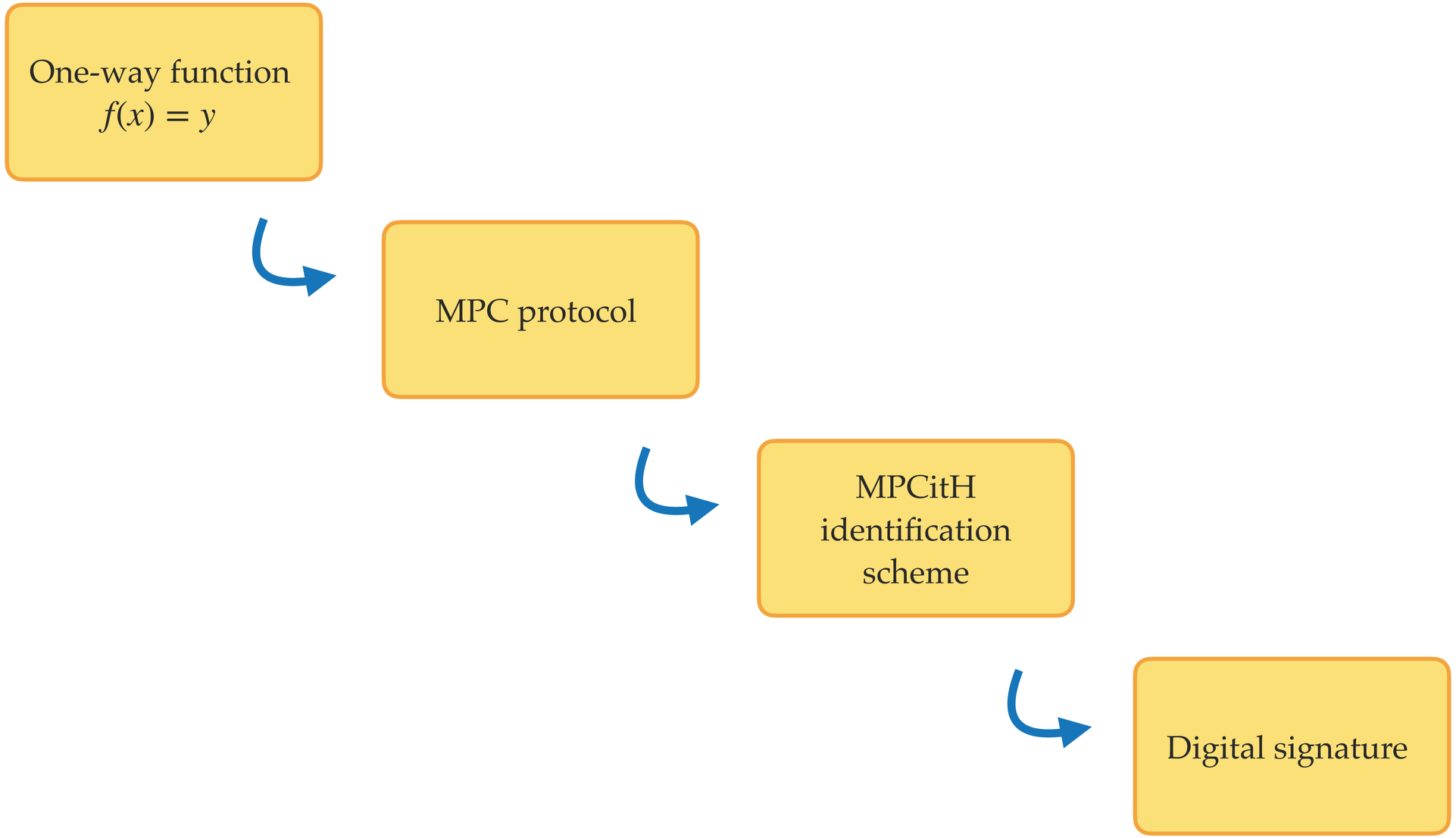
MPCitH construction

Monika Trimoska

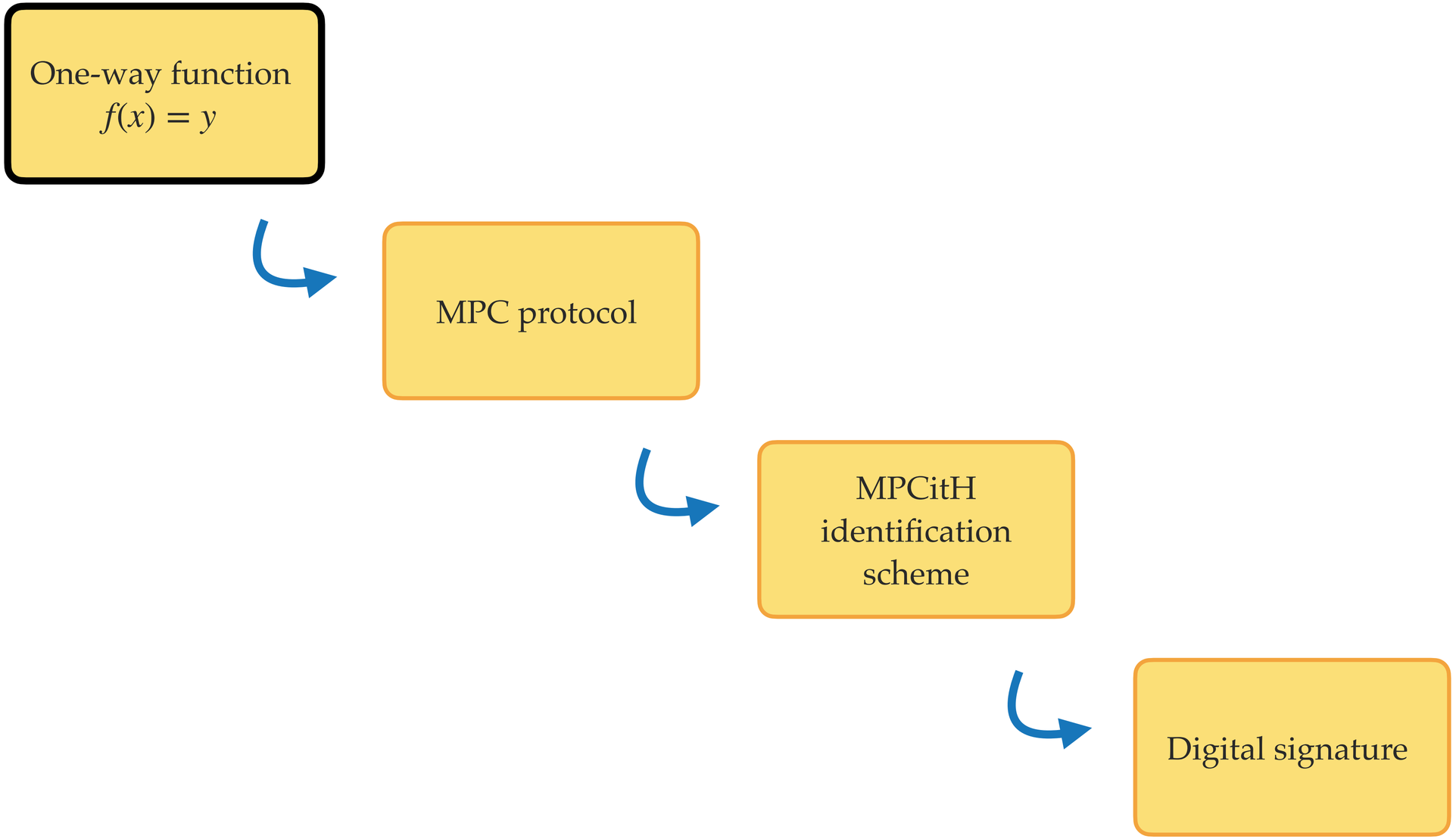
Selected Areas in Cryptology - Part 1
Spring, 2024

TU/e

The MPCitH construction



The MPCitH construction



The MPCitH construction

One-way function
$f(x) = y$

Zero-knowledge proof of knowledge: the prover needs to prove that they know a secret x s.t. $f(x) = y$, for a public y, without revealing any information about x (as in Lecture 3).

