Stoats: Short Two-round vOle-bAsed Threshold Signatures

Lennart Braun, Université Paris Cité, CNRS, IRIF, France

Geoffroy Couteau, Université Paris Cité, CNRS, IRIF, France

Kelsey Melissaris, Université Paris Cité, CNRS, IRIF, France

Mahshid Riahinia, DIENS, École normale supérieure, CNRS, France

Peter Rindal, Category Labs, New York City, USA

Elahe Sadeghi, University of Texas at Austin, Texas, USA

Presented on 2026-10-07 @ NIST TCPT3

Our Submission

Threshold Schnorr/ECDSA with

- **Two rounds**
- **Low communication**
  - 96B for Schnorr
  - 128B for ECDSA
- **Fast Signing**
  - ~1.10 ms per signing session
- **UC security**
- **Deterministic + stateless signing**
- **Two party**
- **Setup supports bounded # of sessions**
  - 2 seconds for dozen millions signing sessions

Tradeoff: spend ~2 seconds in setup for bounded # of signing sessions for optimum performance. A good balance: In real-world applications, 2s of setup cost (without regenerating sk) for dozen millions of signing sessions is a fairly acceptable maintenance cost

Schnorr Signing Recap

$\langle \mathbb{G}, q, G \rangle$
$H: \{0,1\}^* \to \mathbb{F}_q$

$Setup$:

- $sk \xleftarrow{\$} \mathbb{F}_q^*$
- $pk \leftarrow sk \cdot G$

$Sign(m)$:

- $r \xleftarrow{\$} \mathbb{F}_q$
- $R \leftarrow r \cdot G$
- $e \leftarrow H(pk, R, m)$
- $s \leftarrow e \cdot sk + r$
- $\sigma = (R, s)$

$Verify\big(m, \sigma = (R, s)\big)$:

- $e \leftarrow H(pk, R, m)$
- Check $e \cdot pk + R = s \cdot G$

2-Party Schnorr Signing Idea



$sk = sk_0 + sk_1$
$pk = pk_0 + pk_1$



$\langle \mathbb{G}, q, G \rangle$
$H: \{0,1\}^* \to \mathbb{F}_q$

$sk_0 \xleftarrow{\$} \mathbb{F}_q^*$
$pk_0 \leftarrow sk_0 \cdot G$

$sk_1 \xleftarrow{\$} \mathbb{F}_q^*$
$pk_1 \leftarrow sk_1 \cdot G$

$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$r_0 \xleftarrow{\$} \mathbb{F}_q$
$R_0 \leftarrow r_0 \cdot G$
$R \leftarrow R_0 + R_1$
$e \leftarrow H(pk, R, m)$
$s_0 \leftarrow e \cdot sk_0 + r_0$

(R, s_0)

$e \leftarrow H(pk, R, m)$
$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

2-Party Schnorr Signing Idea



$$sk_0 \xleftarrow{\$} \mathbb{F}_q^*$$
$$pk_0 \leftarrow sk_0 \cdot G$$

Traditional fix: commit-open: add round + not deterministic
Alternative: authenticated deterministically-fixed nonces at setup



$$sk_1 \xleftarrow{\$} \mathbb{F}_q^*$$
$$pk_1 \leftarrow sk_1 \cdot G$$

$$\langle \mathbb{G}, q, G \rangle$$
$$H: \{0,1\}^* \rightarrow \mathbb{F}_q$$

$$r_0 \xleftarrow{\$} \mathbb{F}_q$$

$$R_1 = r_1 \cdot G$$

$$r_1 \xleftarrow{\$} \mathbb{F}_q$$

$$R_0 \leftarrow r_0 \cdot G$$
$$R \leftarrow R_0 + R_1$$
$$e \leftarrow H(pk, R, m)$$
$$s_0 \leftarrow e \cdot sk_0 + r_0$$

$$(R, s_0)$$

$$e \leftarrow H(pk, R, m)$$
$$s_1 \leftarrow e \cdot sk_1 + r_1$$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

