COUNT ME IN! EXTENDABILITY FOR THRESHOLD RING SIGNATURES

PKC '22



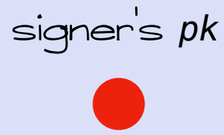


D. F. ARANHA, M. HALL-ANDERSEN, A. NITULESCU, E. PAGNIN, AND S. YAKOUBOV

Digital Signatures

SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`



Ring Signatures (RS)

SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

signer's *pk*

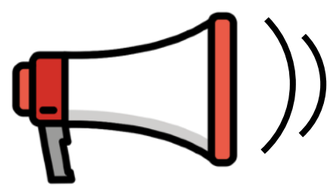
(msg, sgn)

RING SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

ring of *n* potential signers

(msg, sgn)



Threshold Ring Signatures (TRS)

SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

signer's *pk*

(msg, sgn)

RING SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

ring of *n* potential signers

(msg, sgn)

THRESHOLD RING SIGNATURE SCHEME

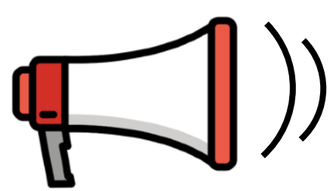
`Setup` `KeyGen` `Sign` `Verify`

t out of *n* signers

(msg, sgn)

`interaction among signers`

`agreement on the ring of public keys`



Extendable Threshold Ring Signatures (ExTRS)

SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

signer's *pk*

(msg, sgn)

RING SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

ring of *n* potential signers

(msg, sgn)

THRESHOLD RING SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

t out of *n* signers

(msg, sgn)

EXTENDABLE THRESHOLD RING SIGNATURE SCHEME

`Setup` `KeyGen` `Sign` `Verify`

t out of *n* signers

(msg, sgn1)

Extend

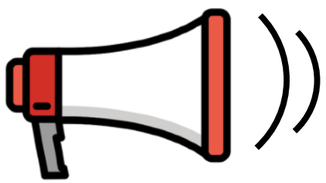
t out of ***n'*** signers

(msg, sgn')

Join

t+1 out of ***n'*** signers

(msg, sgn+)



count me in!

me too!

I endorse this!

`dynamic ring growth`
`no interaction among signers needed`

Our Contribution

✓ **Extendability** (enlarge the set of potential signers of a given signature)

Ring Signatures (RS)

Same-Message Linkable RS

Threshold Ring Signatures

✓ Formal **Syntax** & **Security** Models

✓ **Constructions** & **Implementations**

ERS from Signatures of Knowledge (SoK)

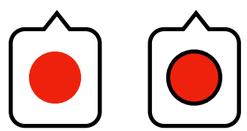
SMLERS black-box from ERS and SoK

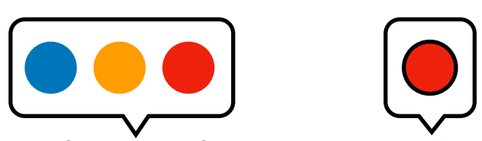
ETRS black-box from SMLERS and from dLog

Threshold Ring Signatures (Syntax)

`Setup(sec.par) → pp`

`KeyGen(pp) → (pk, sk)`





`Sign(m,{pk_i}_{i∈R}, sk) → s`

`Verify(t, m, s, {pk_i}_{i∈R}) → 0/1`

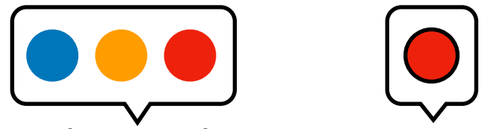
at least `t` secret keys were used to generate the signature `s` for `m`

Bresson, Stern, and Szydlo: “Threshold ring signatures and applications to ad-hoc groups”, CRYPTO, 2002.
Tsang, Wei, Chan, Au, Liu, Wong: “Separable linkable threshold ring signatures”, Indocrypt, 2004.
Melchor, Cayrel, Gaborit, Laguillaumie: “A new efficient threshold ring signature scheme based on coding theory”, IEEE-ToIT, 2011.
Munch-Hansen, Orlandi, Yakoubov: "Stronger Notions and a More Efficient Construction of Threshold Ring Signatures." 2020/678.

