UC4Free!

Existing Threshold Signatures are UC Secure

Full paper

Jan Bobolz, University of Edinburgh 🇬🇧
Elizabeth Crites, Parity Technologies 🇬🇧
Markulf Kohlweiss, University of Edinburgh 🇬🇧 and Input|Output 🌍
Akira Takahashi, JPMorgan AI Research & AlgoCRYPT CoE 🇺🇸

? What **compositional** guarantees do **existing** threshold signature schemes satisfy?

! Existing **game-based** constructions are **UC-secure**

Why do we care?

- 🧮 **NIST call** accepts both game-based proofs and UC proofs.
  - What should we prefer?
    Are game-based proofs **less composable?** No!
- 🔑 Understand security of existing constructions better.
- 📣 Message to tSig community: game-based is **fine**.

Why do we care?

Scheme	UF Level	Adaptive	Assumptions	$\mathsf{rnd}_{\mathrm{sign}}$	$\mathsf{rnd}_{\mathrm{com}}$
Threshold $\mathrm{BLS}_{\mathrm{I}}$ [Bol03]	UF-0		GDH	1	1
[BTZ22, BCK+22]	UF-1		t_{correct}-VCDH		
Threshold $\mathrm{BLS}_{\mathrm{II,III}}$ [DR24]	UF-0	✓	DDH, Co-CDH	1	1
FROST [KG20]	UF-0	–	–	2	2
[BTZ22, BCK+22]	SUF-3		OMDL		
[CKK+25]	UF-0	✓ (1/2)	AOMDL		
FROST2 [CKM21]	UF-0		OMDL	2	2
[BTZ22, BCK+22]	SUF-2		OMDL		
[CKK+25]	UF-0	✓ (1/2)	AOMDL		
FROST3 [RRJ+22, CGRS23]	UF-0		OMDL	2	2
[CKK+25]	UF-0	✓ (1/2)	AOMDL		
FROST-Mask [Che25]	UF-0	✓	AOMDL	2	2
Sparkle+ [CKM23]	UF-0		DL	3	2
[CKM23]	UF-0	✓ (1/2)	AOMDL		
[CFOS25]	UF-0		CDL		
Twinkle [BLT+24]	UF-0	✓	AOMCDH	3	1
[BLT+24]	UF-0	✓	DDH	3	1
Gargos [BDLR25a]	UF-0	✓	DDH	3	1
Snap [KRT24]	UF-1	✓	DL	4	2
Crackle [KRT24]	UF-1	✓	DL	5	3
Glacius [BDLR25b]	UF-0	✓	DL	5	1
Finally [BCdP+25]	UF-0		MLWE, MSIS	3	1
Threshold ML-DSA [CdPE+26]	UF-0		ML-DSA, MLWE	3	1
T-Raccoon [DKM+24]	UF-0		MLWE, MSIS	3	1
T-Raccoon-KRT [KRT24]	UF-1		MLWE, MSIS	3	1
T-Raccoon-KRT-4-rnd	UF-1	✓	MLWE, MSIS	4	1
T-Raccoon-KRT-5-rnd	UF-1	✓	MLWE, MSIS	5	2
T-Raccoon-IA [PKN+25]	UF-1		MLWE, MSIS	3	1

All UC secure!

Games vs UC

The game-based ideology

No ppt can provoke error event

$\underline{\mathcal{O}.\texttt{Sign}(m)}$

$\text{SGN} := \text{SGN} \cup \{m\}$

return $\text{Sign}(sk, m)$

$\underline{\texttt{Game}_{\mathcal{A}}(\lambda)}$

$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda, pk)$

return 1 if $\text{Verify}(pk, m, \sigma)$ **and** $m \notin \text{SGN}$

📒 *Secure* if for all ppt $\mathcal{A}$, $\Pr[\texttt{Game}_{\mathcal{A}}(\lambda) = 1]$ is negligible in λ

(verification consistency & responsiveness sacrificed for simplicity)

The UC ideology: Ideal world

No forgeries in the ideal world by definition

$\mathcal{P}^*.\texttt{KeyGen}()$
return $pk \leftarrow \mathcal{S} :: \texttt{KeyGen}(\mathcal{P})$

