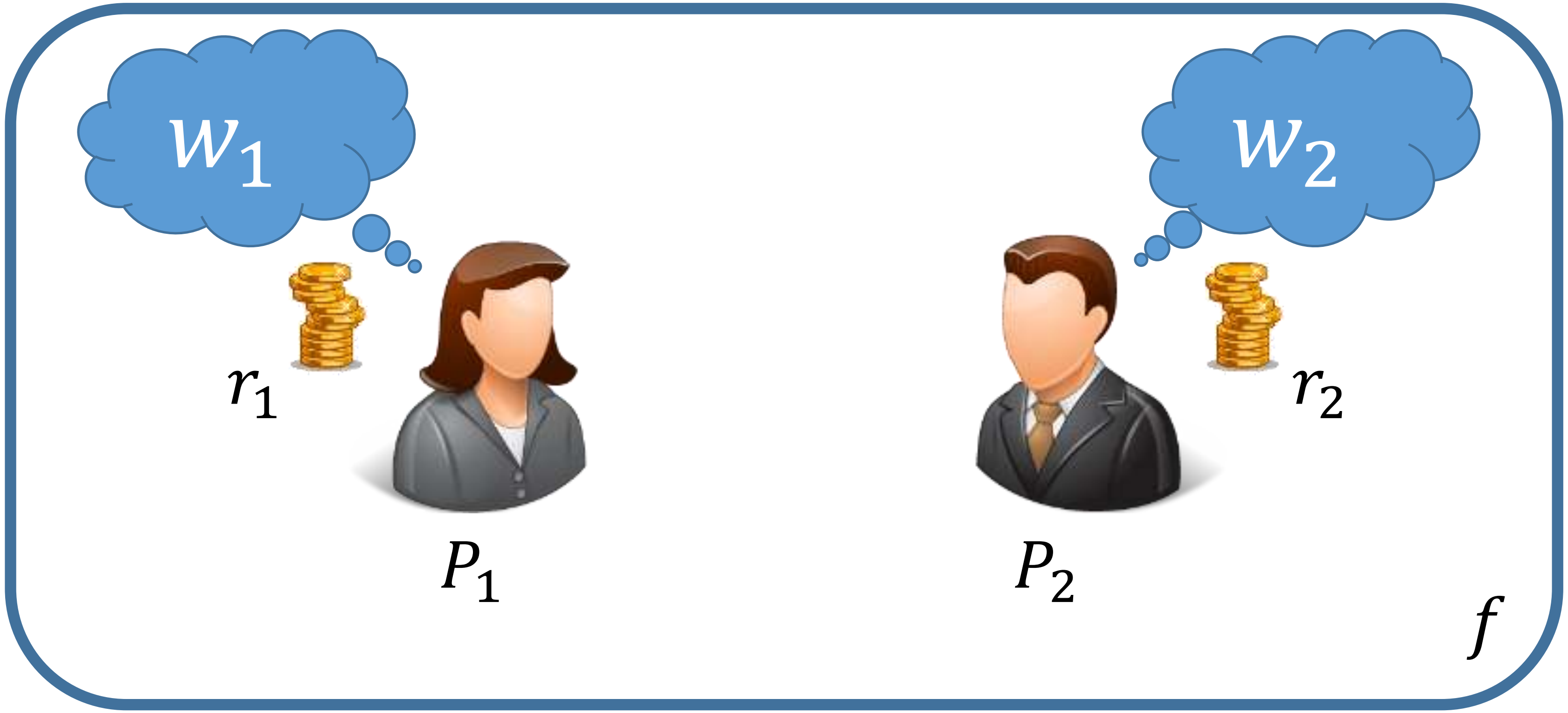
real distribution

ideal distribution

Assuming the existence of one-way functions (OWFs), every **NP** (in fact, **IP**) language has a computational zero-knowledge proof system [GMW86]

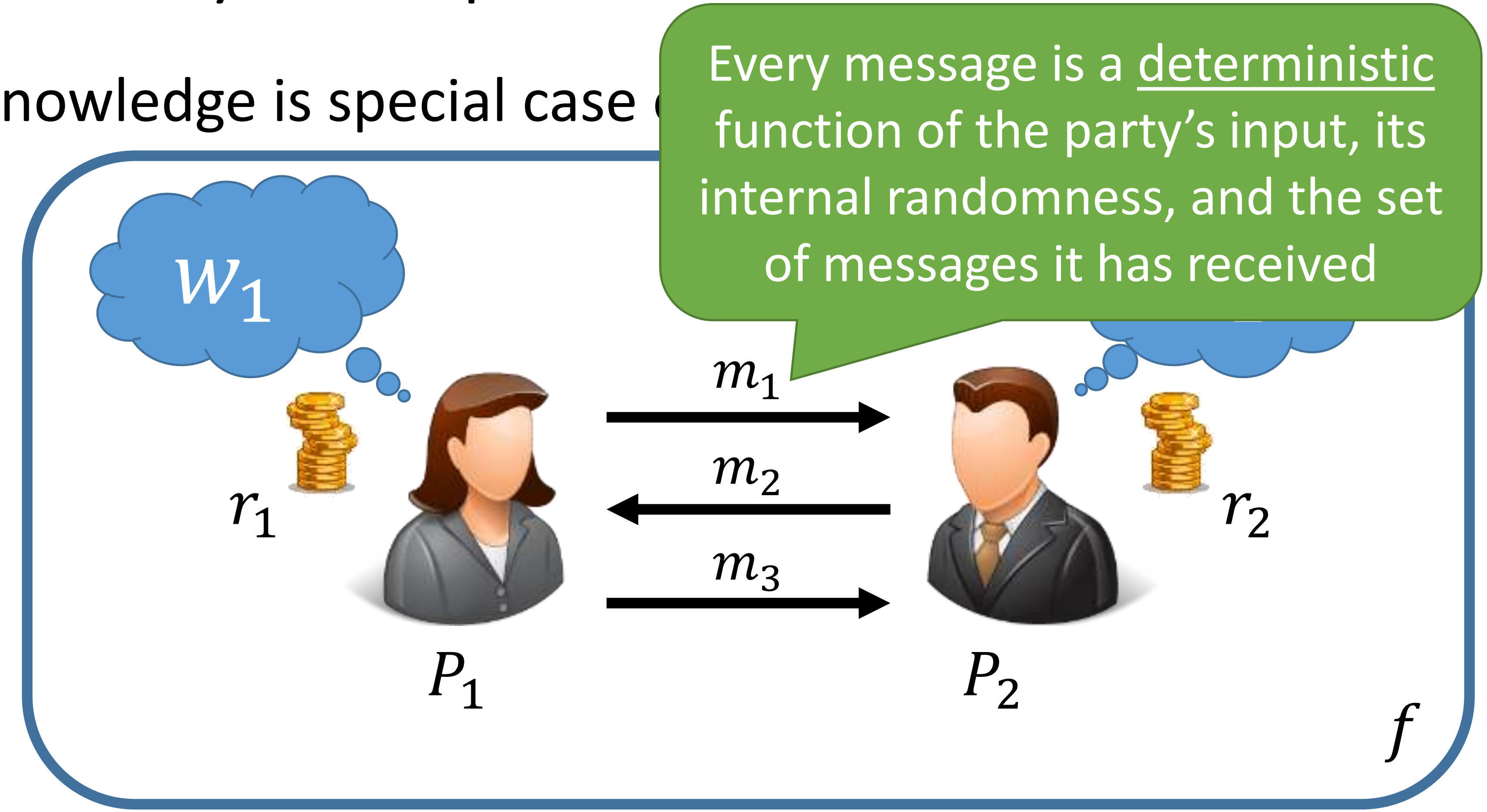
Two-Party Computation

Zero knowledge is special case of <u>two-party computation</u>



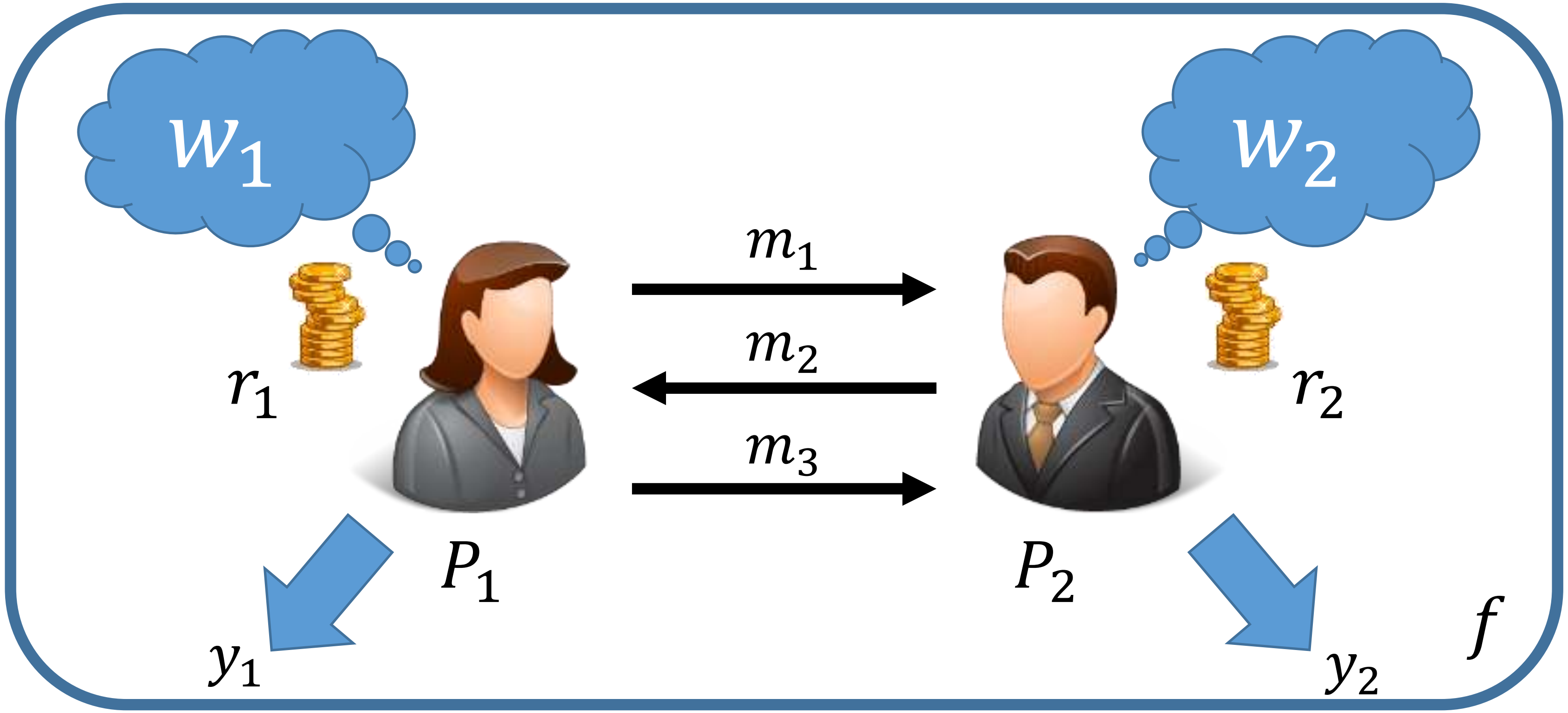
Two-Party Computation

Zero knowledge is special case



Two-Party Computation

Zero knowledge is special case of <u>two-party computation</u>

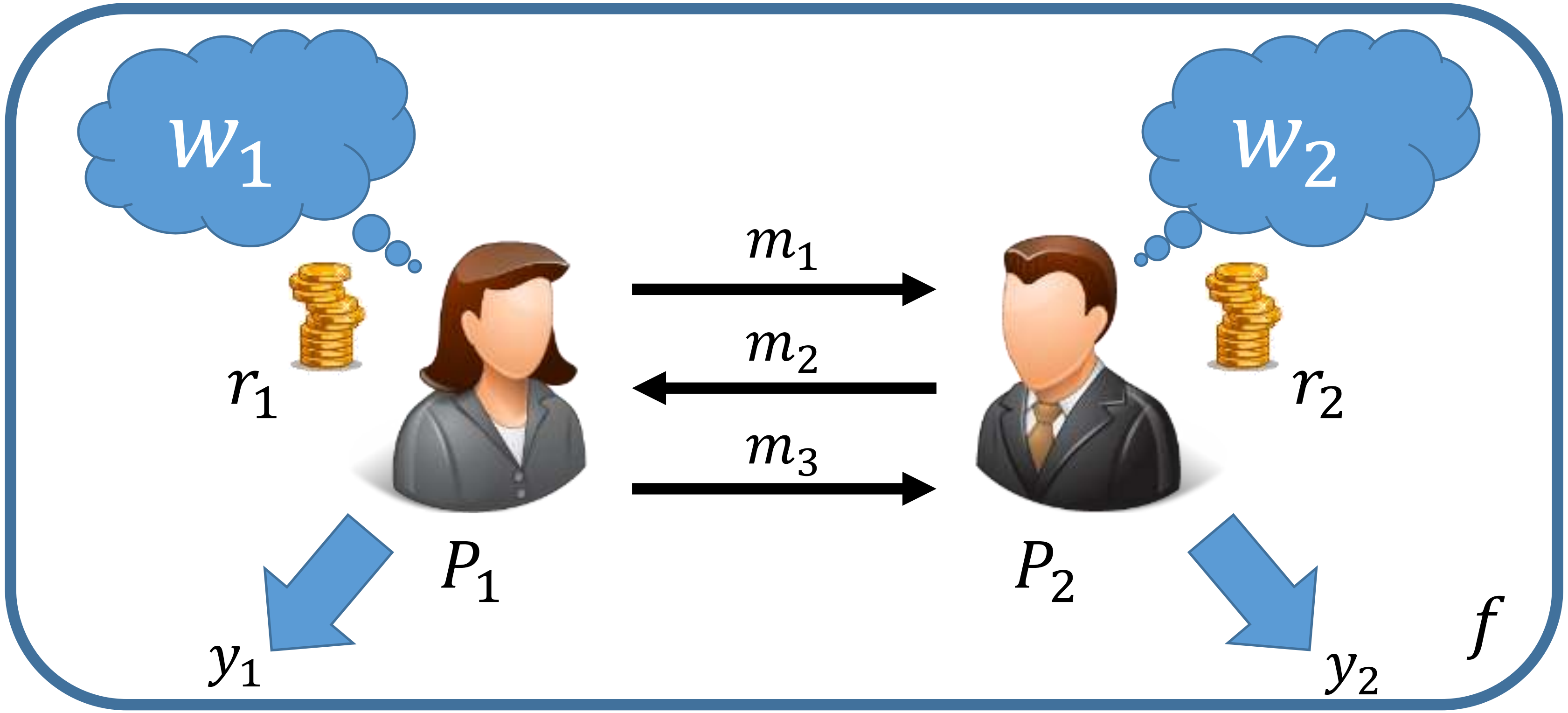


Correctness:

$$y_1 = f(w_1, w_2) = y_2$$

Two-Party Computation

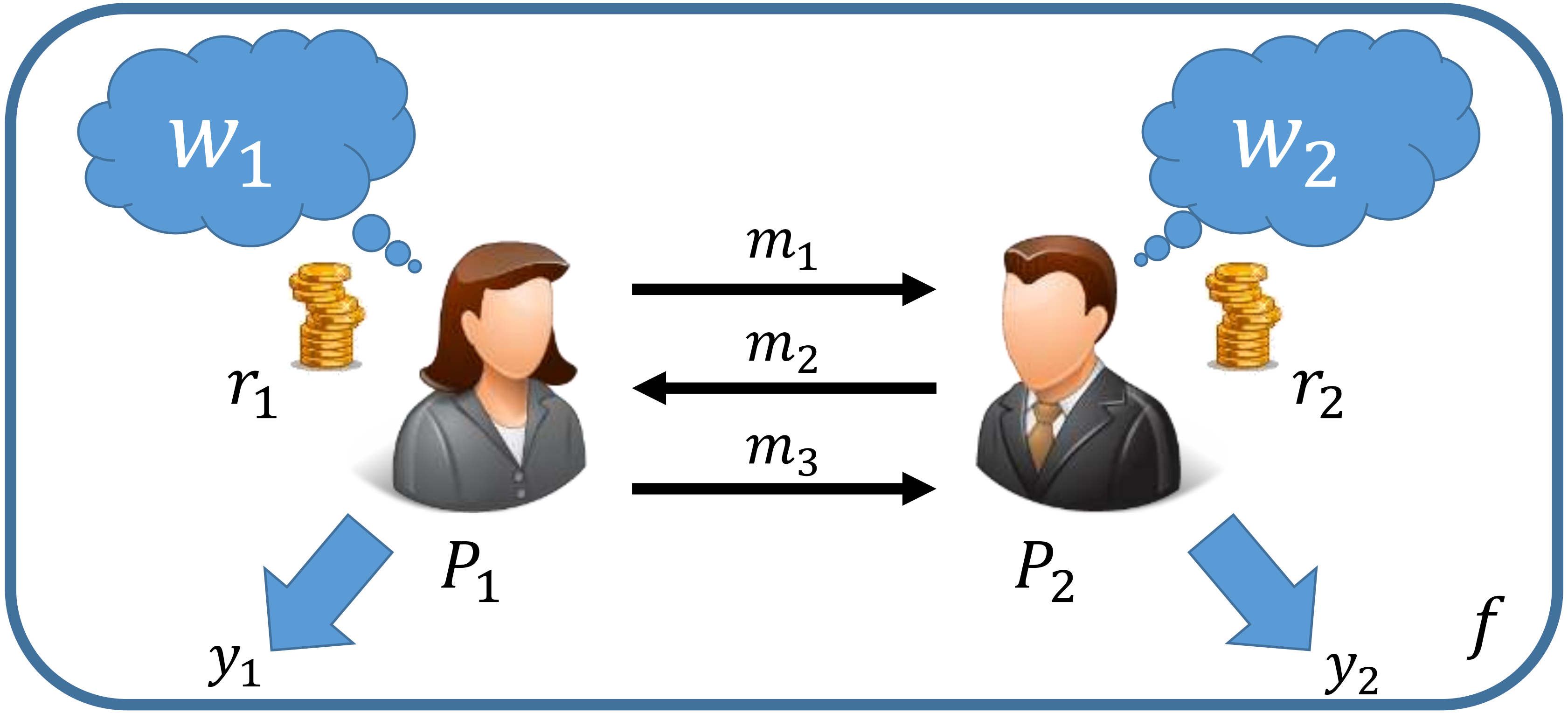
Zero knowledge is special case of <u>two-party computation</u>