Extendable Threshold Ring Signatures (Syntax)

`Setup(sec.par) → pp`

`KeyGen(pp) → (pk, sk)`



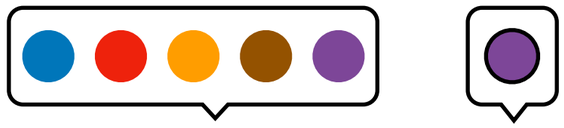
`Sign(m,{pk_i}_{i∈R}, sk) → s`

`Verify(t, m, s, {pk_i}_{i∈R}) → 0/1`

at least `t` secret keys were used to generate the signature `s` for `m`

`Extend(m, s, {pk_i}_{i∈R1}, {pk_i}_{i∈R2}) → s'`

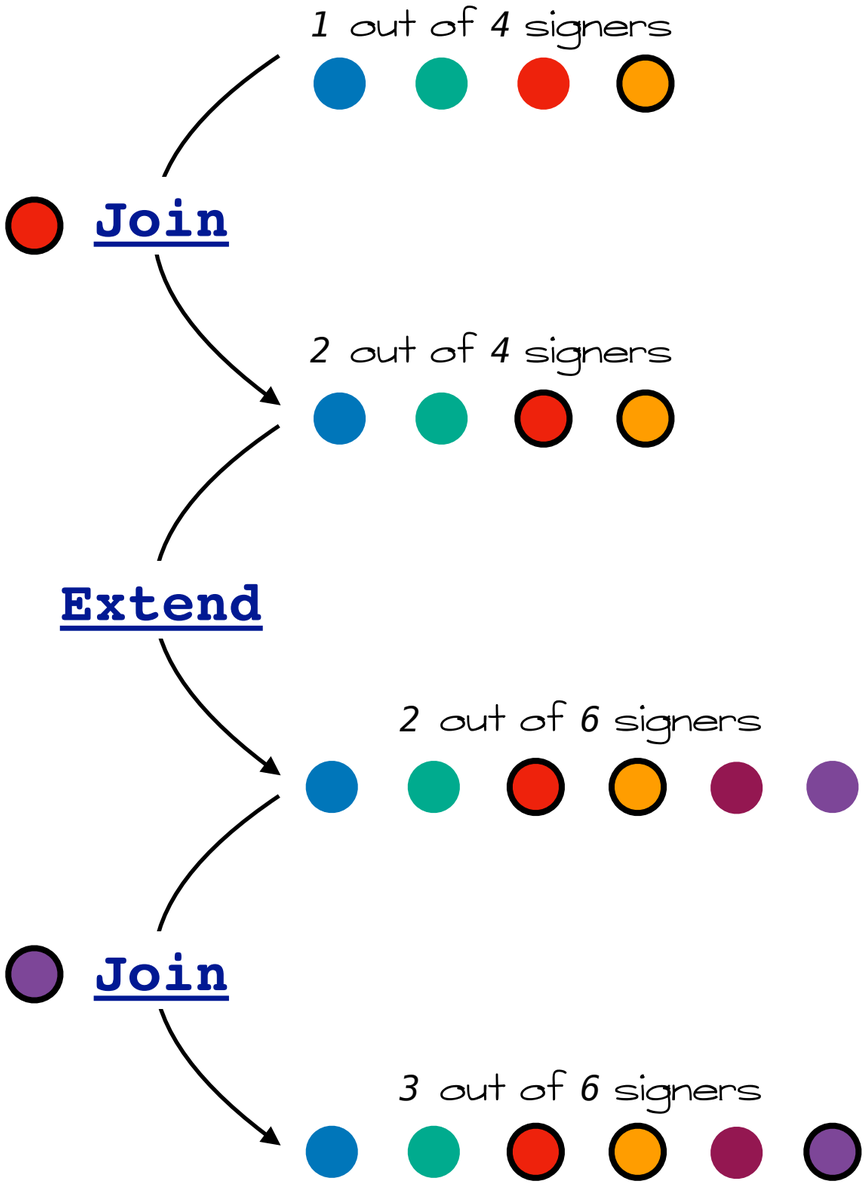
the new signature `s'` verifies for the same threshold `t` as `s` did and for the larger ring