$\mathcal{P}^*.\texttt{Sign}(m)$
assert $\mathcal{P}^*.\texttt{KeyGen}$ **has been called**
$\text{SGN} := \text{SGN} \cup \{m\}$
$\sigma \leftarrow \mathcal{S} :: \texttt{Sign}(m)$
return σ

$\mathcal{P}.\texttt{Verify}(pk', m, \sigma)$ **for all** $\mathcal{P}$
if $pk' = pk$ **and** σ **was output by** $\texttt{Sign}(m)$
  return 1
if $pk' = pk$ **and** $m \notin \text{SGN}$ **then**
  return 0
$b \leftarrow \mathcal{S} :: \texttt{Verify}(pk', m, \sigma)$
return b

Correctness by definition

Unforgeability by definition

The UC ideology: Real world

No error events in the ideal world by definition

$\underline{\mathcal{P}^*.\texttt{KeyGen}()}$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
return pk

$\underline{\mathcal{P}^*.\texttt{Sign}(m)}$
assert $\mathcal{P}^*.\texttt{KeyGen}$ **has been called**
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
return σ

$\underline{\mathcal{P}.\texttt{Verify}(pk', m, \sigma)}$ **for all** $\mathcal{P}$
$b \leftarrow \mathsf{Verify}(pk', m, \sigma)$
return b

$\mathcal{P}^*.\texttt{KeyGen}()$
$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
return pk

$\mathcal{P}^*.\texttt{Sign}(m)$
assert $\mathcal{P}^*.\texttt{KeyGen}$ **has been called**
$\sigma \leftarrow \text{Sign}(sk, m)$
return σ

$\mathcal{P}.\texttt{Verify}(pk', m, \sigma)$ **for all** $\mathcal{P}$
$b \leftarrow \text{Verify}(pk', m, \sigma)$
return b

Calls for corrupt $\mathcal{P}$

😈 Adversary $\mathcal{A}$

Collusion

Calls for honest $\mathcal{P}$

Real world

Ideal world

📒 *Secure* if $\forall \mathcal{A}$, $\exists \mathcal{S}$ s.t. no ppt env can distinguish.

🌍Environment

Calls for honest $\mathcal{P}$

Collusion

$\mathcal{P}^*.\texttt{KeyGen}()$
return $pk \leftarrow \mathcal{S} :: \texttt{KeyGen}(\mathcal{P})$

$\mathcal{P}^*.\texttt{Sign}(m)$
assert $\mathcal{P}^*.\texttt{KeyGen}$ **has been called**
$\text{SGN} := \text{SGN} \cup \{m\}$
$\sigma \leftarrow \mathcal{S} :: \texttt{Sign}(m)$
return σ

$\mathcal{P}.\texttt{Verify}(pk', m, \sigma)$ **for all** $\mathcal{P}$
if σ **was output by** $\texttt{Sign}(m)$
  return 1
if $m \notin \text{SGN}$ **and** $pk' = pk$ **then**
  return 0
$b \leftarrow \mathcal{S} :: \texttt{Verify}(pk', m, \sigma)$
return b

🤖 Simulator $\mathcal{S}$

“Don’t care” queries,
Calls for corrupt $\mathcal{P}$

Signatures: Games vs UC

Composition

Game-based

- 🧩 **Larger protocols** can use KeyGen, Sign, Verify as **subroutine**
  - Proof: bound forgery bad event (reduction to signature game)

Game-based

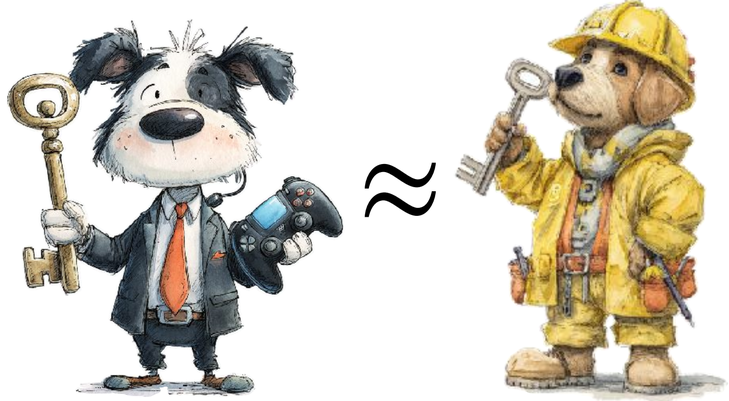
UC

- 🧩 **Larger protocols** use ideal functionality as **subroutine**
  - No forgeries! Super convenient.
  - UC security: ideal subroutine can be replaced by real signature scheme.