$$\text{View}_{P_i}(w_1, w_2; r_1, r_2) = (w_i, r_i, \{m_i\})$$

Two-Party Computation

Zero knowledge is special case of <u>two-party computation</u>



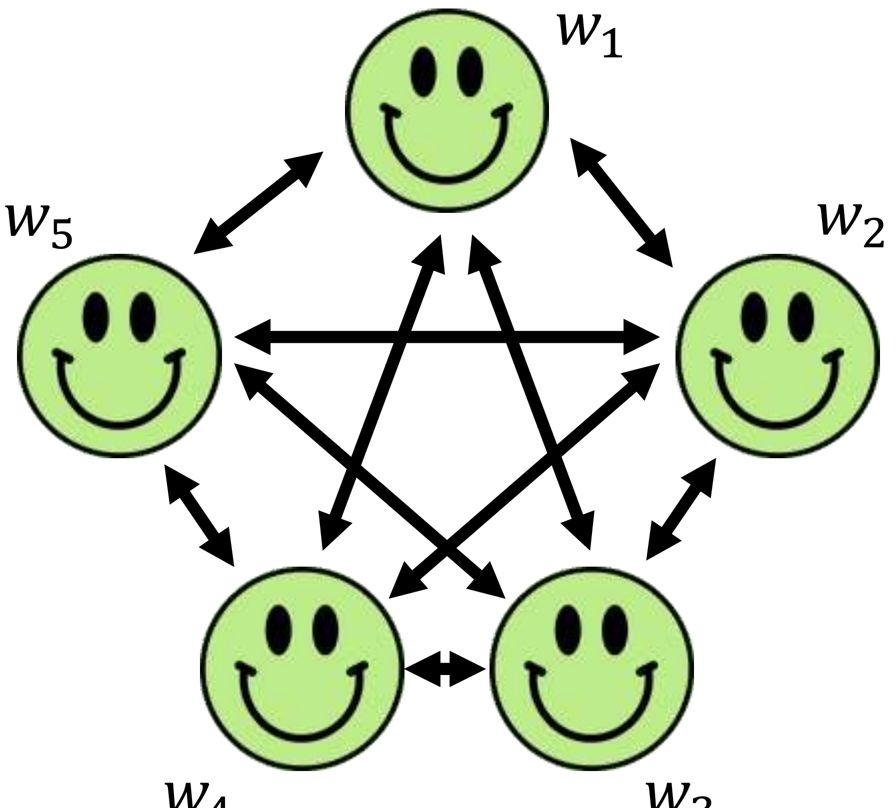
$$\mathrm{View}_{P_i}(w_1, w_2; r_1, r_2) = (w_i, r_i, \{m_i\})$$

$$y_i = \Pi_{f,i}\left(\mathrm{View}_{P_i}(\boldsymbol{w}\,; \boldsymbol{r})\right)$$

Multiparty Computation (MPC)

Correctness: For all inputs $\boldsymbol{w}$ and all $i \in [n]$

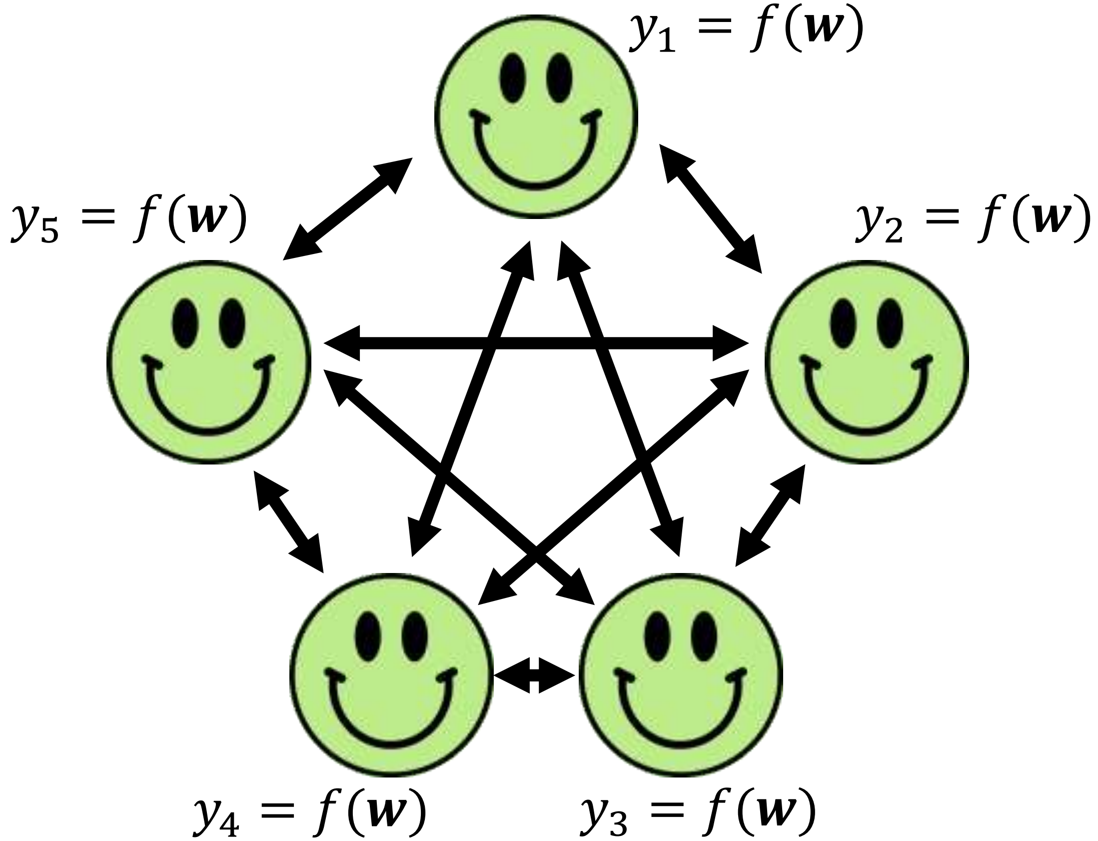
$$\Pr\left[\Pi_{f,i}\left(\mathrm{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r})\right) = f(\boldsymbol{w})\right] = 1$$



Multiparty Computation (MPC)

Correctness: For all inputs $\boldsymbol{w}$ and all $i \in [n]$

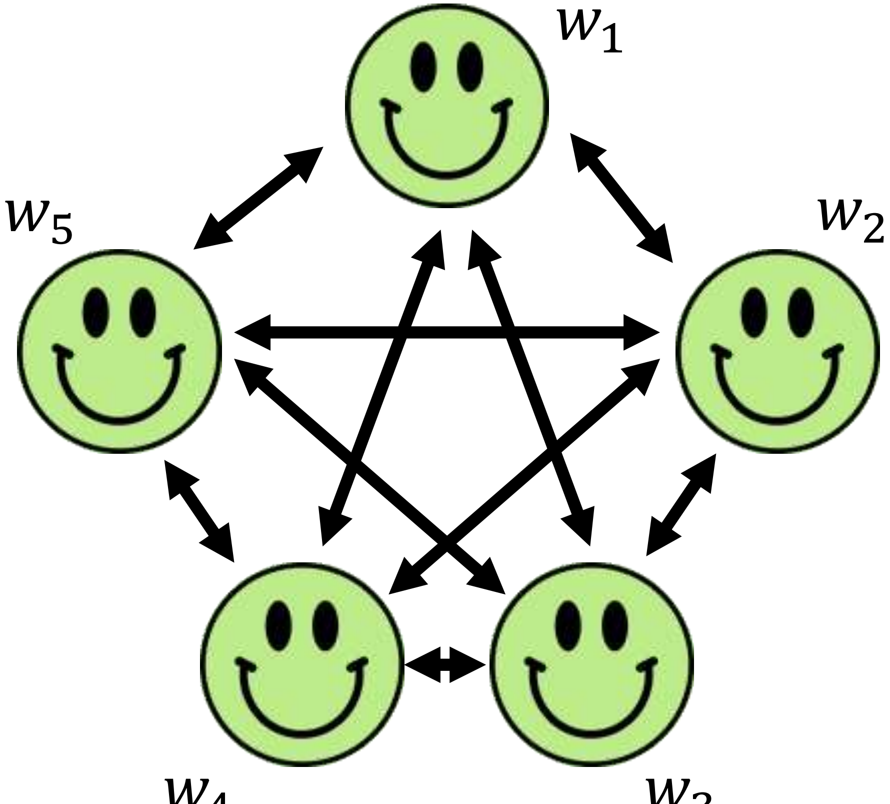
$$\Pr\left[\Pi_{f,i}\left(\text{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r})\right) = f(\boldsymbol{w})\right] = 1$$



Multiparty Computation (MPC)

***t*-Privacy**: For all $T \subseteq [n]$ where $|T| \leq t$, there exists an efficient simulator $\mathcal{S}_T$ such that for all inputs $\boldsymbol{w}$:

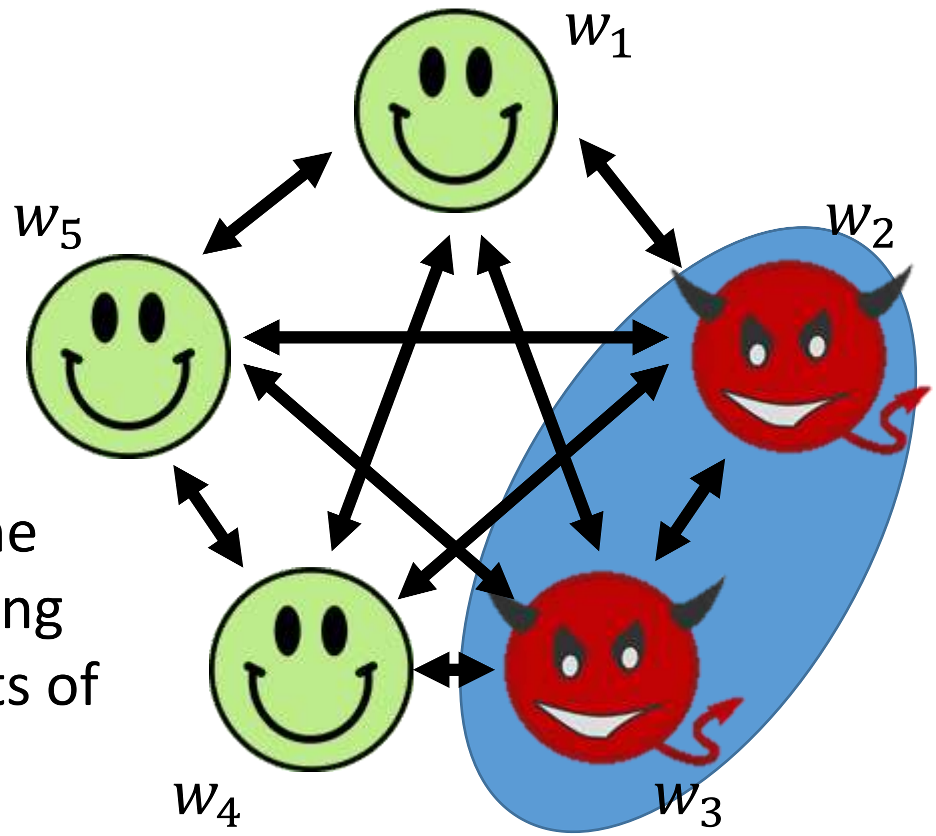
$$\left\{\mathrm{View}_{P_i}(\boldsymbol{w}\,; \boldsymbol{r})\right\}_{i\in T} \equiv \mathcal{S}_T\big(f, \{w_i\}_{i\in A}, f(\boldsymbol{w})\big)$$



Multiparty Computation (MPC)

***t*-Privacy**: For all $T \subseteq [n]$ where $|T| \leq t$, there exists an efficient simulator $\mathcal{S}_T$ such that for all inputs $\boldsymbol{w}$:

$$\{\mathrm{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r})\}_{i\in T} \equiv \mathcal{S}_T\big(f, \{w_i\}_{i\in A}, f(\boldsymbol{w})\big)$$



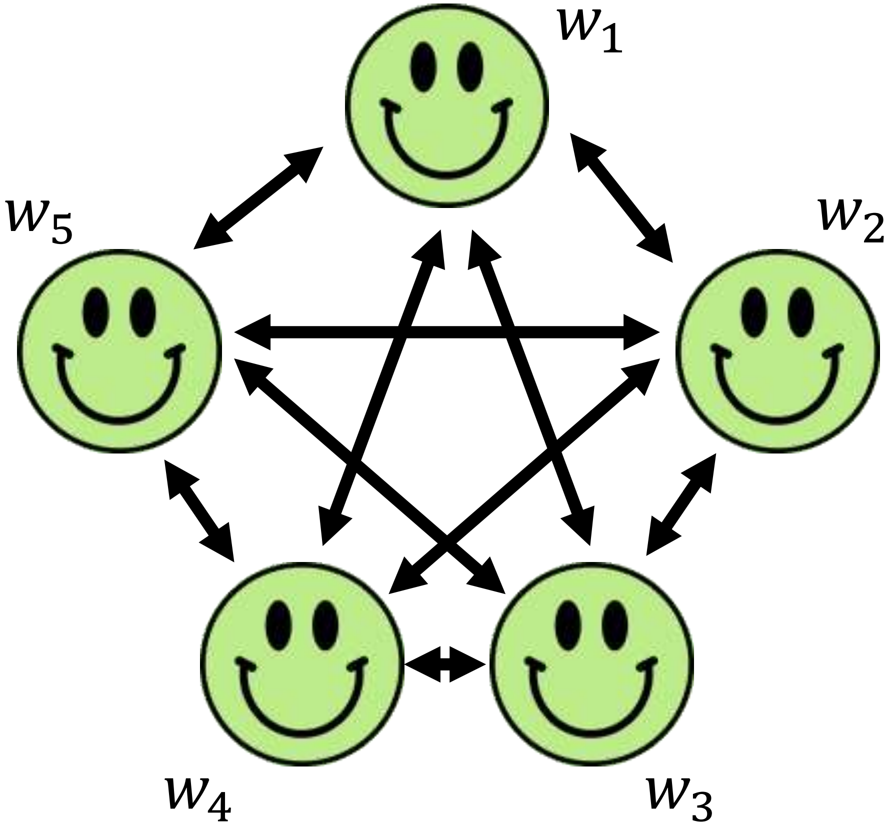
Views of any t-subset of the parties do not reveal anything more about the private inputs of any other party

Multiparty Computation (MPC)

***t*-Robustness**: For all $T \subseteq [n]$ where $|T| \leq t$, and for all f where $f(\boldsymbol{w}) = 0$ for all $\boldsymbol{w}$, then

$$\Pr\left[\Pi_{f,i}\left(\mathrm{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r})\right) = 1\right] = 0$$

for all $i \in [n] \setminus T$ even if the players in T have been *arbitrarily* corrupted

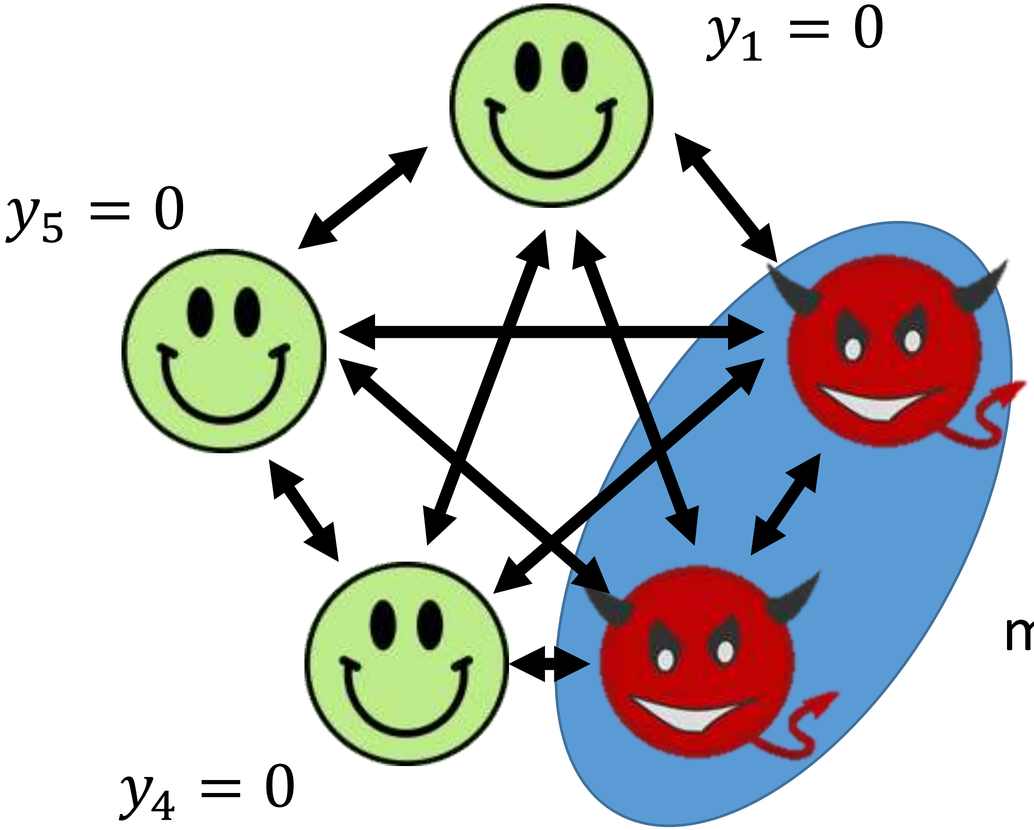


Multiparty Computation (MPC)

