Revisiting Generic Conversion from Non-Adaptive to Adaptively Secure IBS: Tightness and an Extension

Sanjit Chatterjee[1] **Tapas Pandit**[2]

[1]Indian Institute of Science Bangalore

[2]Plaksha University, Mohali

December 21, 2024

Outline

- IBS - Abstract Definition and Security Models
- A Quick Review of Generic Adaptive Constructions of IBS
- Issues in Pan and Wagner's IBS Constructions (PQC 2021)
- Addressing the Identified Issues
- Conclusion

Identity-Based Signature (IBS)

Algorithms:	Correctness:
• $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$ • $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$ • $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$ • $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$	• $\forall(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have $\boxed{\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.}$

Identity-Based Signature (IBS)

Algorithms:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk_{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk_{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

Correctness:

- $\forall(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk_{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk_{id}}), \mathsf{id}) = 1.$$

Ch

$\mathcal{A}$

Identity-Based Signature (IBS)

Algorithms:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

Correctness:

- $\forall (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\boxed{\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.}$$

public parameters (pp) $\longrightarrow$


Ch


$\mathcal{A}$

Identity-Based Signature (IBS)

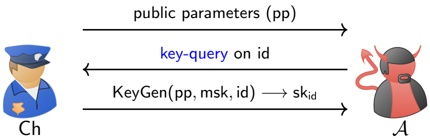
<u>Algorithms</u>:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

<u>Correctness</u>:

- $\forall (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.$$



Identity-Based Signature (IBS)

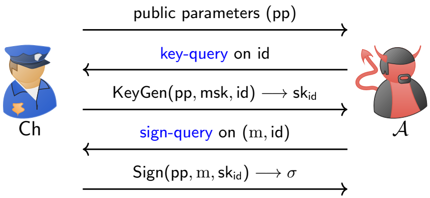
Algorithms:

- $\mathsf{Setup}(\kappa) \rightarrow (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \rightarrow \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

Correctness:

- $\forall(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\boxed{\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.}$$



Identity-Based Signature (IBS)

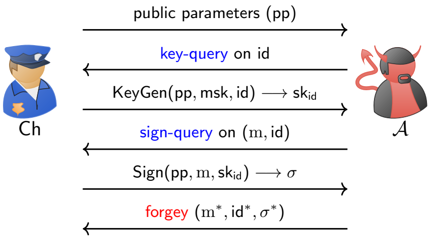
Algorithms:

- $\mathsf{Setup}(\kappa) \rightarrow (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \rightarrow \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

Correctness:

- $\forall (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.$$



Identity-Based Signature (IBS)

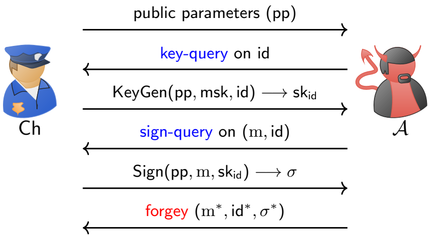
Algorithms:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

Correctness:

- $\forall(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\kappa)$, $\forall \mathsf{id} \in \mathcal{ID}$, $\forall \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$, and $\forall \mathrm{m} \in \mathcal{M}$, we have

$$\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}), \mathsf{id}) = 1.$$



$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa) := \Pr\left[\mathsf{Ver}(\mathsf{pp}, \mathrm{m}^*, \sigma^*, \mathsf{id}^*) = 1 \wedge \mathsf{id}^* \notin \mathcal{Q}_{\mathsf{key}} \wedge (\mathrm{m}^*, \mathsf{id}^*) \notin \mathcal{Q}_{\mathsf{sign}}\right]$$

- The scheme is EUF-ID-CMA secure, if for all ppt $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa)$ is negligible.

EUF-ID-CMA: More Details

Algorithms:

- $\mathsf{Setup}(\kappa) \rightarrow (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \rightarrow \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

$\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa)$:

```
Q_key := ∅, Q_sign := ∅, L_sk := ∅
(pp, msk) ← Setup(1^κ)
(id*, m*, σ*) ← A^{O_sign, O_key}(1^κ, pp)
if id* ∈ Q_key or (id*, m*) ∈ Q_sign then
    return 0
end if
return Ver(pp, id*, m*, σ*)
```

EUF-ID-CMA: More Details

Algorithms:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{EUF\text{-}ID\text{-}CMA}}(\kappa)$:

$\mathcal{Q}_{\mathsf{key}} := \emptyset, \mathcal{Q}_{\mathsf{sign}} := \emptyset, \mathcal{L}_{\mathsf{sk}} := \emptyset$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\{\mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{key}}\}}(1^\kappa, \mathsf{pp})$
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**
  return 0
end if
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}^*, \mathrm{m}^*, \sigma^*)$

