Tightly Secure Chameleon Hash Functions in the Multi-User Setting and Their Applications

Xiangyu Liu, Shengli Liu, and Dawu Gu

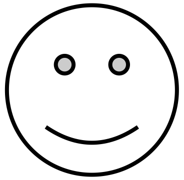
Shanghai Jiao Tong University

China

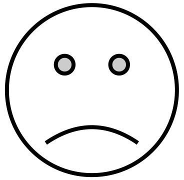
Chameleon Hash Functions (CHF)

$$(\mathsf{hk}, td) \leftarrow \mathsf{KGen}$$

$$\boxed{m}\boxed{r} \xrightarrow{H_{\mathsf{hk}}(\cdot)} \boxed{h}$$



have td: **easy** to find collision



no td: **hard** to find collision

A Collision:

find $(m_1, r_1) \neq (m_2, r_2)$ s.t.

$$H_{\mathsf{hk}}(m_1, r_1) = H_{\mathsf{hk}}(m_2, r_2)$$

Properties of CHF

For random (hk, td) from KGen:

- **Collision Resistance:** hard to find (m_1, r_1, m_2, r_2) s.t. $m_1 \neq m_2$ and $H_{\mathsf{hk}}(m_1, r_1) = H_{\mathsf{hk}}(m_2, r_2)$.
- **Strong Collision Resistance:** hard to find (m_1, r_1, m_2, r_2) s.t. $(m_1, r_1) \neq (m_2, r_2)$ and $H_{\mathsf{hk}}(m_1, r_1) = H_{\mathsf{hk}}(m_2, r_2)$.

- **Random Trapdoor Collision (RTC):** if r_1 is chosen uniformly at random, then r_2 (the output of $\mathsf{TdColl}(\cdot)$) enjoys a uniform distribution.

Existing CHFs

- the claw–free premutation
- the factoring assumption by Shamir and Tauman
- the RSA[n,n] assumption
- the very smooth hash

Sigma protocols

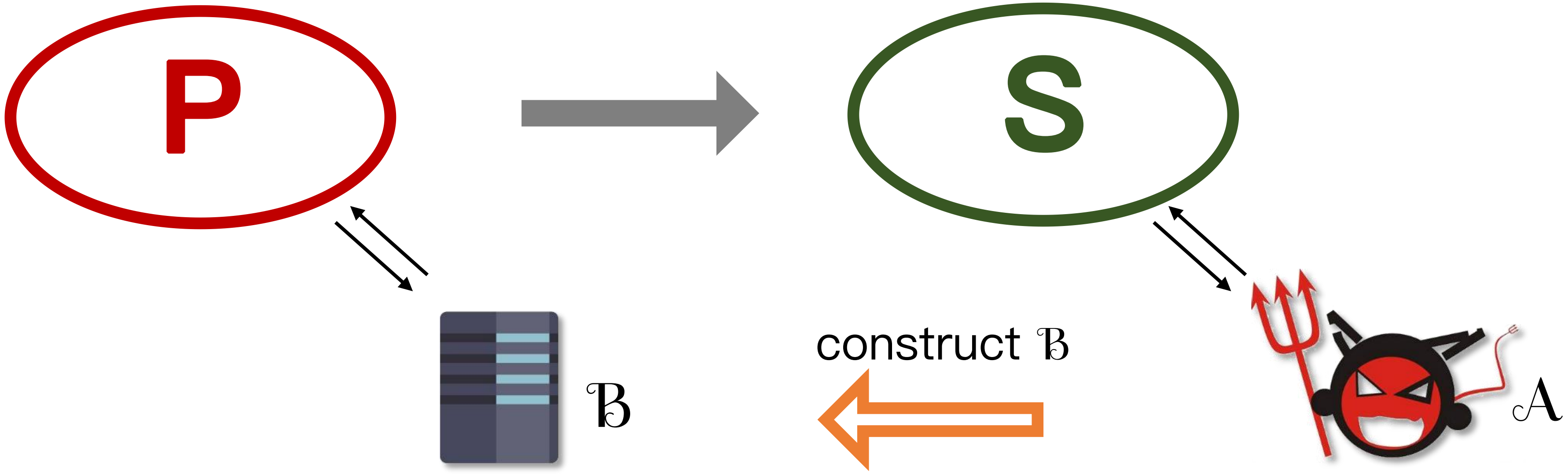
- the Micali–Shamir protocol
- the Okamoto protocol
- the HS identification protocol