VOLE-PCF: on-demand authenticated pseudorandomness



$(\mathbf{u}, \mathbf{v}) \in \mathbb{F}_p^n \times \mathbb{F}_p^n$

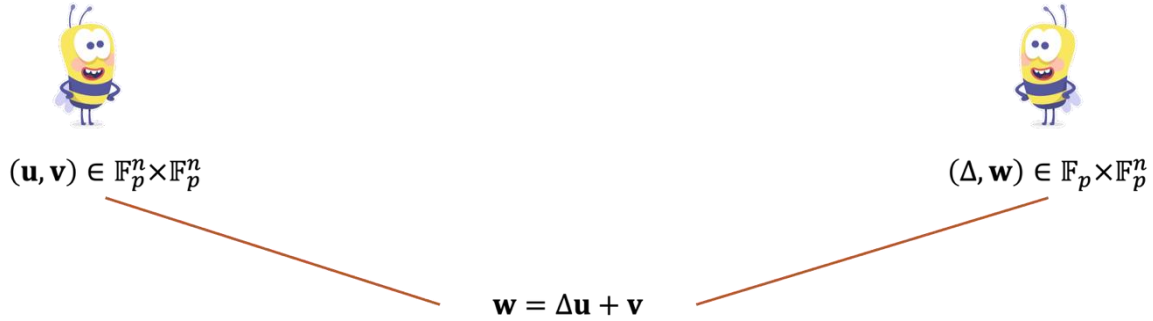


$(\Delta, \mathbf{w}) \in \mathbb{F}_p \times \mathbb{F}_p^n$

VOLE correlation

$\mathbf{w} = \Delta\mathbf{u} + \mathbf{v}$

VOLE-PCF: on-demand authenticated pseudorandomness



$$KeyGen(\Delta) \rightarrow \left(k_0^{PCF}, k_1^{PCF}\right)$$

$$Eval(k_b, b, x) \rightarrow \begin{cases} (u_x, v_x) \in \mathbb{F}_p : \mathrm{b} = 0 \\ w_x \in \mathbb{F}_p : \mathrm{b} = 1 \end{cases}$$

VOLE-PCF: on-demand authenticated pseudorandomness

Keys are short

$KeyGen(\Delta) \rightarrow \left(k_0^{PCF}, k_1^{PCF}\right)$

$Eval(k_b, b, x) \rightarrow \begin{cases} (u_x, v_x) \in \mathbb{F}_p : \mathrm{b} = 0 \\ w_x \in \mathbb{F}_p : \mathrm{b} = 1 \end{cases}$



$(\mathbf{u}, \mathbf{v}) \in \mathbb{F}_p^n \times \mathbb{F}_p^n$



$(\Delta, \mathbf{w}) \in \mathbb{F}_p \times \mathbb{F}_p^n$

$\mathbf{w} = \Delta\mathbf{u} + \mathbf{v}$

Locally generates correlations

VOLE-PCF: on-demand authenticated pseudorandomness





$(\mathbf{u}, \mathbf{v}) \in \mathbb{F}_p^n \times \mathbb{F}_p^n$

$(\Delta, \mathbf{w}) \in \mathbb{F}_p \times \mathbb{F}_p^n$

$\mathbf{w} = \Delta\mathbf{u} + \mathbf{v}$

$$KeyGen(\Delta) \rightarrow \left(k_0^{PCF}, k_1^{PCF}\right)$$

$$Eval(k_b, b, x) \rightarrow \begin{cases} (u_x, v_x) \in \mathbb{F}_p : \mathrm{b} = 0 \\ w_x \in \mathbb{F}_p : \mathrm{b} = 1 \end{cases}$$

Correctness:

$$w_x = \Delta u_x + v_x$$

VOLE-PCF: on-demand authenticated pseudorandomness

No info about Δ

$KeyGen(\Delta) \rightarrow \left(k_0^{PCF}, k_1^{PCF}\right)$

$Eval(k_b, b, x) \rightarrow \begin{cases} (u_x, v_x) \in \mathbb{F}_p : \mathrm{b} = 0 \\ w_x \in \mathbb{F}_p : \mathrm{b} = 1 \end{cases}$