$\mathcal{O}_{\mathsf{key}}(\mathsf{id})$:

if $\mathsf{id} \notin \mathcal{Q}_{\mathsf{key}}$ **then**
  $\mathcal{Q}_{\mathsf{key}} := \mathcal{Q}_{\mathsf{key}} \cup \{\mathsf{id}\}$
end if
if $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}) \in \mathcal{L}_{\mathsf{sk}}$ **then**
  return $\mathsf{sk}_{\mathsf{id}}$
end if
$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
$\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})\}$
return $\mathsf{sk}_{\mathsf{id}}$

$\mathcal{O}_{\mathsf{sign}}(\mathsf{id}, \mathrm{m})$:

if $(\mathsf{id}, \mathrm{m}) \notin \mathcal{Q}_{\mathsf{sign}}$ **then**
  $\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(\mathsf{id}, \mathrm{m})\}$
end if
if $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}) \notin \mathcal{L}_{\mathsf{sk}}$ **then**
  $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
  $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})\}$
end if
return $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$

EUF-ID-CMA: More Details

Algorithms:

- $\mathsf{Setup}(\kappa) \to (\mathsf{pp}, \mathsf{msk})$
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$
- $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pp}, \mathrm{m}, \sigma, \mathsf{id}) = \begin{cases} 1 & \text{accept} \\ 0 & \text{reject.} \end{cases}$

$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{EUF\text{-}ID\text{-}CMA}}(\kappa)$:

$\mathcal{Q}_{\mathsf{key}} := \emptyset, \mathcal{Q}_{\mathsf{sign}} := \emptyset, \mathcal{L}_{\mathsf{sk}} := \emptyset$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\kappa})$
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\{\mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{key}}\}}(1^{\kappa}, \mathsf{pp})$
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**
  return 0
end if
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}^*, \mathrm{m}^*, \sigma^*)$

$\mathcal{O}_{\mathsf{key}}(\mathsf{id})$:

if $\mathsf{id} \notin \mathcal{Q}_{\mathsf{key}}$ **then**
  $\mathcal{Q}_{\mathsf{key}} := \mathcal{Q}_{\mathsf{key}} \cup \{\mathsf{id}\}$
end if
if $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}) \in \mathcal{L}_{\mathsf{sk}}$ **then**
  return $\mathsf{sk}_{\mathsf{id}}$
end if
$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
$\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})\}$
return $\mathsf{sk}_{\mathsf{id}}$

$\mathcal{O}_{\mathsf{sign}}(\mathsf{id}, \mathrm{m})$:

if $(\mathsf{id}, \mathrm{m}) \notin \mathcal{Q}_{\mathsf{sign}}$ **then**
  $\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(\mathsf{id}, \mathrm{m})\}$
end if
if $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}) \notin \mathcal{L}_{\mathsf{sk}}$ **then**
  $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
  $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})\}$
end if
return $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$

- According to the definition of IBS, a user with identity id generates signatures on $(\mathsf{id}, \mathrm{m})$ for different messages m using the same private key $\mathsf{sk}_{\mathsf{id}}$, and this environment is correctly captured by the EUF-ID-CMA model [BNN04, LPLL20].

Non-Adaptive Model: EUF-naCMA

$\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-naCMA}}(\kappa)$:

$(\mathcal{Q}_{\mathsf{key}}, \mathcal{Q}_{\mathsf{sign}}) \leftarrow \mathcal{A}(1^\kappa)$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$
for $\mathsf{id} \in \mathcal{Q}_{\mathsf{key}}$ **do**
    $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
    $\mathcal{L}_{\mathsf{sk}} := \mathcal{L}_{\mathsf{sk}} \cup \{\mathsf{sk}_{\mathsf{id}}\}$
end for
for $(\mathsf{id}, \mathrm{m}) \in \mathcal{Q}_{\mathsf{sign}}$ **do**
    $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$
    $\mathcal{L}_{\mathsf{sign}} := \mathcal{L}_{\mathsf{sign}} \cup \{\sigma\}$
end for
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathcal{L}_{\mathsf{sk}}, \mathcal{L}_{\mathsf{sign}})$
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**
    return 0
end if
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}, \mathrm{m}, \sigma)$

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-naCMA}}(\kappa) := \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-naCMA}}(\kappa) = 1\right]$$

- The scheme is EUF-naCMA secure, if for all ppt $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-naCMA}}(\kappa)$ is negligible.

Generic Constructions of IBS: Adaptive Security

Several generic techniques have been proposed to construct IBS from different primitives:

- Bellare, Namprempre, and Neven (2004, 2009) proposed two generic techniques for IBS:
  1. Using digital signatures at two levels: one for generating public parameters and the master secret key, and another for generating keys for individual identities.
  2. Based on a standard identification scheme combined with a trapdoor sampleable relation (TSR), followed by the Fiat-Shamir transform.

Generic Constructions of IBS: Adaptive Security

Several generic techniques have been proposed to construct IBS from different primitives:

- Bellare, Namprempre, and Neven (2004, 2009) proposed two generic techniques for IBS:
    1. Using digital signatures at two levels: one for generating public parameters and the master secret key, and another for generating keys for individual identities.
    2. Based on a standard identification scheme combined with a trapdoor sampleable relation (TSR), followed by the Fiat-Shamir transform.
- However, these generic approaches lack tight security reductions.