- CHF_{claw}
- CHF_{st}
- CHF_{rsa-n}
- CHF_{vsh}
- CHF_{ms}
- CHF_{oka}
- $\mathrm{CHF}_{\mathrm{hs}}$

Bellare and Ristov [BR14] proved that CHFs and Sigma protocols are equivalent.

Provable Security

We construct a cryptographic scheme **S** based on the problem **P**.

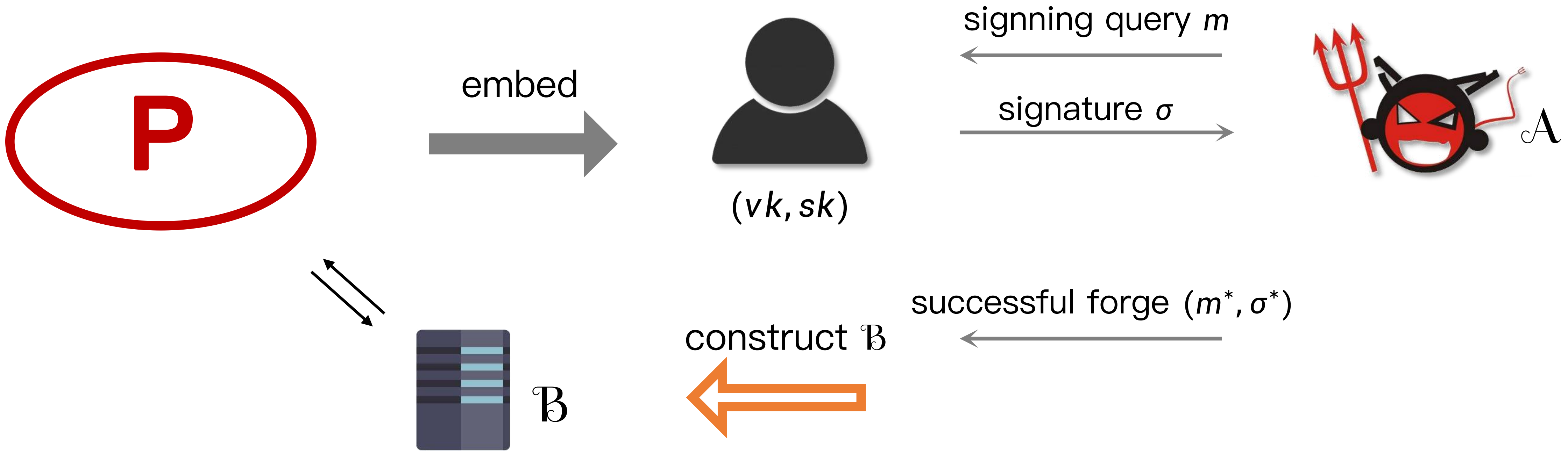


PPT algorithm $\mathcal{B}$ successfully solves **P**
(with probability ϵ')

PPT adversary $\mathcal{A}$ successfully attacks
(with probability ϵ)