***t*-Robustness**: For all $T \subseteq [n]$ where $|T| \leq t$, and for all f where $f(\boldsymbol{w}) = 0$ for all $\boldsymbol{w}$, then

$$\Pr\left[\Pi_{f,i}\left(\mathrm{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r})\right) = 1\right] = 0$$

for all $i \in [n] \setminus T$ even if the players in T have been *arbitrarily* corrupted



If there are no inputs $\boldsymbol{w}$ to f where $f(\boldsymbol{w}) = 1$, then a malicious adversary corrupting up to t parties cannot cause an honest party to output 1

Zero-Knowledge from Two-Party Computation

Zero knowledge is special case of two-party computation

- Given a statement x for an **NP** relation R, define the function
$$f_x(w) = R(x, w)$$
- We require a 1-private, 1-robust two-party computation protocol Π_{f_x} for f_x
- The prover and verifier execute Π_{f_x}
  - Prover's input: the witness w
  - Verifier's input: none
- The verifier accepts if the output of Π_{f_x} is 1

Zero-Knowledge from Two-Party Computation

Zero knowledge is special case of two-party computation

- General two-party computation with robustness against malicious adversaries requires oblivious transfer (OT) [Yao86, GMW87] and thus, <u>cannot</u> be instantiated from one-way functions

On the other hand, zero knowledge for **NP** is known from one-way functions (OWFs) [GMW86]

- Constructions very inefficient – relies on running a Karp reduction to an **NP**-complete problem (e.g., 3-coloring)

This talk: constructing zero-knowledge for **NP** from OWFs + black-box use of *any* (semi-honest) MPC protocol

"MPC in the Head" [IKOS07]

Let $R(x, w)$ be an **NP** relation and define the function

$$f_x(w_1, \dots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n),$$

where $n \geq 3$

Key idea:

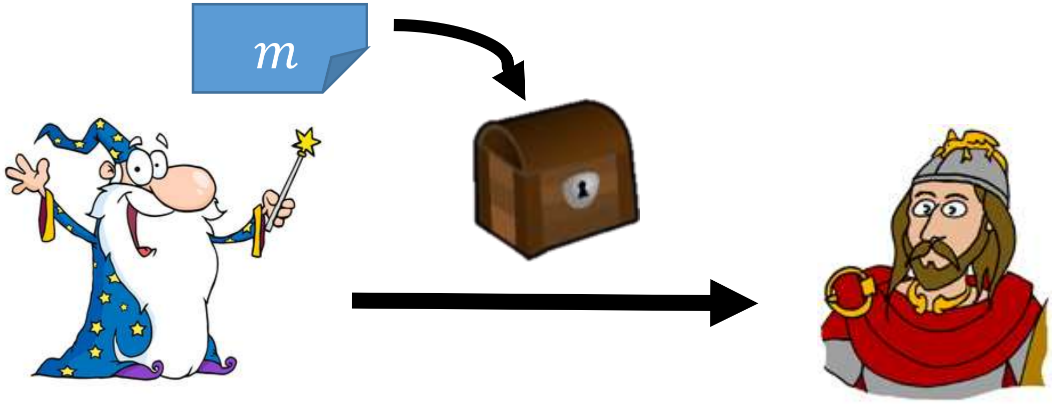
- Prover "simulates" an n-party MPC protocol Π_{f_x} for the function f_x
- Verifier checks that the simulation is correct

Key advantage: relies only on OWFs and <u>semi-honest</u> secure MPC

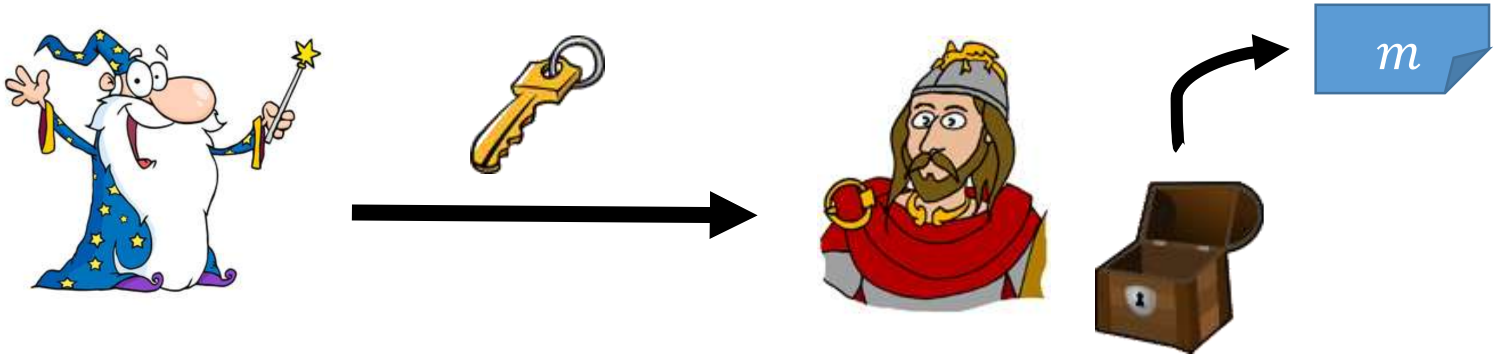
“MPC in the Head” [IKOS07]

Key cryptographic primitive: commitment scheme

$\text{Commit}(m; r) \to c$



$\text{Open}(c, r) \to m$



“MPC in the Head” [IKOS07]

Key cryptographic primitive: commitment scheme

- **Perfectly binding**: each commitment can be opened in exactly one way

$$\forall r_0, r_1 : \mathrm{Commit}(m_0\,; r_0) = \mathrm{Commit}(m_1\,; r_1) \Rightarrow m_0 = m_1$$

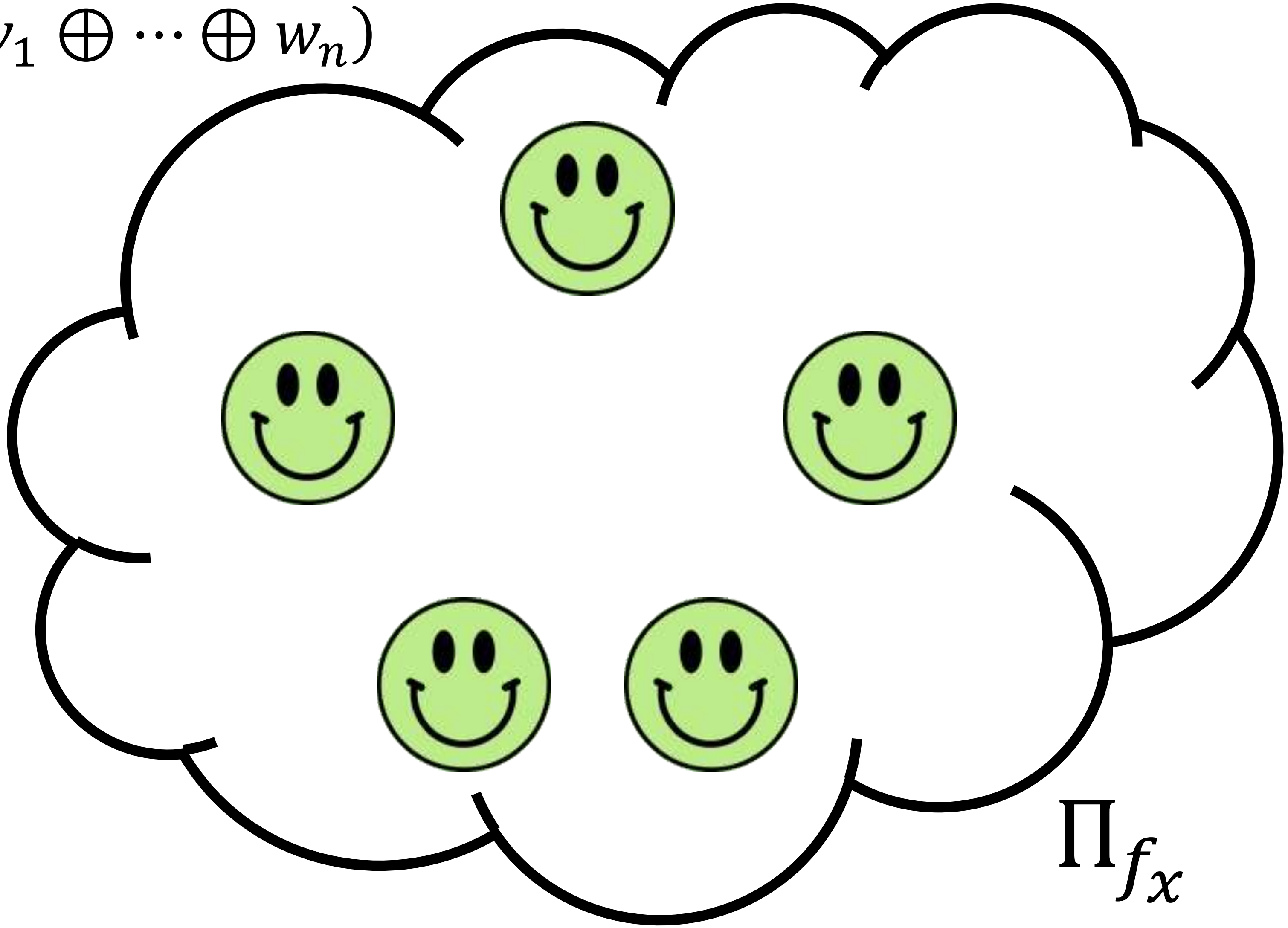
- **Computationally hiding**: commitment hides committed value to any bounded adversary:

$$\mathrm{Commit}(m_0\,; r) \approx_c \mathrm{Commit}(m_1\,; r)$$

- Non-interactive commitments can be constructed from any injective OWF [Blu81]
- Interactive commitments can be constructed from any OWF

“MPC in the Head” [IKOS07]

$$f_x(w_1, \dots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n)$$



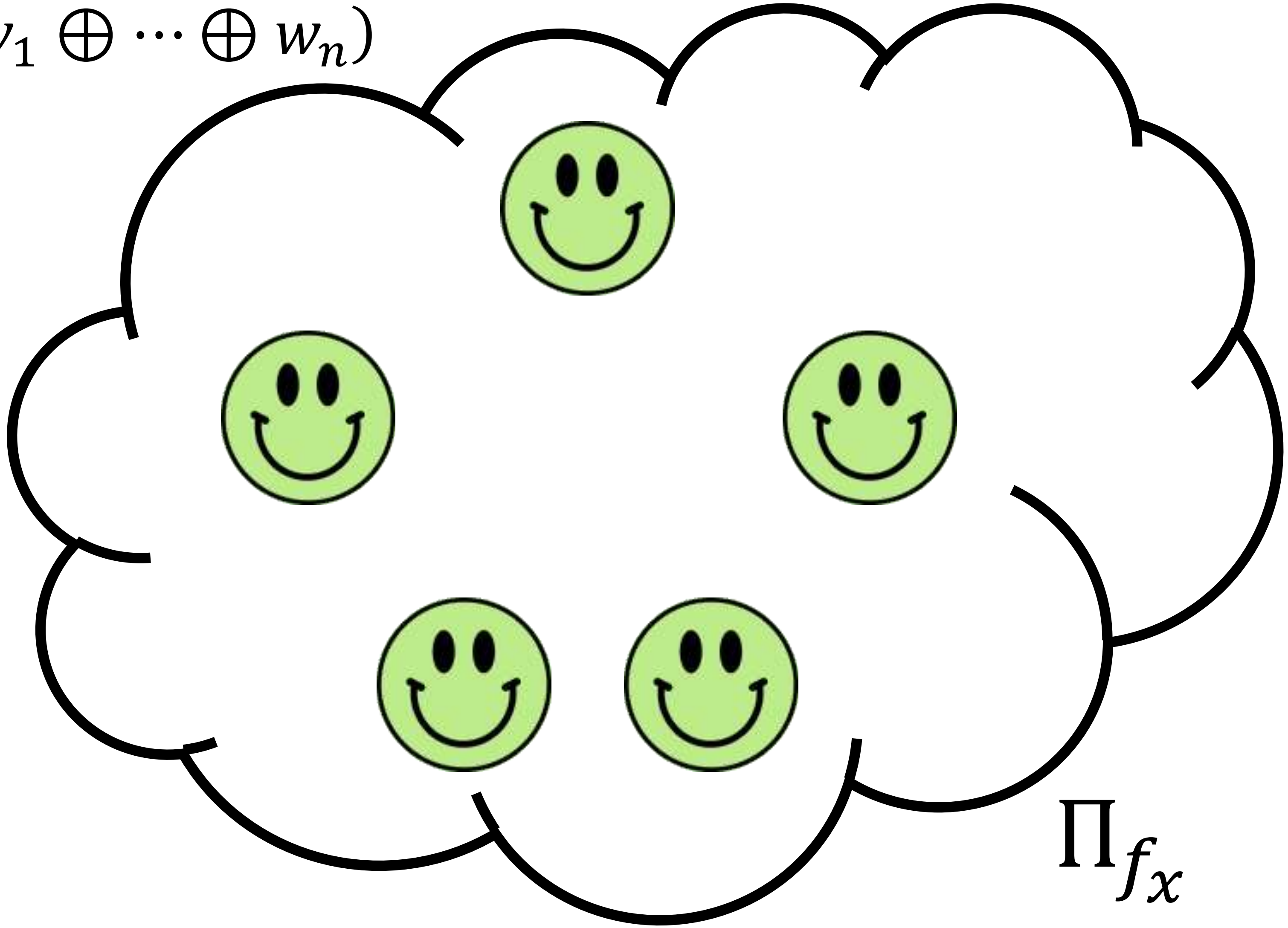
$R(x, w)$ is an **NP** relation

“MPC in the Head” [IKOS07]