$(\mathbf{u}, \mathbf{v}) \in \mathbb{F}_p^n \times \mathbb{F}_p^n$



$(\Delta, \mathbf{w}) \in \mathbb{F}_p \times \mathbb{F}_p^n$

$\mathbf{w} = \Delta\mathbf{u} + \mathbf{v}$

VOLE-PCF: on-demand authenticated pseudorandomness

(u_x, v_x) by *Eval*

$\approx$

(u_x, v_x) for $u_x \xleftarrow{\$} \mathbb{F}_p^n, v_x = w_x - \Delta \cdot u_x$



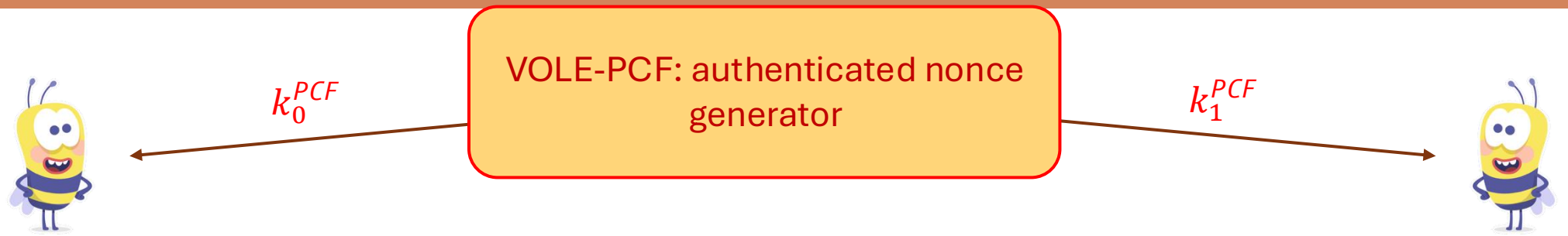
$(\Delta, \mathbf{w}) \in \mathbb{F}_p \times \mathbb{F}_p^n$

$\mathbf{w} = \Delta\mathbf{u} + \mathbf{v}$

$KeyGen(\Delta) \rightarrow \left(k_0^{PCF}, k_1^{PCF}\right)$

$$Eval(k_b, b, x) \rightarrow \begin{cases} (u_x, v_x) \in \mathbb{F}_p : \mathrm{b} = 0 \\ w_x \in \mathbb{F}_p : \mathrm{b} = 1 \end{cases}$$

2-Party Schnorr Signing



2-Party Schnorr Signing

$sk = sk_0 + sk_1$



k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$R \leftarrow R_0 + R_1$

$e \leftarrow H(pk, R, m)$

$s_0 \leftarrow e \cdot sk_0 + r_0$

(R_0, V_0, s_0)

$w_0 \leftarrow Eval(k_1^{Eval}, 1, m)$

check $V_0 + w_0 \cdot G = \Delta \cdot R_0$

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

2-Party Schnorr Signing

$sk = sk_0 + sk_1$



k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$R \leftarrow R_0 + R_1$

$e \leftarrow H(pk, R, m)$

$s_0 \leftarrow e \cdot sk_0 + r_0$

(R_0, V_0, s_0)

By correctness

$w_0 \leftarrow Eval(k_1^{E}$ $1, m)$

check $V_0 + w_0 \cdot G = \Delta \cdot R_0$

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

2-Party Schnorr Signing

$sk = sk_0 + sk_1$



k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$R \leftarrow R_0 + R_1$

$e \leftarrow H(pk, R, m)$

$s_0 \leftarrow e \cdot sk_0 + r_0$

(R_0, V_0, s_0)

By correctness

$w_0 \leftarrow Eval(k_1^{E}$ $1, m)$

check $V_0 + w_0 \cdot G = \Delta \cdot R_0$

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

Maliciously-chosen (r_0, v_0) will result on leakage of Δ

2-Party Schnorr Signing

$sk = sk_0 + sk_1$



k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$r_0 \approx x \xleftarrow{\$} \mathbb{F}_p^n$

By correctness

(R_0, V_0, s_0)