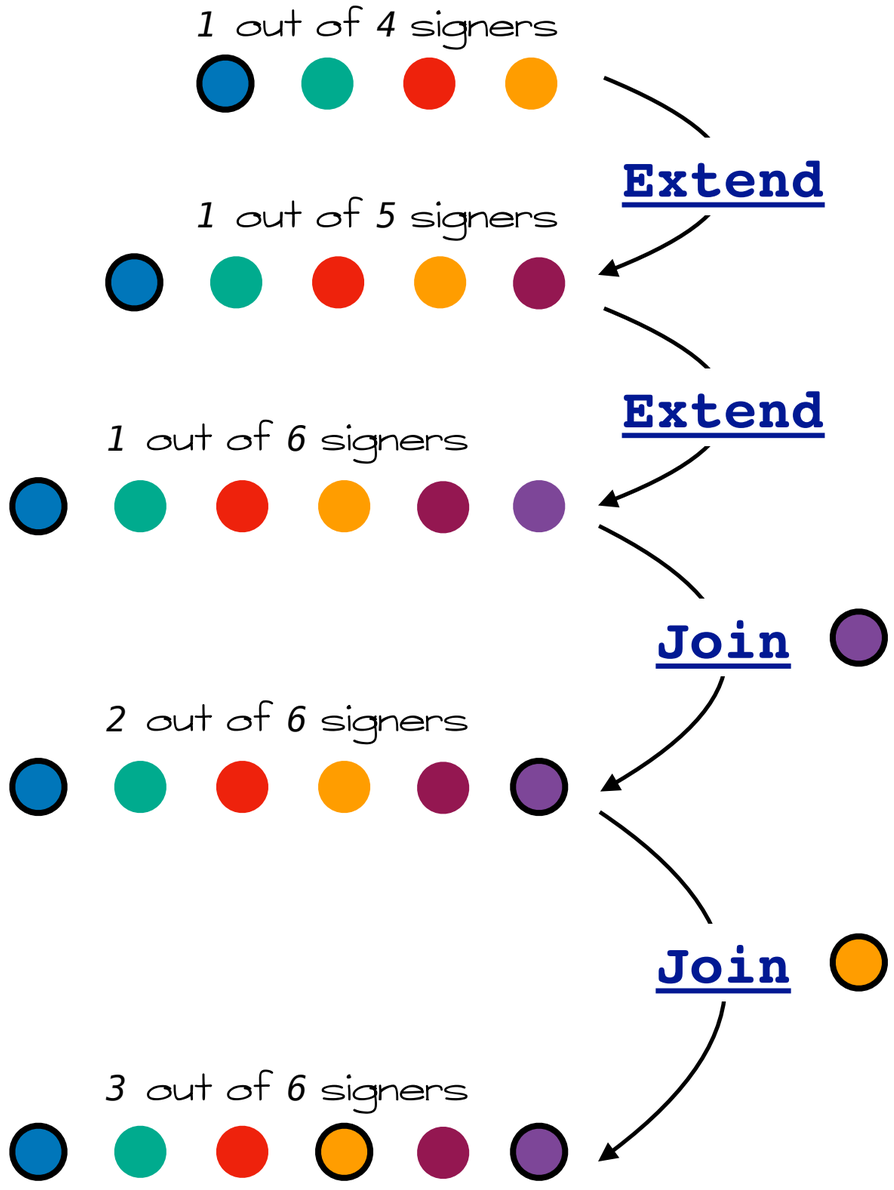


`Join(m, s, {pk_i}_{i∈R}, sk) → s'`

the new signature `s'` verifies for threshold `t_s + 1`

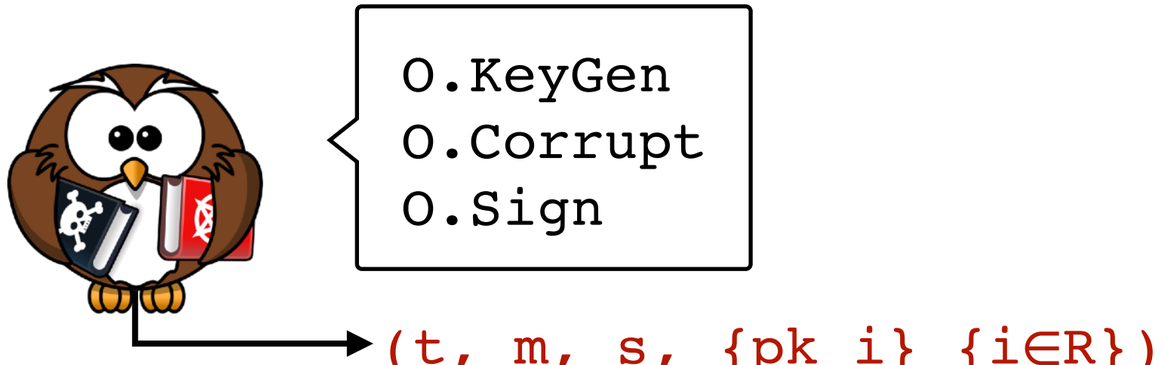
Visualizing Extendibility (Ladders)