MPC protocol

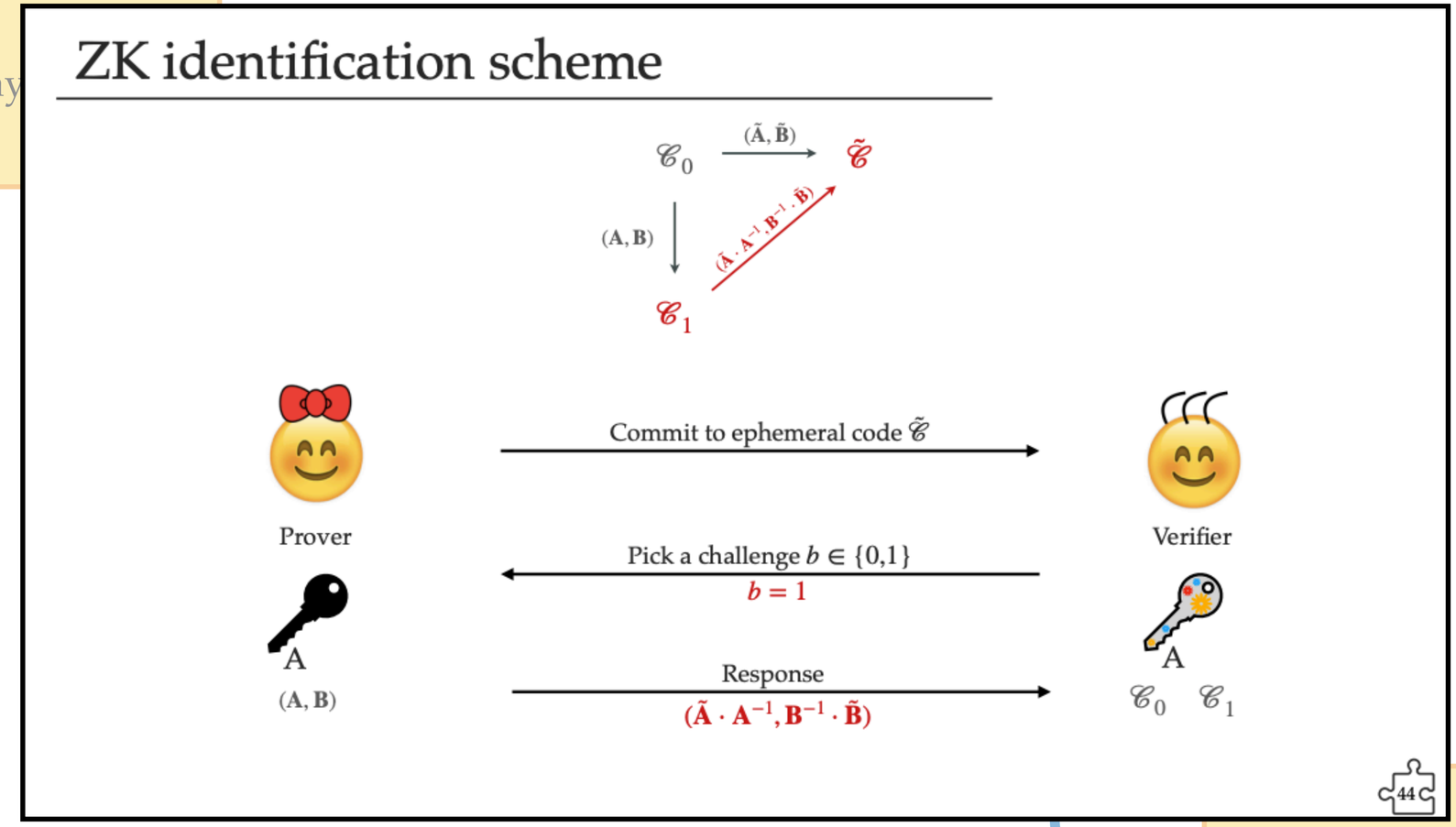
MPCitH identification scheme

Digital signature

The MPCitH construction

One-way

Zero-knowledge proof of knowledge: the prover



Digital signature

The syndrome decoding problem (recall)

The syndrome decoding problem

Given a syndrome $\mathbf{s} = \mathbf{H}\mathbf{x}$, find $\mathbf{x}$ such that $\mathrm{wt}(\mathbf{x}) = t$.

The syndrome decoding problem (recall)

The syndrome decoding problem

Given a syndrome $\mathbf{s} = \mathbf{H}\mathbf{x}$, find $\mathbf{x}$ such that $\mathrm{wt}(\mathbf{x}) = t$.

$$\underset{\mathbf{H}}{\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}} \quad \underset{\mathbf{x}}{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}} = \underset{\mathbf{s}}{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}$$

Find $\mathbf{x}$ of minimum weight.

The MPCitH construction

One-way function
$f(x) = y$

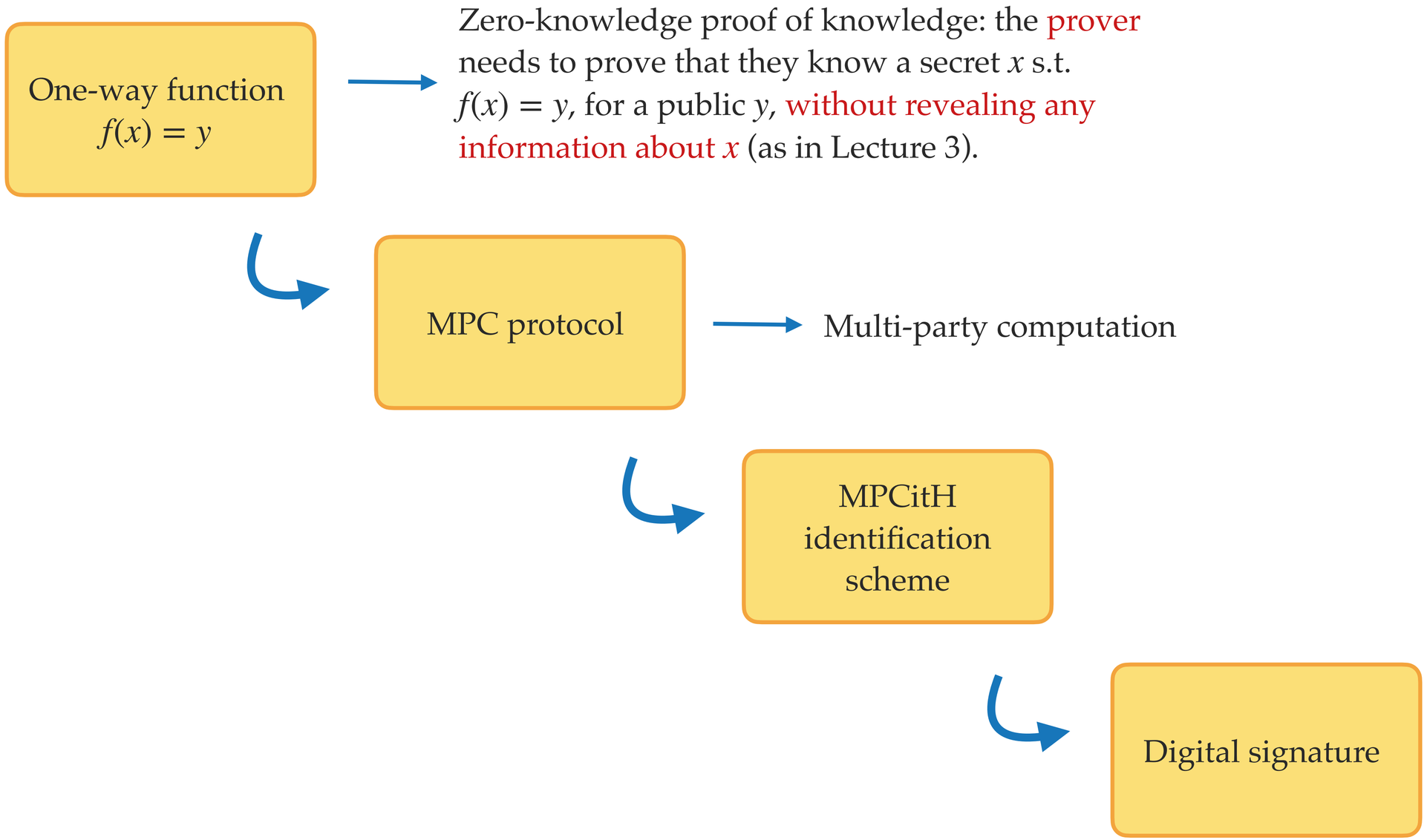
Zero-knowledge proof of knowledge: the prover needs to prove that they know a secret x s.t. $f(x) = y$, for a public y, without revealing any information about x (as in Lecture 3).

