VOLE-based ZK and VOLEitH

Università Bocconi MILANO

Emmanuela Orsini

AIT AUSTRIAN INSTITUTE OF TECHNOLOGY

Sebastian Ramacher

DTU

Carsten Baum
Christian Majenz

Lennart Braun
Lawrence Roy
Peter Scholl

Aalto University

Michael Klooß

COSIC

Cyprien Delpech de Saint Guilhem

IBM Research

Ward Beullens

TU Graz

Shibam Mukherjee
Christian Rechberger

Based on

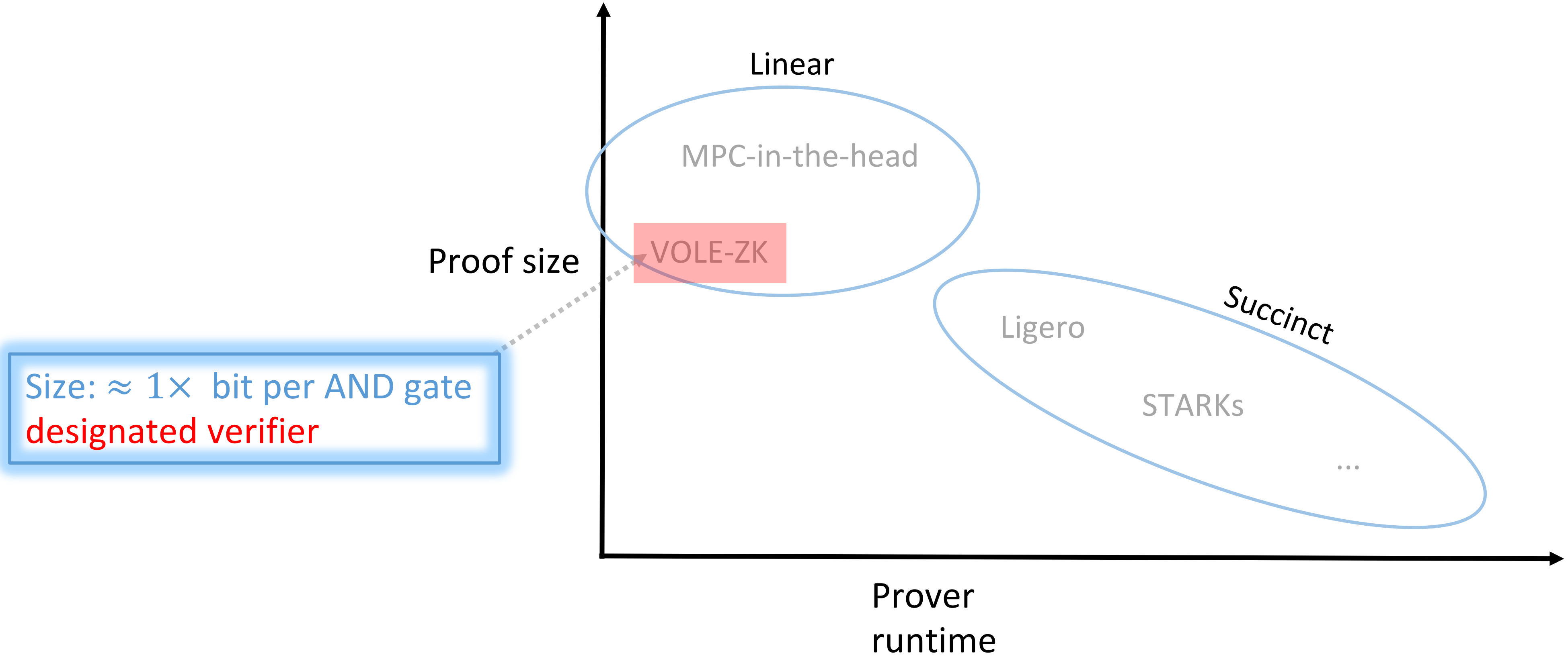
Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures From VOLE-in-the-Head
Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß,
Emmanuela Orsini, Lawrence Roy, Peter Scholl
CRYPTO 2023

FAEST: Algorithm Specifications
Carsten Baum, Lennart Braun, Cyprien de Saint Guilhem, Michael Klooß, Christian Majenz,
Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger,
Lawrence Roy, Peter Scholl

One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures
Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher,
Christian Rechberger, Lawrence Roy, Peter Scholl
In submission

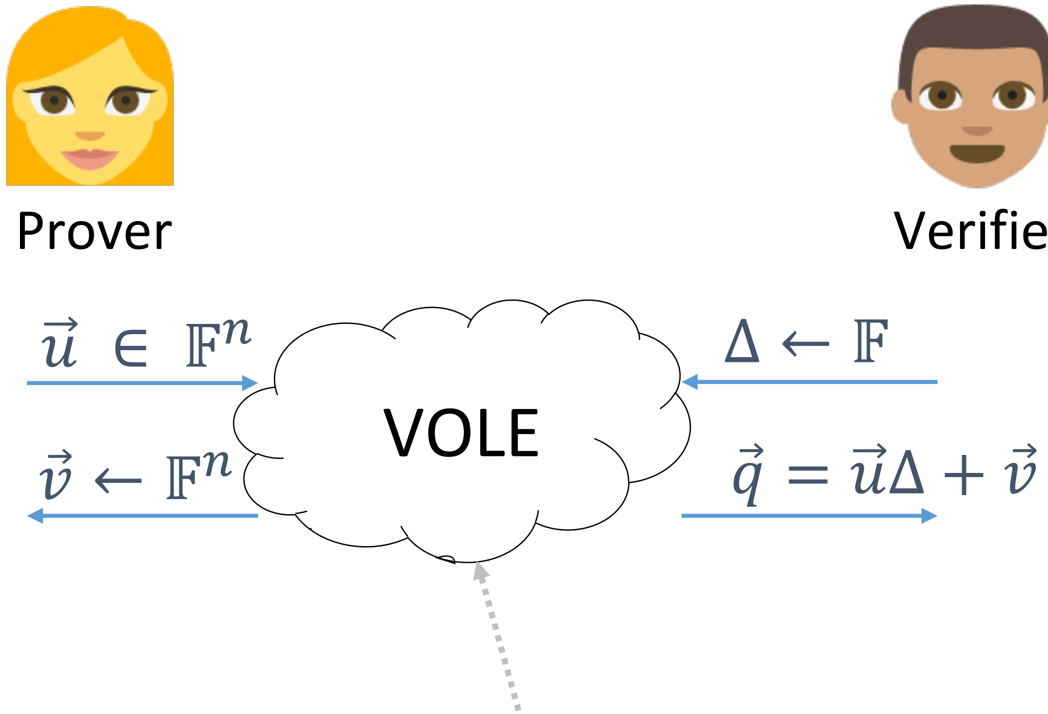
No VOLEs were harmed while making this presentation
Thanks to Lance and Lennart for the slides