Generic Constructions of IBS: Adaptive and Tight Security

- Note that tightly secure cryptographic schemes offer better concrete security assurance than their non-tight counterparts.
- Zhang et al. (2019) proposed a generic construction of IBS using two digital signatures: one secure in the single-user setting and the other in the multi-user setting.
- Later, Lee et al. (2020) showed that the same construction achieves tight security in the EUF-ID-CMA model.
- However, the construction is not efficient as each signature includes the underlying public key.

Generic Constructions of Pan and Wagner (2021)

This work claims to realize tightly EUF-ID-CMA secure IBS schemes from lattices using a two-stage approach:

1. First, construct an IBS scheme from lattices achieving a tight reduction in a **non-adaptive** security model.
2. Then, lift such scheme to tight **adaptive security** (EUF-ID-CMA) using two generic approaches:
   - One based on chameleon hashes in the standard model (SM).
   - The other based on hash functions in the random oracle model (ROM).

Generic Constructions of Pan and Wagner (2021)

This work claims to realize tightly EUF-ID-CMA secure IBS schemes from lattices using a two-stage approach:

1. First, construct an IBS scheme from lattices achieving a tight reduction in a **non-adaptive** security model.
2. Then, lift such scheme to tight **adaptive security** (EUF-ID-CMA) using two generic approaches:
   - One based on chameleon hashes in the standard model (SM).
   - The other based on hash functions in the random oracle model (ROM).

Generic Constructions of Pan and Wagner (2021)

This work claims to realize tightly EUF-ID-CMA secure IBS schemes from lattices using a two-stage approach:

1. First, construct an IBS scheme from lattices achieving a tight reduction in a **non-adaptive** security model.
2. Then, lift such scheme to tight **adaptive security** (EUF-ID-CMA) using two generic approaches:
   - One based on chameleon hashes in the standard model (SM).
   - The other based on hash functions in the random oracle model (ROM).

Note: The above result has recently been extended (SPMC-ACNS23) to the QROM.

Issues in Adaptive Model of PW21

$\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-ID-CMA-PW}}(\kappa)$:

$\mathcal{Q}_{\mathsf{key}} := \emptyset, \mathcal{Q}_{\mathsf{sign}} := \emptyset$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\{\mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{key}}\}}(1^\kappa, \mathsf{pp})$
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**
  return 0
end if
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}^*, \mathrm{m}^*, \sigma^*)$

$\mathcal{O}_{\mathsf{key}}(\mathsf{id})$:

$\mathcal{Q}_{\mathsf{key}} := \mathcal{Q}_{\mathsf{key}} \cup \{\mathsf{id}\}$
return $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$

$\mathcal{O}_{\mathsf{sign}}(\mathsf{id}, \mathrm{m})$:

$\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(\mathsf{id}, \mathrm{m})\}$
$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
return $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$

Issues in Adaptive Model of PW21

$\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-ID-CMA-PW}}(\kappa)$:

$\mathcal{Q}_{\mathsf{key}} := \emptyset, \mathcal{Q}_{\mathsf{sign}} := \emptyset$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\{\mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{key}}\}}(1^\kappa, \mathsf{pp})$
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**
  return 0
end if
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}^*, \mathrm{m}^*, \sigma^*)$

$\mathcal{O}_{\mathsf{key}}(\mathsf{id})$:

$\mathcal{Q}_{\mathsf{key}} := \mathcal{Q}_{\mathsf{key}} \cup \{\mathsf{id}\}$
return $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$

$\mathcal{O}_{\mathsf{sign}}(\mathsf{id}, \mathrm{m})$:

$\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(\mathsf{id}, \mathrm{m})\}$
$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
return $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$

Issue in EUF-ID-CMA-PW

1. For answering two (or more) signing queries on the same identity but for different messages, a fresh key is generated each time.
2. When $\mathcal{A}$ makes a sign query followed by a key query on the same identity, the returned signing key is different from the one used to generate the signature.

Issues in Adaptive Model of PW21

$\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-ID-CMA-PW}}(\kappa)$:	$\mathcal{O}_{\mathsf{key}}(\mathsf{id})$:
$\mathcal{Q}_{\mathsf{key}} := \emptyset, \mathcal{Q}_{\mathsf{sign}} := \emptyset$	$\mathcal{Q}_{\mathsf{key}} := \mathcal{Q}_{\mathsf{key}} \cup \{\mathsf{id}\}$
$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$	**return** $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
$(\mathsf{id}^*, \mathrm{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\{\mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{key}}\}}(1^\kappa, \mathsf{pp})$	$\mathcal{O}_{\mathsf{sign}}(\mathsf{id}, \mathrm{m})$:
if $\mathsf{id}^* \in \mathcal{Q}_{\mathsf{key}}$ or $(\mathsf{id}^*, \mathrm{m}^*) \in \mathcal{Q}_{\mathsf{sign}}$ **then**	$\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(\mathsf{id}, \mathrm{m})\}$
return 0	$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$
end if	**return** $\mathsf{Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})$
return $\mathsf{Ver}(\mathsf{pp}, \mathsf{id}^*, \mathrm{m}^*, \sigma^*)$	

Issue in EUF-ID-CMA-PW

1. For answering two (or more) signing queries on the same identity but for different messages, a fresh key is generated each time.
2. When $\mathcal{A}$ makes a sign query followed by a key query on the same identity, the returned signing key is different from the one used to generate the signature.