$$f_x(w_1, \dots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n)$$

$$w = w_1 \oplus w_2 \oplus w_3 \oplus w_4 \oplus w_5$$

uniformly random strings



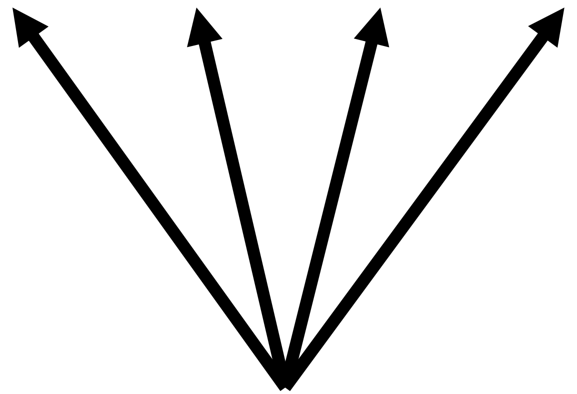
Π_{f_x}

Step 1: Secret share the witness

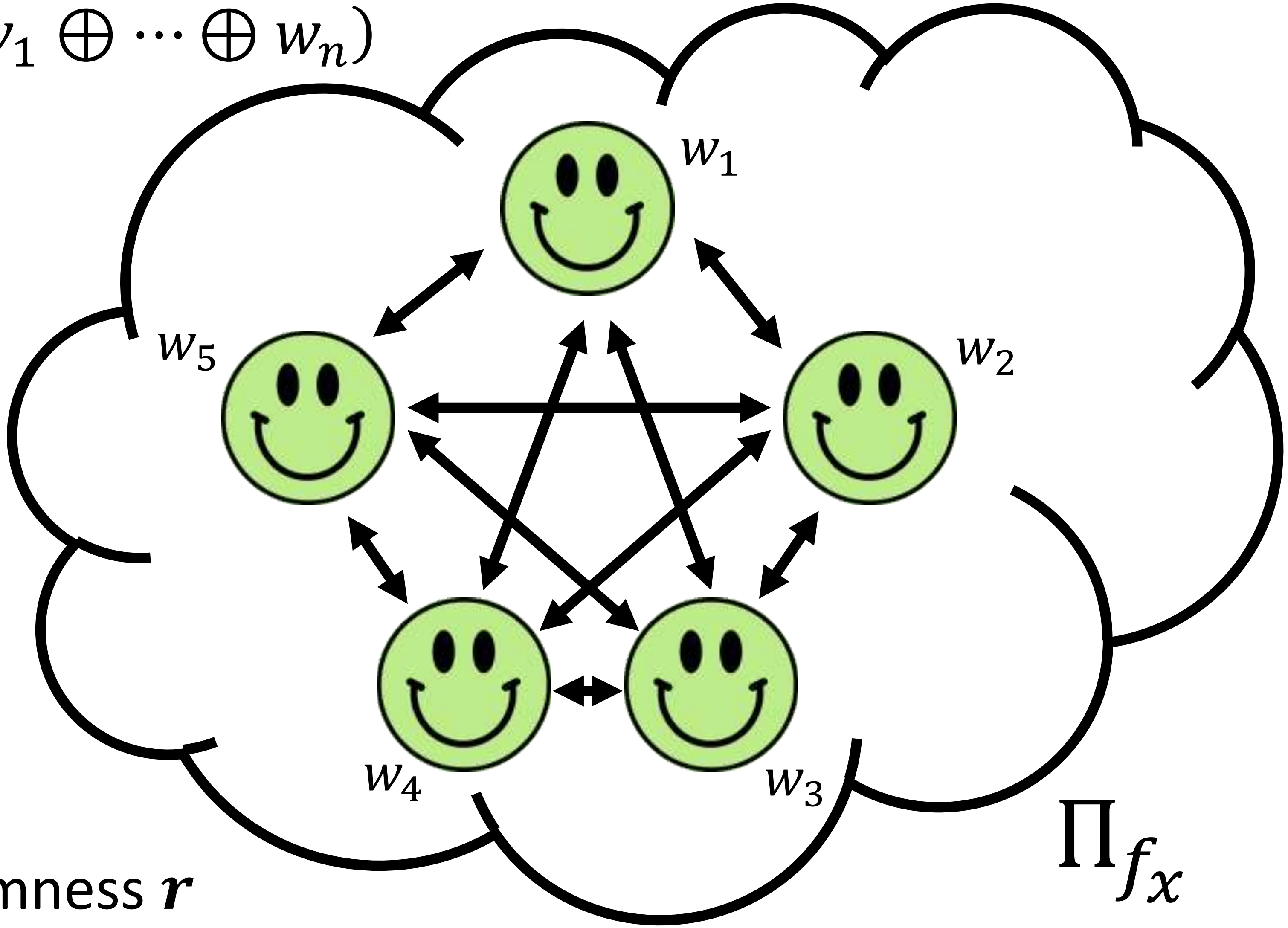
“MPC in the Head” [IKOS07]

$$f_x(w_1, \dots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n)$$

$$w = w_1 \oplus w_2 \oplus w_3 \oplus w_4 \oplus w_5$$



uniformly random strings

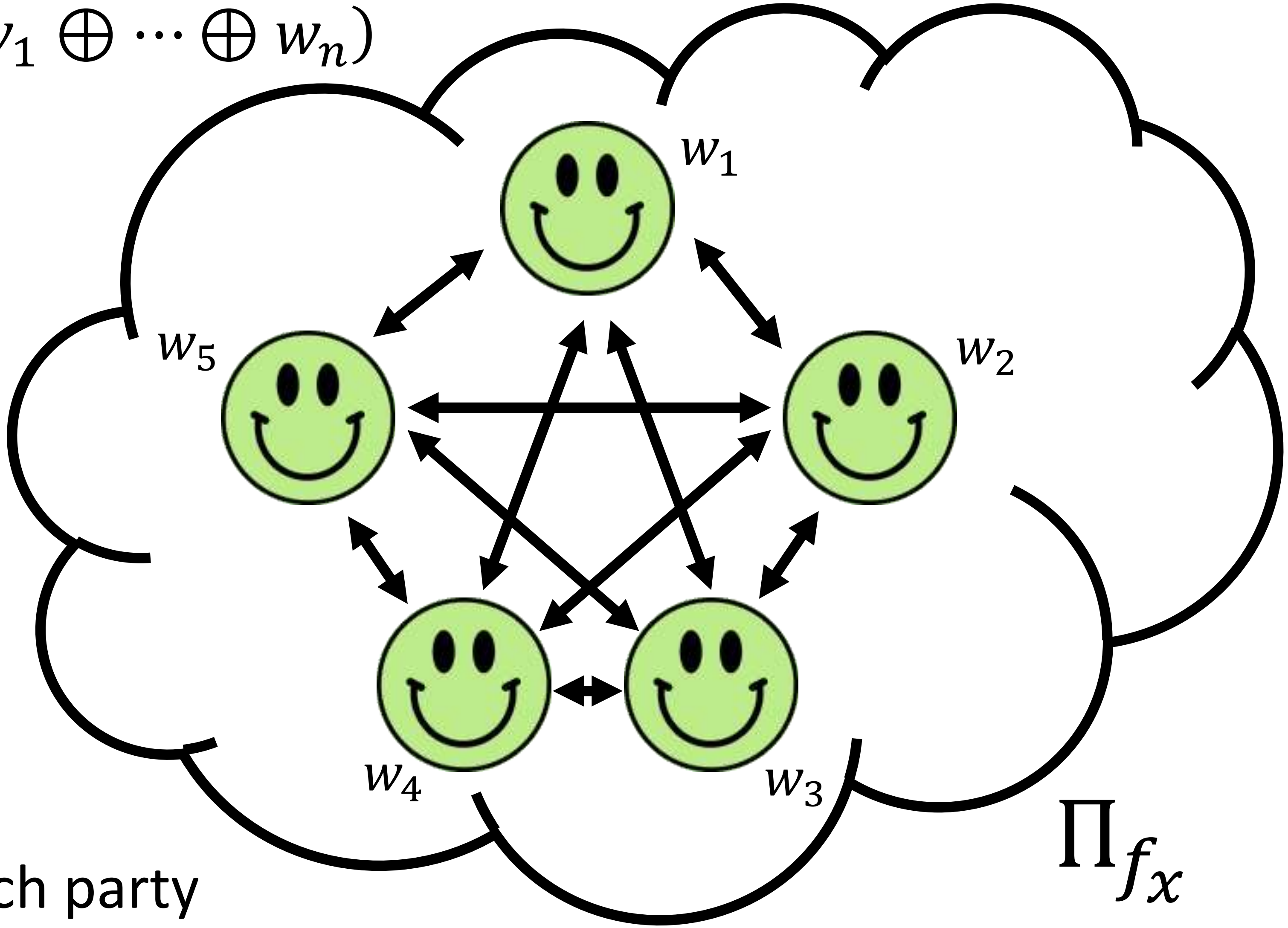


Step 2: Simulate Π_{f_x} using randomness $\boldsymbol{r}$

“MPC in the Head” [IKOS07]

$$f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n)$$

$$\begin{aligned} c_1 &= \mathrm{Commit}\big(\mathrm{View}_{P_1}(\boldsymbol{w}; \boldsymbol{r}); r_1'\big) \\ &\vdots \\ c_n &= \mathrm{Commit}\big(\mathrm{View}_{P_n}(\boldsymbol{w}; \boldsymbol{r}); r_n'\big) \end{aligned}$$



Step 3: Commit to the view of each party

“MPC in the Head” [IKOS07]

public input: statement x

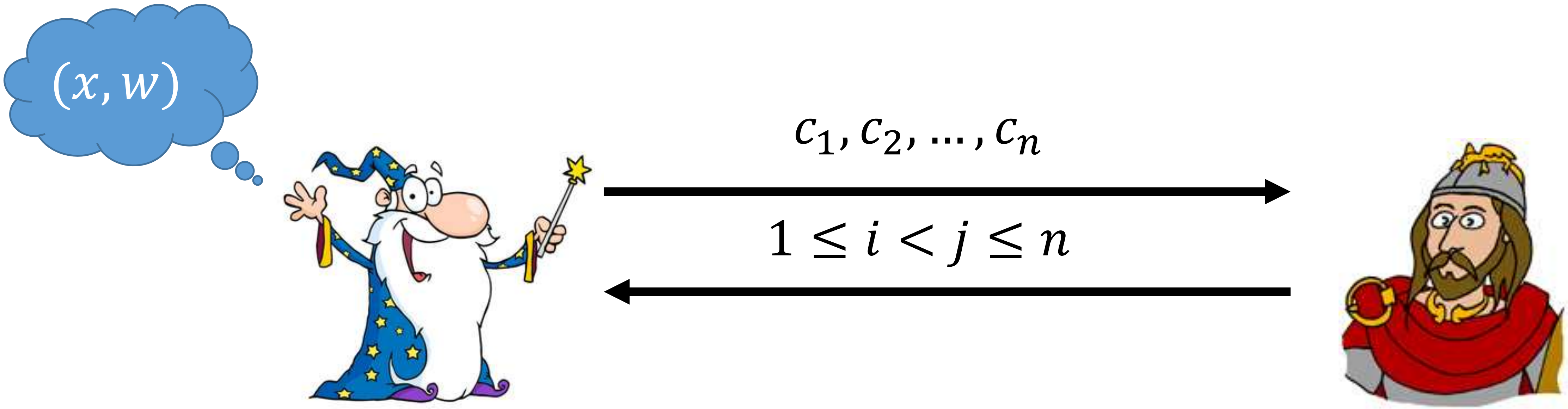
(x, w)

$c_1, c_2, \ldots, c_n$

Step 4: Prover sends the commitments to the verifier

“MPC in the Head” [IKOS07]

public input: statement x



Step 5: Verifier challenges prover to open two of the views (at random)

“MPC in the Head” [IKOS07]

public input: statement x

$c_1, c_2, \dots, c_n$ $\longrightarrow$

$1 \le i < j \le n$ $\longleftarrow$

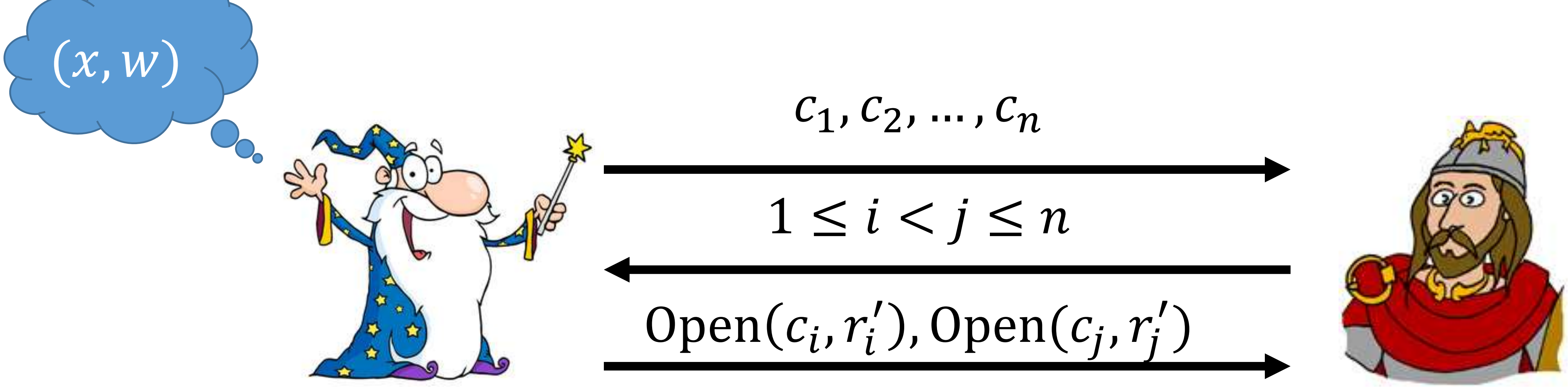
$\mathrm{Open}(c_i, r_i'), \mathrm{Open}(c_j, r_j')$ $\longrightarrow$

Step 6: Prover opens up commitments to requested views

“MPC in the Head” [IKOS07]

Verification conditions:

1. Commitments are correctly opened
2. The outputs of both P_i and P_j is 1
3. The views View_{P_i} and View_{P_j} are consistent with an honest execution of Π_{f_x}



Step 7: Verifier checks the proof

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Completeness:

- Suppose $R(x, w) = 1$
- Prover is honest so $w = w_1 \oplus \cdots \oplus w_n$
- By construction, $f_x(w_1, \ldots, w_n) = R(x, w) = 1$
- Perfect correctness of Π_{f_x} implies that all parties in honest execution output $f_x(w_1, \ldots, w_n) = 1$

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Soundness:

- Suppose $R(x, w) = 0$ for all w
- By perfect correctness of Π_{f_x}, for *all* choices of $w_1, \ldots, w_n$, parties in an honest execution of Π_{f_x} will output 0
- Either all outputs are 0 or there is at least one pair of views that are inconsistent
- Verifier rejects with probability at least $1/n^2$ (commitments are perfectly binding)

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Can be amplified by running the protocol multiple times (κn^2 times to achieve negligible soundness error $2^{-\kappa}$)

Soundness:

- Suppose $R(x, w) = 0$ for all w
- By perfect correctness of Π_{f_x}, … honest execution of Π_{f_x} will output 0
- Either all outputs are 0 or there is at leas… e pair of views that are inconsistent
- Verifier rejects with probability at least $1/n^2$ (commitments are perfectly binding)

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Zero-Knowledge:

- Suppose that $R(x, w) = 1$
- View of verifier consists of committed views to all parties, and views $\mathrm{View}_{P_i}(\boldsymbol{w}; \boldsymbol{r})$ and $\mathrm{View}_{P_j}(\boldsymbol{w}; \boldsymbol{r})$ (which include w_i and w_j) for two of the parties
- When $n \geq 3$, w_i, w_j are uniformly random strings
- By 2-privacy of Π_{f_x}, $\mathrm{View}_{P_i}(\boldsymbol{w}; \boldsymbol{r})$ and $\mathrm{View}_{P_j}(\boldsymbol{w}; \boldsymbol{r})$ can be simulated given just $f_x, w_i, w_j, f_x(\boldsymbol{w}) = 1$

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Zero-Knowl

- Suppo
- View o o all parties, and views View_{P_i} de w_i and w_j) for two of the parties
- When $n \geq$ w_j are uniformly random strings
- By 2-privacy of Π_{f_x}, $\text{View}_{P_i}(\boldsymbol{w}; \boldsymbol{r})$ and $\text{View}_{P_j}(\boldsymbol{w}; \boldsymbol{r})$ can be simulated given just $f_x, w_i, w_j, f_x(\boldsymbol{w}) = 1$

Since prover is honest here, the proof only requires privacy against *semi-honest* parties

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose the commitment scheme is perfectly binding and computationally hiding and that Π_{f_x} is perfectly correct and is 2-private (against semi-honest adversaries), then this protocol is a zero-knowledge proof for the NP-relation R.

Concrete instantiations:

- Information-theoretic: 5-party BGW protocol [BGW88]
- Computational (based on OT): 3-party GMW protocol [GMW87]
- … and many more

"MPC in the Head" [IKOS07]

Using an n-party MPC protocol, the soundness error is $1 - 1/n^2$

Consequence: achieving negligible soundness $2^{-\kappa}$ requires $\Omega(\kappa)$ repetitions of the protocol

Can we obtain negligible soundness error without performing the $\Omega(\kappa)$ repetitions of the protocol?