UC

Signatures: Games vs UC

Strength



Game-based ≈ UC

- 🥳 Well-known result for signature schemes [Can04]:
  UC-secure ⇔ (game-based) correct ∧ (game-based) unforgeable

- Proof (⇐): Simulator just runs scheme's KeyGen, Sign, Verify.
  Only way for environment to distinguish real/ideal:
  If real $\mathsf{Verify}(\ldots) = 1$ but ideal $\mathtt{Verify}$ outputs 0 (unlikely b/c unforgeability),
  or if real $\mathsf{Verify}(\ldots) = 0$ but ideal $\mathtt{Verify}$ outputs 1 (unlikely b/c correctness).

- Proof (⇒): From forger, build environment that calls $\mathtt{Verify}$ on forgery.
  Real says 1, ideal says 0.
  (similar for correctness)

Let’s do this for threshold signatures

Game-based
threshold-signature definition

≈

UC
ideal threshold-signature functionality

Challenges for UC threshold signatures

💡 Intuition: same [Can04] proof template should apply.

❓ What's the **right game**?

Many notions of "unforgeable"

❓ What's the **right ideal functionality**?

Many modeling decisions

The “right” game

Game-based threshold unforgeability

$\underline{\mathcal{O}.\mathtt{Sign}_k(\mathscr{P}, eid, T, m, ctx, \mathbf{pm}_{k-1})}$ for $k \in [rnd_{\text{sign}}]$

if $k = rnd_{\text{com}}$ **then**

$ssid := \text{ssid}(T, m, ctx, (\mathbf{pm}_\ell)_{\ell<k})$

$\text{SGN}[m, ssid] := \text{SGN}[m, ssid] \cup \{\mathscr{P}\}$

$(pm_k, st_{\mathscr{P},eid}) \leftarrow \text{Sign}_k(sk_{\mathscr{P}}, st_{\mathscr{P},eid}, T, m, ctx, \mathbf{pm}_{k-1})$

return pm_k

$\underline{\mathtt{Game}^{\text{UF}}_{\mathscr{A}}(\lambda)}$

$(pk, (sk_{\mathscr{P}})_{\mathscr{P}\in\mathbf{P}}) \leftarrow \text{KeyGen}(1^\lambda)$

$(m, \sigma) \leftarrow \mathscr{A}^{\mathcal{O}}(1^\lambda, pk)$

assert $\text{Verify}(pk, m, \sigma)$

return 1 if *not trivial forgery*

e.g., UF-2:

$\nexists ssid : \ |\text{SGN}[m, ssid] \cup \mathbf{C}| \geq t$

Game-based threshold unforgeability

$\underline{\mathtt{Game}^{\mathrm{UF}}_{\mathscr{A}}(\lambda)}$

$(pk, (sk_{\mathscr{P}})_{\mathscr{P}\in\mathbf{P}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$

$(m, \sigma) \leftarrow \mathscr{A}^{\mathcal{O}}(1^\lambda, pk)$

assert $\mathsf{Verify}(pk, m, \sigma)$

return 1 if ***not trivial forgery***

e.g., UF-2:

$\nexists ssid : \ |\mathrm{SGN}[m, ssid] \cup \mathbf{C}| \geq t$

$\underline{\mathtt{Game}^{\mathrm{SUF}}_{\mathscr{A}}(\lambda)}$

$(pk, (sk_{\mathscr{P}})_{\mathscr{P}\in\mathbf{P}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$

$(m, \sigma_1, \ldots, \sigma_n) \leftarrow \mathscr{A}^{\mathcal{O}}(1^\lambda, pk)$

assert $\sigma_i \neq \sigma_j$ **for all** $i \neq j$

assert $\mathsf{Verify}(pk, m, \sigma_i)$ **for all** i

return 1 if ***not trivial forgery***

e.g., SUF-2

$\left|\{ssid : \ |\mathrm{SGN}[m, ssid] \cup \mathbf{C}| \geq t\}\right| < n$

The “right” functionality

A straight translation from game to UC

(verification consistency & strong unforgeability & responsiveness sacrificed for simplicity)