MPC protocol

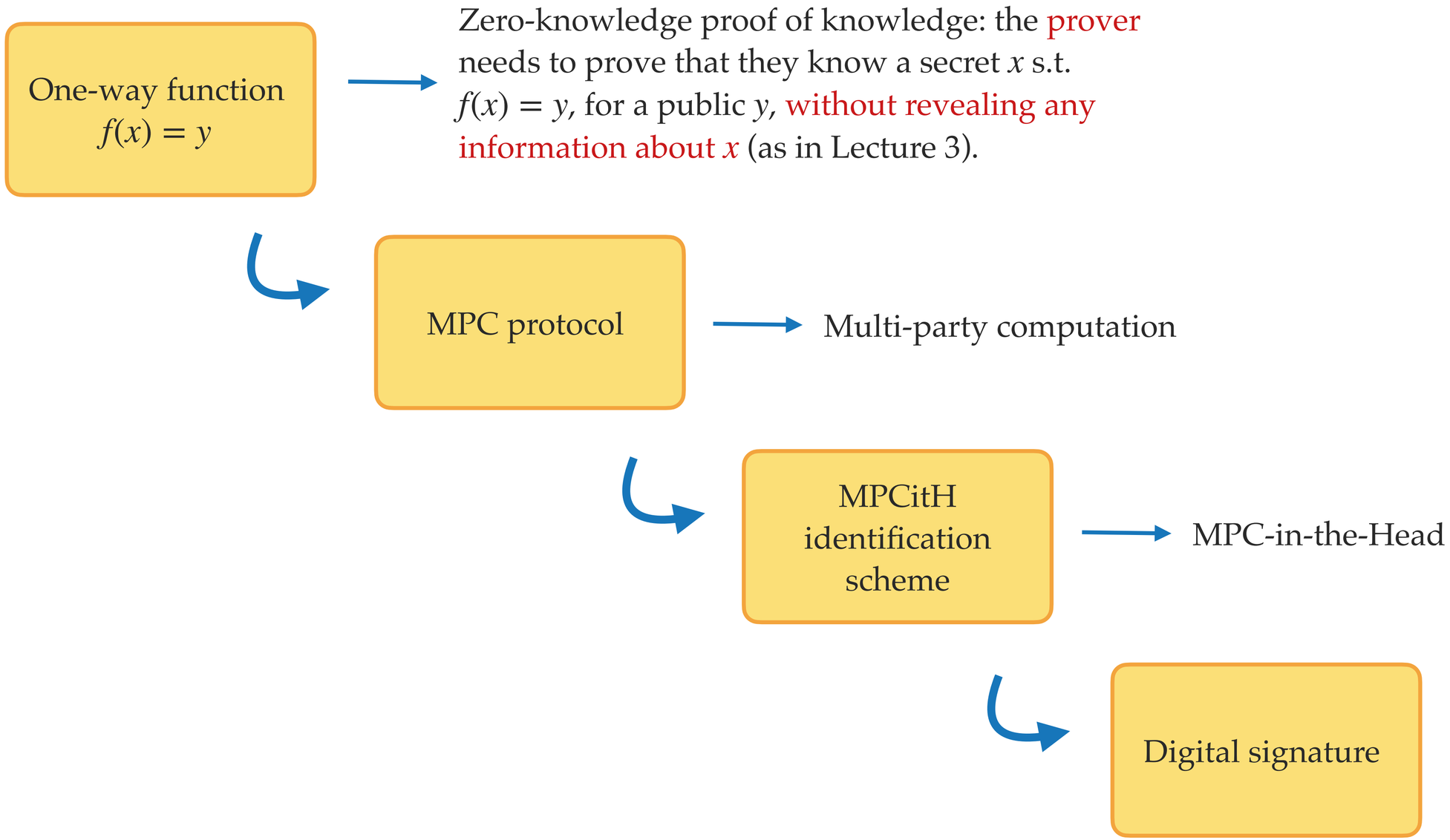
MPCitH identification scheme

Digital signature

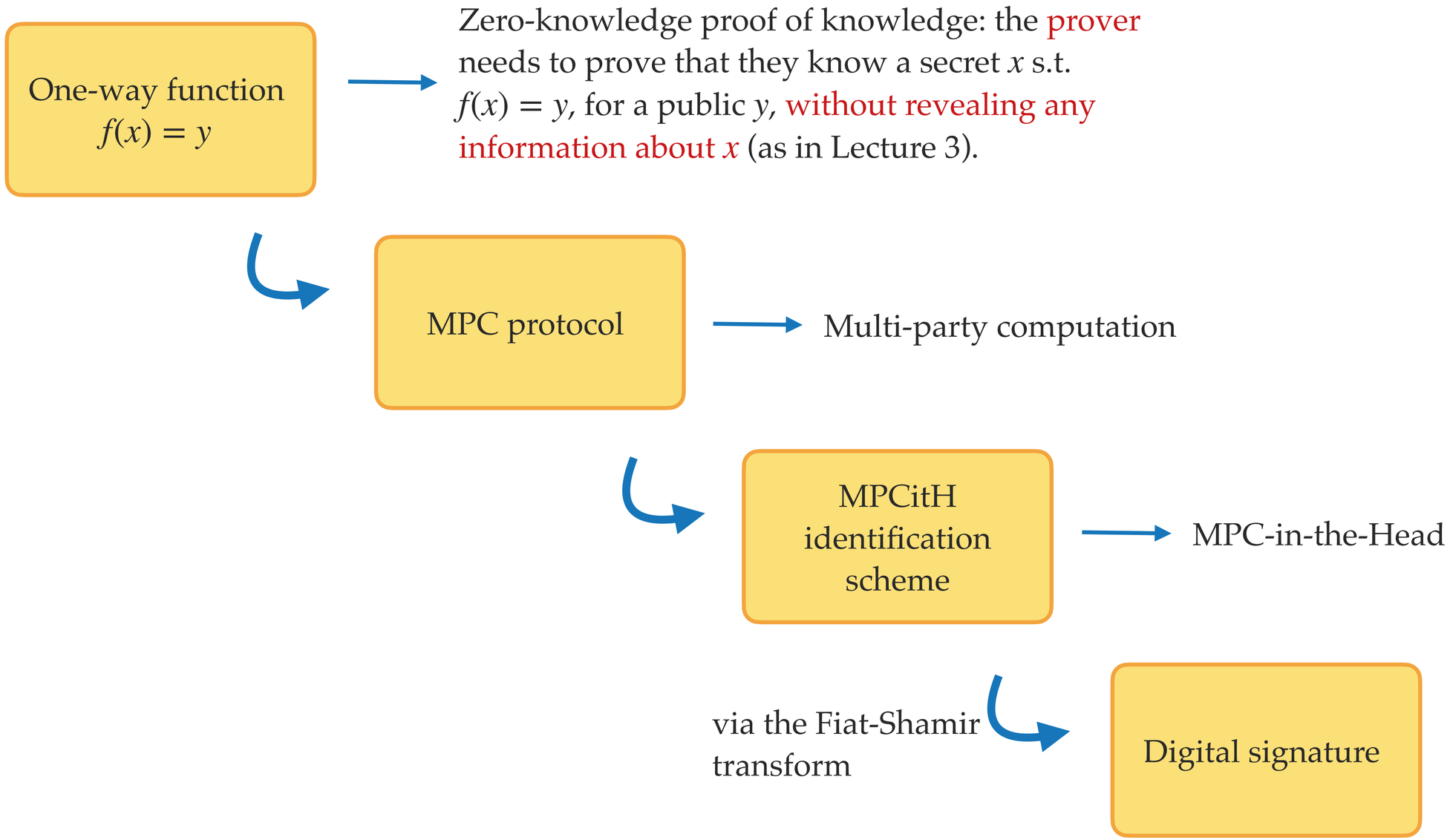
The MPCitH construction



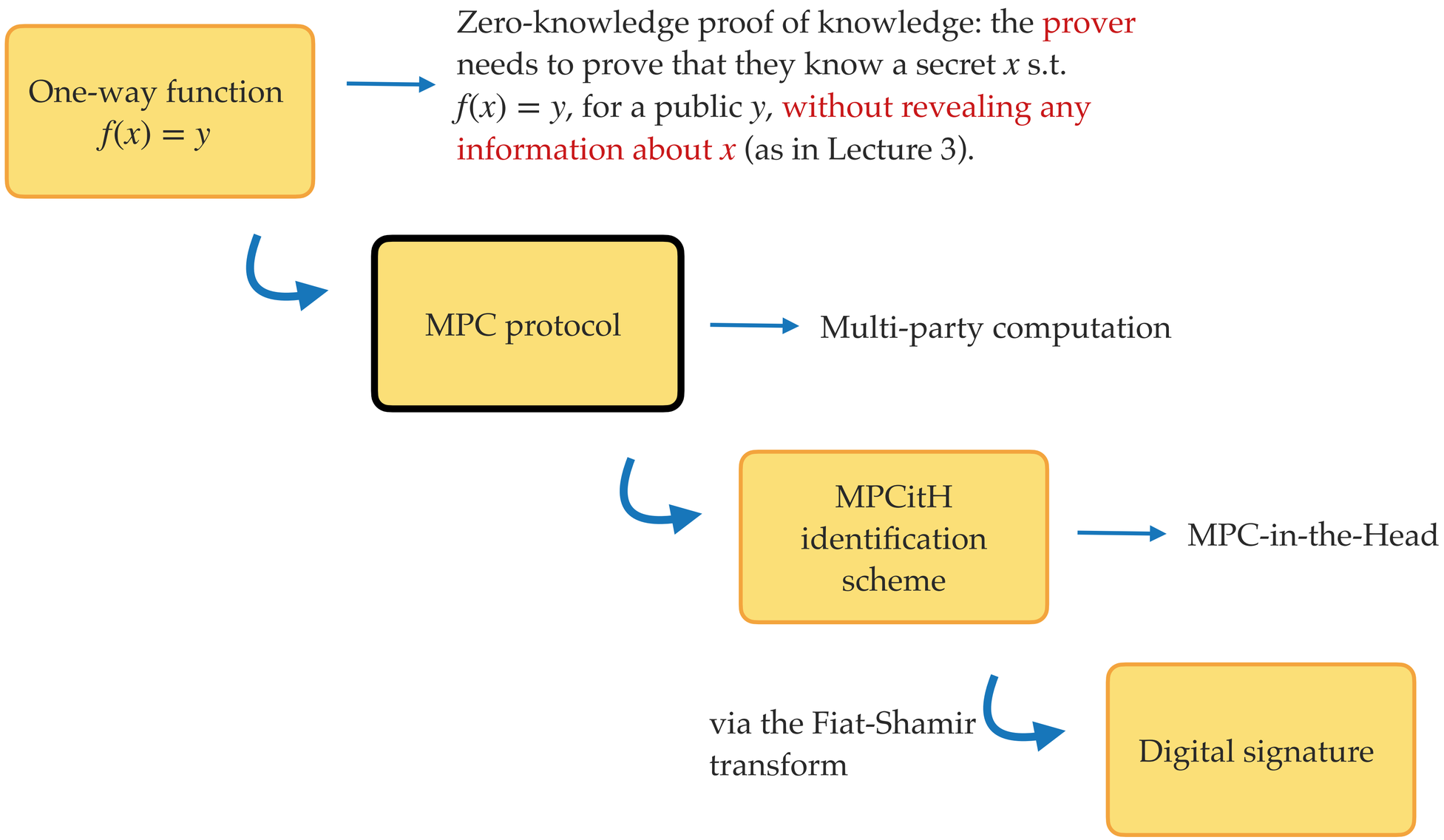
The MPCitH construction



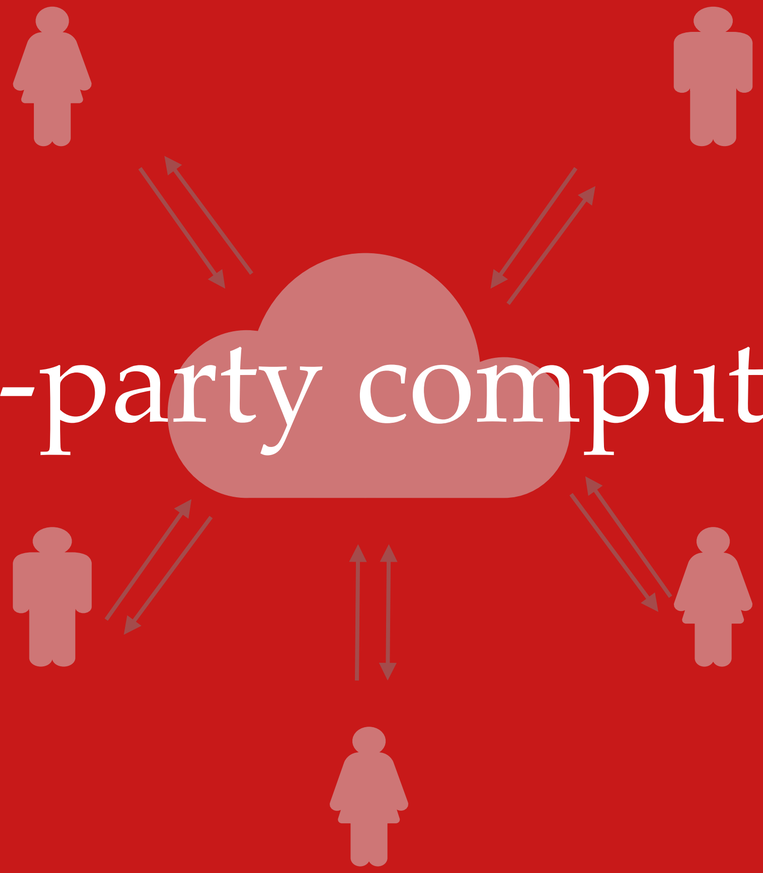
The MPCitH construction



The MPCitH construction



Multi-party computation

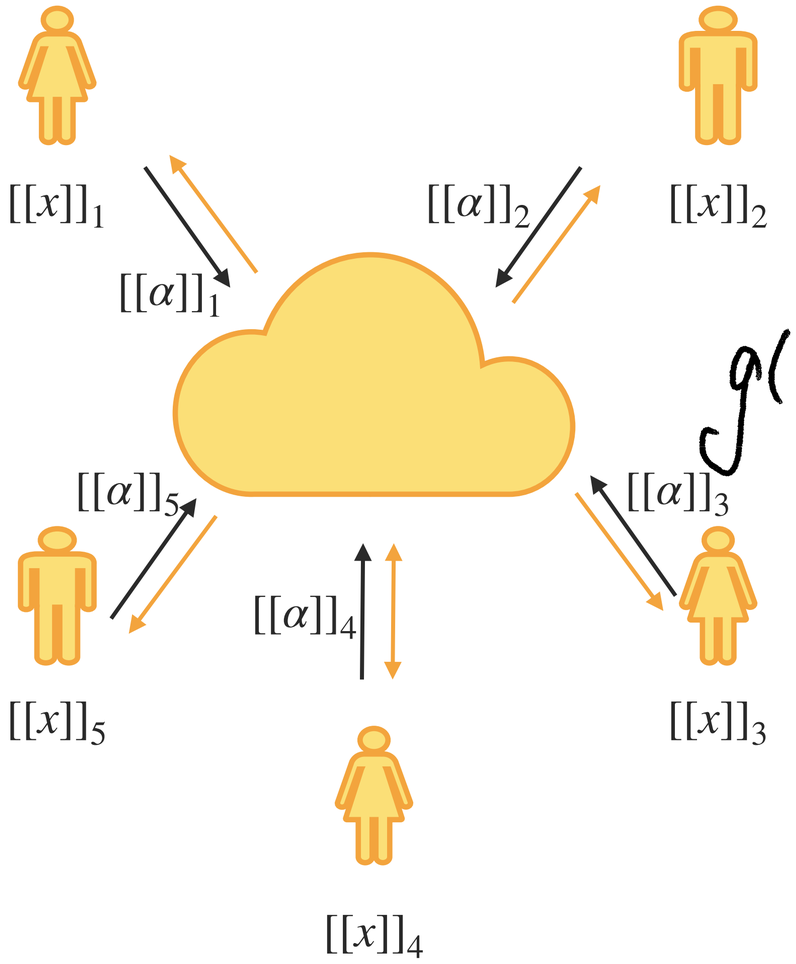


MPC: Multi-party computation

$X = [[x]]_1 + [[x]]_2 + \cdots + [[x]]_N$

$f(x) = y$

Jointly compute



$$g(x) = \begin{cases} \text{Accept if } f(x) = y \\ \text{Reject if } f(x) \neq y \end{cases}$$

Toy example: discrete log

$h = g^x$

Jointly check whether $g^x = h$.

$$x = [[x]]_1 + [[x]]_2 + \dots + [[x]]_N$$

$$[[h]]_i = g^{[[x]]_i}$$

$$\prod_{i=1}^{N} [[h]]_i = h$$

Real example: the SD problem

The syndrome decoding problem

Given a syndrome $\mathbf{s} = \mathbf{Hx}$, find $\mathbf{x}$ such that $\mathrm{wt}(\mathbf{x}) = t$.

Not a ‘linear problem’ (otherwise, it would be easy).

Real example: the SD problem

The syndrome decoding problem

Given a syndrome $\mathbf{s} = \mathbf{Hx}$, find $\mathbf{x}$ such that $\mathrm{wt}(\mathbf{x}) = t$.

Not a 'linear problem' (otherwise, it would be easy).