“MPC in the Head” [IKOS07]

Using an n-party MPC protocol, the soundness error is $1 - 1/n^2$

Soundness error is large because verifier checks only a *single* view

Can reduce the soundness error by having the prover open up more views (e.g., $t = \Theta(\kappa)$ views)

- Zero-knowledge maintained as long as Π_{f_x} is t-private
- Soundness amplification will rely on leveraging robustness of Π_{f_x}

"MPC in the Head" [IKOS07]

Using an n-party MPC protocol, the soundness error is $1 - 1/n^2$

Soundness error is large because

Without robustness, even if the prover open $n-1$ views, the soundness error can still be $O\left(\frac{1}{n}\right)$

Can reduce the soundness error ews (e.g., $t = \Theta(\kappa)$ views)

- Zero-knowledge maintained as long as Π_{f_x} is t-priva
- Soundness amplification will rely on leveraging robustness of Π_{f_x}

"MPC in the Head" [IKOS07]

$n = \Theta(\kappa)$ $t = \Theta(n)$

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

$c_1, c_2, \ldots, c_n$ →

← $T \subseteq [n]$ where $|T| = t$

$\{\mathrm{Open}(c_i, r_i')\}_{i \in [T]}$ →

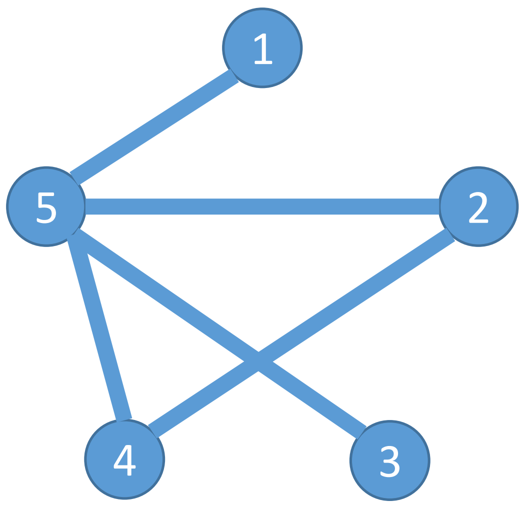
Verifier can now ask for t openings *without* compromising zero-knowledge

“MPC in the Head” [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

To analyze soundness, define the inconsistency graph G for the prover’s simulated MPC protocol:

- Nodes correspond to parties
- An edge between i and j denotes an inconsistency between View_{P_i} and View_{P_j}

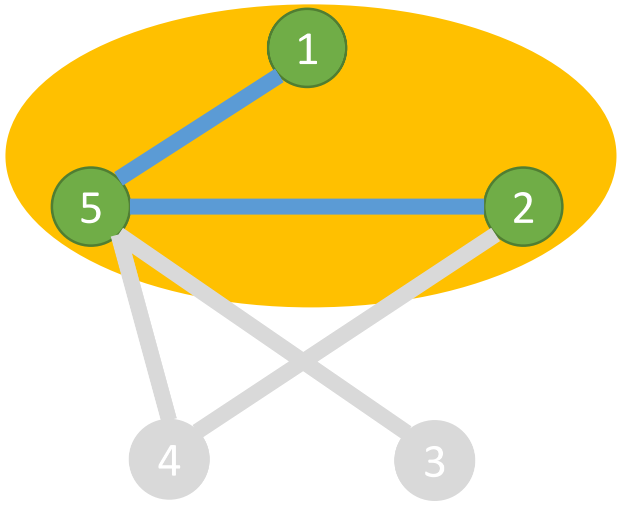


“MPC in the Head” [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

To analyze soundness, define the inconsistency graph G for the prover’s simulated MPC protocol:

Verifier chooses some subset of nodes and rejects if induced subgraph on those nodes contains an edge



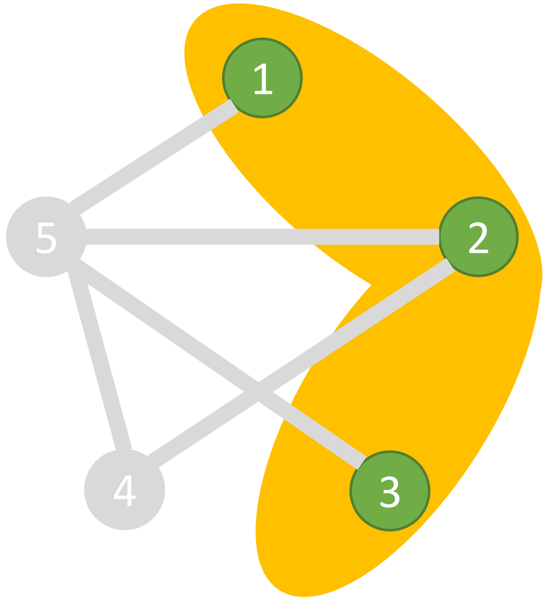
Verifier rejects

“MPC in the Head” [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

To analyze soundness, define the inconsistency graph G for the prover’s simulated MPC protocol:

Verifier chooses some subset of nodes and rejects if induced subgraph on those nodes contains an edge

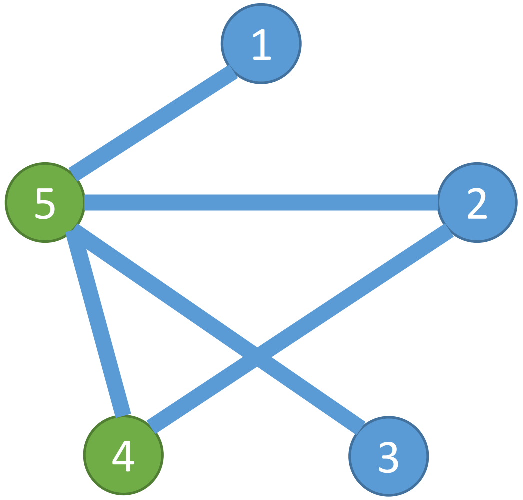


Verifier may accept

"MPC in the Head" [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

Case 1: Suppose G contains a vertex cover B of size at most t



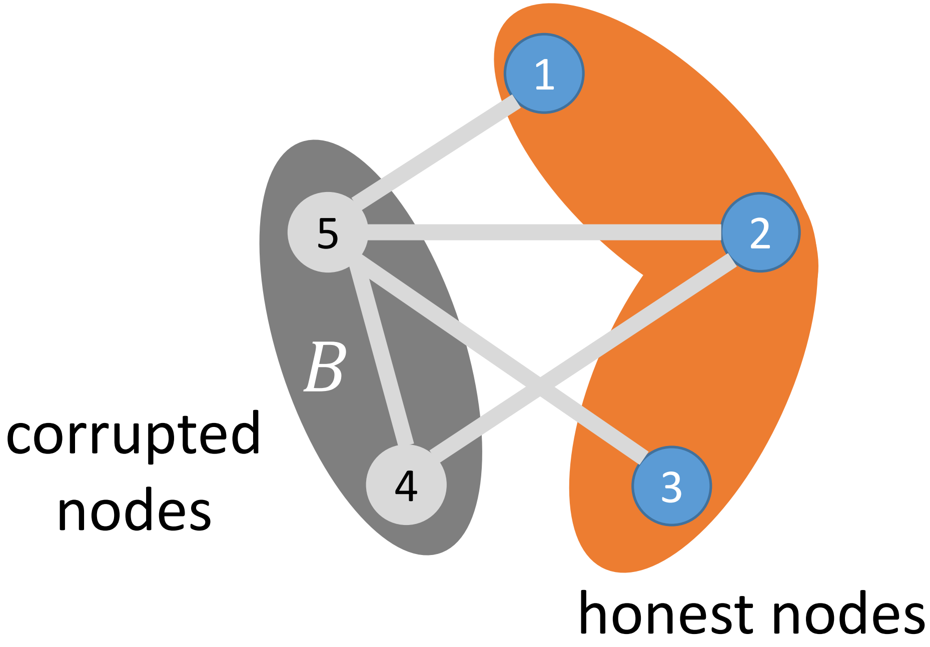
$B = \{4,5\}$

small number of corrupted parties ⇒ most parties are honest and will output 0 by robustness

"MPC in the Head" [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

Case 1: Suppose G contains a vertex cover B of size at most t



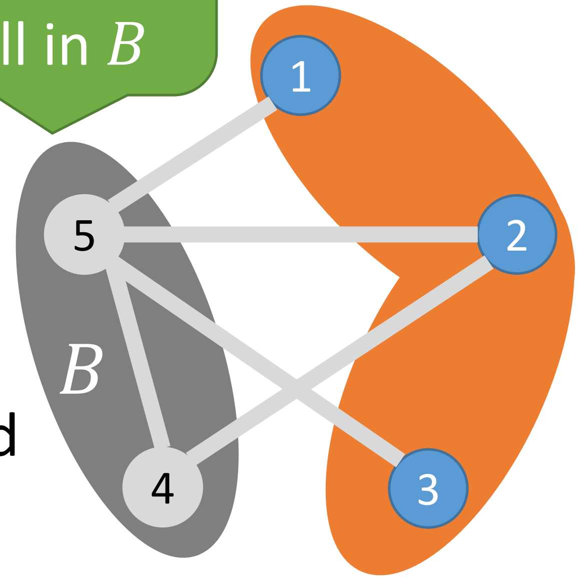
- By definition, views of all nodes not in B are *consistent* (i.e., correspond to an honest protocol execution)
- Π_{f_x} is t-robust, so all nodes not in B output 0 on a false statement

“MPC in the Head” [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

Failure only if <u>all</u> nodes chosen by verifier fall in B

ose G contains a vertex cover B of size at most t

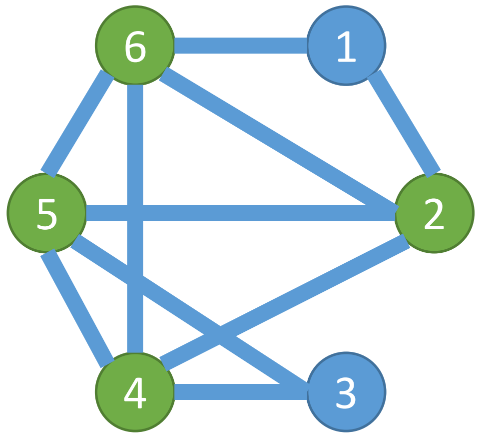


- By definition, views of all nodes not in B are *consistent* (i.e., correspond to an honest protocol execution)
- Π_{f_x} is t-robust, so all nodes not in B output 0 on a false statement
- Verifier can only accept if $T \subseteq B$, so soundness error is bounded by $(t/n)^t = 2^{-\Omega(n)} = 2^{-\Omega(\kappa)}$

"MPC in the Head" [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

Case 2: Suppose the minimum vertex cover of G has size greater than t

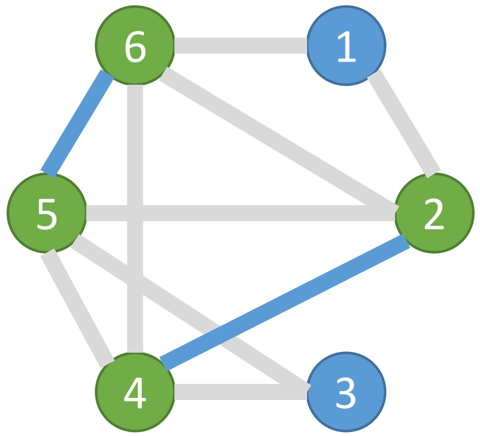


large number of corrupted parties ⇒ likely to be detected by verifier

“MPC in the Head” [IKOS07]

Suppose Π_{f_x} is an n-party MPC protocol that is t-private and t-robust

Case 2: Suppose the minimum vertex cover of G has size greater than t



- Then G has a matching of size greater than $t/2$
- Verifier accepts only if no edges in G between any of the nodes in T, and in particular, no edges in the matching
- Since $t = \Theta(n)$, the verifier misses all edges in the matching with probability $2^{-\Omega(n)} = 2^{-\Omega(\kappa)}$

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose that the following holds:

- the commitment scheme is perfectly binding and computationally hiding,
- Π_{f_x} is t-private (against semi-honest adversaries), and t-robust (against malicious adversaries) n-party protocol for f_x.

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-verifier zero-knowledge proof for the NP-relation R with soundness error $2^{-\kappa}$.

Relies only on OWFs (for the commitments) and black-box access to Π_{f_x}.

“MPC in the Head” [IKOS07]

Theorem [IKOS07]. Suppose that the following holds:

- the commitment scheme is perf … ionally hiding,
- Π_{f_x} is t-private (against semi-ho … ust (against malicious adversaries) n-party protocol …

Can be boosted to zero-knowledge by having the verifier commit to its queries using *a statistically-hiding* commitment scheme [Ros04]

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-verifier zero-knowledge proof for the NP-relation R with soundness error $2^{-\kappa}$.

Relies only on OWFs (for the commitments) and black-box access to Π_{f_x}.

"MPC in the Head" [IKOS07]

Theorem [IKOS07]. Suppose that the following holds:

- the commitment scheme is perfectly ally hiding,
- Π_{f_x} is t-private (against semi-honest (against malicious adversaries) n-pa

If $t = \Theta(\kappa)$ and $n = \Theta(t)$, then this protocol is an honest-ve zero-knowledge proof for the NP-relation R with soundness error $2^{-\kappa}$.

Concrete parameters: for 2^{-80} soundness error, can use $(n, t, r) = (92{,}64{,}64)$

Relies only on OWFs (for the commitments) and black-box access to Π_{f_x}.

ZKBoo [GMO16]

$c_1, c_2, \dots, c_n$ →

← $S \subseteq [n]$ where $|S| = t_1$

$\{\mathrm{Open}(c_i, r_i')\}_{i \in [S]}$ →

For concrete soundness targets (e.g., 2^{-80}), most efficient instantiation of IKOS is to use *simple*, *non-robust* multiparty computation protocol and amplify soundness by repeating the protocol

ZKBoo [GMO16]

$c_1, c_2, \dots, c_n$

$S \subseteq [n]$ where $|S| = t_1$

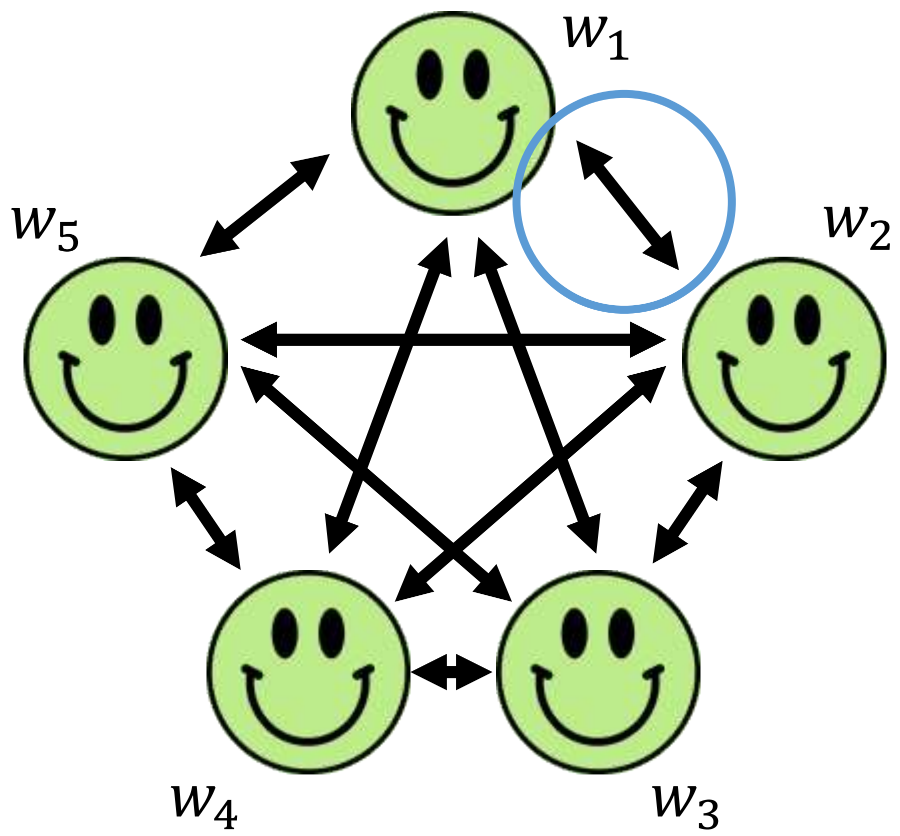
n-party BGW protocol obtaining soundness error 2^{-80} requires $n = 1122, t = 374$

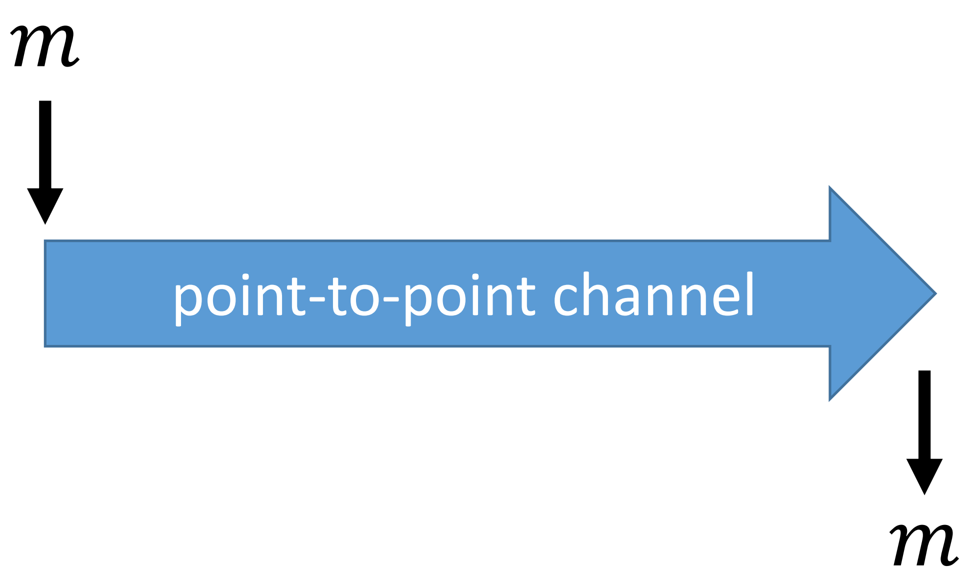
iterating a 3-party protocol with 2-privacy yields proofs which contain 274 bits per multiplication gate for 2^{-80} soundness

For concrete soundness targets (e.g., 2^{-80}), most efficient instantiation of IKOS is to use *simple*, *non-robust* multiparty computation protocol and amplify soundness by repeating the protocol

ZKBoo [GMO16]