Families of ZK Proofs

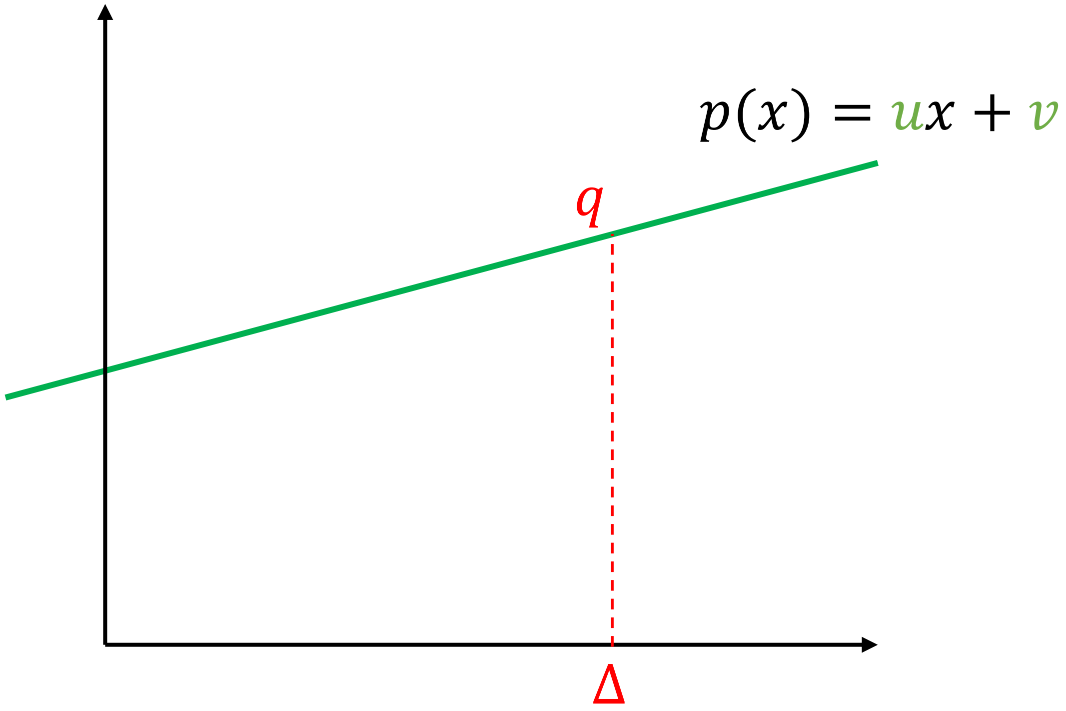


VOLE-ZK
– in the designated verifier setting

Background: VOLE
(vector oblivious linear evaluation)



Can be instantiated with OT, HE, LPN…



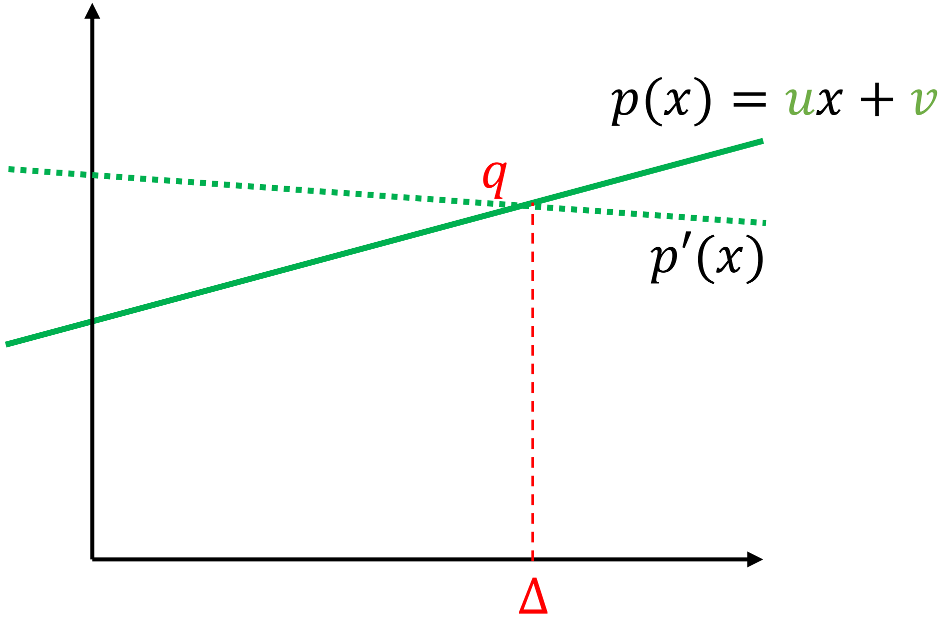
ZK from VOLE (designated verifier)

[BMRS 21, WYKW 21]

Use VOLE as a linear commitment to $\vec{u}$

To open

- Prover sends (u, v), Verifier checks if $q = u\Delta + v$
- Hiding: since v is random
- Binding: opening to $u' \neq u$ requires guessing Δ — prob. $1/|\mathbb{F}|$
  (as long as Δ is kept secret from Prover $\longrightarrow$ D.V.)