- In either case, the security model deviates from the standard EUF-ID-CMA model and, therefore, does not accurately capture the real protocol environment.

Reduction Outline of PW21

- Both reductions are invalid in EUF-ID-CMA, though they remain valid in their proposed model, EUF-ID-CMA-PW.
- The issue is illustrated using their chameleon hash-based construction.

Reduction Outline of PW21

- Both reductions are invalid in EUF-ID-CMA, though they remain valid in their proposed model, EUF-ID-CMA-PW.
- The issue is illustrated using their chameleon hash-based construction.
- Let $\text{IBS}' = (\text{IBS'.Setup}, \text{IBS'.KeyGen}, \text{IBS'.Sign}, \text{IBS'.Ver})$ be a non-adaptively secure primitive identity-based signature scheme with identity space $\mathcal{ID}'$ and message space $\mathcal{M}'$.
- Let $\text{CHF} = (\text{CHGen}, \text{CHash}, \text{CHColl})$ be a chameleon hash function.

Reduction Outline of PW21

- Both reductions are invalid in EUF-ID-CMA, though they remain valid in their proposed model, EUF-ID-CMA-PW.
- The issue is illustrated using their chameleon hash-based construction.
- Let $\mathsf{IBS}' = (\mathsf{IBS'.Setup}, \mathsf{IBS'.KeyGen}, \mathsf{IBS'.Sign}, \mathsf{IBS'.Ver})$ be a non-adaptively secure primitive identity-based signature scheme with identity space $\mathcal{ID}'$ and message space $\mathcal{M}'$.
- Let $\mathsf{CHF} = (\mathsf{CHGen}, \mathsf{CHash}, \mathsf{CHColl})$ be a chameleon hash function.
- Let $\mathsf{IBS} = (\mathsf{IBS.Setup}, \mathsf{IBS.KeyGen}, \mathsf{IBS.Sign}, \mathsf{IBS.Ver})$ denote the target IBS scheme constructed using chameleon hash functions and the primitive IBS scheme IBS'.

Reduction Outline of PW21 (Cont.)

$\underline{\mathsf{IBS.Setup}(\kappa)}$:

$(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{CHGen}(\kappa)$
$(\mathsf{pp}', \mathsf{msk}') \leftarrow \mathsf{IBS'.Setup}(\kappa)$
$\mathsf{pp} := (\mathsf{pp}', \mathsf{hk})$ and $\mathsf{msk} := \mathsf{msk}'$
return $(\mathsf{pp}, \mathsf{msk})$

$\underline{\mathsf{IBS.KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})}$:

$r \xleftarrow{\mathrm{U}} \mathsf{SaltSp}$
$\mathsf{id}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathsf{id}; r)$
$\mathsf{sk}_{\mathsf{id}'} \leftarrow \mathsf{IBS'.KeyGen}(\mathsf{pp}', \mathsf{msk}', \mathsf{id}')$
$\mathsf{sk}_{\mathsf{id}} := (\mathsf{sk}_{\mathsf{id}'}, r)$
return $\mathsf{sk}_{\mathsf{id}}$

$\underline{\mathsf{IBS.Sign}(\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}})}$:

parse $\mathsf{sk}_{\mathsf{id}}$ as $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{id}'}, r)$
$s \xleftarrow{\mathrm{U}} \mathsf{SaltSp}$
$\mathrm{m}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathrm{m}; s)$
$\sigma' \leftarrow \mathsf{IBS'.Sign}(\mathsf{pp}', \mathrm{m}', \mathsf{sk}_{\mathsf{id}'})$
$\sigma := (\sigma', r, s)$
return σ

$\underline{\mathsf{IBS.Ver}(\mathsf{pp}, \mathsf{id}, \mathrm{m}, \sigma)}$:

parse σ as $\sigma = (\sigma', r, s)$
$\mathsf{id}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathsf{id}; r)$
$\mathrm{m}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathrm{m}; s)$
return $\mathsf{IBS'.Ver}(\mathsf{pp}', \mathsf{id}', \mathrm{m}', \sigma')$

Reduction Outline of PW21 (Cont.)