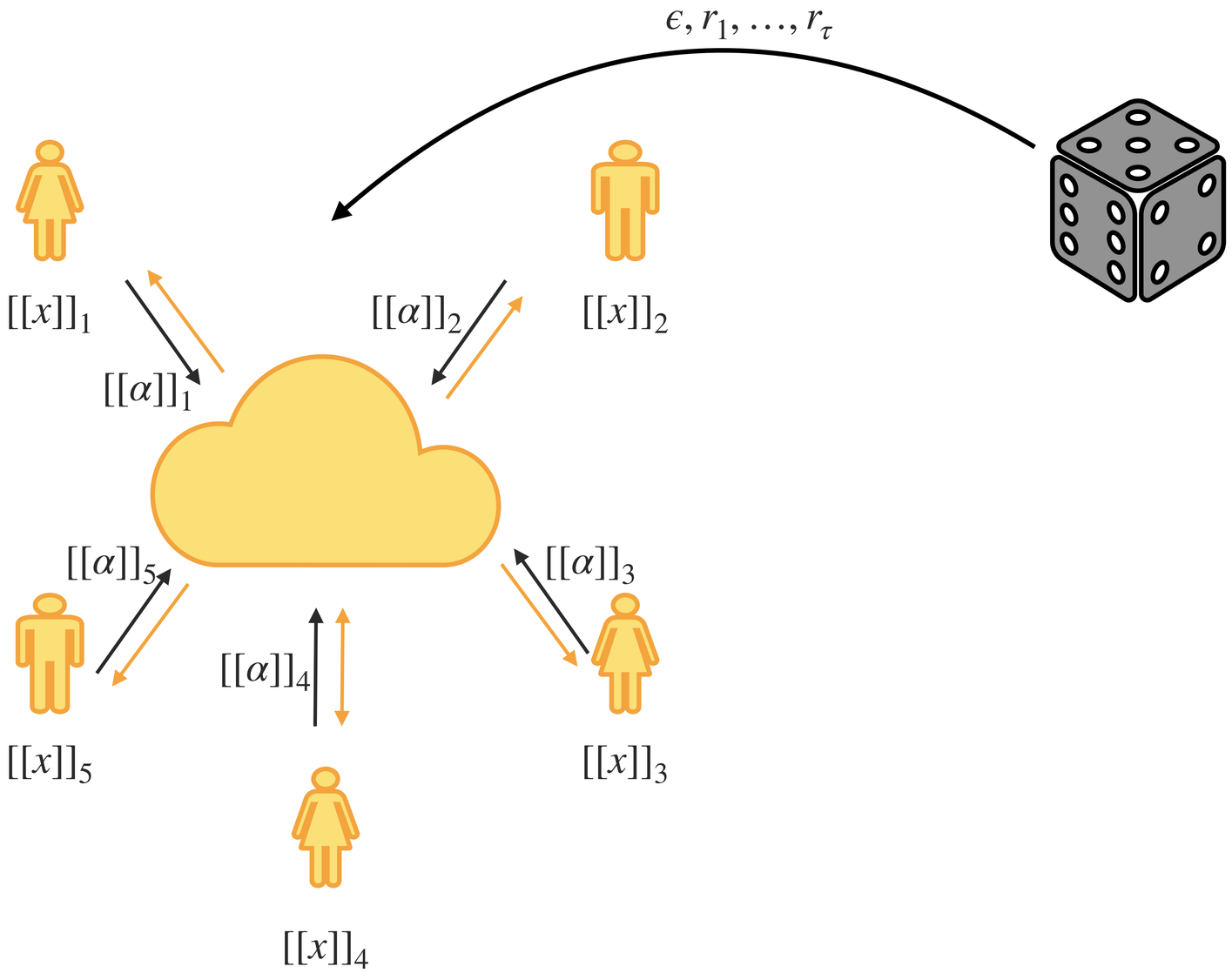
because of the weight constraint

1) Finding x s.t. $Hx=s$ is easy

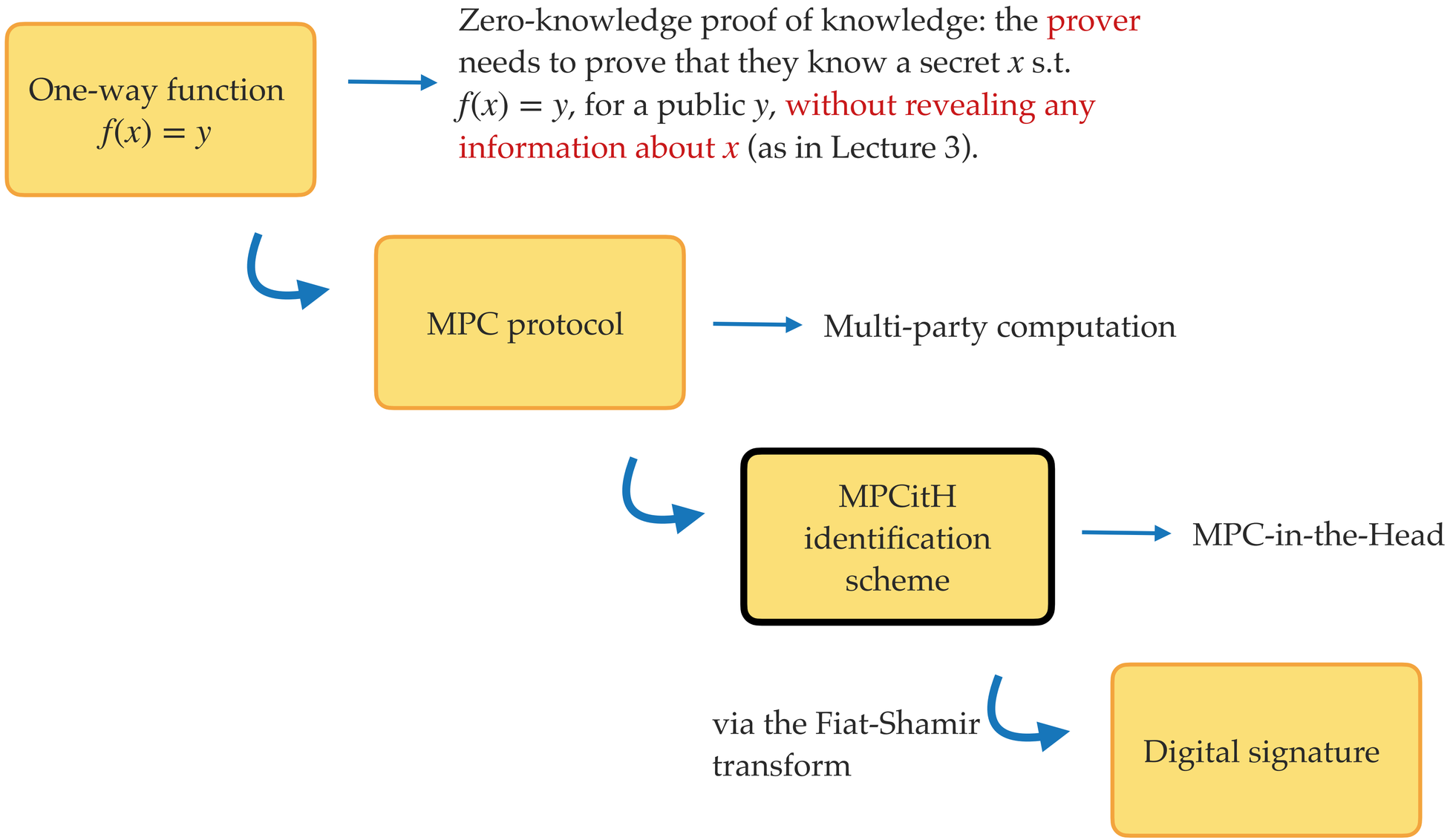
2) Finding x s.t. $\mathrm{hw}(x)=t$ is easy.

Finding x that satisfies both constraints is hard

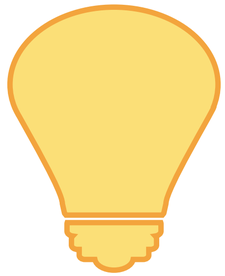
Broadcast model with oracle



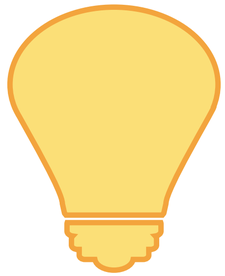
The MPCitH construction



MPCitH

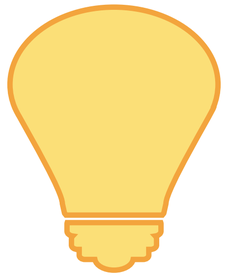


MPCitH



→ Properties that the underlying MPC model needs to have:

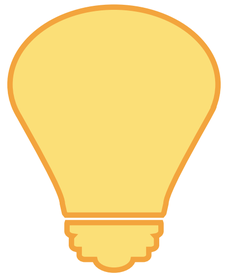
MPCitH



→ Properties that the underlying MPC model needs to have:

- $(N-1)$-private: the views of any $N-1$ parties do not reveal any information on x.
- Semi-honest: weak notion in MPC, but enough for the MPCitH application.

MPCitH



→ Properties that the underlying MPC model needs to have:

- $(N-1)$-private: the views of any $N-1$ parties do not reveal any information on x.
- Semi-honest: weak notion in MPC, but enough for the MPCitH application.

→ First instantiation: the PICNIC family (from symmetric primitives).