Security loss: $L = \frac{\epsilon}{\epsilon'}$

Signature in the Single User Setting

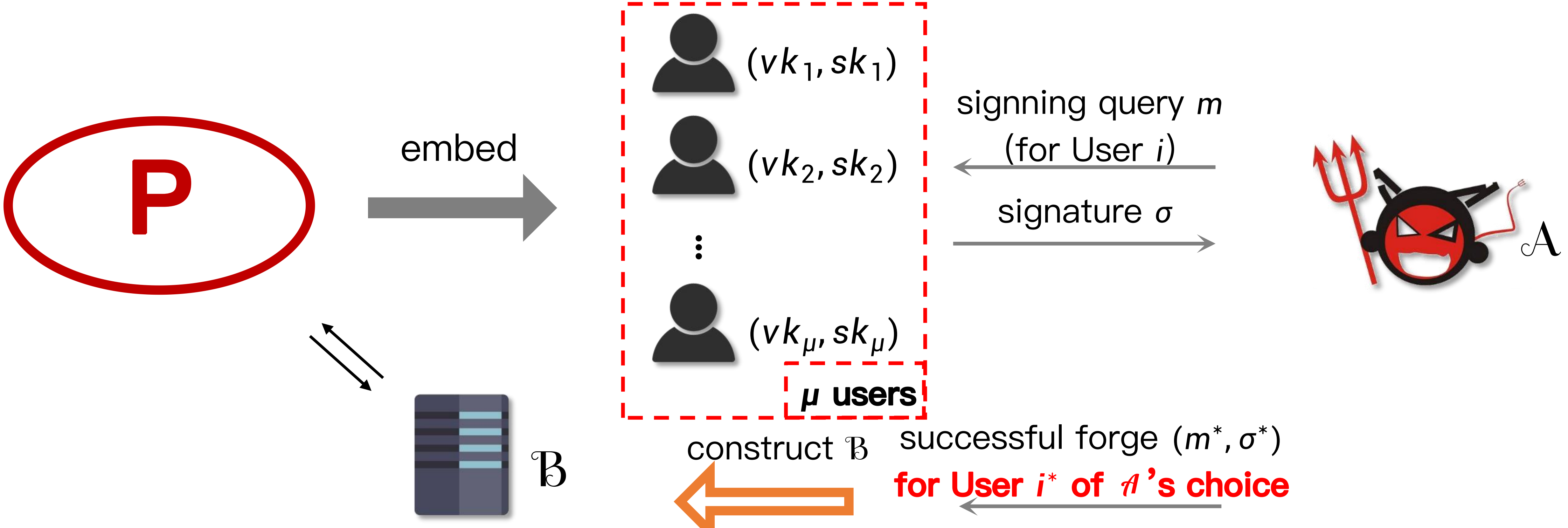


PPT algorithm $\mathcal{B}$ successfully solves **P** (with probability ϵ')

PPT adversary $\mathcal{A}$ successfully attacks (with probability ϵ)

Security loss: $L = \frac{\epsilon}{\epsilon'}$

Signature in the Multi-User Setting



PPT algorithm $\mathcal{B}$ successfully solves **P** (with probability ϵ')

PPT adversary $\mathcal{A}$ successfully attacks (with probability ϵ)

Tight Security: constant security loss
Loose Security: security loss depends on μ

Hybrid Argument:
security loss at least μ

Advantages of Tight Security

- μ (the total number of users) can be as large as 2^{30}!

To achieve **the same security level**, tightly secure schemes have:

- **Smaller elements**
- **Lower bandwidth**
- **Faster computations**

YES!!

Security of CHF in the Multi-User Setting

For μ random pairs $(\mathsf{hk}_i, td_i)_{i\in\mu}$ from KGen:

- Multi-User Collision Resistance: hard to find $i^* \in [\mu]$ and (m_1, r_1, m_2, r_2) s.t. $m_1 \neq m_2$ and $H_{\mathsf{hk}_{i^*}}(m_1, r_1) = H_{\mathsf{hk}_{i^*}}(m_2, r_2)$.
- Strong Multi-User Collision Resistance: hard to find $i^* \in [\mu]$ and (m_1, r_1, m_2, r_2) s.t. $(m_1, r_1) \neq (m_2, r_2)$ and $H_{\mathsf{hk}_{i^*}}(m_1, r_1) = H_{\mathsf{hk}_{i^*}}(m_2, r_2)$.

— Not all constructions achieve tight security in the multi-user setting!!