$w_0 \leftarrow Eval(k_1^{E}$ $1, m)$

check $V_0 + w_0 \cdot G = \Delta \cdot R_0$

Maliciously-chosen (r_0, v_0) will result on leakage of Δ

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

2-Party Schnorr Signing



$sk = sk_0 + sk_1$

k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$R \leftarrow R_0 + R_1$

$e \leftarrow H(pk, R, m)$

$s_0 \leftarrow e \cdot sk_0 + r_0$

$(R_0, s_0 \oplus H(V_0))$

$w_0 \leftarrow Eval(k_1^{\mathrm{PCF}}, 1, m)$

$V_0 = \Delta \cdot R_0 - w_0 \cdot G$

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

2-Party Schnorr Signing



$sk = sk_0 + sk_1$

k_0^{PCF}

$(k_0^{PCF}, k_1^{PCF}) \leftarrow KeyGen(\Delta)$

k_1^{PCF}



$R_1 = r_1 \cdot G$

$r_1 \xleftarrow{\$} \mathbb{F}_q$

$(r_0, v_0) \leftarrow Eval(k_0^{PCF}, 0, m)$

$R_0 \leftarrow r_0 \cdot G$

$V_0 \leftarrow v_0 \cdot G$

$R \leftarrow R_0 + R_1$

$e \leftarrow H(pk, R, m)$

$s_0 \leftarrow e \cdot sk_0 + r_0$

$(R_0, s_0 \oplus H(V_0))$

$w_0 \leftarrow Eval(k_1^{PCF}, 1, m)$

$V_0 = \Delta \cdot R_0 - w_0 \cdot G$

$e \leftarrow H(pk, R, m)$

$s_1 \leftarrow e \cdot sk_1 + r_1$

output $\sigma = (R_0 + R_1, s_0 + s_1)$

Reduce comm. with a "hashing trick"