ZK from VOLE (designated verifier)

[BMRS 21, WYKW 21]

Use VOLE as a linear commitment to $\vec{u}$

Prover has $(u_1, v_1), (u_2, v_2)$

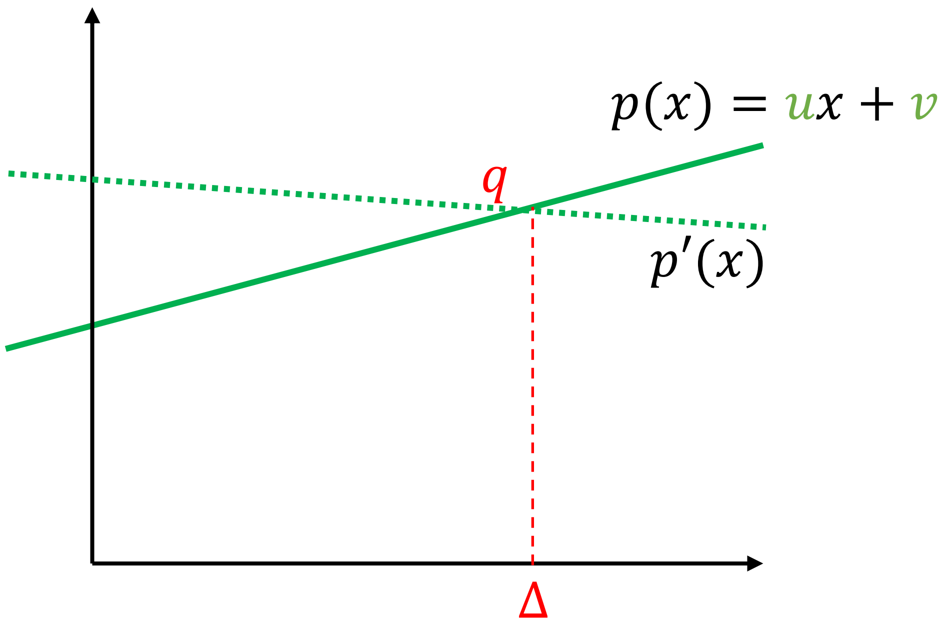
Verifier has $(\Delta, q_1 = u_1\Delta + v_1, q_2 = u_2\Delta + v_2)$

For $\alpha, \beta \in \mathbb{F}$ we have

Computed locally by Verifier

$$q = \alpha q_1 + q_2 + \beta\Delta$$
$$= (\alpha u_1 + u_2 + \beta)\Delta + (\alpha v_1 + v_2)$$

Computed locally by Prover



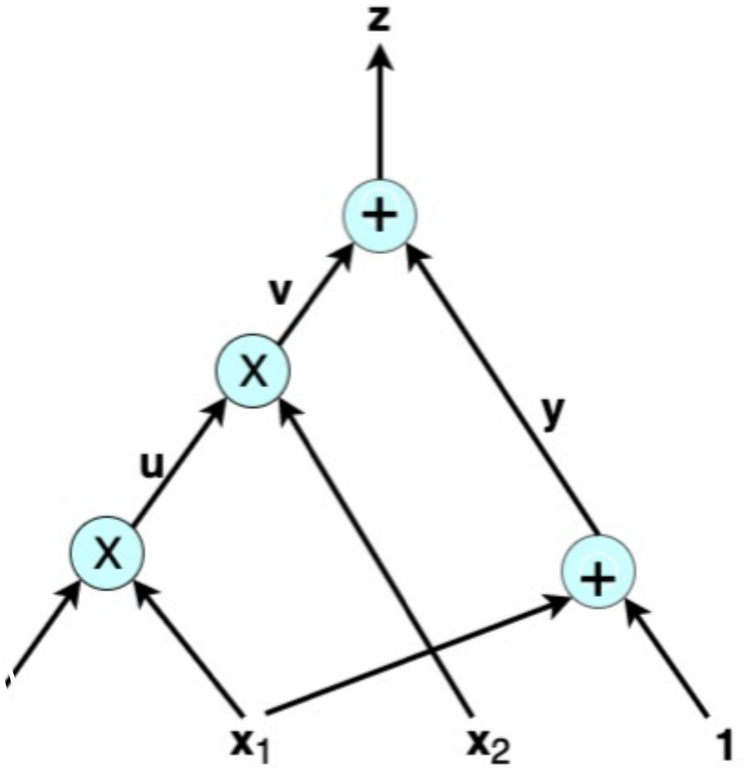
ZK from VOLE via Commit-and-Prove

[BMRS 21, WYKW 21]

- Commit to witness $\vec{w}$
  - ➢Get random VOLE with secret $\vec{u}$
  - ➢Prover sends $\vec{\delta} = \vec{w} - \vec{u}$ to Verifier
  - ➢Both add $\vec{\delta}$ to VOLE commitment to $\vec{u}$

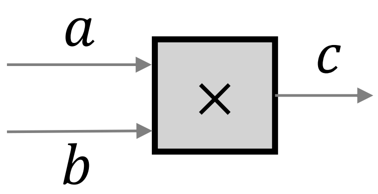
- Evaluate C gate-by-gate:
  - ➢Linear gates: easy

  - ➢Multiplication: ???