Hard to achieve tight security

- the claw-free premutation
- the factoring assumption by Shamir and Tauman
- the RSA[n,n] assumption
- the very smooth hash
- the Micali-Shamir protocol

- CHF_{claw}
- CHF_{st}
- CHF_{rsa-n}
- CHF_{vsh}
- CHF_{ms}

Achieving Tight Security

CHF_{dl} the Discrete Logarithm assumption

CHF_{rsa} the RSA assumption

Random self-reducibility

CHF_{fac} the factoring assumption Embed the factoring problem instance $N(=pq)$ into the public parameter (without knowing p and q)

Random self-reducibility: given one DL (or RSA) problem instance (g, g^x) (or (x, x^e)) one can create multiple instance (g, g^{x_i}) (or $(x_i, {x_i}^e)$)

CHF_{dl}

$\underline{\mathsf{Setup}(1^\lambda):}$ $(\mathbb{G}, q, g) \leftarrow \mathsf{GGen}(1^\lambda)$ Define $\mathcal{M} := \mathbb{Z}_q$, $\mathcal{R} := \mathbb{Z}_q$, $\mathcal{Y} := \mathbb{G}$ Return $\mathsf{pp}_{\mathsf{CHF}} := (\mathbb{G}, q, g, \mathcal{M}, \mathcal{R}, \mathcal{Y})$ $\underline{\mathsf{KGen}(\mathsf{pp}_{\mathsf{CHF}}):}$ $x \xleftarrow{\$} \mathbb{Z}_q$; $X := g^x$ Return $(hk := X, td := x)$	$\underline{\mathsf{Eval}(hk, m, r):}$ $h := hk^m \cdot g^r$ Return h $\underline{\mathsf{TdColl}(td, m_1, r_1, m_2):}$ $r_2 := td \cdot (m_1 - m_2) + r_1 \mod q$ Return r_2