IBS.Setup(κ):

1: $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{CHGen}(\kappa)$
2: $(\mathsf{pp}', \mathsf{msk}') \leftarrow \mathsf{IBS'.Setup}(\kappa)$
3: $\mathsf{pp} := (\mathsf{pp}', \mathsf{hk})$ and $\mathsf{msk} := \mathsf{msk}'$
4: **return** $(\mathsf{pp}, \mathsf{msk})$

IBS.KeyGen(pp, msk, id):

1: $r \xleftarrow{\mathrm{U}} \mathsf{SaltSp}$
2: $\mathsf{id}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathsf{id}; r)$
3: $\mathsf{sk}_{\mathsf{id}'} \leftarrow \mathsf{IBS'.KeyGen}(\mathsf{pp}', \mathsf{msk}', \mathsf{id}')$
4: $\mathsf{sk}_{\mathsf{id}} := (\mathsf{sk}_{\mathsf{id}'}, r)$
5: **return** $\mathsf{sk}_{\mathsf{id}}$

IBS.Sign($\mathsf{pp}, \mathrm{m}, \mathsf{sk}_{\mathsf{id}}$):

1: parse $\mathsf{sk}_{\mathsf{id}}$ as $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{id}'}, r)$
2: $s \xleftarrow{\mathrm{U}} \mathsf{SaltSp}$
3: $\mathrm{m}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathrm{m}; s)$
4: $\sigma' \leftarrow \mathsf{IBS'.Sign}(\mathsf{pp}', \mathrm{m}', \mathsf{sk}_{\mathsf{id}'})$
5: $\sigma := (\sigma', r, s)$
6: **return** σ

IBS.Ver($\mathsf{pp}, \mathsf{id}, \mathrm{m}, \sigma$):

1: parse σ as $\sigma = (\sigma', r, s)$
2: $\mathsf{id}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathsf{id}; r)$
3: $\mathrm{m}' \leftarrow \mathsf{CHash}(\mathsf{hk}, \mathrm{m}; s)$
4: **return** $\mathsf{IBS'.Ver}(\mathsf{pp}', \mathsf{id}', \mathrm{m}', \sigma')$

1. At the beginning, a simulator S declares $\mathcal{Q}'_{\mathsf{key}}$ and $\mathcal{Q}'_{\mathsf{sign}}$ in the EUF-naCMA game against IBS′ and obtains the corresponding keys and signatures.
2. Specifically, S prepares the i-th entry of $\mathcal{Q}'_{\mathsf{sign}}$ as follows: $\mathsf{id}'_i = \mathsf{CHash}(\mathsf{hk}, 0; r'_i)$ and $\mathrm{m}'_i = \mathsf{CHash}(\mathsf{hk}, 0; s'_i)$, where r'_i and s'_i are random salts.
3. Let σ'_i denote the signature that S obtains for $(\mathsf{id}'_i, \mathrm{m}'_i)$ from its challenger.

Reduction Outline of PW21 (Cont.)

$\underline{\text{IBS.Setup}(\kappa)}$:	$\underline{\text{IBS.Sign}(\text{pp}, \text{m}, \text{sk}_{\text{id}})}$:
$(\text{hk}, \text{td}) \leftarrow \text{CHGen}(\kappa)$	parse sk_{id} as $\text{sk}_{\text{id}} = (\text{sk}_{\text{id}'}, r)$
$(\text{pp}', \text{msk}') \leftarrow \text{IBS'.Setup}(\kappa)$	$s \xleftarrow{U} \text{SaltSp}$
$\text{pp} := (\text{pp}', \text{hk})$ and $\text{msk} := \text{msk}'$	$\text{m}' \leftarrow \text{CHash}(\text{hk}, \text{m}; s)$
return (pp, msk)	$\sigma' \leftarrow \text{IBS'.Sign}(\text{pp}', \text{m}', \text{sk}_{\text{id}'})$
	$\sigma := (\sigma', r, s)$
	return σ
$\underline{\text{IBS.KeyGen}(\text{pp}, \text{msk}, \text{id})}$:	$\underline{\text{IBS.Ver}(\text{pp}, \text{id}, \text{m}, \sigma)}$:
$r \xleftarrow{U} \text{SaltSp}$	parse σ as $\sigma = (\sigma', r, s)$
$\text{id}' \leftarrow \text{CHash}(\text{hk}, \text{id}; r)$	$\text{id}' \leftarrow \text{CHash}(\text{hk}, \text{id}; r)$
$\text{sk}_{\text{id}'} \leftarrow \text{IBS'.KeyGen}(\text{pp}', \text{msk}', \text{id}')$	$\text{m}' \leftarrow \text{CHash}(\text{hk}, \text{m}; s)$
$\text{sk}_{\text{id}} := (\text{sk}_{\text{id}'}, r)$	**return** $\text{IBS'.Ver}(\text{pp}', \text{id}', \text{m}', \sigma')$
return sk_{id}	

4. Later, when $\mathcal{A}$ makes the i-th signature query on some message $(\text{id}_i, \text{m}_i)$, S utilizes the trapdoor td to correctly map $(\text{id}'_i, \text{m}'_i)$.
5. Specifically, using td, S finds r_i and s_i such that $\text{CHash}(\text{hk}, \text{id}_i; r_i) = \text{id}'_i$ and $\text{CHash}(\text{hk}, \text{m}_i; s_i) = \text{m}'_i$, and returns $\sigma_i = (\sigma'_i, r_i, s_i)$ to $\mathcal{A}$.
6. A similar approach is applicable in preparing $\mathcal{Q}'_{\text{key}}$ and answering key queries.