Building PCF

Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

s : short, uniform seed



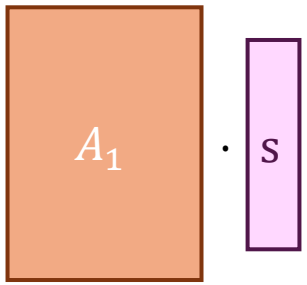
Building PCF

Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

A : generating matrix of a random linear code



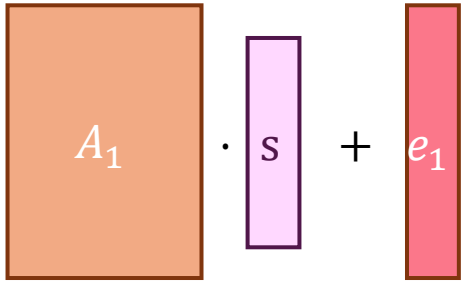
Building PCF

Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

e : sparse error vector

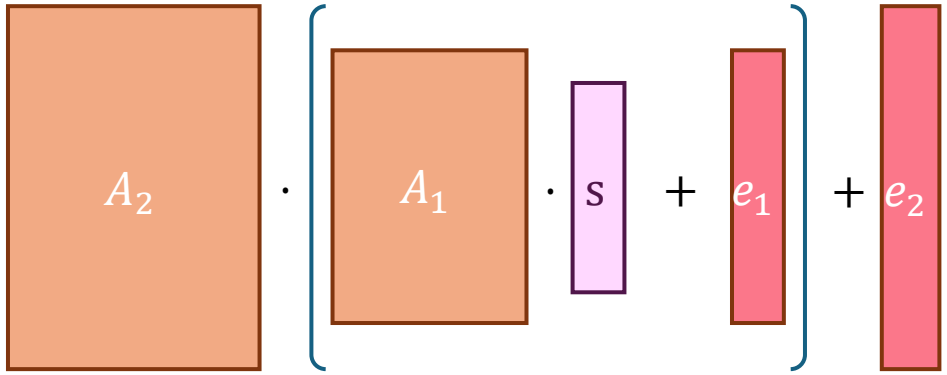