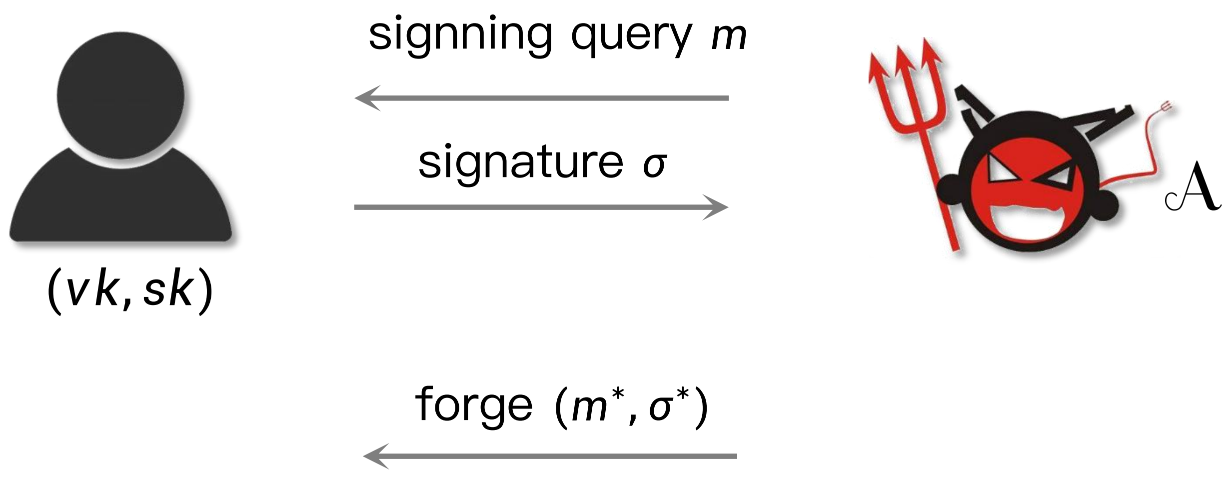
CHF_{rsa}

$\underline{\mathsf{Setup}(1^\lambda):}$ $(N,p,q,e,d) \leftarrow \mathsf{RSAGen}(1^\lambda)$ $\ell := L(\lambda)$ Define $\mathcal{M} := \{0,1\}^\ell$, $\mathcal{R} := \mathbb{Z}_N^*$, $\mathcal{Y} := \mathbb{Z}_N^*$ Return $\mathsf{pp}_{\mathsf{CHF}} := (N, e, \mathcal{M}, \mathcal{R}, \mathcal{Y})$ $\underline{\mathsf{KGen}(\mathsf{pp}_{\mathsf{CHF}}):}$ $x \xleftarrow{\$} \mathbb{Z}_N^*$; $X := x^e \mod N$ Return $(hk := X, td := x)$	$\underline{\mathsf{Eval}(hk, m, r):}$ $h := hk^m \cdot r^e \mod N$ Return h $\underline{\mathsf{TdColl}(td, m_1, r_1, m_2):}$ $r_2 := td^{m_1 - m_2} \cdot r_1 \mod N$ Return r_2

CHF_{fac}

$\underline{\mathsf{Setup}(1^\lambda):}$ $(N,p,q) \leftarrow \mathsf{FacGen}(1^\lambda)$ $\ell := \mathsf{poly}(\lambda)$ Define $\mathcal{M} := \{0,1\}^\ell$, $\mathcal{R} := \mathbb{Z}_N^+$, $\mathcal{Y} := \mathbb{QR}_N$ Return $\mathsf{pp}_{\mathsf{CHF}} := (N, \mathcal{M}, \mathcal{R}, \mathcal{Y})$ $\underline{\mathsf{KGen}(\mathsf{pp}_{\mathsf{CHF}}):}$ For $k \in [\ell]$: $\quad s_k \xleftarrow{\$} \mathbb{Z}_N^*$; $u_k := s_k^2 \mod N$ $hk := (u_1, ..., u_\ell)$; $td := (s_1, ..., s_\ell)$ Return (hk, td)	$\underline{\mathsf{Eval}(hk, m, r):}$ Parse $hk = (u_1, ..., u_\ell)$ $h := \prod_{k=1}^{\ell} u_k^{m_k} \cdot r^2 \mod N$ Return h $\underline{\mathsf{TdColl}(td, m_1, r_1, m_2):}$ Parse $td = (s_1, ..., s_\ell)$ $r_2 := \prod_{k=1}^{\ell} s_k^{m_{1,k} - m_{2,k}} \cdot r_1$ $r_2 := \min\{r_2, N - r_2\}$ Return r_2

Applications in Signatures



In the Single User Setting:

EUF–CMA: hard to forge a (valid) pair (m^*, σ^*) for new message m^*.

S–EUF–CMA: hard to forge a new (valid) pair (m^*, σ^*).

In the Multi–User Setting:

MU–EUF–CMA: hard to forge a (valid) pair (m^*, σ^*) for new message m^* under v

S–MU–EUF–CMA: hard to forge a new (valid) pair (m^*, σ^*) under vk_{i^*}.

Applications in Signatures

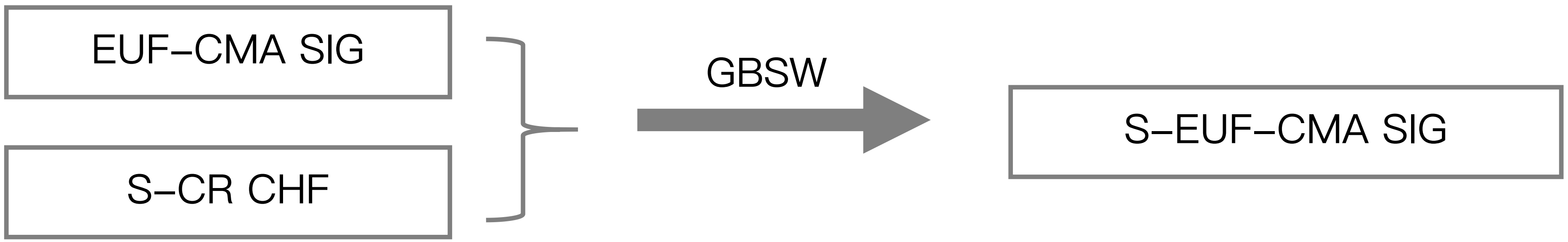
tightly S–MU–CR CHF + ***tightly*** MU–EUF–CMA SIG $\xrightarrow{\text{GBSW}}$ ***tightly*** S–MU–EUF–CMA SIG $\xrightarrow{\text{double–key}}$ ***tightly*** S–MU–EUF–CMA$^{\text{corr}}$ SIG

tightly MU–CR CHF + ***tightly*** MU–EUF–NCMA SIG → ***tightly*** MU–EUF–CMA online/offline SIG