MPCitH identification scheme

$h = g^x$

Prover

Verifier

$Com([[h]]_1, \ldots, Com([[h]]_N)$

$[[h]]_1 \ldots [[h]]_N$

$\downarrow$

$g^{[[x]]_1}$

Pick $c \in [1; N]$

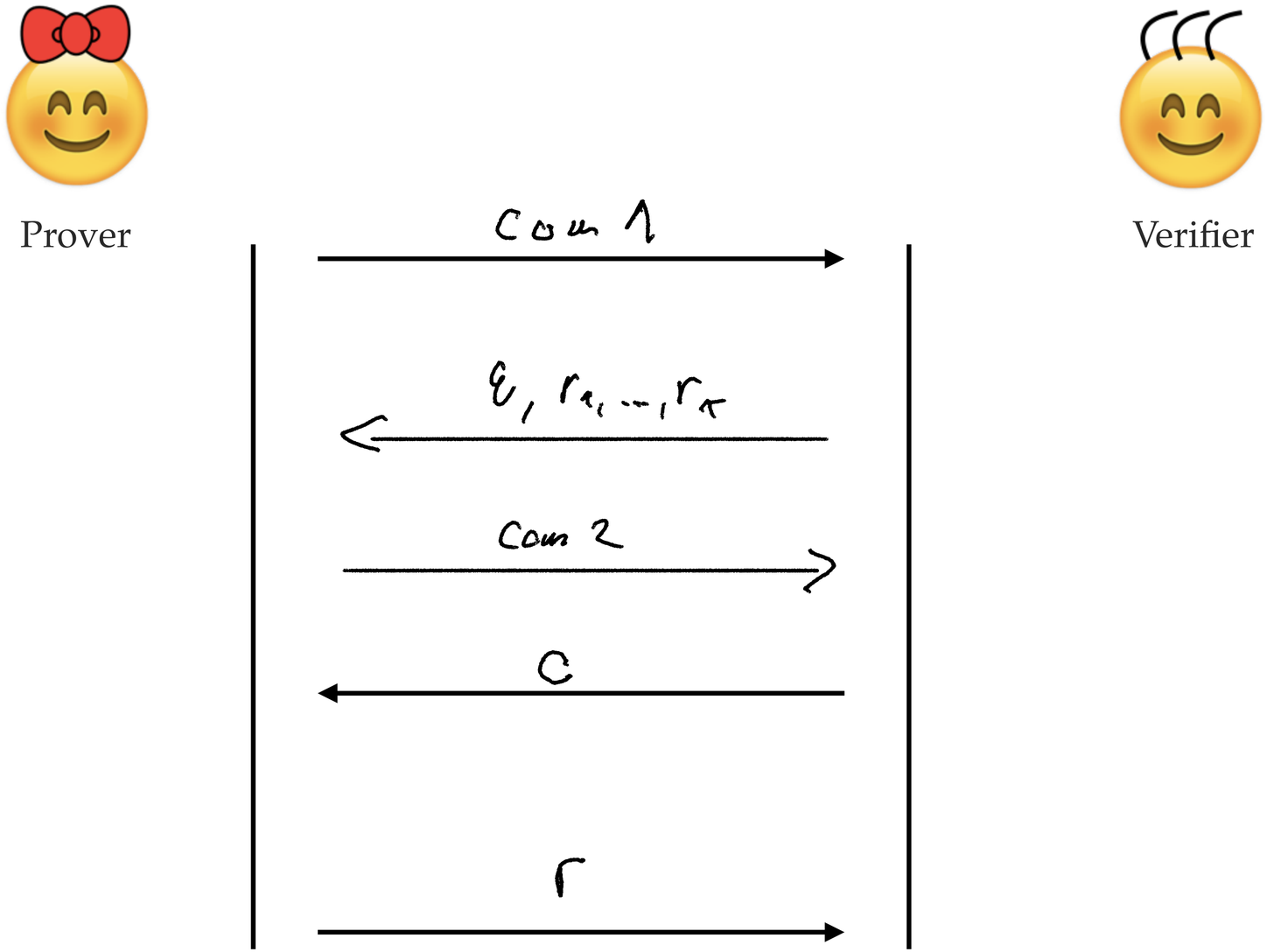
c

Check $g^{[[x]]_i} = [[h]]_i$ for $i \in I$

$[[x]]_i \quad i \in I \quad i \neq c$

Check $\prod_{i=1}^{N} [[h]]_i = h$

MPCitH identification scheme



Opening the black boxes

Checking the weight constraint

Let $t_1, \dots, t_q$ the elements of $\mathbb{F}_q$

Prove that we know $\mathbf{x}$ s.t. $hw(\mathbf{x}) = t$.

$S \cdot Q = F \cdot P$

F → public

$S(t_i) = x_i$

$Q(t_i) = 0$ for every $x_i \neq 0$

Prove that you know Q, P s.t. $S \cdot Q = F \cdot P$ evaluates to 0 on all t_i $i \in [1; n]$

Checking the weight constraint

$$g_i = \prod_{\substack{1 \le j \le n \\ j \neq i}} (f_i - f_j)^{-1}$$

$$P(x) = \frac{Q(x)\, S(x)}{F(x)}$$

$$F(x) = \prod_{i=1}^{n} (x - f_i)$$

↳ 'the vanishing polynomial' of the set $\{f_1, \dots, f_n\}$

$$S(x) = \sum_{i=1}^{n} g_i x_i \frac{F(x)}{x - f_i}$$

→ obtained through Lagrange interpolation of the coordinates of x s.t. $S(f_i) = x_i$

$$Q(x) = \prod_{i \in E} (x - f_i)$$

→ the subset of indices of the nonzero coordinates of x. $|E| = t \cdot \deg(Q) = t$

Toy example

$q=7,\ n=6,\ x=(1,0,0,2,0,0),$ so $t=2$

Using Lagrange interpolation to obtain S.

$$g_1=(1-2)^{-1}(1-3)^{-1}(1-4)^{-1}(1-5)^{-1}(1-6)^{-1}=6\cdot3\cdot2\cdot5\cdot4=6$$

killing this when $s(1)$

$$g_4=(4-1)^{-1}(4-2)^{-1}(4-3)^{-1}(4-5)^{-1}(4-6)^{-1}=3$$

$$S(X)=g_1\cdot x_1\frac{F(X)}{X-1}+g_4\cdot x_4\frac{F(X)}{X-4}=$$

$$=\frac{6\,(X-1)(X-2)(X-3)(X-4)(X-5)(X-6)}{(X-1)}+$$

$(X-1)\to 1$ is not a root anymore

$$+\frac{3\cdot2\,(X-1)(X-2)(X-3)(X-4)(X-5)(X-6)}{(X-4)}=$$

$(X-4)\to 4$ is not a root

Toy example

$q=7$, $n=6$, $x=(1,0,0,2,0,0)$, so $t=2$

Check that $S(f_i)=x_i$:

$$S(f_1)-S(1) = \underbrace{6}_{6} \cdot \underbrace{6\cdot 5\cdot 4\cdot 3\cdot 2}_{6} = 1$$

1 is root to the second polynomial in the sum, but not to the first

$$S(f_4)-S(4) = 3\cdot 2\cdot 3\cdot 2\cdot 1\cdot 6\cdot 5 =$$

$$= 3\cdot 2\cdot 5 = 2$$

Toy example

$q=7$, $n=6$, $x=(1,0,0,2,0,0)$, so $t=2$

$$Q(X)=(X-1)(X-4)=X^2-5X+4=$$

$$=X^2+2X+4$$

→ Q is indeed of degree t.

$$F(X)=(X-1)(X-2)(X-3)(X-4)(X-5)(X-6)=$$

$$=X^6+6$$

$$P(X)=\frac{Q(X)\,S(X)}{F(X)}=5X+5$$

↗ Sage

Toy example $q=7,\ n=6,\ x=(1,0,0,2,0,0)$, so $t=2$

Checking that $S \cdot Q = F \cdot P$ by evaluating $S \cdot Q - F \cdot P$ on random points, making sure we obtain 0 each time (because if we check once, we might get a 'false positive'). This is done in an MPC setting, by evaluating additive shares of the polynomial $S \cdot Q - F \cdot P$. Here (having all shares), we can clearly see that

$$S \cdot Q = 5x^7 + 5x^6 + 2x + 2 = F \cdot P,$$

so $S \cdot Q - F \cdot P$ is the zero polynomial. Hence, there exists a Q of degree at most t, that is a 'complement' of S, which is obtained by Lagrange interpolation of x. Hence, $hw(x) \leq t$.

Verifying multiplication: Beaver triples

Goal: having shared values $[[x]], [[y]], [[z]] \in \mathbb{F}_q$ verify that $z = x \cdot y$

Use a random auxiliary triple $a, b, c \in \mathbb{F}_q$, where $c = a \cdot b$.

Beaver's algorithm checks both multiplications, losing secrecy of a, b, c.

- Get random $\epsilon \in \mathbb{F}_q$
- Parties locally compute $[[\alpha]] = \epsilon[[x]] + [[a]]$ and $[[\beta]] = [[y]] + [[b]]$.
- Parties broadcast $[[\alpha]]$ and $[[\beta]]$ shares to open α and β.
- Parties locally compute $[[v]] = \epsilon[[z]] - [[c]] + \alpha[[b]] + \beta[[a]] - \alpha\beta$
- Parties broadcast $[[v]]$ shares to open v and accept if $v = 0$.

Security properties (recall)

Completeness

If the statement is true, an honest prover is always able to convince an honest verifier.