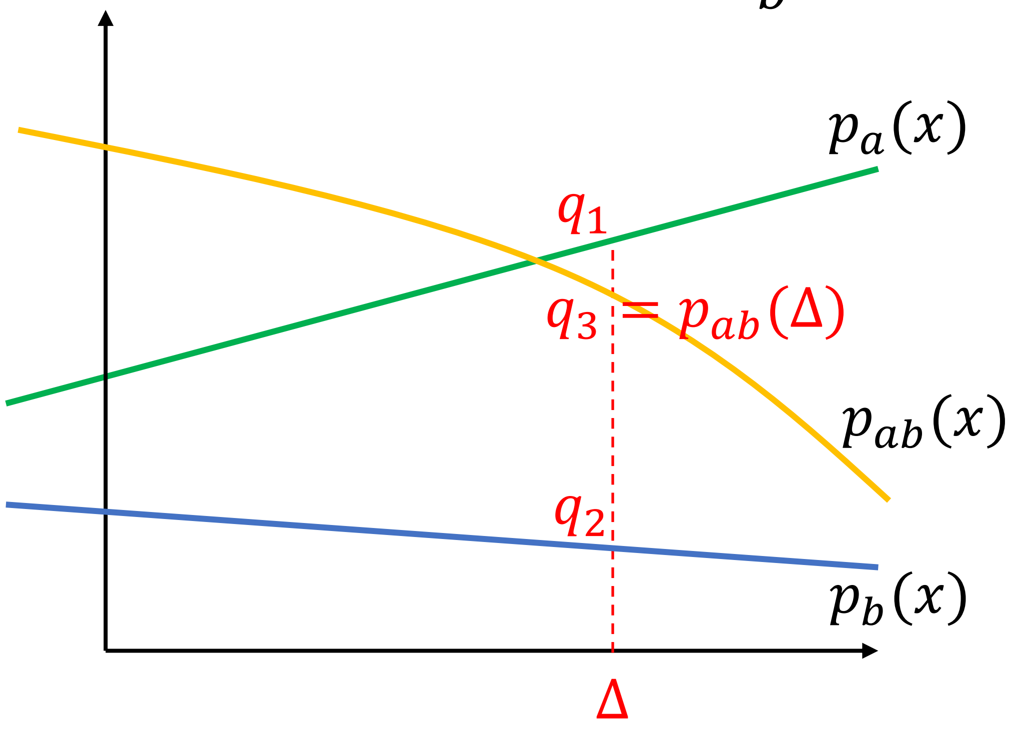


Multiplication check (QuickSilver)

[DIO 21, YSWW 21]



- Multiply two lines $\Rightarrow$ quadratic polynomial
  - $p_{ab}(x) = d_0 + d_1 x + abx^2$
- Commit to output $c \Rightarrow p_c(x) = v + cx$
- $p_{ab}(x) - xp_c(x)$ should be degree-1 if $ab = c$
  - Open and check
  - First, mask with random deg-1 commitment



Cost analysis for VOLE-ZK

- LPN-based VOLE generates ℓ random VOLEs with $o(\ell)$ communication

- Per multiplication gate:
  - Commit to c
    - $1\times$ VOLE element
  - Open masked commitment
    - Can be amortized to constant size over all mults (check random combination of polynomials)

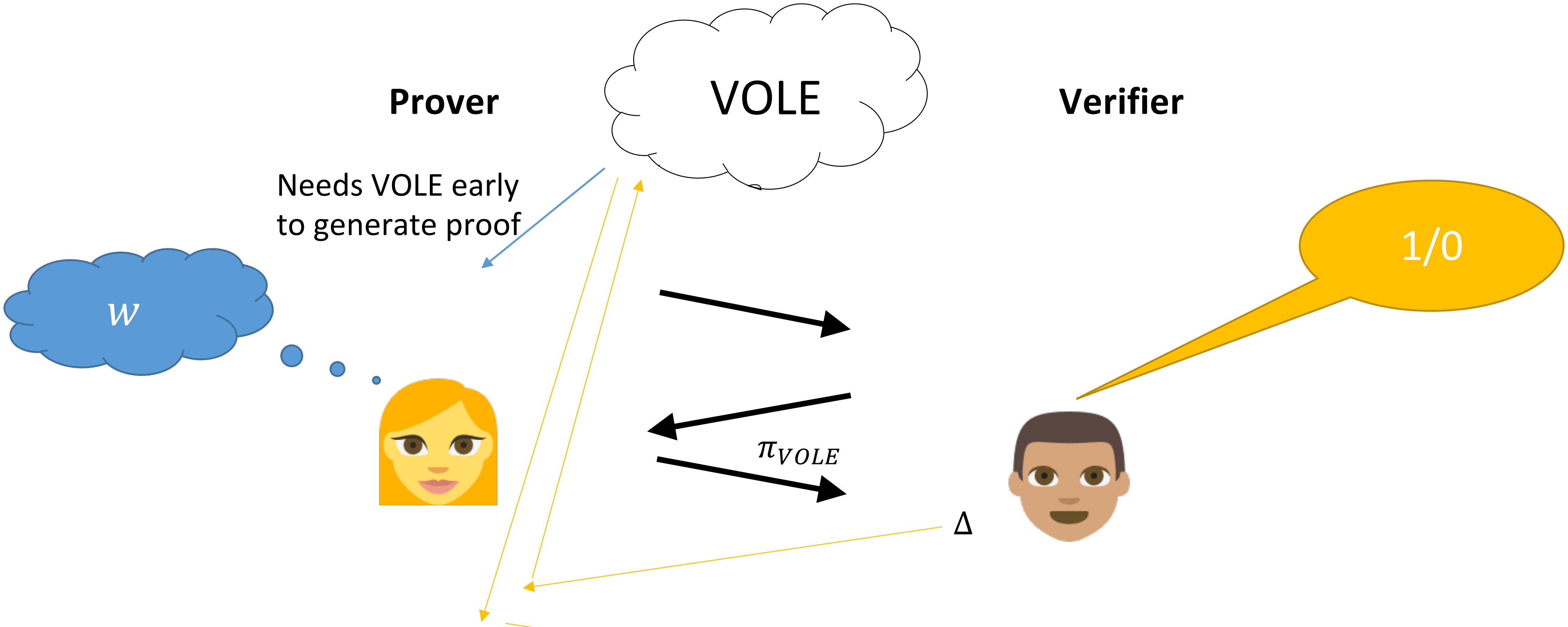
- For circuit:
  - $m+n$ field elements for m inputs and n mult. gates

To open

- Prover sends (u, v), Verifier checks if $q = u\Delta + v$
- Hiding: since v is random
- Binding: opening to $u' \neq u$ requires guessing Δ — prob. $1/|\mathbb{F}|$
  (as long as Δ is kept secret from Prover $\longrightarrow$ D.V.)

Really necessary???

Is secret Δ necessary?