Building PCF

Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

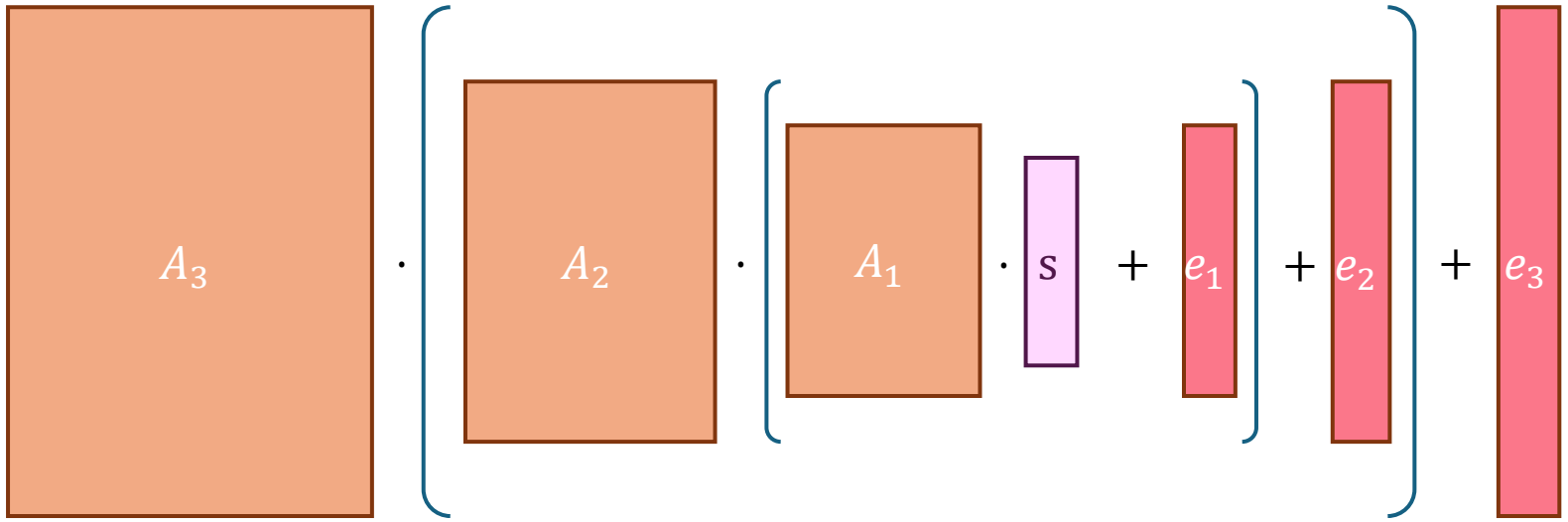


Building PCF

Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

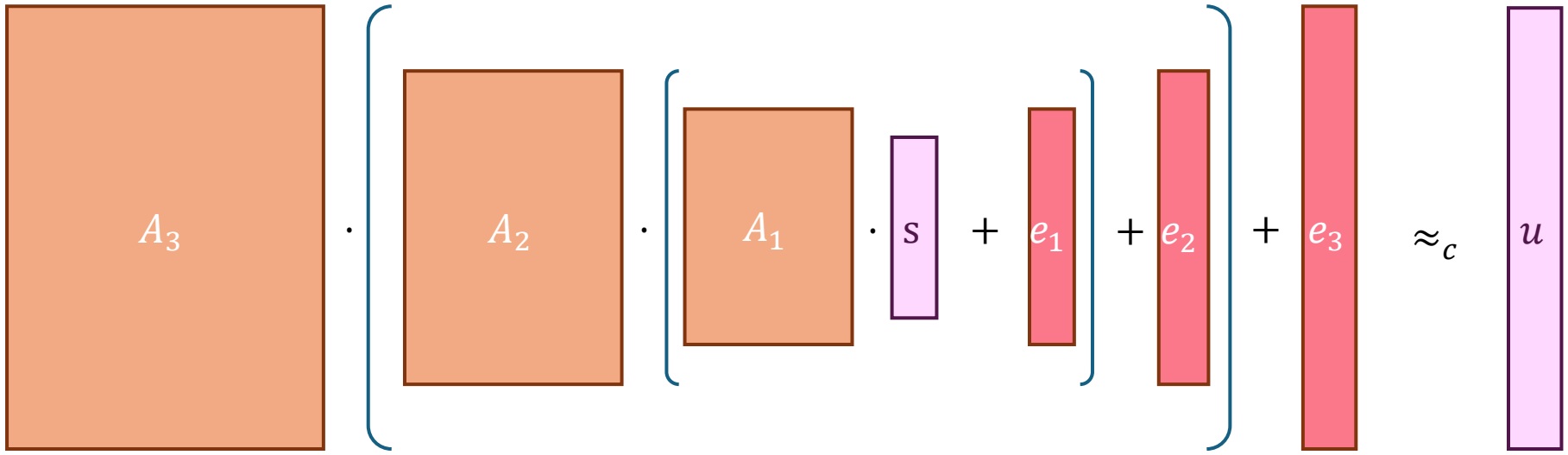


Building PCF

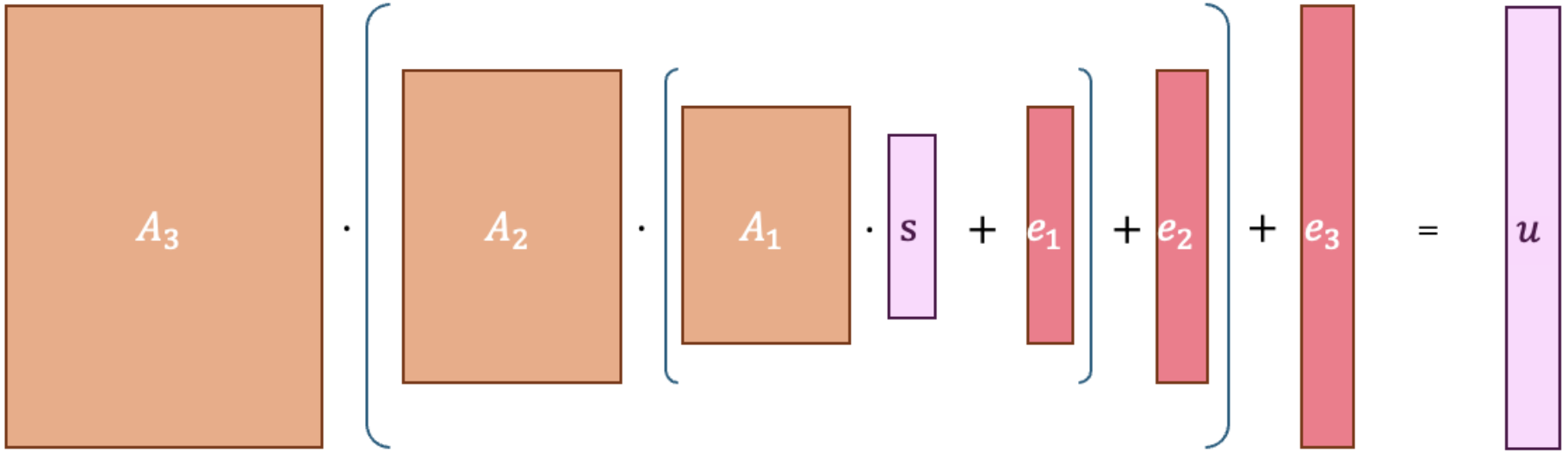
Purpose: generate shares of $\langle \Delta \cdot$ a long pseudorandom vector $\rangle$

Approach: sparse-LPN based PCF of [BCMRS'25]