Soundness

A dishonest prover cannot convince an honest verifier other than with a small probability.

Soundness

A dishonest prover cannot convince an honest verifier other than with a small probability.

2-Special soundness

Having obtained two valid transcripts with the same commitment and a different challenge, we can extract a solution for the underlying problem.

Zero-knowledge

Anyone observing the transcript (including the verifier) learns nothing other than the fact that the statement is true.

Security properties

Completeness

Security properties

Completeness ✓

Security properties

→ Completeness ✓

→ Soundness

Security properties

→ Completeness ✓

→ Soundness

Two ways of cheating:

- → Guessing an ϵ_i or manipulating the multiplication test: $\frac{1}{q}$.
- → Guessing the second challenge: $\frac{1}{N}$.

Security properties

→ Completeness ✓

→ Soundness ✓

Two ways of cheating:

→ Guessing an ϵ_i or manipulating the multiplication test: $\frac{1}{q}$.

→ Guessing the second challenge: $\frac{1}{N}$.

Security properties

→ Completeness ✓

→ Soundness ✓

Two ways of cheating:

→ Guessing an ϵ_i or manipulating the multiplication test: $\frac{1}{q}$.

→ Guessing the second challenge: $\frac{1}{N}$.

→ Zero-knowledge

Security properties

→ Completeness ✓

→ Soundness ✓

Two ways of cheating:

→ Guessing an ϵ_i or manipulating the multiplication test: $\frac{1}{q}$.

→ Guessing the second challenge: $\frac{1}{N}$.

→ Zero-knowledge

As a result of the $(N-1)$-private property of the underlying MPC protocol.

Security properties

→ Completeness ✓

→ Soundness ✓

Two ways of cheating:

→ Guessing an ϵ_i or manipulating the multiplication test: $\frac{1}{q}$.

→ Guessing the second challenge: $\frac{1}{N}$.

→ Zero-knowledge ✓

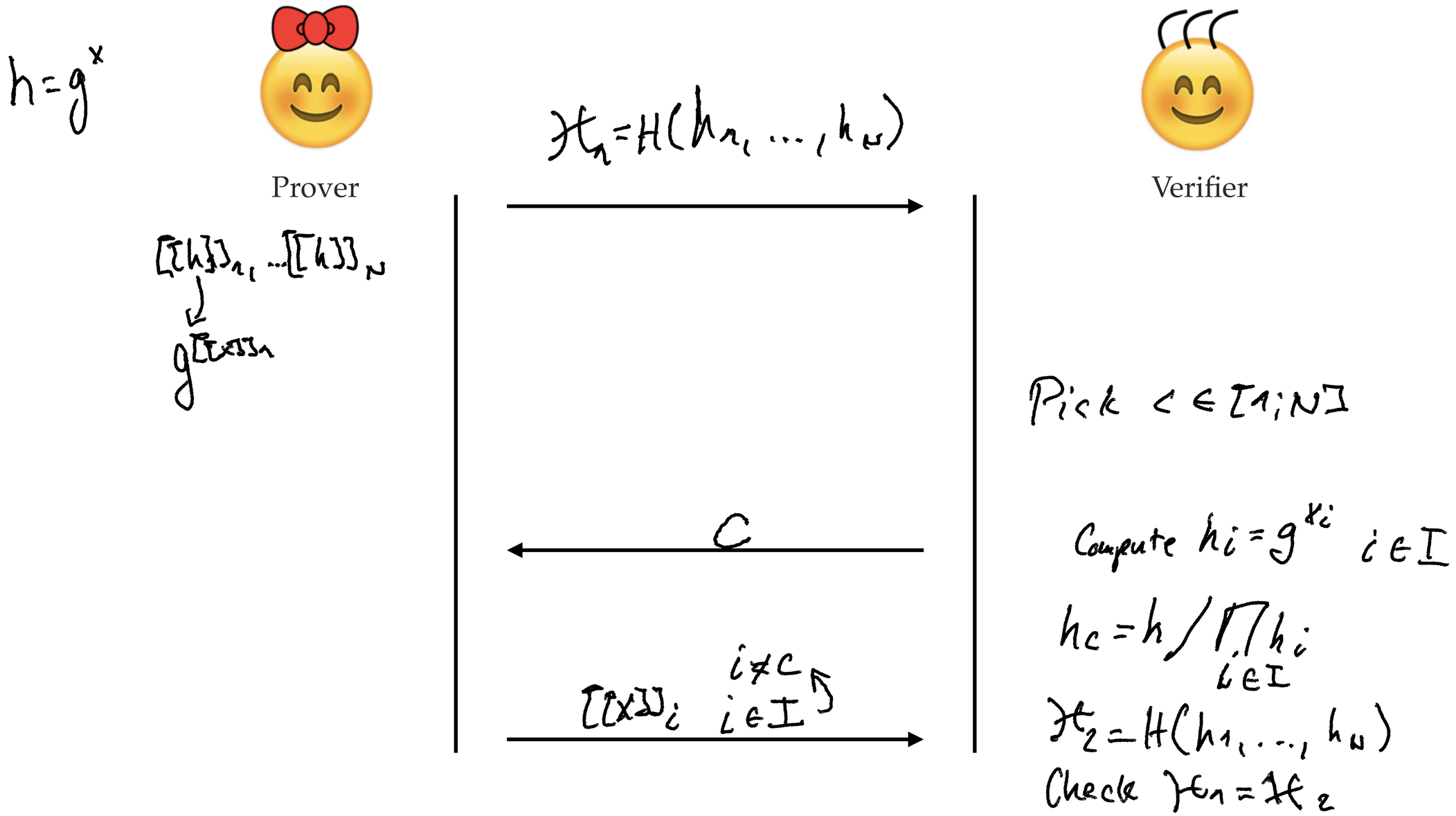
As a result of the $(N-1)$-private property of the underlying MPC protocol.