Issues in the Reductions of PW21

- **Case I:** $\mathcal{A}$ makes the i-th and j-th signature queries on the same identity as (id, m_i) and (id, m_j):
  - As per the protocol, the same secret key $\mathsf{sk}_{\mathsf{id}}$ has to be used to generate the two signatures.
  - Hence, the same randomizer r must be included as part of the two returned signatures.
  - However, the randomizers are different as the two queries are respectively mapped to the i-th and j-th elements of $\mathcal{Q}'_{\mathsf{sign}}$:

Issues in the Reductions of PW21

- **Case I:** $\mathcal{A}$ makes the i-th and j-th signature queries on the same identity as (id, m_i) and (id, m_j):
  - As per the protocol, the same secret key $\mathsf{sk}_{\mathsf{id}}$ has to be used to generate the two signatures.
  - Hence, the same randomizer r must be included as part of the two returned signatures.
  - However, the randomizers are different as the two queries are respectively mapped to the i-th and j-th elements of $\mathcal{Q}'_{\mathsf{sign}}$:
    - S first has to compute $r_1 = \mathsf{CHColl}(\mathsf{hk}, \mathsf{td}, 0, r_i, \mathsf{id})$ and $r_2 = \mathsf{CHColl}(\mathsf{hk}, \mathsf{td}, 0, r_j, \mathsf{id})$.
    - Clearly, $r_1 \neq r_2$ with overwhelming probability due to the property of chameleon hash function.

Issues in the Reductions of PW21

- **Case II:** $\mathcal{A}$ first makes a signature query on some identity $\hat{\mathsf{id}}$, followed by a key query on $\hat{\mathsf{id}}$:
  - According to the protocol, the secret key returned for the key query on $\hat{\mathsf{id}}$ must be the same as the one used to respond to the preceding signature query.
  - This implies that the salt component r in both responses must be identical.
  - However, in the reduction presented in [PW21], the two queries produce different randomizers r.
- Thus, it violates the actual protocol environment as well as the standard EUF-ID-CMA model.

Issues in the Reductions of PW21

- **Case II:** $\mathcal{A}$ first makes a signature query on some identity $\hat{\mathsf{id}}$, followed by a key query on $\hat{\mathsf{id}}$:
  - According to the protocol, the secret key returned for the key query on $\hat{\mathsf{id}}$ must be the same as the one used to respond to the preceding signature query.
  - This implies that the salt component r in both responses must be identical.
  - However, in the reduction presented in [PW21], the two queries produce different randomizers r.
- Thus, it violates the actual protocol environment as well as the standard EUF-ID-CMA model.

Remark: A similar argument is applicable to the ROM-based reduction of [PW21].

Tight Security in a Restrictive Yet Realistic Model

- We show that a tight reduction is possible in a restricted version of the standard EUF-ID-CMA model.
- **WModel I:** This is the same as the EUF-ID-CMA model, except that $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$.
- We show that both generic constructions of [PW21] achieve tight security in **WModel I**.

Tight Security in a Restrictive Yet Realistic Model

- We show that a tight reduction is possible in a restricted version of the standard EUF-ID-CMA model.
- **WModel I:** This is the same as the EUF-ID-CMA model, except that $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$.
- We show that both generic constructions of [PW21] achieve tight security in **WModel I**.
- Note that the original reductions from [PW21] cannot go through this model, even though it appears to be a weak model.
  - For example, if $\mathcal{A}$ makes all signature queries with the same identity, the salt parts r will differ for all the replied signatures – which is a violation w.r.t **WModel I**.
  - On the other hand, our reductions consider q_s^2 signature calls in EUM-naCMA against q_s signature calls in **WModel I** to correctly handle the above scenario.

Lifting Reductions from WModel-I to EUF-ID-CMA

WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

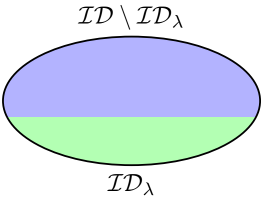
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



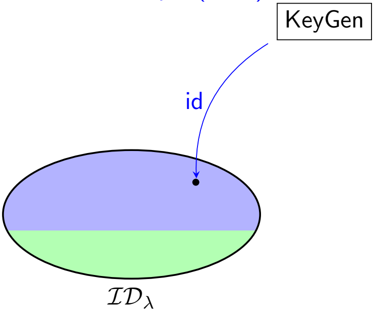
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

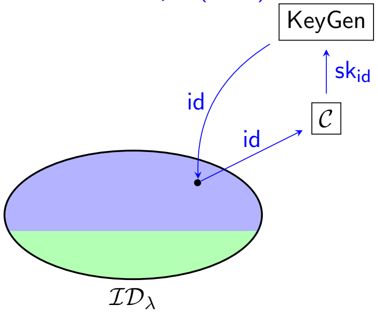
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