F-TS3 (simplified)

$\mathscr{P}.\texttt{KeyGen}()$ **for all** $\mathscr{P} \in \mathbf{P}$

return $pk \leftarrow \mathcal{S} :: \texttt{KeyGen}(\mathscr{P})$

$\mathscr{P}.\texttt{Sign}_k(eid, T, m, ctx, \mathbf{pm}_{k-1})$ **for** $k \in [rnd_{\text{sign}}]$, $\mathscr{P} \in \mathbf{P}$

assert $\mathscr{P}.\texttt{KeyGen}$ **has been called**

if $k = rnd_{\text{com}}$ **then**

$ssid := \text{ssid}(T, m, ctx, (\mathbf{pm}_\ell)_{\ell<k})$

$\text{SGN}[m, ssid] := \text{SGN}[m, ssid] \cup \{\mathscr{P}\}$

$pm' \leftarrow \mathcal{S} :: \texttt{Sign}(\mathscr{P}, T, m, ctx, pm_{k-1})$

return pm'

$\mathscr{P}.\texttt{Combine}(pk', T, m, ctx, (\mathbf{pm_k})_{k=1}^n)$ **for all** $\mathscr{P}$

$\sigma \leftarrow \mathcal{S} :: \texttt{Combine}(pk', T, m, ctx, (\mathbf{pm_k})_{k=1}^n)$

return σ

$\mathscr{P}.\texttt{Verify}(pk', m, \sigma)$ **for all** $\mathscr{P}$

if σ **was output by** $\mathscr{P}'.\texttt{Combine}(\cdot, m, \cdot)$ **s.t.** ***correctness/robustness*** **is expected then**

return 1

if $pk' = pk$ **and** ***not trivial forgery*** **then**

return 0

$b \leftarrow \mathcal{S} :: \texttt{Verify}(pk', m, \sigma)$

return b

Proof of equivalence

Game-based Threshold Sigs

≈

UC Threshold Sigs

Theorem 2. *The protocol* Π-TS3$[\mathbf{P}, \mathbf{t}, \Sigma_{\mathsf{TS}}, \mathsf{cor}]$ *(uniform-)UC-realizes* $\mathcal{F}$-TS3$[\mathbf{P}, \mathbf{t}, \mathsf{lvl}, \mathbf{rnd}, \mathsf{ssid}, \mathsf{cor}]$ *in the* $\mathcal{F}$-Gen$[\mathbf{P}, \mathbf{t}, \Sigma_{\mathsf{TS}}::\mathsf{Setup}, \Sigma_{\mathsf{TS}}::\mathsf{KeyGen}]$ *hybrid model in the presence of* $\mathcal{G}$-Setup$[\Sigma_{\mathsf{TS}}::\mathsf{Setup}]$ *if and only if the threshold signature scheme* $\Sigma_{\mathsf{TS}} = (\mathbf{rnd}, \mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Sign}_k\}_{k=1}^{\mathsf{rnd}_{\mathsf{sign}}}, \mathsf{Verify}, \mathsf{ssid})$ *is* $(\mathsf{cor}, \mathbf{P}, \mathbf{t})$*-correct (Definition 1),* $(\mathsf{cor}, \mathbf{P}, \mathbf{t})$*-robust (Definition 2), and* $(\mathsf{cor}, \mathsf{lvl}, \mathbf{P}, \mathbf{t})$*-unforgeable (Definition 3).*

Proof: similar to [Can04] in spirit.
Simulator runs **normal algorithms honestly**. Environment **can only distinguish by** breaking correctness or robustness or unforgeability

🤨 Isn't UC *supposed to be stronger* than games?

Simulation-based F-Sign	Our F-TS3
F-Sign: MPC-compute the (plain signature scheme's) Sign function on secret-shared key	**F-TS3**: Full tSig API, but enforces unforgeability
🤖 Simulator is given σ, must force protocol to output σ	🤖 Simulator just executes round functions honestly
😈 Distribution of σ cannot be influenced by adversary	😈 Adversary can have benign influence
🔐 Overall security argument: DKG + F-Sign + Schnorr unforgeability = secure	🔐 Overall security argument: F-TS3 = secure

🤨 Isn’t UC *supposed to be stronger* than games?