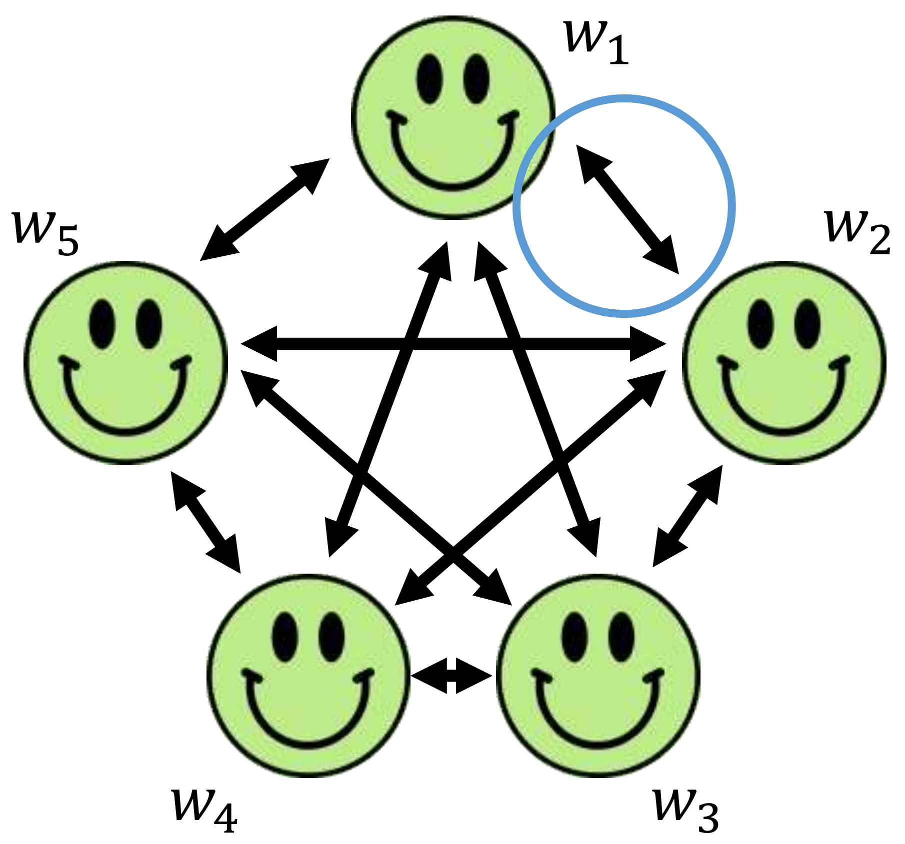
Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

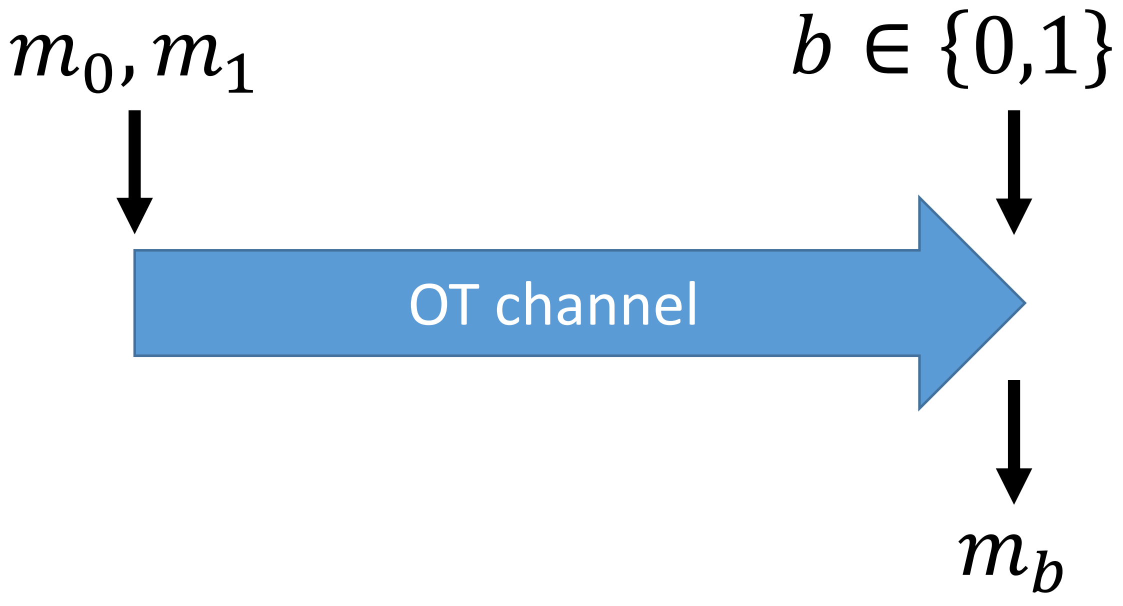




ZKBoo [GMO16]

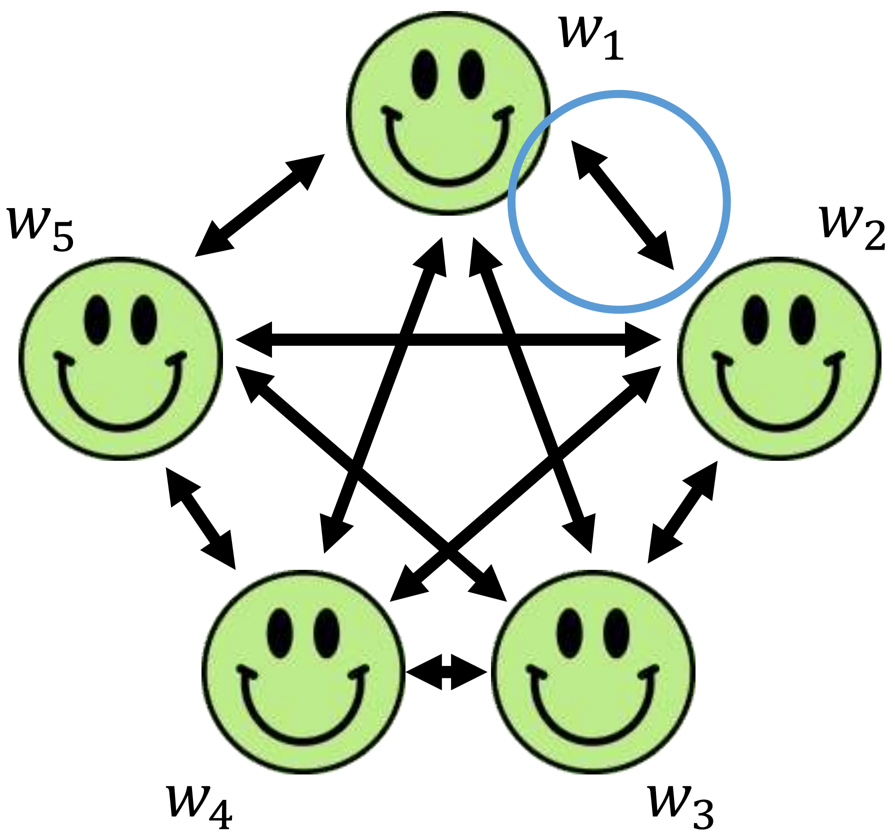
Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

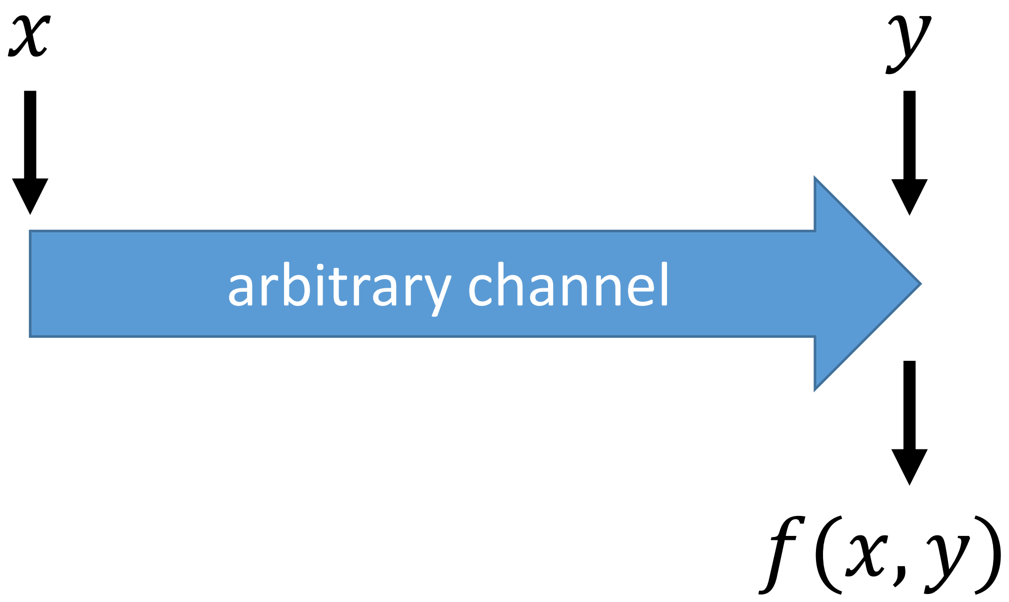




ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

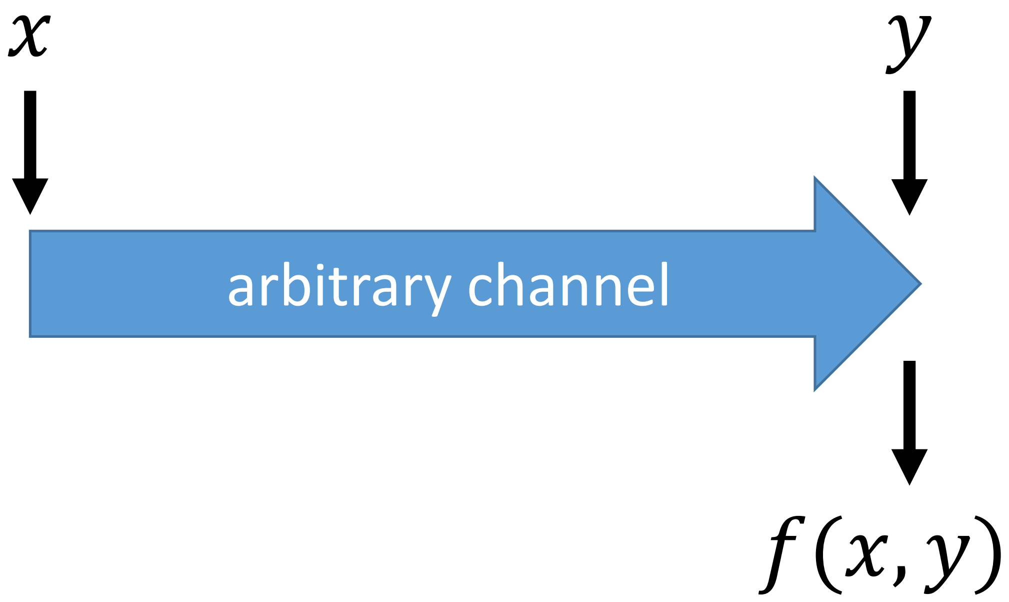




ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In MPC setting, channels are implemented using secure two-party computation
- In “MPC-in-the-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function f)

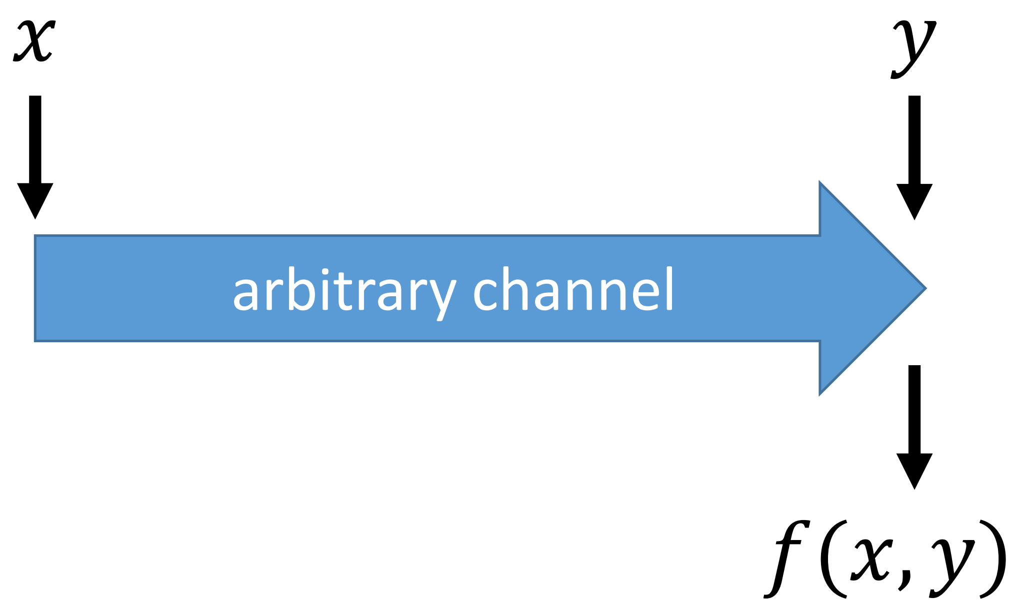


ZKBoo [GMO16]

Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In I... im... -party cor...
- In “MPC-... e-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function f)

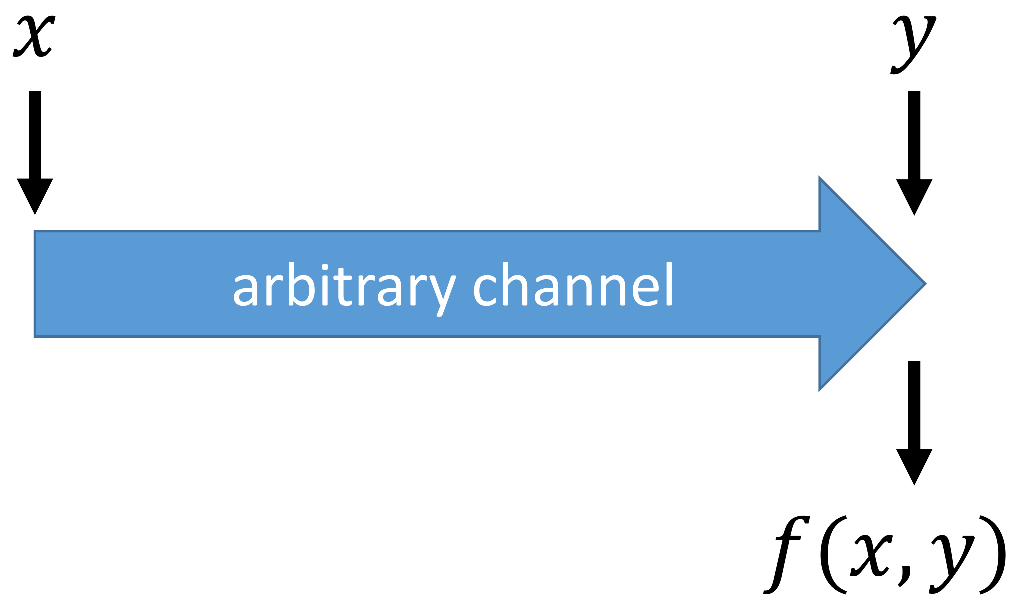
does not trivialize the problem since protocol must still provide privacy



ZKBoo [GMO16]

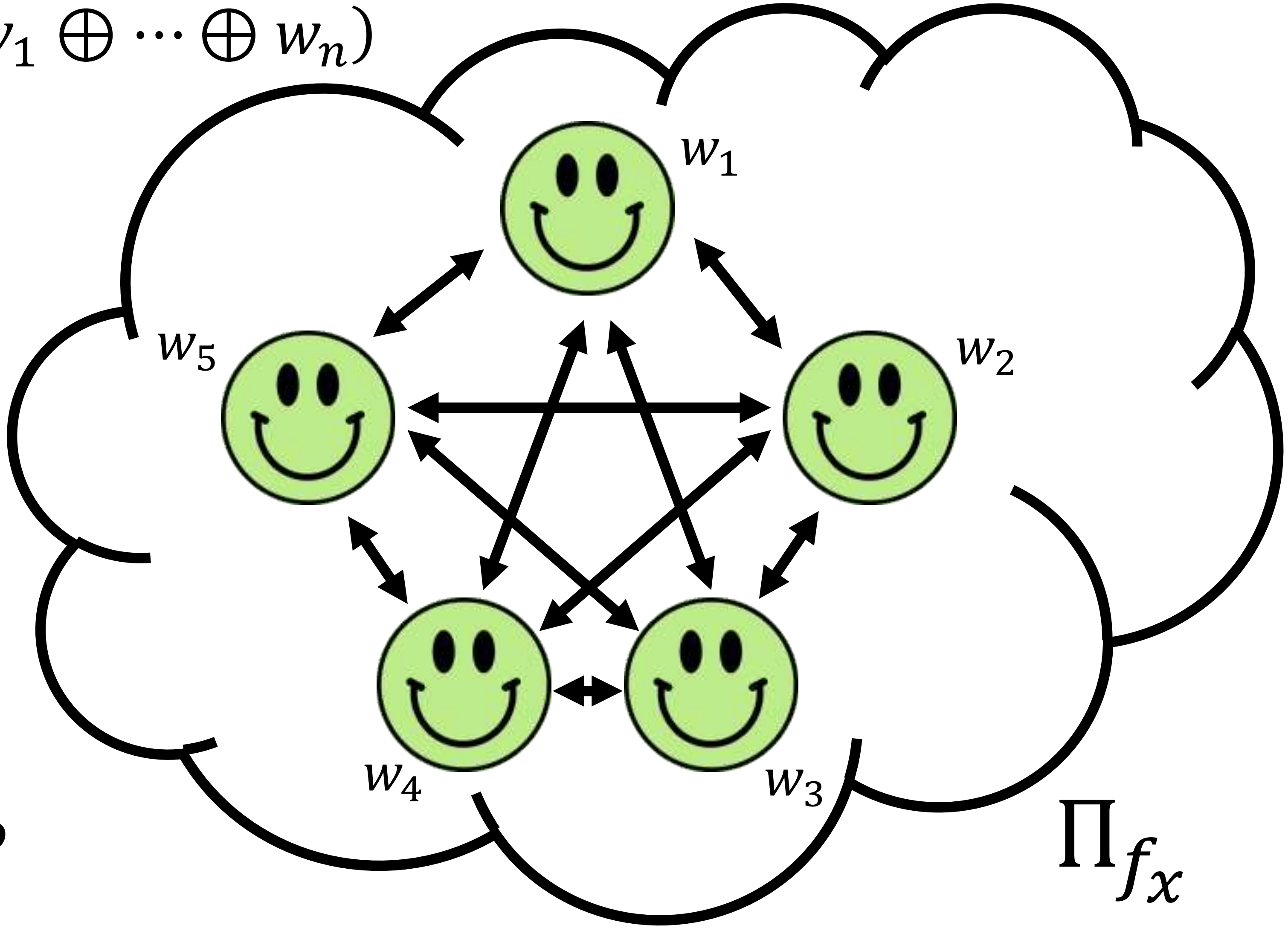
Emulating an MPC protocol *cheaper* than running the MPC protocol in the standard model

- In MPC setting, channels are implemented using secure two-party computation
- In “MPC-in-the-head,” can model them as *ideal* functionalities (e.g., as an oracle to the function f)
- New design space for MPC protocols



ZKBoo [GMO16]

$$f_x(w_1, \ldots, w_n) = R(x, w_1 \oplus \cdots \oplus w_n)$$



How to construct Π_{f_x}?