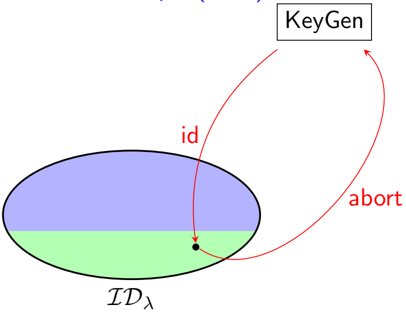
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

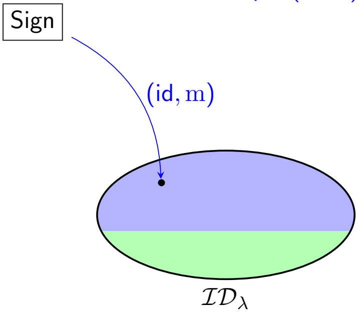
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

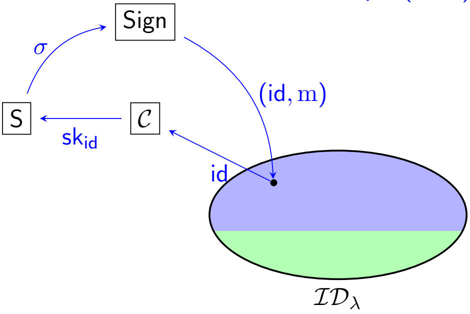
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

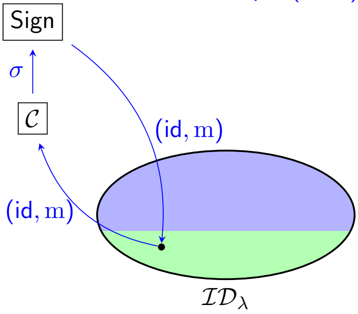
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

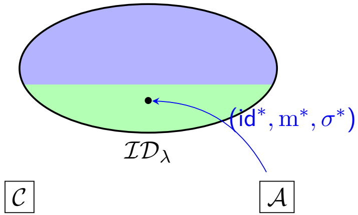
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



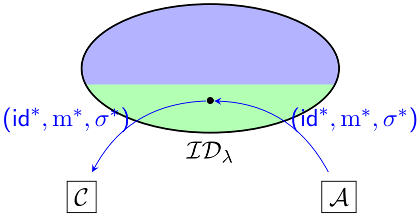
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

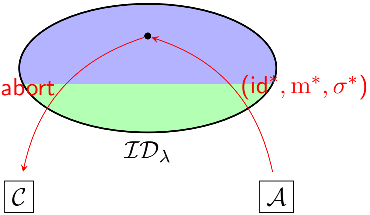
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{C}$ – EUF-naCMA Challenger

$\mathcal{A}$ – EUF-ID-CMA Attacker

S – Simulator

Coron's Technique (2000)



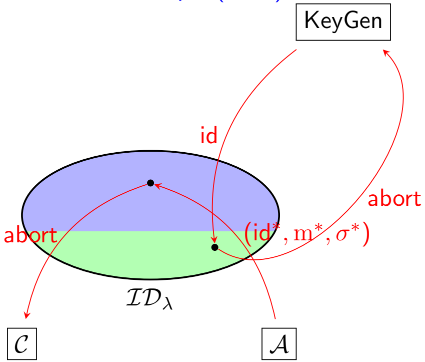
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



Lifting Reductions from WModel-I to EUF-ID-CMA

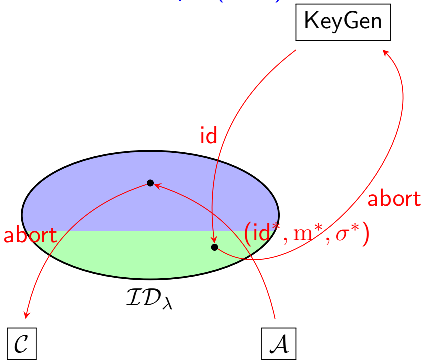
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



$$\text{Adv}_{\text{S}}^{\text{WModel-I}}(\kappa) \quad \geq \quad \boxed{(1-\lambda)^{q_k} \cdot \lambda} \cdot \text{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa)$$

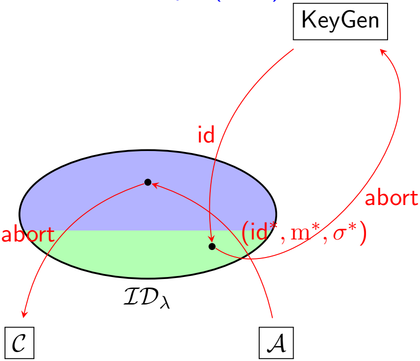
WModel I: $\mathcal{Q}_{\mathsf{key}} \cap \mathcal{Q}_{\mathsf{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



$$\mathsf{Adv}_{\mathsf{S}}^{\mathsf{WModel\text{-}I}}(\kappa) \geq \boxed{(1-\lambda)^{q_k} \cdot \lambda} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathrm{EUF\text{-}ID\text{-}CMA}}(\kappa)$$

$(1-\lambda)^{q_k} \cdot \lambda$ attains the maximum value of $1/(e \cdot q_k)$ when $\lambda = 1/(1+q_k)$.