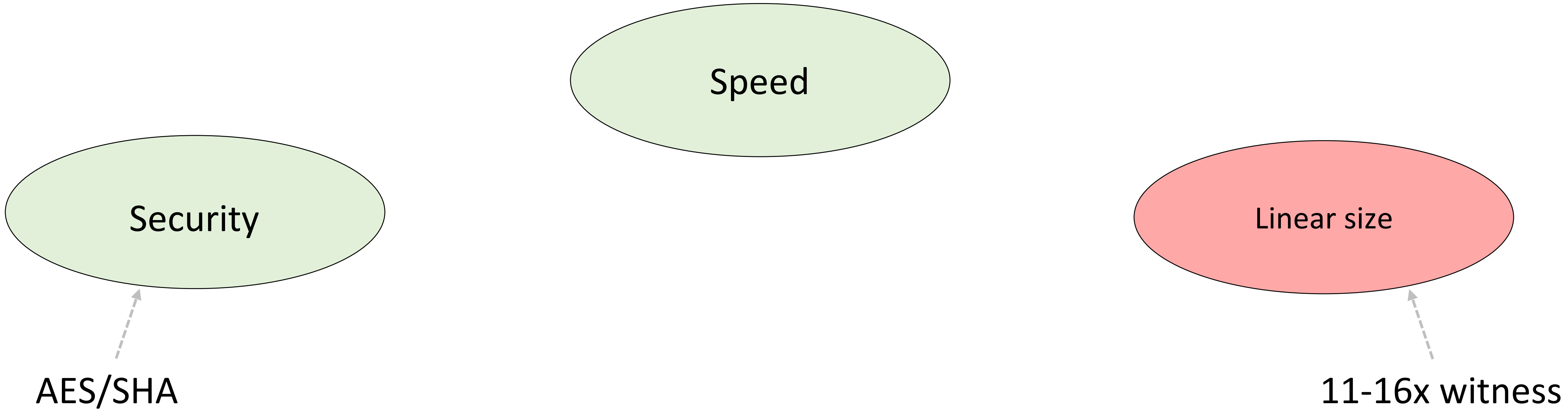
Needs VOLE shares only in the end
Prover can even learn Δ

Fiat-Shamir goes through!

VOLE-in-the-Head

– Adding public verifiability

VOLE-in-the-Head: a general tool for making VOLE-ZK proofs publicly verifiable

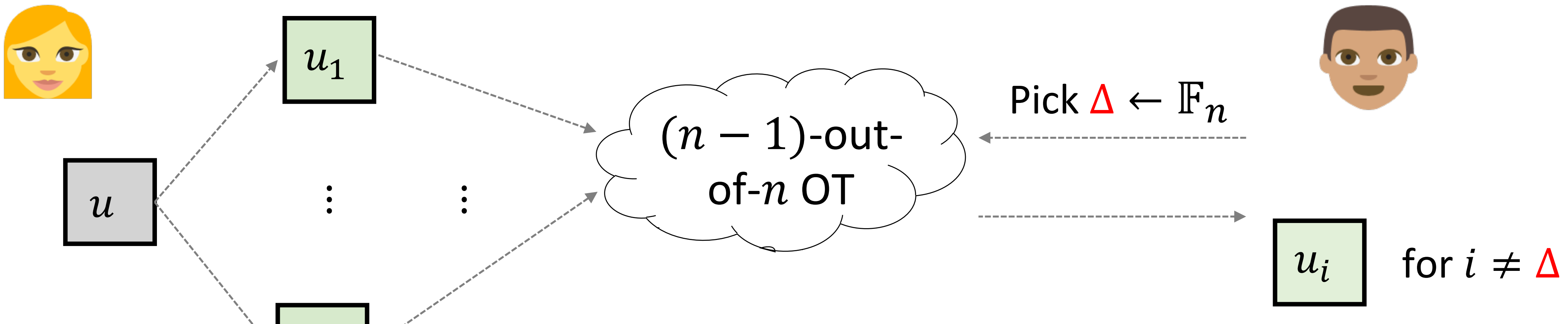


Application: FAEST post-quantum signature scheme

How to do VOLE? Warm-up: using OT

Key observation: n-out-of-n secret sharing + OT $\Rightarrow$ VOLE in $\mathbb{F}_n$

(from SoftSpokenOT [Roy 22])