(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)


$w_1^{(0)}$


$w_2^{(0)}$

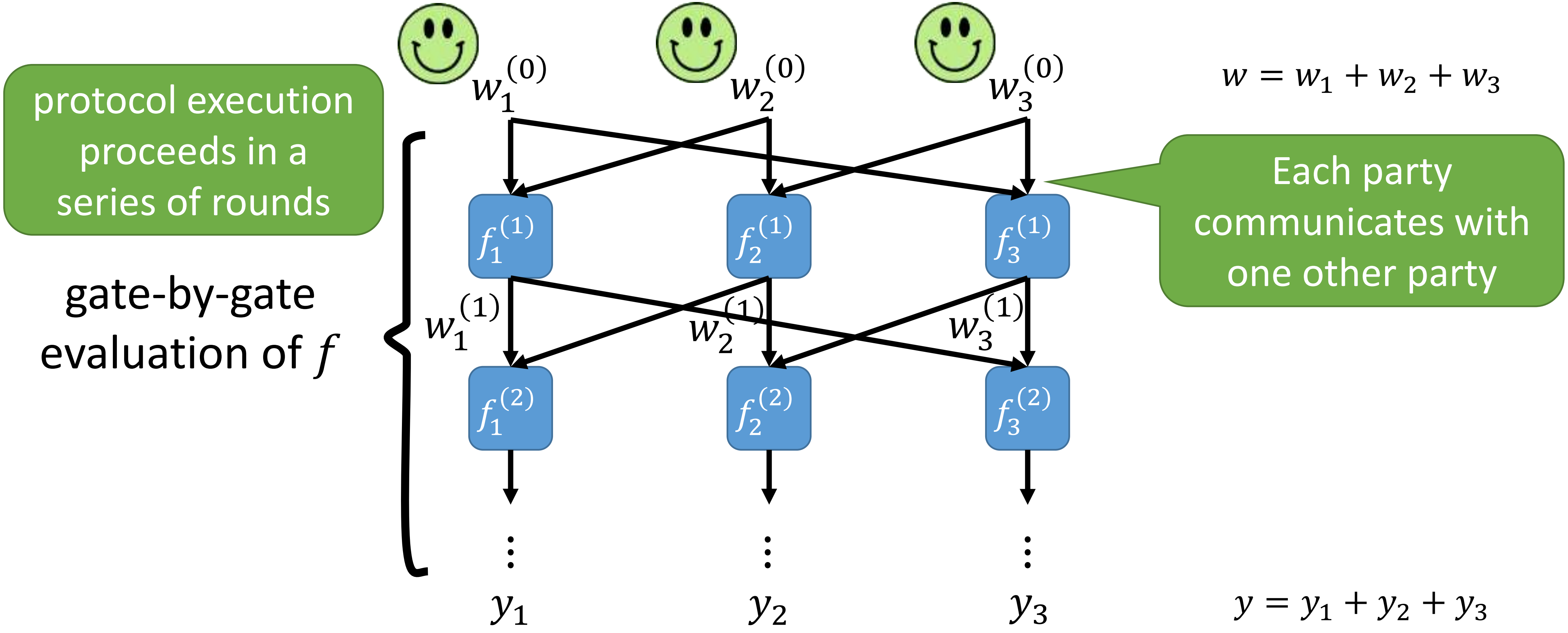

$w_3^{(0)}$

$w = w_1 + w_2 + w_3$

Function evaluation on secret shared-inputs

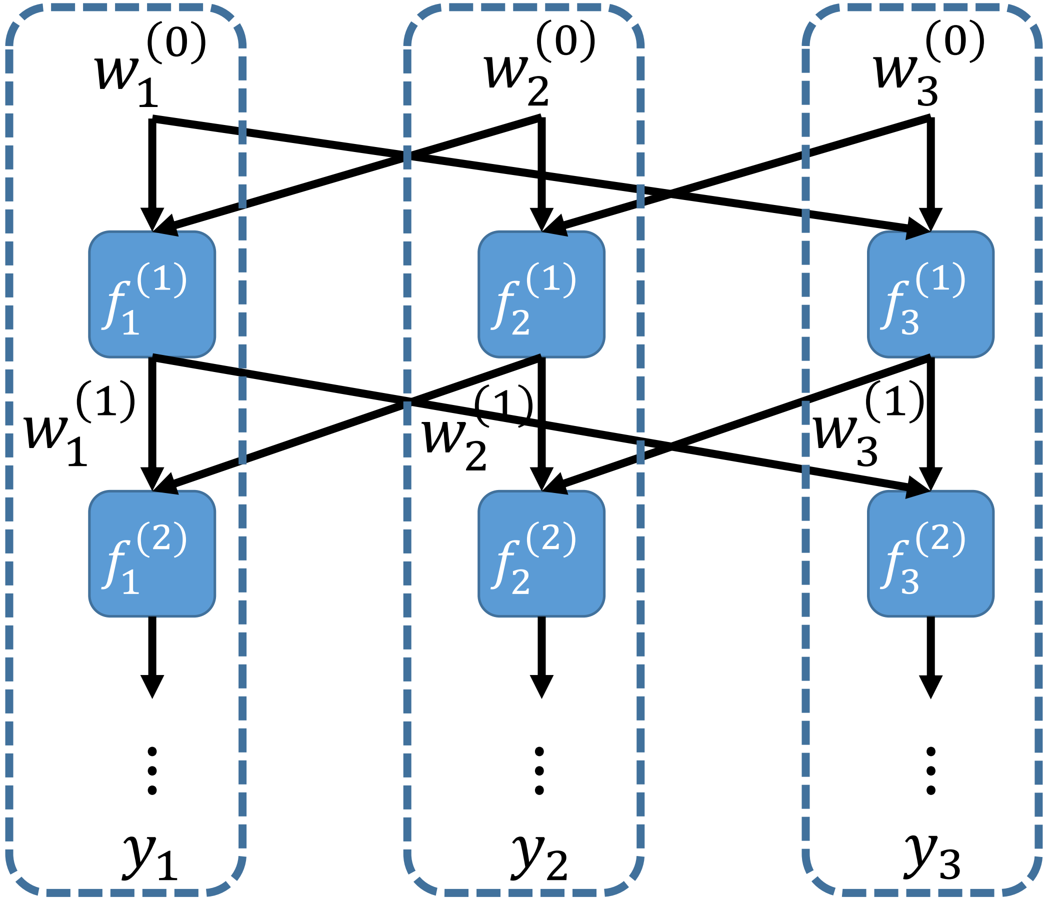
(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)



(2, 3)-Function Decompositions [GMO16]

A variant of the GMW protocol (can also be viewed as a function decomposition)

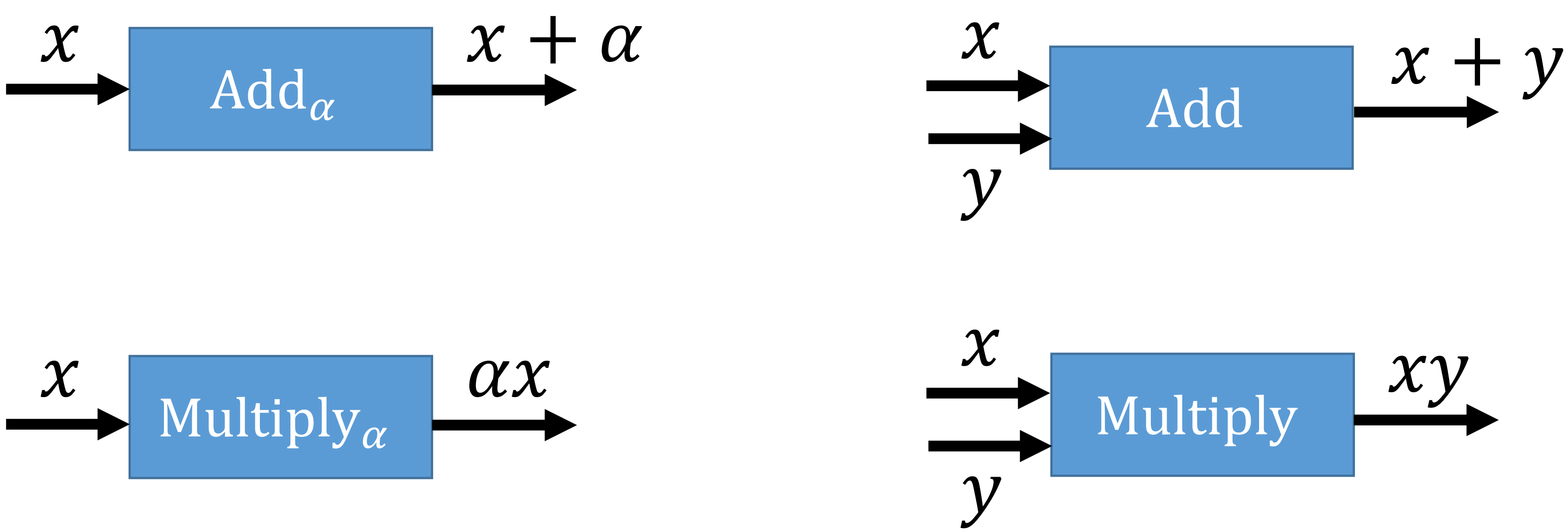


$$\mathrm{View}_{P_i}(\boldsymbol{w}\,;\boldsymbol{r}) = \left\{w_i^{(0)}, \dots, w_i^{(N)}\right\}$$

protocol is 2-private

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

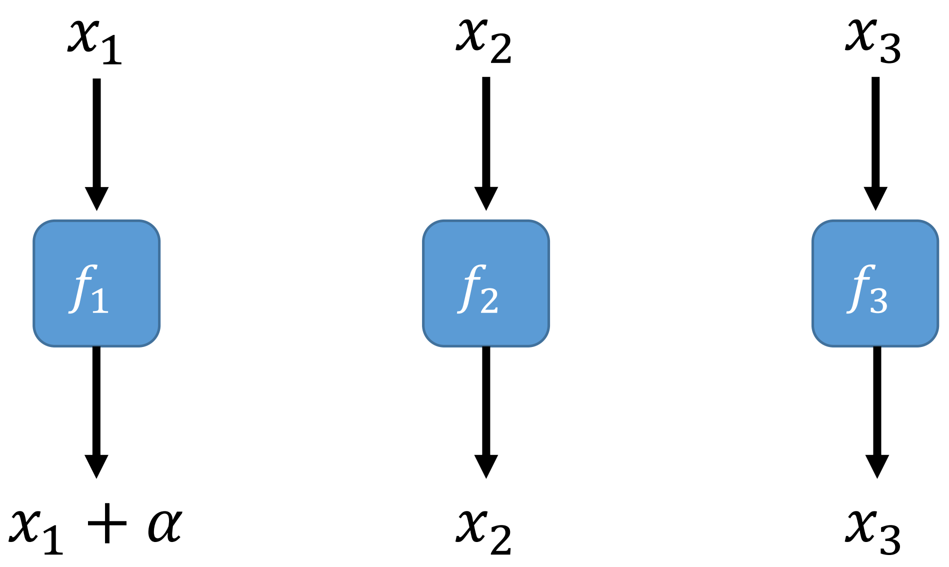


(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

no interaction needed

$$x = x_1 + x_2 + x_3$$



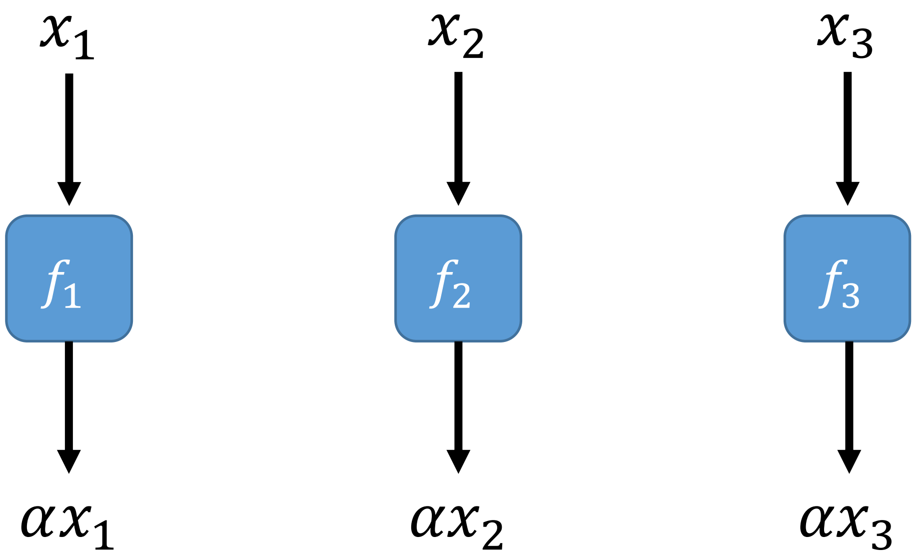
$$(x_1 + \alpha) + x_2 + x_3 = x + \alpha$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

no interaction needed

$x = x_1 + x_2 + x_3$



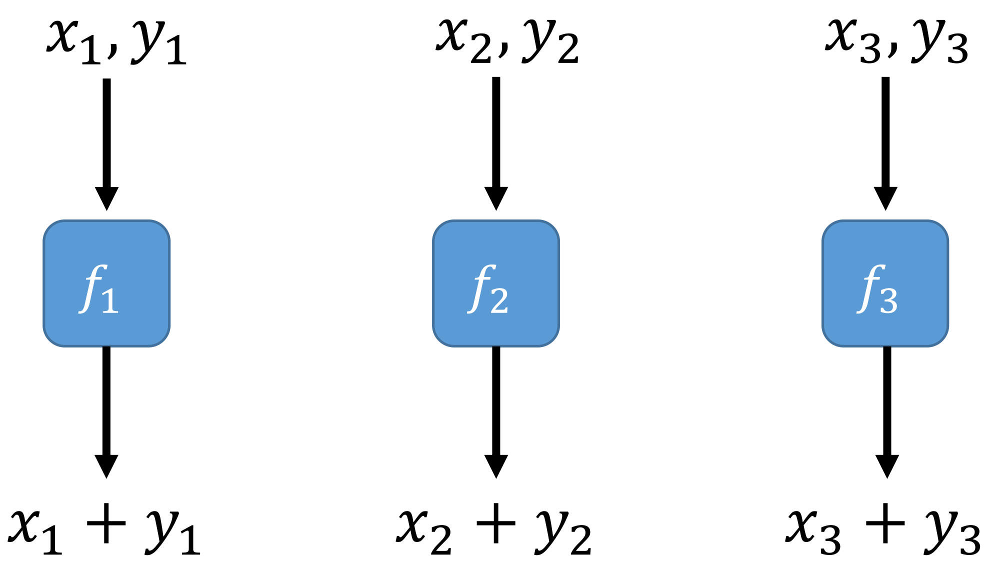
$\alpha x_1 + \alpha x_2 + \alpha x_3 = \alpha x$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

no interaction needed

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$



$$(x_1 + y_1) + (x_2 + y_2) + (x_3 + y_3) = x + y$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$