Optimizations

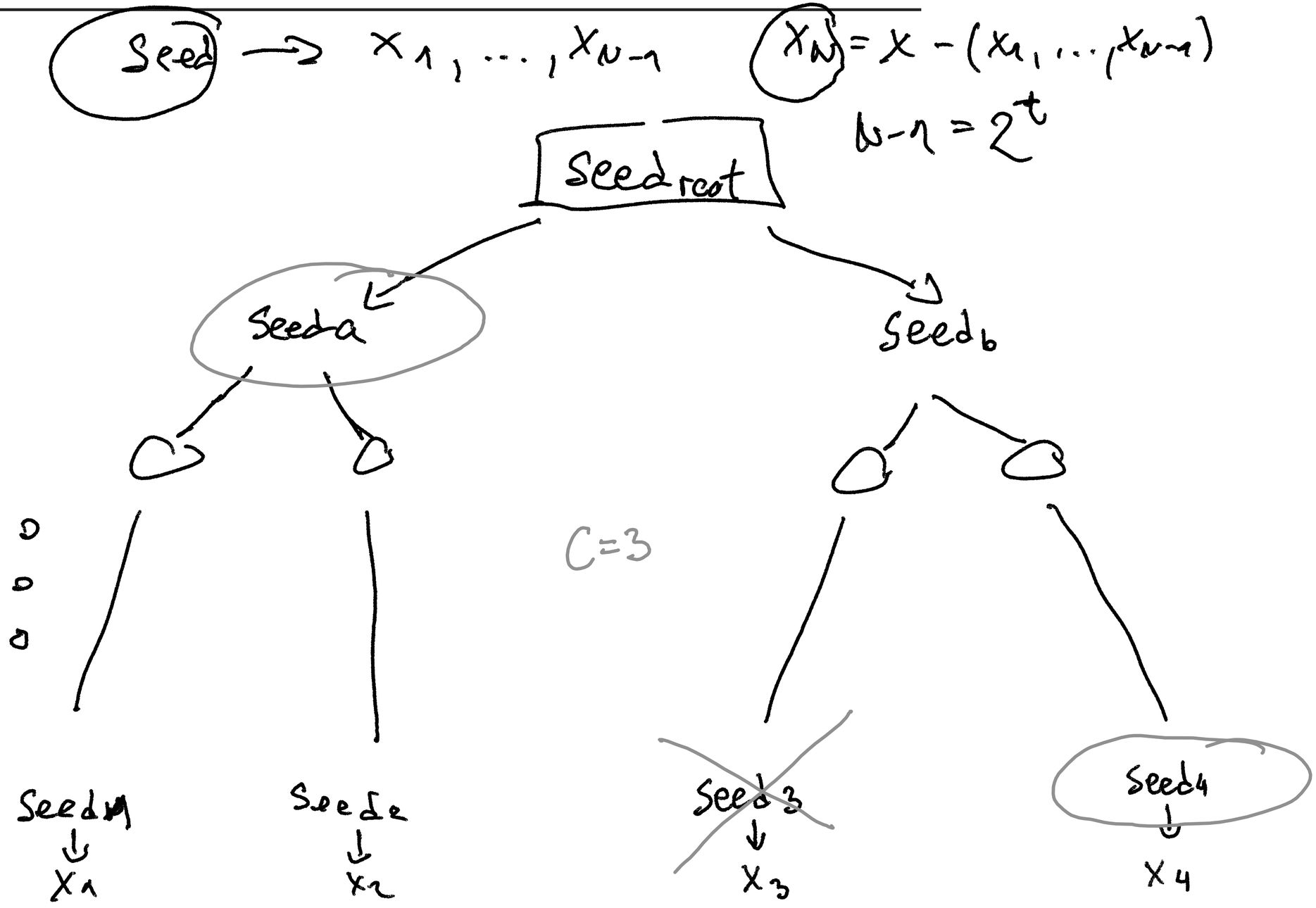
Optimizations

- Hashing the commitments
- Seed tree
- Hypercube

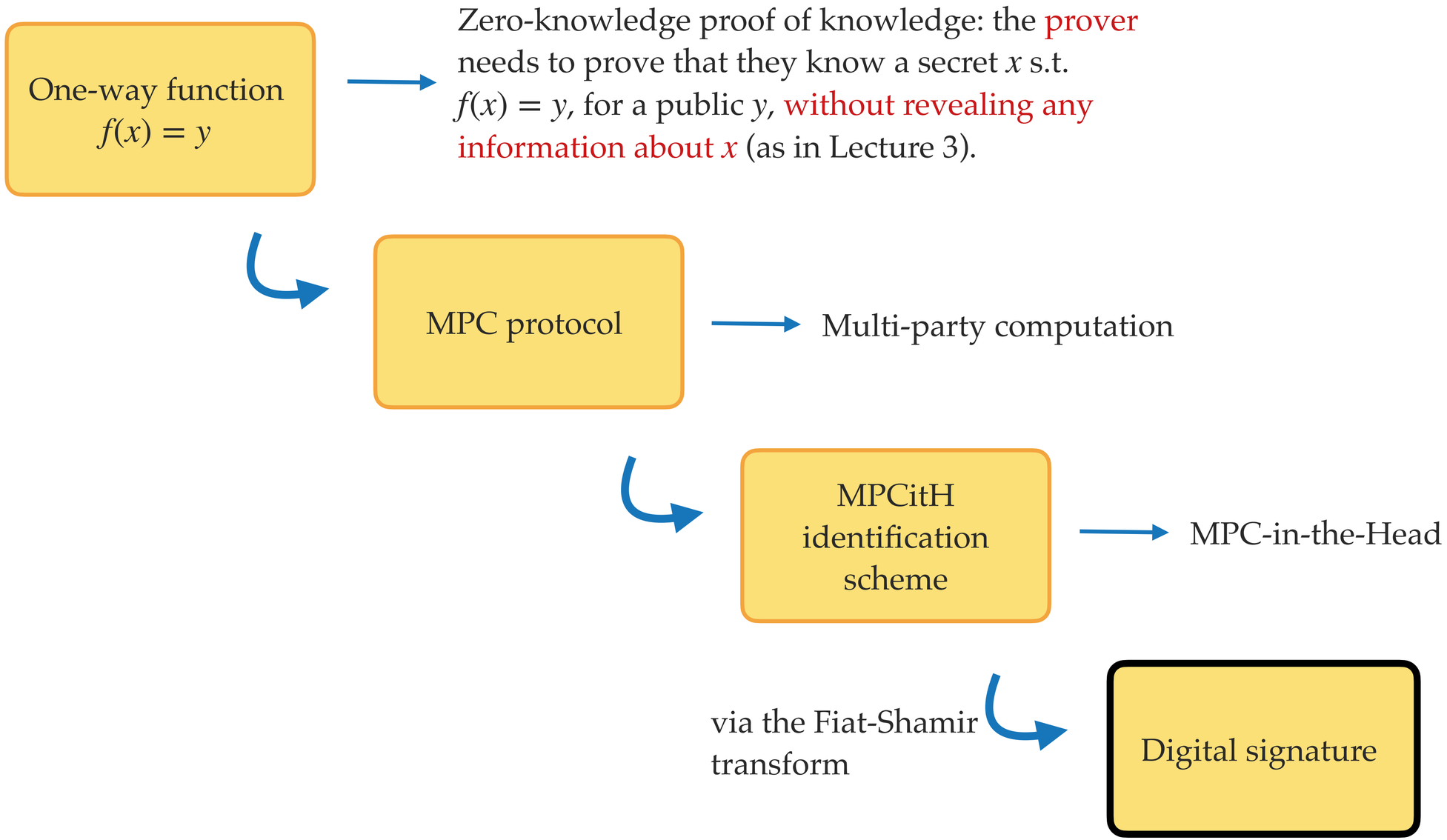
Hashing the commitments



Seed tree


Seed → x1, ..., xN-1
xN = x - (x1, ..., xN-1)
N-1 = 2^t
Seed_root
Seed_a
Seed_b
C=3
Seed_1
x1
Seed_2
x2
Seed_3
x3
Seed_4
x4

The MPCitH construction



The SDitH signature scheme

Signing

Signature:

1. Generate random sharing $[\![x_A]\!]$, $[\![P]\!]$, $[\![Q]\!]$, $[\![a]\!]$, $[\![b]\!]$, $[\![c]\!]$
2. Commit the parties' shares:

$$[\![x_A]\!]_i, [\![P]\!]_i, [\![Q]\!]_i, [\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i \xrightarrow{\text{Commit}} \mathsf{com}_i$$

3. Derive the first challenge (randomness of MPC protocol):

$$\mathsf{com}_1, \ldots, \mathsf{com}_N \xrightarrow{\text{Hash}} h_1 \to r, \varepsilon$$

4. Simulate the MPC protocol:

$$[\![x_A]\!], [\![P]\!], [\![Q]\!], [\![a]\!], [\![b]\!], [\![c]\!], r, \varepsilon \xrightarrow{\text{MPC}} [\![\alpha]\!], [\![\beta]\!], [\![v]\!]$$

5. Derive the second challenge (index of non-opened party):

$$h_1, [\![\alpha]\!], [\![\beta]\!], [\![v]\!] \xrightarrow{\text{Hash}} h_2 \to I$$

6. Build the signature from

$$h_1,\ h_2,\ \left\{[\![x_A]\!]_i, [\![P]\!]_i, [\![Q]\!]_i, [\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i\right\}_{i \in I},\ \left\{\mathsf{com}_i, [\![\alpha]\!]_i, [\![\beta]\!]_i, [\![v]\!]_i\right\}_{i \notin I}$$

Verification

Verification:

1. Recompute the commitments, for parties $i \in I$ (with I obtained from h_2):

$$[\![x_A]\!]_i, [\![P]\!]_i, [\![Q]\!]_i, [\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i \xrightarrow{\text{Commit}} \mathsf{com}_i$$

2. Recompute the first challenge (randomness of MPC protocol):

$$\mathsf{com}_1, \ldots, \mathsf{com}_N \xrightarrow{\text{Hash}} h_1 \to r, \varepsilon$$

3. Simulate the MPC protocol, for parties $i \in I$:

$$[\![x_A]\!]_i, [\![P]\!]_i, [\![Q]\!]_i, [\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i, r, \varepsilon \xrightarrow{\text{MPC}} [\![\alpha]\!]_i, [\![\beta]\!]_i, [\![v]\!]_i$$

4. Recompute the second challenge (index of non-opened party):

$$h_1, [\![\alpha]\!], [\![\beta]\!], [\![v]\!] \xrightarrow{\text{Hash}} h_2$$

5. Check that recomputed h_1, h_2 match the signature.

MPCitH in the NIST competition

SDitH	→	Syndrome decoding problem in the Hamming metric
RYDE	→	Syndrome decoding problem in the rank metric
PERK	→	Permuted kernel problem
MQOM	→	MQ problem (Lecture 1)
MiRitH	→	MinRank problem
MIRA	→	MinRank problem
Biscuit	→	MQ problem (with additional structure)

MPCitH in the NIST competition

SDitH

RYDE

PERK

MQOM

MiRitH

MIRA

Biscuit

Example.

Parameter Set	MPCitH Parameters						Sizes (in bytes)			
	N	ℓ	τ	η	t	p	pk	sk	Sig. Avg	Sig. Max
SDitH-L1-hyp	2^8	–	17	4	3	$2^{-71.2}$	120	404	8 241	8 260
SDitH-L3-hyp	2^8	–	26	4	3	$2^{-72.4}$	183	616	19 161	19 206
SDitH-L5-hyp	2^8	–	34	4	4	$2^{-94.8}$	234	812	33 370	33 448
SDitH-L1-thr	q	3	6	4	7	$2^{-166.2}$	120	404	10 117	10 424
SDitH-L3-thr	q	3	9	4	10	$2^{-241.5}$	183	616	24 918	25 603
SDitH-L5-thr	q	3	12	4	13	$2^{-308.5}$	234	812	43 943	45 160