ETRS: Security Model

UNFORGEABILITY



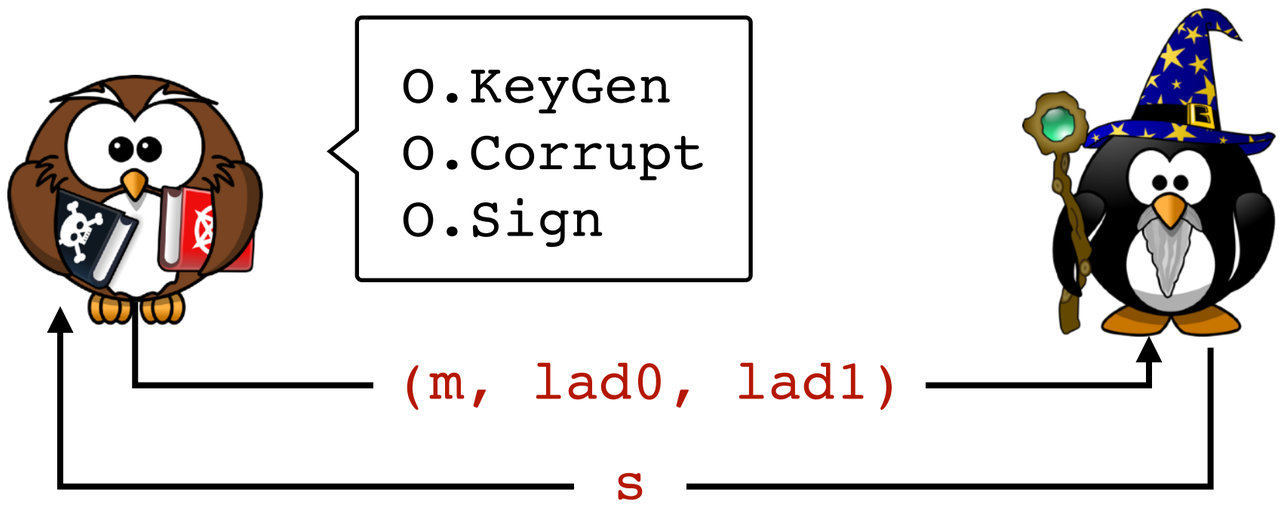
$\mathscr{A}$ wins the EUF game if:

1- `Verify(t,m,s,R) = 1`

2- # corrupted signers in R < t

3- # `O.Sign` queries for m < t

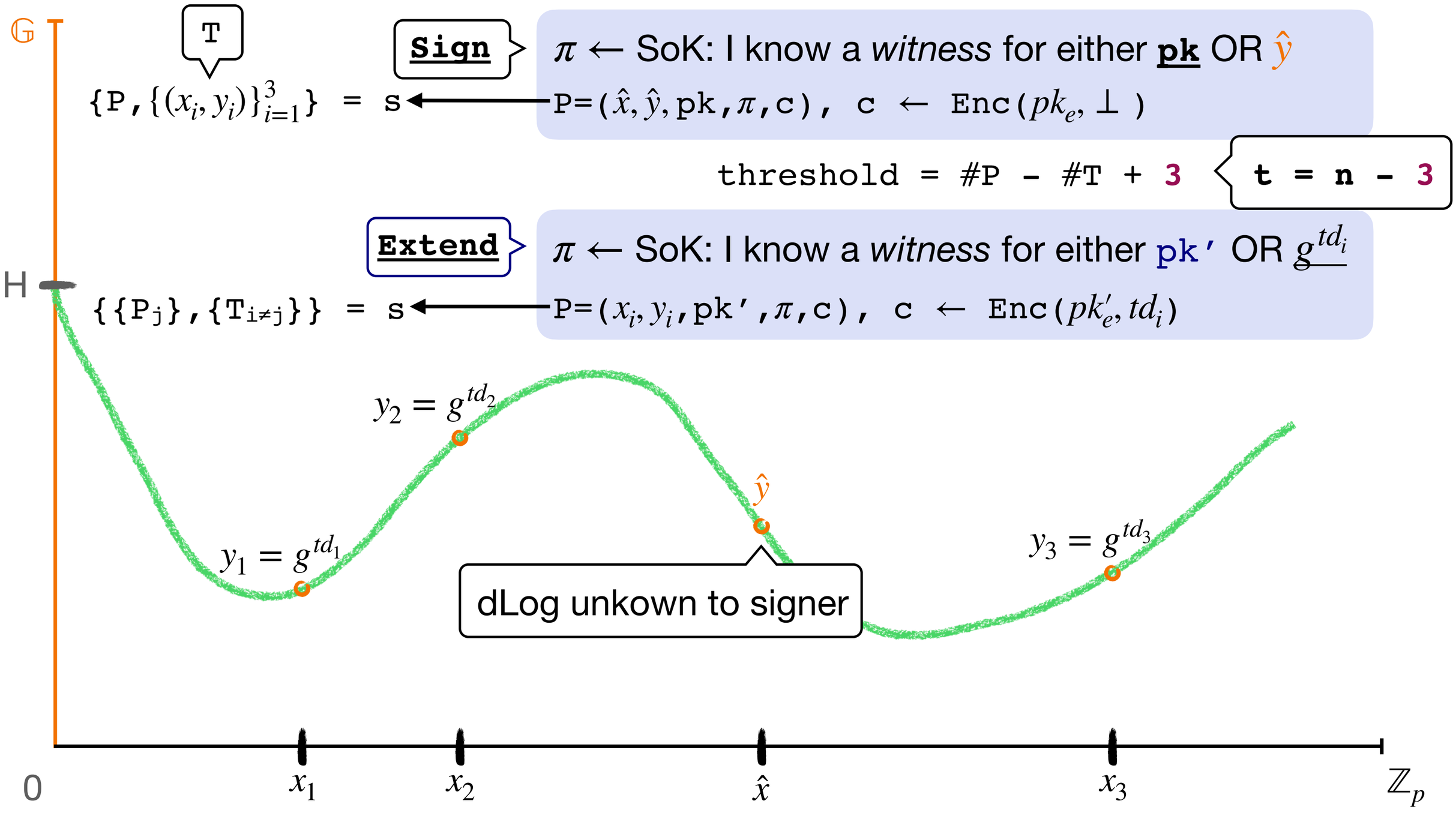
ANONYMITY & ANONYMOUS EXTENDABILITY



$\mathscr{A}$ wins if it correctly guesses what ladder has been chosen to generate s

Our Construction: ETRS From dLog (and PKE)

interpolation in the exponent: polynomial of degree **3** on $\{H, g^{td_1}, g^{td_2}, g^{td_3}\}$



Our Construction: ETRS From dLog (and PKE)

interpolation in the exponent: polynomial of degree **3** on $\{H, g^{td_1}, g^{td_2}, g^{td_3}\}$