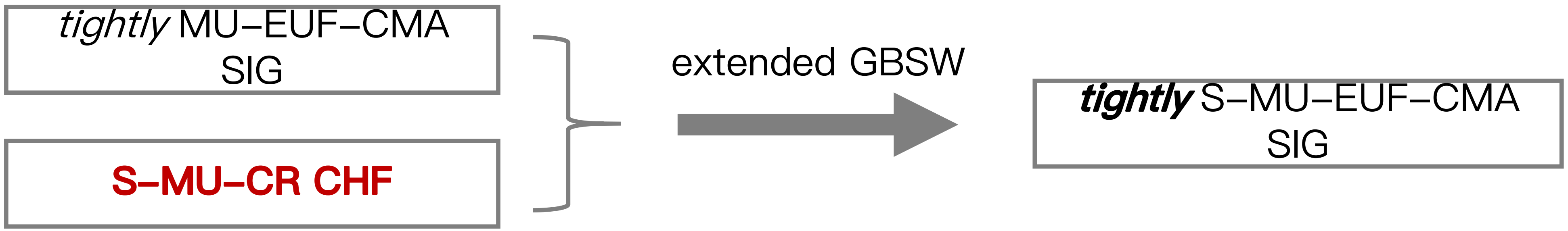
tightly MU–CR CHF + ***tightly*** MU–EUF–CMA SIG → ***tightly*** MU–EUF–CMA chameleon/proxy SIG

Extended GBSW Transform

GBSW Transform [SPW07]:



Extended GBSW Transform (using our tightly secure S–MU–CR CHF):



Online/Offline Signatures

Offline Phase:

no m given

Output st

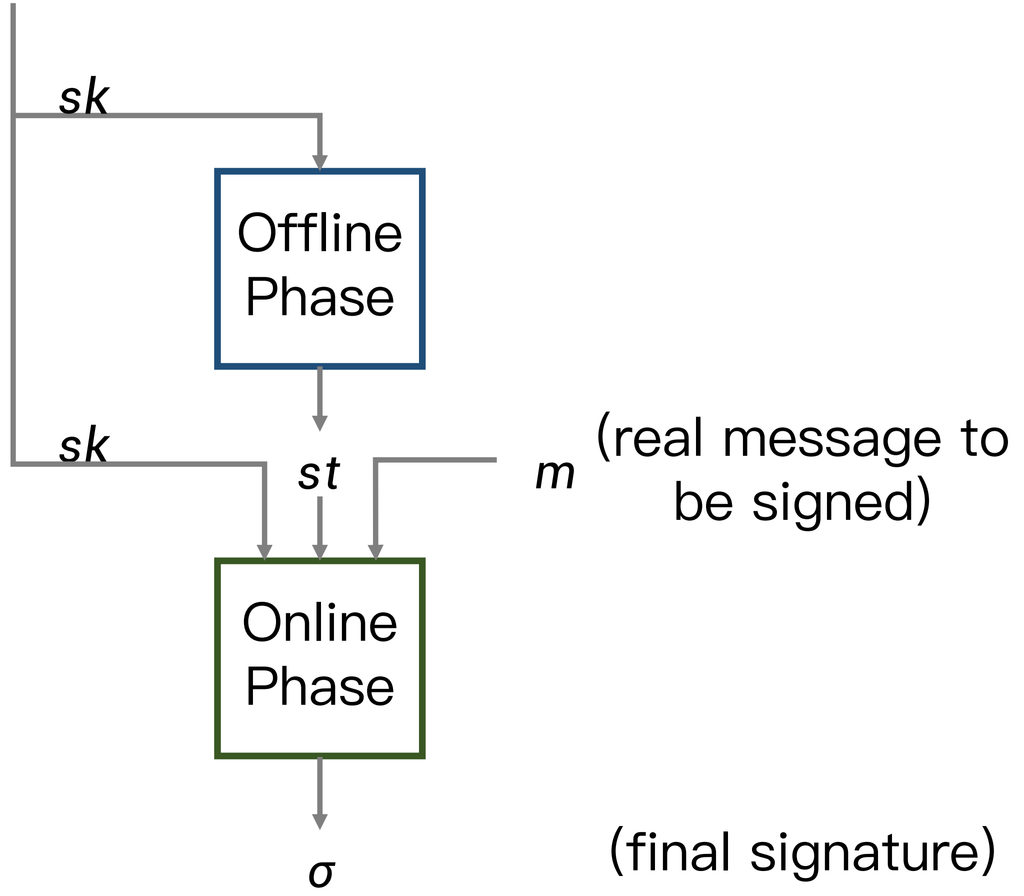
} pre−computation

Online Phase:

given sk, st, and m

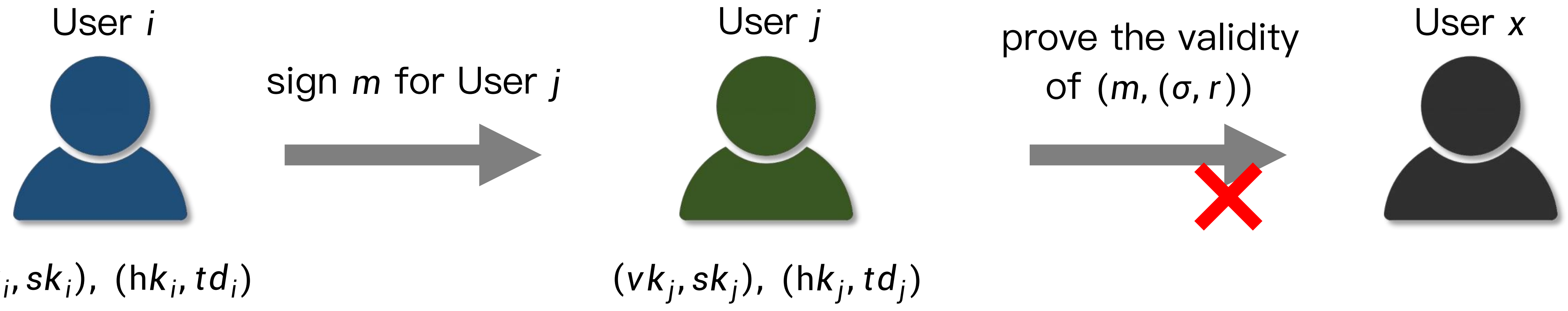
Output signature σ

} **highly efficient**



Chameleon Signatures

Chameleon signatures provide non–transferability



choose random r

$\sigma \leftarrow Sign(vk_i, H_{\mathrm{hk}_j}(m, r))$ $\xrightarrow{\sigma, r}$

It is hard for User j to convince a third party the validity of $(m, (\sigma, r))$.

Proxy Signatures

choose random m', r'

$\sigma \leftarrow Sign(vk,\ H_{hk}(m', r'))$

$\xrightarrow{\sigma, r'}$

find a collision

$H_{hk}(m, r) = H_{hk}(m', r')$

$\xrightarrow{\sigma, r}$

Hash-and-Sign Paradigm

$(vk, sk) \leftarrow \mathsf{SIG.KGen}$ $(\mathsf{h}k, td) \leftarrow \mathsf{CHF.KGen}$

First Phase

$(sk, \mathsf{h}k)$

sign for $H_{\mathsf{h}k}(m', r')$

(m', r' are chosen randomly)

Output (σ, m', r')

Second Phase

(td)

given the real message m to be signed

find a collision by TdColl

Output signature (σ, r)

Conclusion

- Security notion of (strong) collision resistance for CHFs
- Present three constructions, CHF_{dl}, CHF_{rsa}, CHF_{fac}, and prove their S-MU-CR security
- Extended GBSW transform
- Further applications in signatures

Thank you!

Questions?