Idea: use a short, uniform seed to generate a long, uniform vector

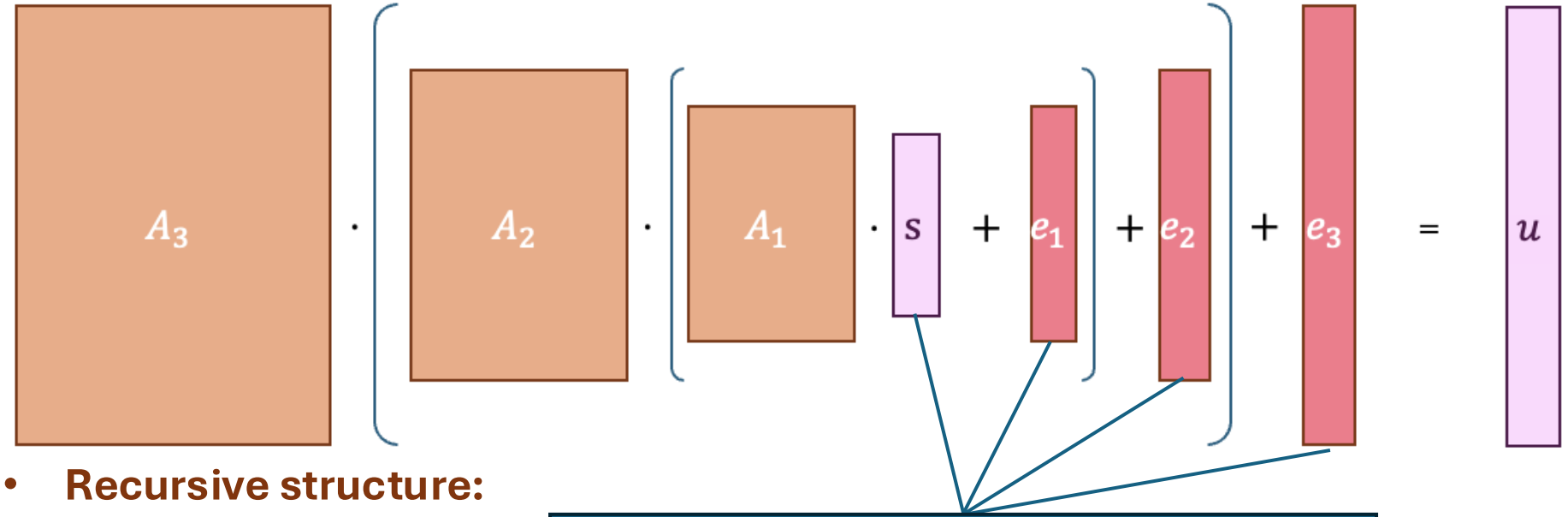


Building PCF



- **Recursive structure:**
  Each expansion can be up to a limited amount due to existing attacks

Building PCF



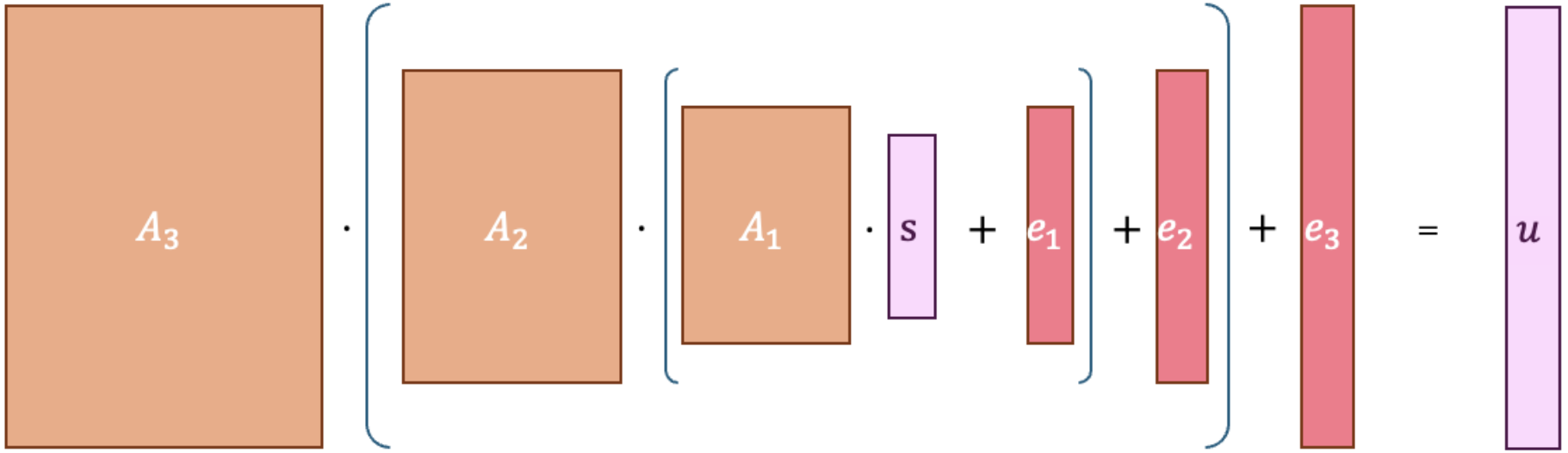
- **Recursive structure:**

  Each expansion ca ng attacks

- **Building VOLE-PCF:**

shares of $\Delta \cdot s$ and $\Delta \cdot e_i \equiv$ shares of $\Delta \cdot u$