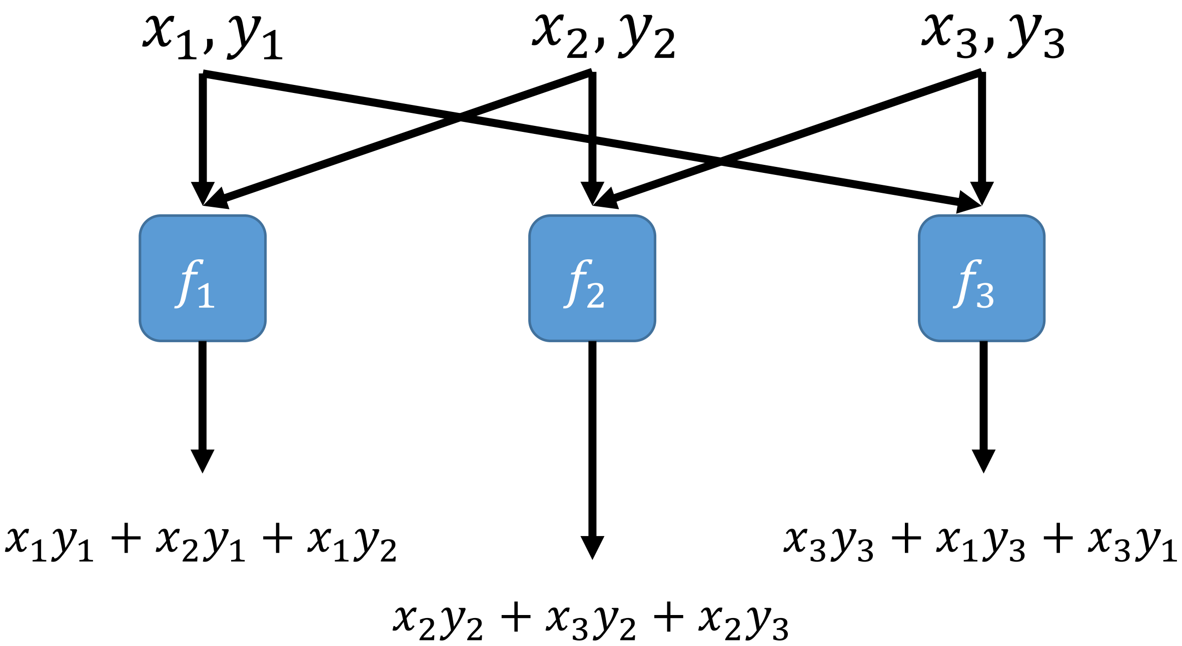
x_1, y_1 $\quad$ x_2, y_2 $\quad$ x_3, y_3

$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1, y_1, x_2, y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2, y_2, x_3, y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1, y_1, x_3, y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$



$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1,y_1,x_2,y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2,y_2,x_3,y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1,y_1,x_3,y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$

x_1, y_1 x_2, y_2 x_3, y_3

cannot be 2-private: information about x_3, y_3 revealed

$x_1y_1 + x_2y_1 + x_1y_2$ $x_3y_3 + x_1y_3 + x_3y_1$

$x_2y_2 + x_3y_2 + x_2y_3$

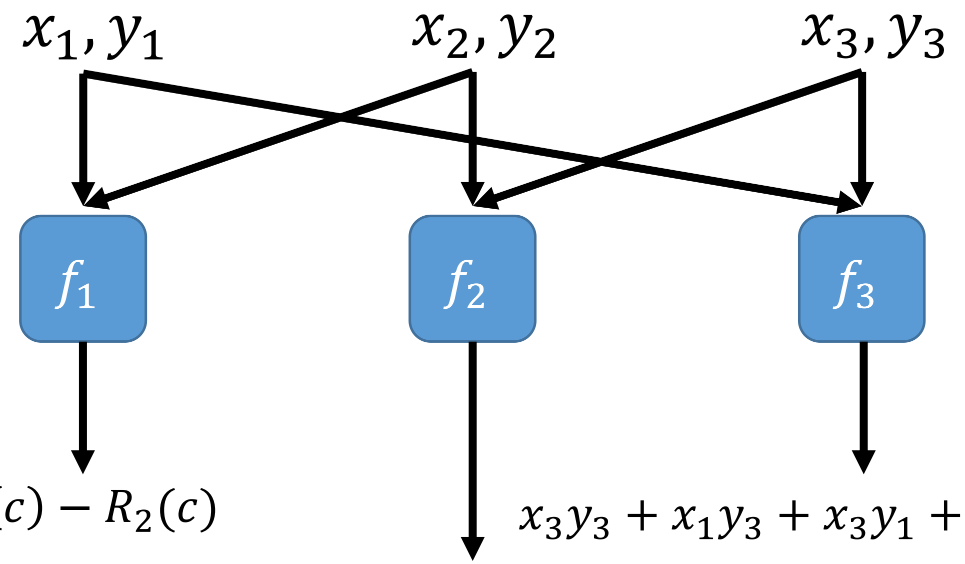
$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1, y_1, x_2, y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2, y_2, x_3, y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1, y_1, x_3, y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$



$$x_1y_1 + x_2y_1 + x_1y_2 + R_1(c) - R_2(c)$$

$$x_3y_3 + x_1y_3 + x_3y_1 + R_3(c) - R_1(c)$$

$$x_2y_2 + x_3y_2 + x_2y_3 + R_2(c) - R_3(c)$$

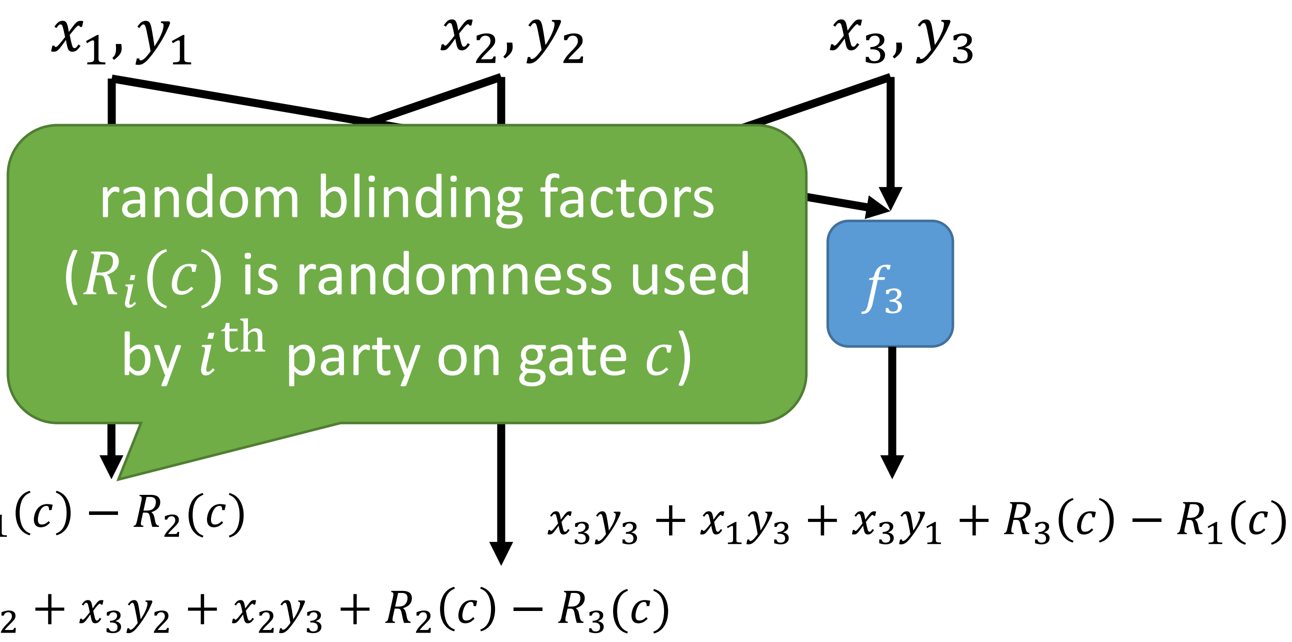
$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1, y_1, x_2, y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2, y_2, x_3, y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1, y_1, x_3, y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$



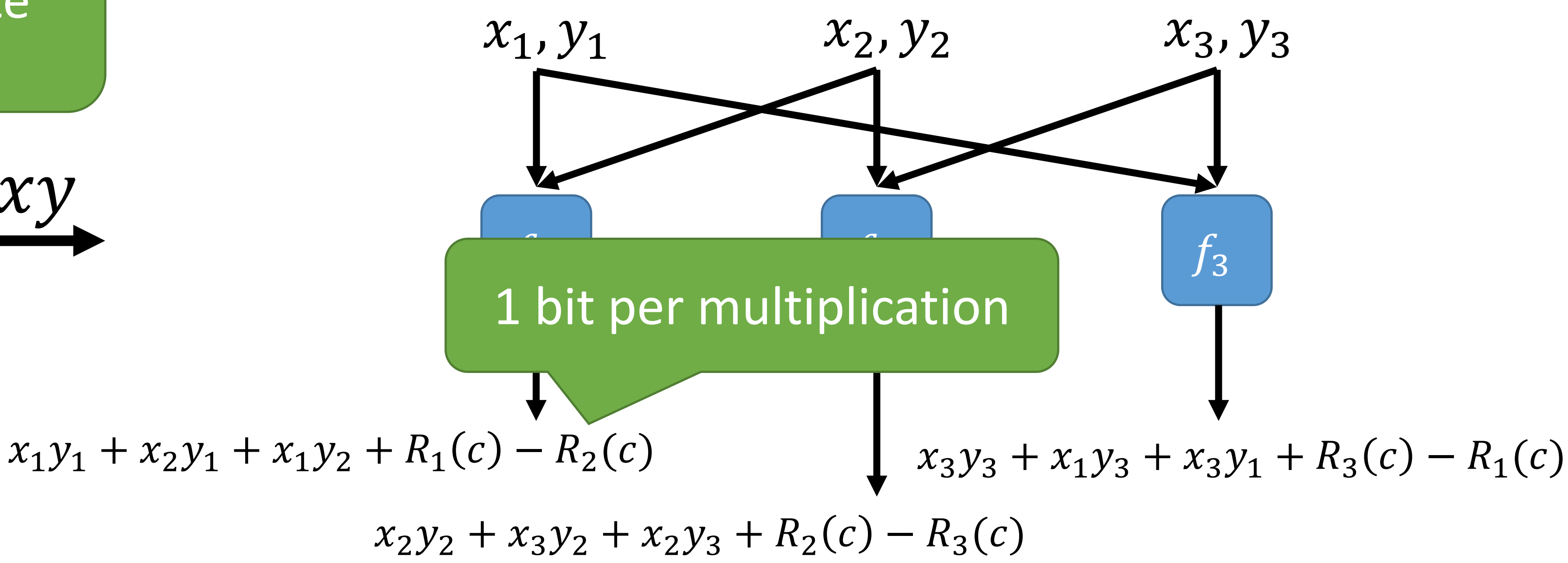
$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1, y_1, x_2, y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2, y_2, x_3, y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1, y_1, x_3, y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

similar to GMW, blind each intermediate product

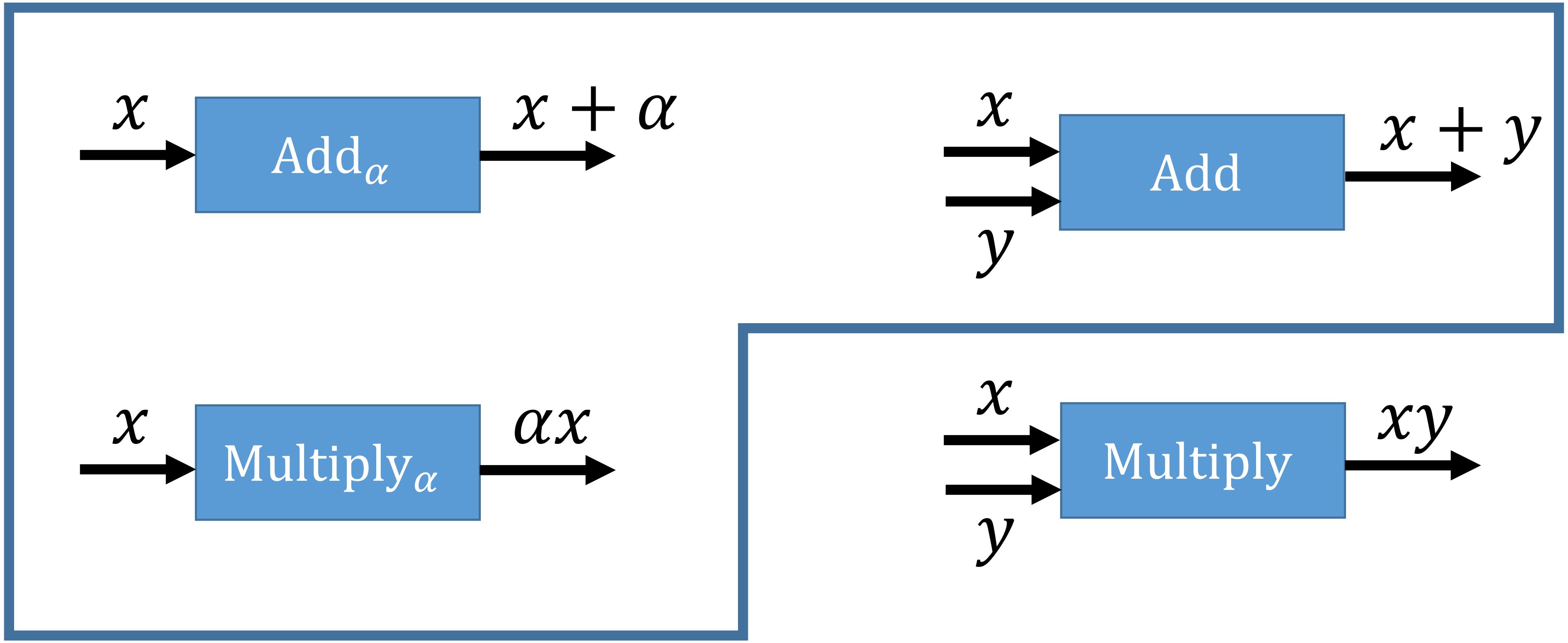
$$x = x_1 + x_2 + x_3$$
$$y = y_1 + y_2 + y_3$$



$$\overbrace{(x_1y_1 + x_2y_1 + x_1y_2)}^{\text{only depends on } x_1,y_1,x_2,y_2} + \overbrace{(x_2y_2 + x_3y_2 + x_2y_3)}^{\text{only depends on } x_2,y_2,x_3,y_3} + \overbrace{(x_3y_3 + x_1y_3 + x_3y_1)}^{\text{only depends on } x_1,y_1,x_3,y_3} = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = xy$$

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$

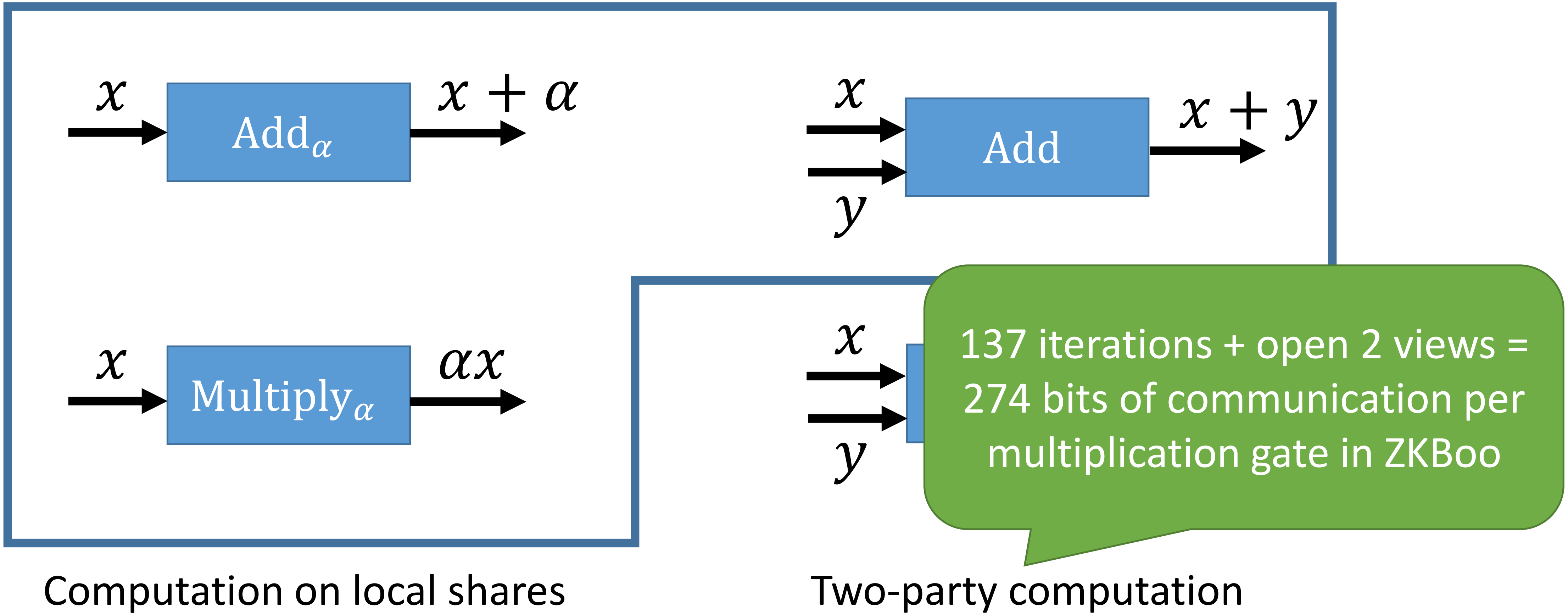


Computation on local shares

Two-party computation

(2, 3)-Function Decompositions [GMO16]

Express f_x as an arithmetic circuit over finite field $\mathbb{F}$



Summary

- "MPC in the head" gives new paradigm for constructing efficient zero-knowledge proof systems
- New directions in designing efficient MPC protocols *for zero-knowledge* can be quite efficient in practice
- Zero-knowledge protocols can also be used for signature schemes (Fiat-Shamir) – including post-quantum signatures!

Open Directions

- Designing new MPC protocols for more efficient zero-knowledge
  - Many theoretical MPC protocols with better communication complexity – shorter proofs and (post-quantum) signatures
- Alternative viewpoints: “MPC in the head” as a PCP with large alphabet (i.e., each party’s view)