Extend

$\pi \leftarrow$ SoK: I know a *witness* for either `pk'` OR g^{td_i}

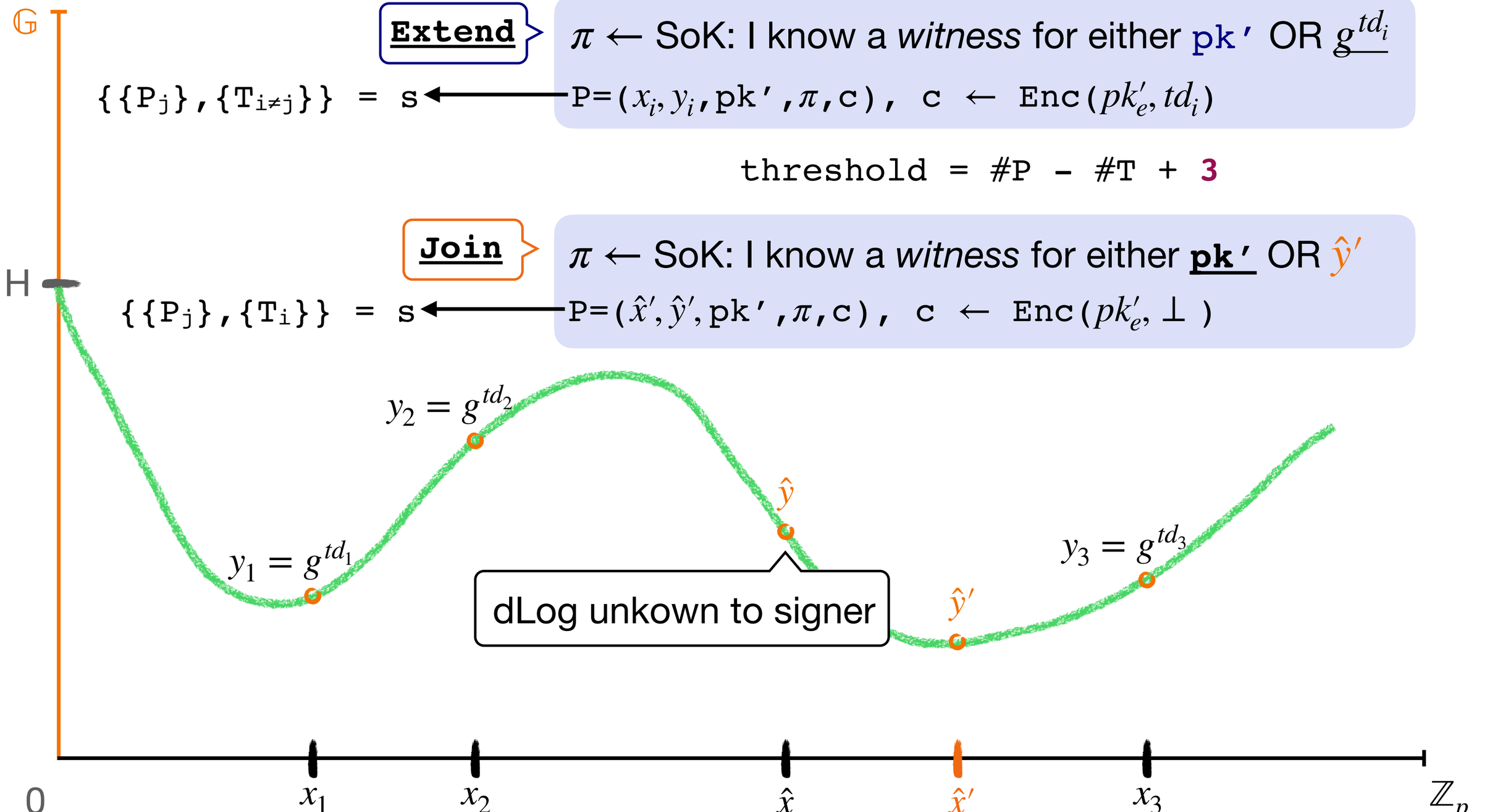
`{{Pⱼ},{Tᵢ≠ⱼ}} = s` ⟵ `P=(`x_i, y_i`,pk',`π`,c), c ←` $Enc(pk'_e, td_i)$

`threshold = #P - #T + 3`

Join

$\pi \leftarrow$ SoK: I know a *witness* for either **`pk'`** OR $\hat{y}'$

`{{Pⱼ},{Tᵢ}} = s` ⟵ `P=(`$\hat{x}', \hat{y}'$`,pk',`π`,c), c ←` $Enc(pk'_e, \perp)$



Our Construction

$$\mathscr{R}_{\mathbb{G}} = \{(\phi, w) = (h, \mathtt{pk}, x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p : \\ : g^x = h \vee g^x = \mathtt{pk}\}$$

KeyGen() $\mapsto$ $(\mathtt{pk}, \mathtt{sk})$

$(\mathtt{pk}_s, \mathtt{sk}_s) \leftarrow \mathbf{ERS}.\mathsf{KeyGen}()$
$(\mathtt{pk}_e, \mathtt{sk}_e) \leftarrow \mathbf{PKE}.\mathsf{KeyGen}()$
return $(\mathtt{pk} = (\mathtt{pk}_s, \mathtt{pk}_e), \mathtt{sk} = (\mathtt{sk}_s, \mathtt{sk}_e))$