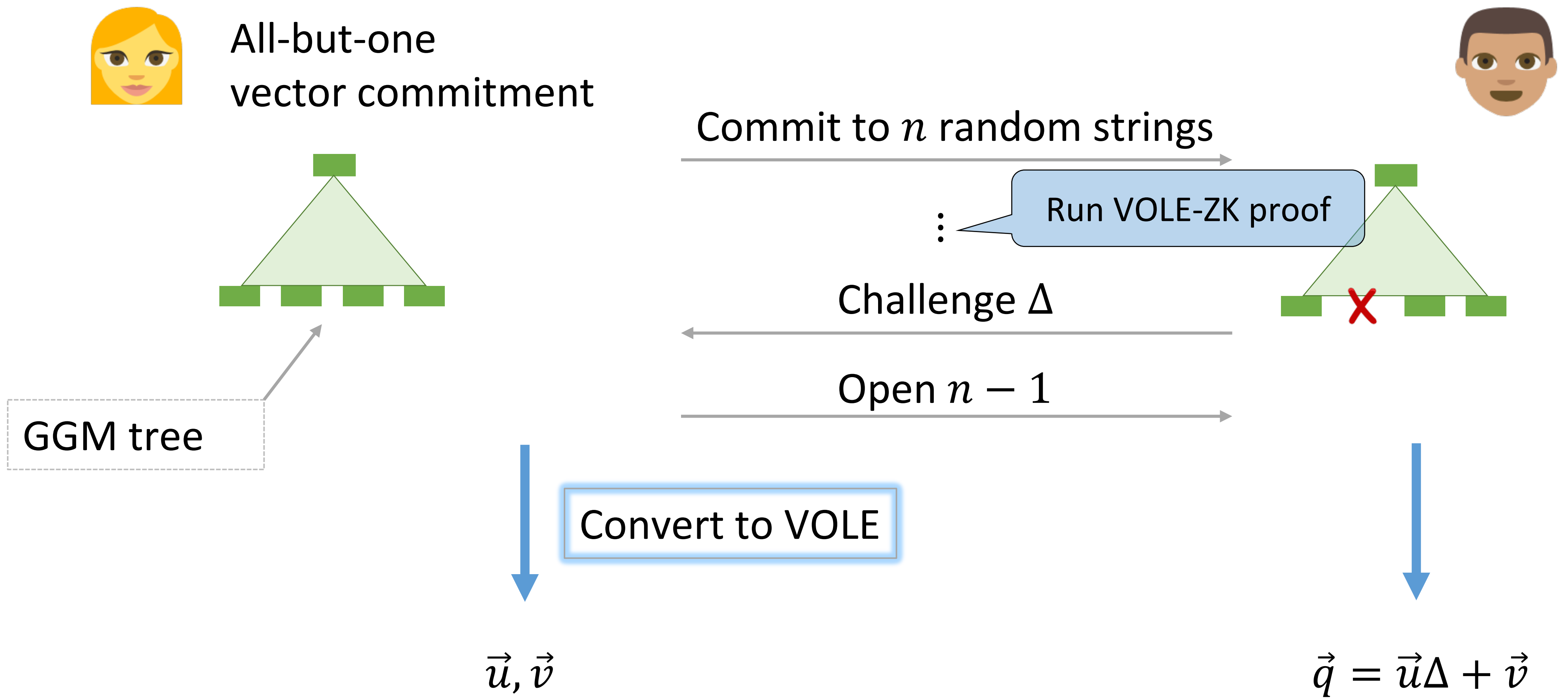
$u = u_1 + \cdots + u_n$

$v = -1 \cdot u_1 - \cdots - n \cdot u_n$ (in $\mathbb{F}_n$)

$$q = \sum_i u_i \cdot (\Delta - i) = u\Delta + v$$

0 when $i = \Delta$

How to do VOLE-in-the-head?



VOLE-in-the-head: some details

- $(n-1)$-out-of-n vector commit $\Rightarrow$ VOLE in $\mathbb{F}_n$
  - ➢ Commitments have soundness error $\frac{1}{n}$ ⍰
  - ➢ What about $\mathbb{F}_m$ for large m?

- For extension fields, $m = n^{\tau}$:
  - ➢ Repeat τ times, with same $u \in \mathbb{F}_n$
  - ➢ Cost over $\mathbb{F}_2$, 11-16 bits per AND (for 128-bit security)

Needs consistency check

More details: consistency checking

- When repeating τ times:
  - ➢Need to ensure prover uses consistent u

- Consistency check from [Roy 22]:

Universal hash function R

←

$\tilde{u} = R\vec{u},\ \tilde{v} = R\vec{v}$

→

Check $R\vec{q} = \tilde{u}\Delta + \tilde{v}$

- Security:
  - ➢Needs new analysis for round-by-round soundness with Fiat-Shamir

FAEST

– Post-quantum signatures from AES

19

Paradigm for ZK-based signatures

- Signature:
  - NIZK proof of knowledge of sk, such that $pk = \mathrm{Enc}_{sk}(x)$

- Challenge: finding a ZK-friendly Enc
  - Custom ciphers: e.g. LowMC, MiMC
  - Other assumptions: code-based, multivariate…

AES: a ZK-friendly OWF?

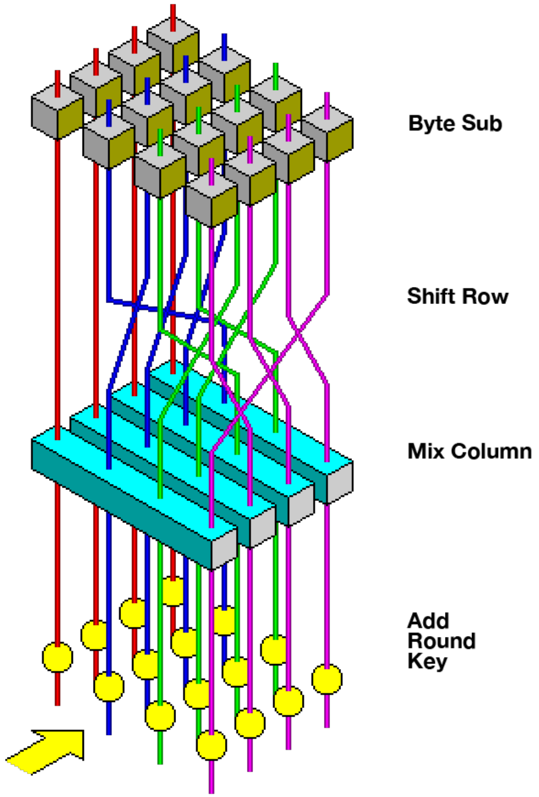
ShiftRows, MixColumns, AddRoundKey:

- All linear over $\mathbb{F}_2$

S-Box:

- Inversion in $\mathbb{F}_{2^8}$
- Prove in ZK as 1 multiplication constraint

$$x \cdot y = 1 \iff y = x^{-1}$$



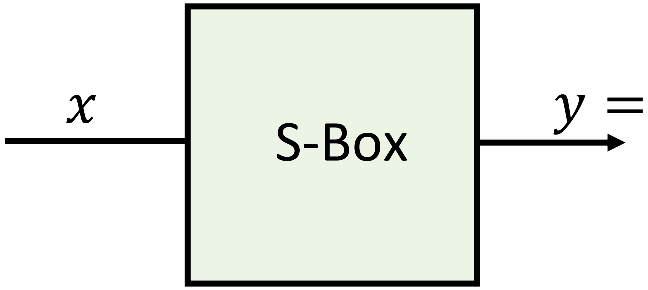
Proving AES-128 in FAEST

Witness: key + internal state of each round (+ key schedule)

- 1600 bits (in $\mathbb{F}_2$)

200 constraints over $\mathbb{F}_{2^8}$:

- 1 per S-box:
  - ➢degree-2 polynomial: $xy = 1$



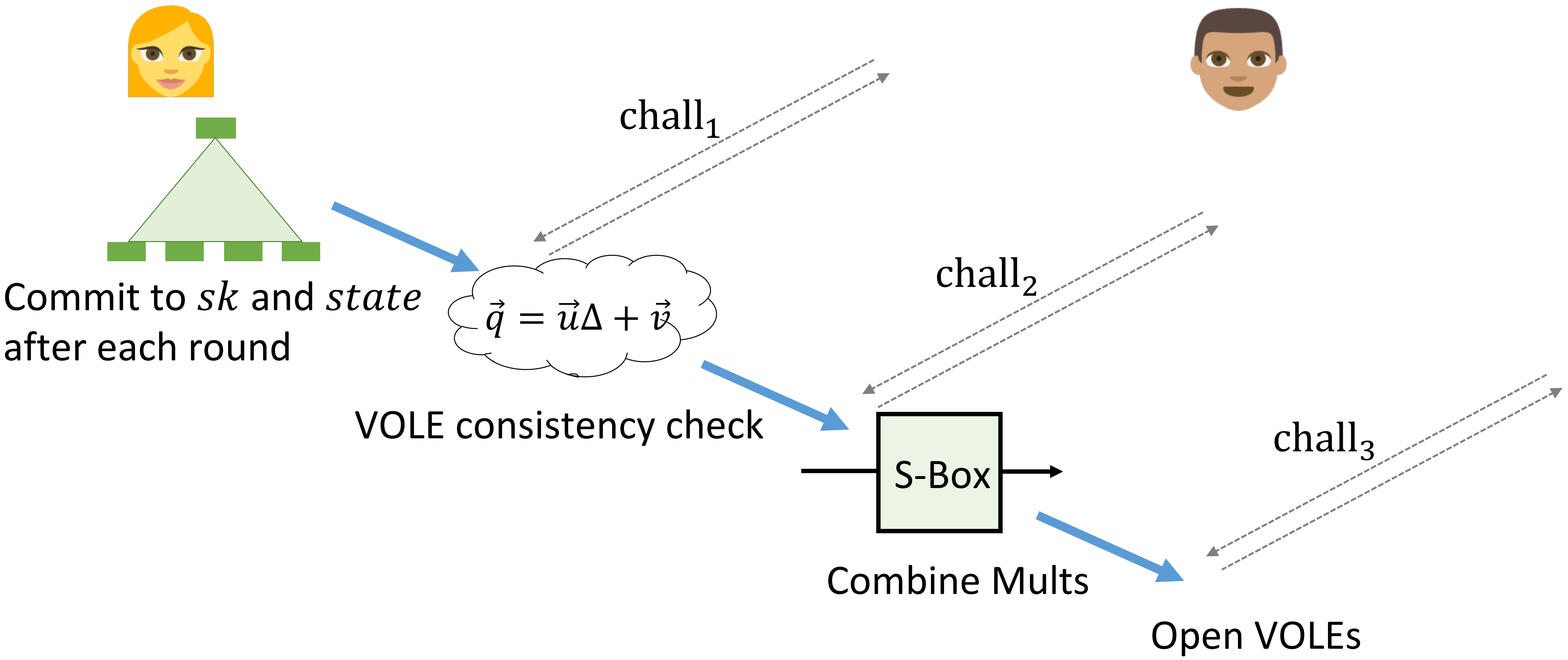
$$y = \begin{cases} 0 & \text{if } x = 0 \\ x^{-1} & \text{o.w.} \end{cases}$$

(in $\mathbb{F}_{2^8}$)

What if $x = 0$?

- ➢Sample k such that this never happens
- ➢1-2 bits less security (for AES-128)

FAEST summary: proving $pk = \mathrm{AES}_{sk}(x)$



23

FAEST performance (AVX2 + AES-NI)

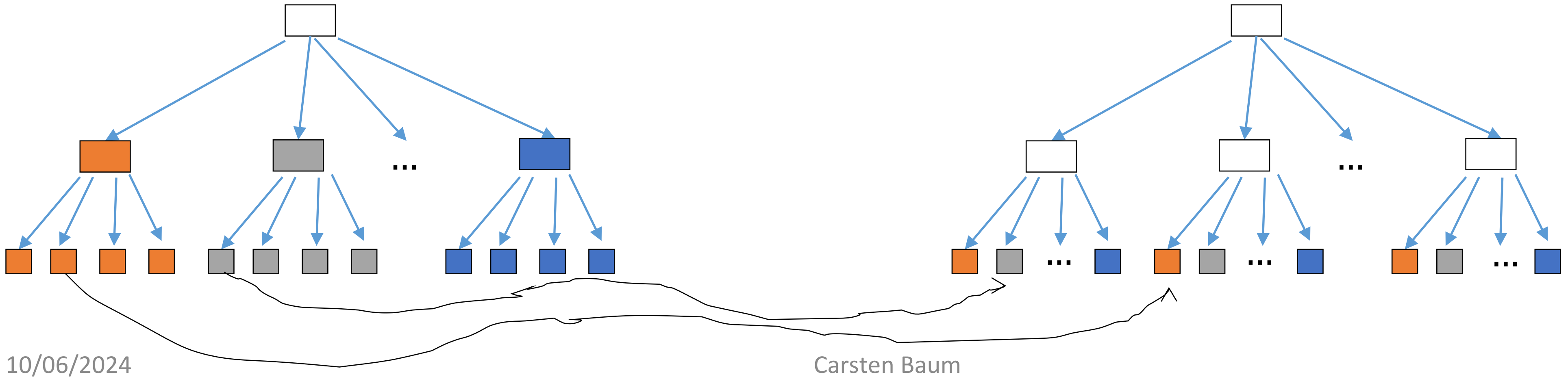
	Sign/Verify	Size
FAEST-128s	$\approx$ 4,4 ms	5.006 B
FAEST-128f	$\approx$ 0,4 ms	6.336 B