- What about **rewinding**? Impossible in UC, no?
  - Indeed, **UC simulator cannot rewind**.
    But F-TS3 simulator doesn’t have to.
  - Logically: reduction to dlog can rewind the UC experiment *as a whole*

Why we aren’t happy yet

- ✅ Get composability guarantees
- ✅ Makes composition proofs easy: forgeries don’t exist
- ❌ Environment (calling protocol) must manage protocol messages
  - Calling protocols must implement lots of management pseudocode
    (though not more than for a game-based definition)
  - Subtly against UC spirit

F-TS1

Easier-to-use, more traditional functionality

(verification consistency & responsiveness sacrificed for simplicity)

F-TS1 (simplified)

$\underline{\mathscr{P}.\texttt{KeyGen}()}$ **for all** $\mathscr{P} \in \mathbf{P}$
return $pk \leftarrow \mathcal{S} :: \texttt{KeyGen}(\mathscr{P})$

$\underline{\mathscr{P}.\texttt{Sign}(T, m)}$ **for** $\mathscr{P} \in \mathbf{P}$
$\text{SGN}[m] := \text{SGN}[m] \cup \{\mathscr{P}\}$
$\sigma \leftarrow \mathcal{S} :: \texttt{Sign}(\mathscr{P}, T, m)$
return σ

$\underline{\mathscr{P}.\texttt{Verify}(pk', m, \sigma)}$ **for all** $\mathscr{P}$
if $pk' = pk$ **and** $|\text{SGN}[m] \cup \mathbf{C}| < t$ **then**
  return 0
$b \leftarrow \mathcal{S} :: \texttt{Verify}(pk', m, \sigma)$
return b

Instantiation

Theorem 4. *Let* $\mathsf{lvl} = \text{UF-1}$. *Let* $\mathsf{rnd}_{\text{com}} = \mathsf{rnd}_{\text{msg}} = \mathsf{rnd}_{\text{T}} = \mathsf{rnd}_{\text{ctx}} = 1$. *Let* $t_{\text{correct}} = t_{\text{robust}} = |\mathbf{P}|+1$ *(i.e., no correctness or robustness guarantees in* $\mathcal{F}$-TS3*). The protocol* Π-$\texttt{TS1}[\mathbf{P}, \mathbf{t}, \mathsf{cor}]$ *(uniform-)UC-realizes* $\mathcal{F}$-$\texttt{TS1}[\mathbf{P}, \mathbf{t}, \mathsf{cor}]$ *in the* $\mathcal{F}$-$\texttt{TS3}[\mathbf{P}, \mathbf{t}, \mathsf{lvl}, \mathbf{rnd}, \mathsf{ssid}, \mathsf{cor}]$ *and* $\mathcal{F}$-$\texttt{Net}$ *hybrid model.*

🍿 What else is in the paper?

	Game/F-TS3	F-TS2	F-TS1	F-TSSync2	F-TSSync1
Realized by game schemes	✅	✅	✅	✅	✅
Network	❌ None	✅ Untrusted	✅ Untrusted	✅ Sync	✅ Sync
Correctness Robustness	✅	❌	❌	✅	✅
Preprocessing	✅	✅	❌	✅	❌
Unforgeability levels	✅ all	✅ most	❌ UF-1	✅ most	❌ UF-1

🍿 What else is in the paper?

- 🧠 Functionalities are **responsive**, support **strong UF**, verification **consistency**
- 🛜 **Simplified** (F-Sig) responsiveness via default algorithms (alternative to aspects of [CDLLR25])
- 📏 Generalized [BCKMTZ22] **hierarchy**
  - More flexibility around consensus ⇒ **BLS is strongly unforgeable**!
  - **Strong unforgeability** via counting (more general)
- 💪 Game-based definition of **robustness**

🏁 Summary

Full paper

- 🦄 5 threshold signature **ideal functionalities**
  - Pick one for your use-case
- 🛠️ Protocols to instantiate them from **existing (game-based) constructions**
- 🧮 **Modeling improvements** for both UC and games