Sign$(\mu, \mathtt{sk}) \mapsto \sigma$

$X \leftarrow_R \binom{\mathbb{Z}_p^*}{n'}$ // pick n' distinct evaluation points
$T := \varnothing; \quad P := \varnothing$
for $x \in X$
  $\mathtt{td} \leftarrow_R \mathbb{Z}_p$ // generate trapdoors for poly. values
  $T \leftarrow T \cup \{(x, \mathtt{td})\}$ // populate trapdoor set
$c \leftarrow \mathsf{Enc}(\mathtt{pk}_e, \perp)$ // no info to pass on
$\hat{x} \leftarrow_R \mathbb{Z}_p^* \setminus X$ // pick a new evaluation point
$(y, \pi) \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \mathtt{sk}, \mathtt{pk}, \mu)$
$P := \{(\hat{x}, y, \mathtt{pk}_s, \pi, c)\}$
return $\sigma := (T, P)$

Join$(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}}, \mathtt{sk}, \sigma) \mapsto \sigma'$

// check if current signer's $\mathtt{pk}_s$ is in P
if $\exists\ (x, y, \mathtt{pk}, \pi, c) \in P$ *s.t.* $\mathtt{pk} = \mathtt{pk}_s$
// remove simulated proof for the signer who wants to join
  $P \leftarrow P \setminus \{(x, y, \mathtt{pk}_s, \pi, c)\}$
// retrieve trapdoor value
  $\mathtt{td} \leftarrow \mathsf{Dec}(\mathtt{sk}_e, c)$
// add eval. point and td to the set of available trapdoors
  $T \leftarrow T \cup \{(x, \mathtt{td})\}$
$c' \leftarrow \mathsf{Enc}(\mathtt{pk}_e, \perp)$ // no info to pass on
$\hat{x} \leftarrow_R \mathbb{Z}_p^* \setminus X$ // pick a new evaluation point
// interpolate a unique representation of the polynomial
$(y', \pi') \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \mathtt{sk}, \mathtt{pk}, \mu)$
$P \leftarrow P \cup \{(\hat{x}, y', \mathtt{pk}_s, \pi', c')\}$
Randomly permute P
return $\sigma := (T, P)$

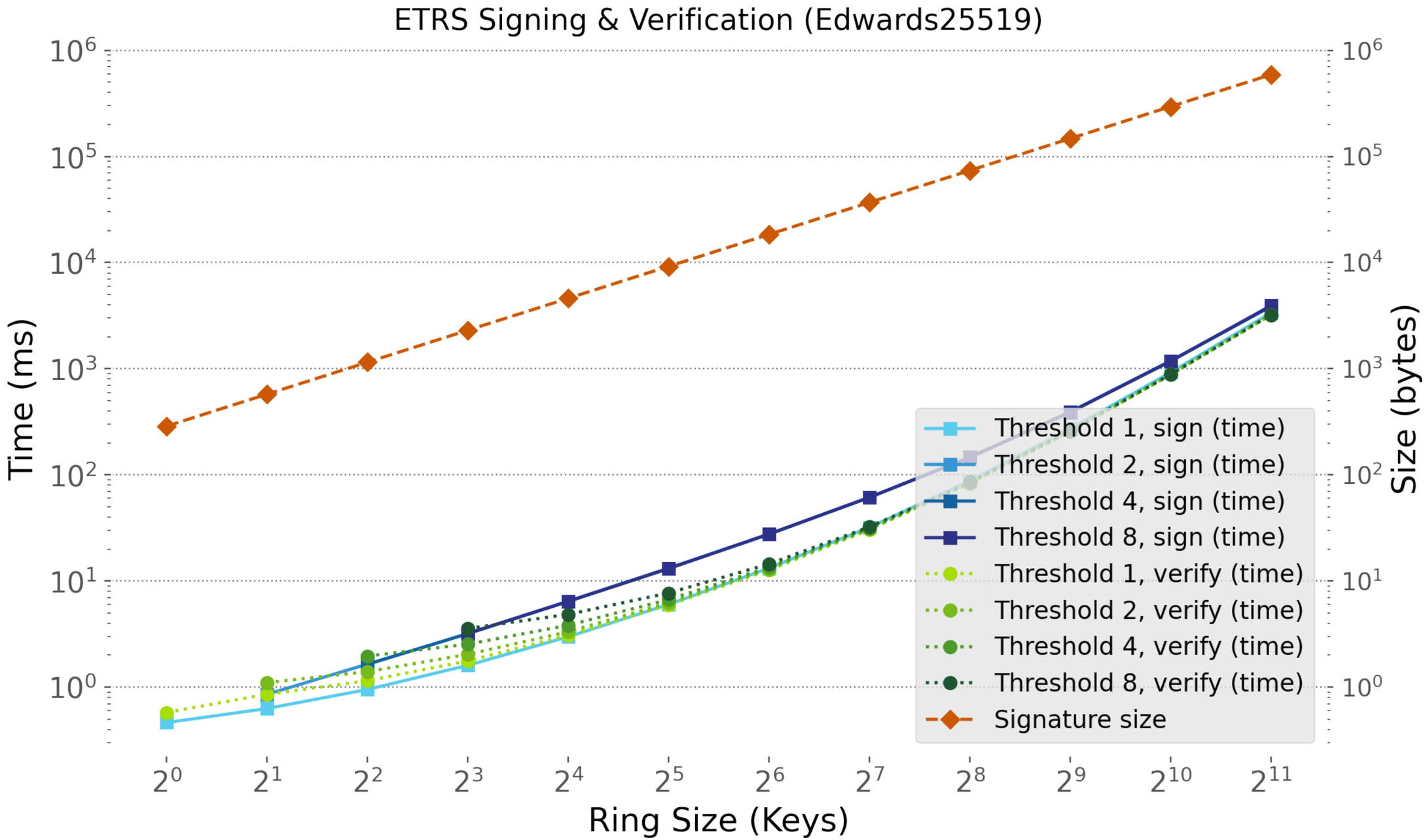
Extend$(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}}, \sigma, \mathtt{pk}) \mapsto \sigma'$