Timings on machine with AMD Ryzen 7 5800H, 3.2–4.4 GHz

Optimizations

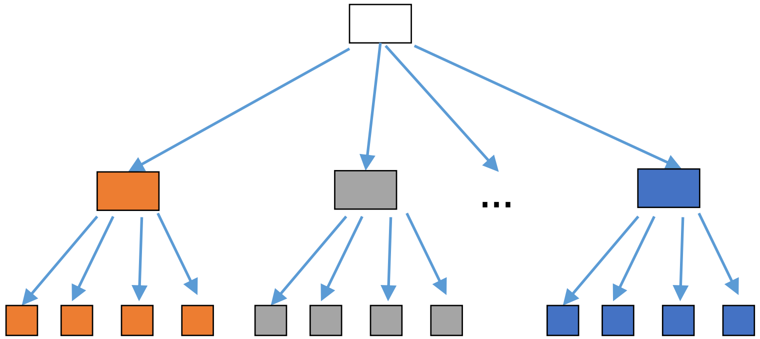
All-but-one
vector commitment

- VOLEitH (like MPCitH) uses τ all-but-one vector commitments from GGM
- Opening them separately is inefficient
- Interleave vector commitments



Optimizations

All-but-one
vector commitment

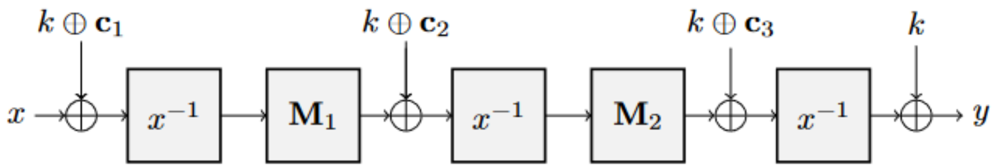


Rejection sampling on vector commitment opening (not every index is valid)

⇒ more work for (dishonest) prover

⇒ can lower τ

Different OWFs - Rain

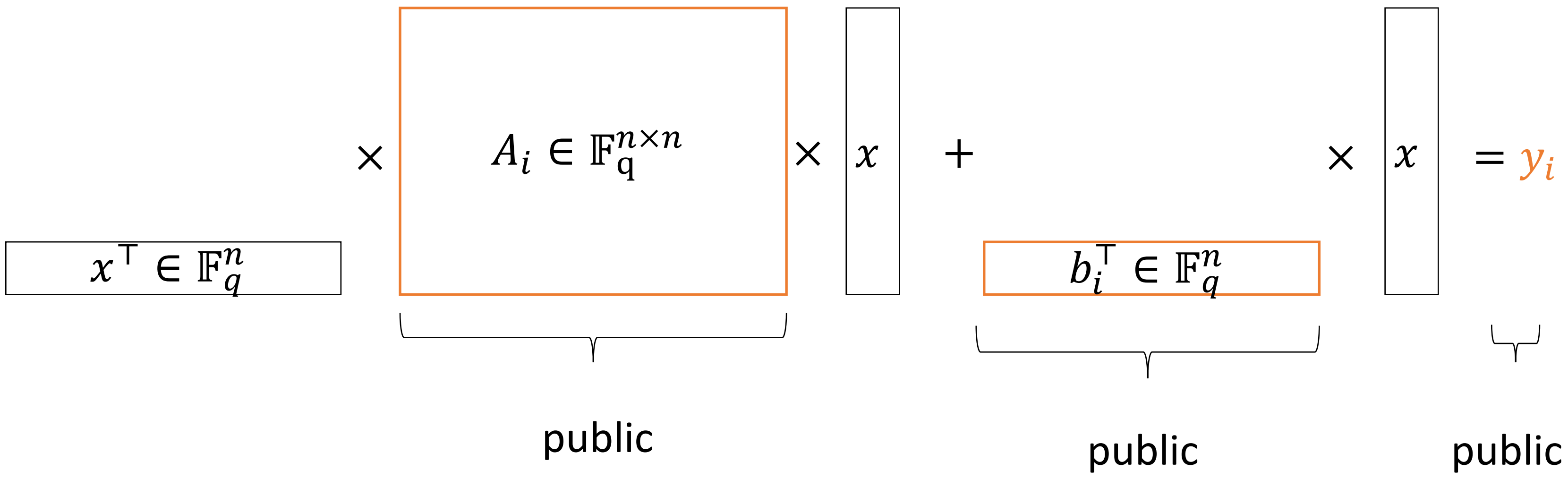


- Linear operations (add k, c_i, multiply with M_i) over $\mathbb{F}_2$
- Inversion over $\mathbb{F}_{2^\lambda}$

Witness: key + output of each S-Box

λ *bits* (key) — $R \cdot \lambda$ *bits* (output of each S-Box)

Different OWFs – MQ polynomials



Can derive A_i, b_i from public seed

$pk = (seed, (y_i)_{i\in[m]}), sk = (x_j)$

Witness only consists of sk. Main bottleneck is computation of $O(mn^2)$ mults in $\mathbb{F}_{2^\lambda}$

FAEST performance (AVX2 + AES-NI)

	Sign/Verify	Size
FAEST-128s	≈ 4,4 ms	5.006 B
FAEST-128f	≈ 0,4 ms	6.336 B
FAEST-fullkey-128s	≈ 4,4 ms	5.006 B
FAEST-fullkey-128f	≈ 0,5 ms	6.336 B
FAESTER-128s	≈ 3,3 ms	4.594 B
FAESTER-128f	≈ 0,4 ms	5.444 B
MandaRain-4-128s	≈ 2,8ms	3.114 B
MandaRain-4-128f	≈ 0,4 ms	3.878 B
KuMQuat-2-L1s	≈ 4,3 ms	2.555 B
KuMQuat-2-L1f	≈ 0,5 ms	3.028 B
SPHINCS+s	≈ 4,4 ms/ 0,4 ms	17.088 B
SPHINCS+f	≈ 88,2 ms/ 0,15 ms	7.856 B

Timings on machine with AMD Ryzen 7 5800H, 3.2–4.4 GHz

4-round Rain OWF (MandaRain-4-128s, MandaRain-4-128f)

MQ OWF (KuMQuat-2-L1s, KuMQuat-2-L1f)

Conclusion

Thank you!

VOLE-ZK proofs:

- Lightweight and fast with linear size
- VOLE-in-the-head: publicly verifiable

FAEST signature:

- Conservative security
- Reasonable performance
- Recent, more aggressive optimizations

https://faest.info