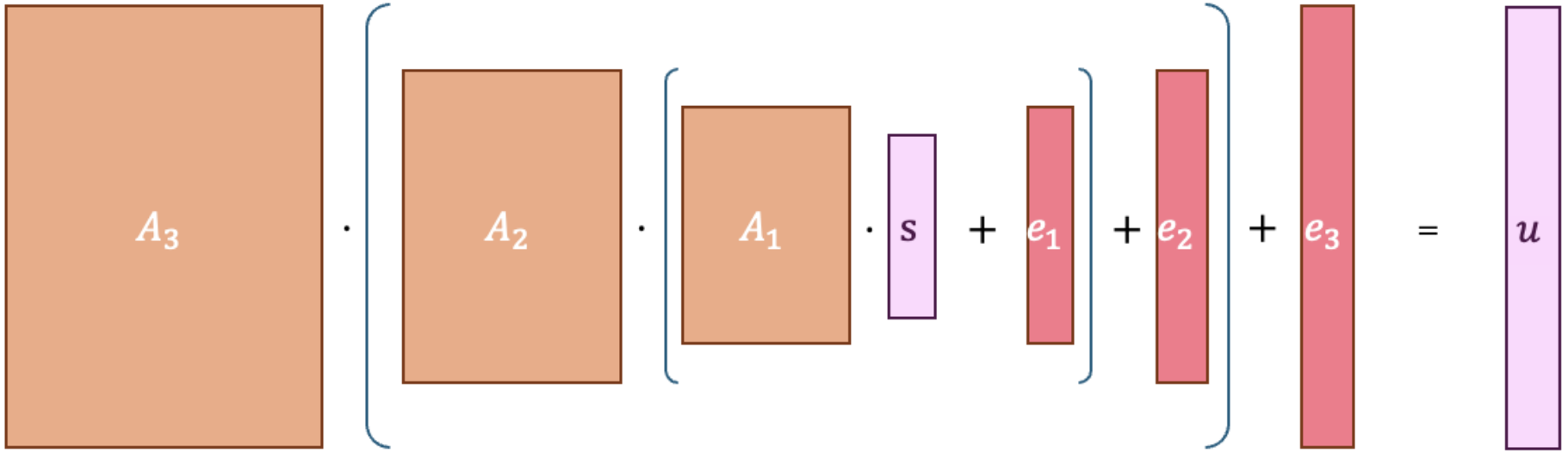
Building PCF



- **Recursive structure:**
  Each expansion can be up to a limited amount due to existing attacks

- **Building VOLE-PCF:**
  Storing $\Delta \cdot s$ in the keys and compressing shares of $\Delta \cdot e_i$ using a DPF

Building PCF



- **Recursive structure:**
  Each expansion can be up to a limited amount due to existing attacks

- **Building VOLE-PCF:**
  Storing $\Delta \cdot s$ in the keys and compressing shares of $\Delta \cdot e_i$ using a DPF

- **Sampling correlations on-the-fly:**
  Adaptively using random oracle in a tree-based fashion to generate rows of the products of these sparse matrices

TL;DL

- Two party threshold Schnorr/ECDSA signature:
  - Low communication: $96B$ for Schnorr, $128B$ for ECDSA
  - Fast signing: $\sim 1.10ms$ on one core of a laptop
  - Realizing UC signing functionalities
  - Stateless + deterministic: resilient against realistic real-world threats
  - $\sim 2s$ setup cost for ~ 12 million signing sessions on one core of a laptop

- All approaches apply to ECDSA with some adaptation:
  - There is an additional multiplicative component
    - Following a careful representation of the ECDSA signing, we can use VOLE-PCF approach in a similar manner

- Statelessness: using sparse-LPN, generation of matrices on-the-fly requires generating each row, requiring the maintenance of a counter.
  - Black-box reduction: starts from a stateful PCF, evaluates it many times, produces one output of a stateless PCF.

Thank You!

Questions?

team-stoats@listes.irif.fr