Lifting Reductions from WModel-I to EUF-ID-CMA

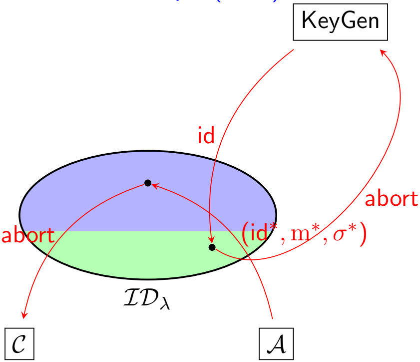
WModel I: $\mathcal{Q}_{\text{key}} \cap \mathcal{Q}_{\text{id}} = \emptyset$

$\mathcal{A}$ – EUF-ID-CMA Attacker

$\mathcal{C}$ – EUF-naCMA Challenger

S – Simulator

Coron's Technique (2000)



$$\begin{aligned}
\text{Adv}_{\text{S}}^{\text{WModel-I}}(\kappa) &\geq \boxed{(1-\lambda)^{q_k} \cdot \lambda} \cdot \text{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa) \\
&\approx \frac{1}{e \cdot q_k} \cdot \text{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa) \\
&\approx \boxed{\frac{1}{q_k}} \cdot \text{Adv}_{\mathcal{A}}^{\text{EUF-ID-CMA}}(\kappa),
\end{aligned}$$

$(1-\lambda)^{q_k} \cdot \lambda$ attains the maximum value of $1/(e \cdot q_k)$ when $\lambda = 1/(1+q_k)$.

Why Tight Security of PW21 is unlikely in EUF-ID-CMA?

- Consider the following two scenarios:
  1. $\mathcal{A}$ first makes a sign query on some $(\mathsf{id}, \mathrm{m})$ followed by a key query on the same identity id.
  2. $\mathcal{A}$ is likely to make sign queries on certain identities but not a follow-up key query.

Why Tight Security of PW21 is unlikely in EUF-ID-CMA?

- Consider the following two scenarios:
  1. $\mathcal{A}$ first makes a sign query on some $(\mathsf{id}, \mathrm{m})$ followed by a key query on the same identity id.
  2. $\mathcal{A}$ is likely to make sign queries on certain identities but not a follow-up key query.
- Hence, before answering a sign query on $(\mathsf{id}, \mathrm{m})$ for which no preceding key query on id exists, the simulator must somehow predict whether a key query on id will be made subsequently.

Why Tight Security of PW21 is unlikely in EUF-ID-CMA?

- Consider the following two scenarios:
  1. $\mathcal{A}$ first makes a sign query on some $(\mathsf{id}, \mathrm{m})$ followed by a key query on the same identity id.
  2. $\mathcal{A}$ is likely to make sign queries on certain identities but not a follow-up key query.
- Hence, before answering a sign query on $(\mathsf{id}, \mathrm{m})$ for which no preceding key query on id exists, the simulator must somehow predict whether a key query on id will be made subsequently.
- If the prediction is incorrect, the simulator has to abort, as otherwise, it will fail to provide a proper simulation of the EUF-ID-CMA security game.

Why Tight Security of PW21 is unlikely in EUF-ID-CMA?

- Consider the following two scenarios:
  1. $\mathcal{A}$ first makes a sign query on some $(\mathrm{id}, \mathrm{m})$ followed by a key query on the same identity id.
  2. $\mathcal{A}$ is likely to make sign queries on certain identities but not a follow-up key query.
- Hence, before answering a sign query on $(\mathrm{id}, \mathrm{m})$ for which no preceding key query on id exists, the simulator must somehow predict whether a key query on id will be made subsequently.
- If the prediction is incorrect, the simulator has to abort, as otherwise, it will fail to provide a proper simulation of the EUF-ID-CMA security game.
- This entails a degradation in the reduction which is proportional to either q_k or q_s.

Conclusion

- Pan and Wagner proposed a generic conversion from non-adaptive to adaptively secure IBS with a tightness claim.
- We identified certain gaps in their approach and proposed new reductions to address these gaps.
- We argued why the technique of [PW21] is unlikely to yield a tight reduction in the EUF-ID-CMA model.
- Additionally, we proposed a functional extension of the Pan-Wagner technique, enabling the registration of multiple devices under the same identity.

Thank you for your kind attention!

Research Associate Position in PQC

- **Duration:** Initially for one year (extendable by another year)
- **Monthly Stipend:** 58,000 + 18% HRA
- **Qualification:** Ph.D. in Computer Science or Mathematics with a strong background in Cryptography.

Interested candidates may contact Dr. Tapas Pandit, Assistant Professor at **Plaksha University, Mohali, Punjab**.
Contact Email: tapas.pandit@plaksha.edu.in