if $\mathtt{pk} \in \{\mathtt{pk}_j\}_{j \in \mathcal{R}}$: **return** $\perp$
$(\hat{x}, \hat{\mathtt{td}}) \leftarrow_R T$ // Pick eval-point and trapdoor
$c' \leftarrow \mathsf{Enc}(\mathtt{pk}_e, \hat{\mathtt{td}})$ // enable future endorsing
// interpolate a unique representation of the polynomial
$(y', \pi') \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \hat{x}, \mathtt{pk}, \mu)$
$T \leftarrow T \setminus \{(\hat{x}, \hat{\mathtt{td}})\}$ // erase used trapdoor
// Add simulated signature to the set of proofs
$P \leftarrow P \cup \{(\hat{x}, y', \mathtt{pk}_s, \pi', c')\}$
Randomly permute P
return $\sigma' := (T, P)$

Verify$(t, \mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}}, \sigma) \mapsto \mathtt{accept}/\mathtt{reject}$

if $\{\mathtt{pk}_j\}_{j \in \mathcal{R}} \neq \{\mathtt{pk}_i\}_{(\cdot,\cdot,\mathtt{pk}_i,\cdot,\cdot) \in P}$:
  return `reject`
// check y's are consistent with a degree n' polynomial
$\mathcal{Z} := \{(0, H)\} \cup \{(x, g^{\mathtt{td}})\}_{(x,\mathtt{td}) \in T}$
$\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(x, y)\}_{(x,y,\mathtt{pk},c,\pi) \in P}$
Pick $\hat{\mathcal{Z}} \subseteq \mathcal{Z}$ *s.t.* $|\hat{\mathcal{Z}}| = n' + 1$
$\mathcal{X} := \{x\}_{(x,y) \in \mathcal{Z}}; \hat{\mathcal{X}} := \{x\}_{(x,y) \in \hat{\mathcal{Z}}}$
for $(x, y) \in \mathcal{Z}$:
  if $y \neq \prod_{(\hat{x},\hat{y}) \in \hat{\mathcal{Z}}} \hat{y}^{L_{(\hat{\mathcal{X}},\hat{x})}(x)}$: **return** `reject`
// Interpolation over the standard set $\{1,\ldots,n'\}$
for $i \in [n']$: $V_i \leftarrow \prod_{(x,y) \in \mathcal{Z}} y^{L_{(\mathcal{X},x)}(i)}$
$\hat{\mu} := (\mu, \{V_i\}_{i \in [n']})$
for $(x, y, \mathtt{pk}_s, \pi, c) \in P$ // check proofs individually
  $\phi := (y, \mathtt{pk}_s)$
  if $\mathbf{SoK}.\mathsf{Verify}(\hat{\mu}, \mathscr{R}_{\mathbb{G}}, \phi, \pi) = \mathtt{reject}$
    return `reject`
if $|T| + |P| \geq t + n'$ **return** `accept`
else return `reject`

ETRS From dLog



THANK YOU FOR YOUR ATTENTION