Stronger Security Notions for Decentralized Traceable Attribute-Based Signatures and More Efficient Constructions

Essam Ghadafi

University College London
e.ghadafi@ucl.ac.uk

CT-RSA 2015

OUTLINE

1 BACKGROUND

2 NEW SECURITY MODEL

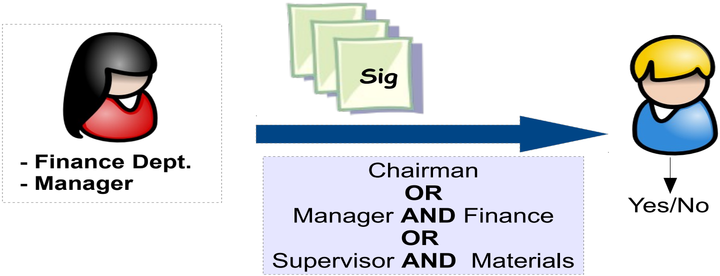
3 OUR GENERIC CONSTRUCTION

4 INSTANTIATIONS

5 EFFICIENCY COMPARISON

Attribute-Based Signatures [Maji et al. 2008]:

- Users have attributes (“Manager”, “Finance Department”, etc.).
- User with attributes $\mathcal{A}$ can sign messages w.r.t. policy $\mathbb{P}$ if $\mathbb{P}(\mathcal{A}) = 1$.
- Verifier only learns that the signature produced by someone with sufficient attributes to satisfy $\mathbb{P}$.



Example Applications:

- **Attribute-Based Messaging:**
  Recipients are assured the sender satisfies a certain policy.

- **Leaking Secrets:**
  - Ring Signatures [RST01] allow a signer to sign a message on behalf of an ad-hoc group.

  ABS allow more expressive predicates for leaking a secret
  $\Rightarrow$ The whistle-blower satisfies some policy vs. the whistle-blower is in the ring.

- **Many other applications:** ...

- **(Perfect) Privacy (Anonymity):**
  The signature hides:
  1. The identity of the signer.
  2. The attributes used in the signing (i.e. how $\mathbb{P}$ was satisfied).

- **Unforgeability:**
  A signer cannot forge signatures w.r.t. signing policies her attributes do not satisfy even if she colludes with other signers.

- **(Perfect) Privacy (Anonymity):**
  The signature hides:
  1. The identity of the signer.
  2. The attributes used in the signing (i.e. how $\mathbb{P}$ was satisfied).

- **Unforgeability:**
  A signer cannot forge signatures w.r.t. signing policies her attributes do not satisfy even if she colludes with other signers.

RELATED WORK ON ATTRIBUTE-BASED SIGNATURES

- Maji et al. 2008 & 2011.
- Shahandashti and Safavi-Naini 2009.
- Li et al. 2010.
- Okamoto and Takashima 2011 & 2012.
- Gagné et al. 2012.
- Herranz et al. 2012.

Additionally provide anonymity revocation mechanism (i.e. an opener) to enforce accountability.

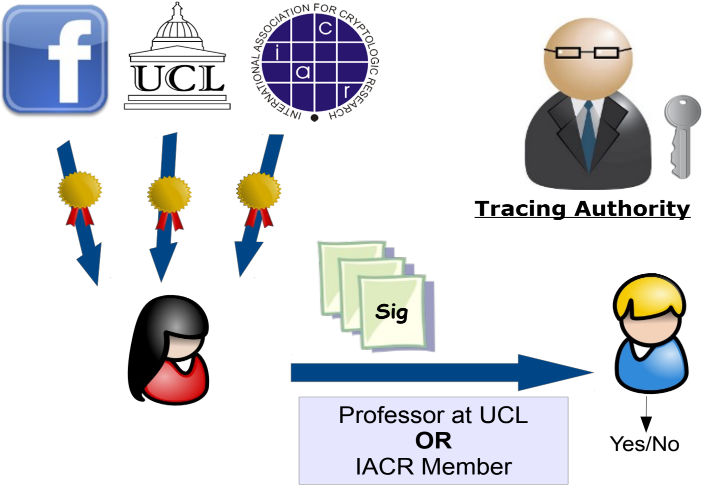
- **Traceable Attribute-Based Signatures (TABS) [Escala et al. 2011]:**
  - A single attribute authority.
  - No judge to verify the opener's decisions.

- **Decentralized Traceable Attribute-Based Signatures (DTABS) [El Kaafarani et al. 2014]:**
  - Multiple attribute authorities. Need not be aware of each other.
  - Signers and attribute authorities can join at any time.
  - Tracing correctness is publicly verifiable.

Additionally provide anonymity revocation mechanism (i.e. an opener) to enforce accountability.

- **Traceable Attribute-Based Signatures (TABS) [Escala et al. 2011]:**
  - A single attribute authority.
  - No judge to verify the opener's decisions.

- **Decentralized Traceable Attribute-Based Signatures (DTABS) [El Kaafarani et al. 2014]:**
  - Multiple attribute authorities. Need not be aware of each other.
  - Signers and attribute authorities can join at any time.
  - Tracing correctness is publicly verifiable.


UCL
INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH
i a c r
Tracing Authority
Sig
Professor at UCL
OR
IACR Member
Yes/No

Besides Correctness [El Kaafarani et al. 2014]:

- **Anonymity:** Signatures hide identity of the signer and attributes used.
- **Full Unforgeability:** Signers cannot sign w.r.t. policies not satisfied by their individual attributes even if they collude. Covers non-frameability.
- **Traceability:** The tracing authority can always identify the signer and prove its decision.

OUR CONTRIBUTION

1. A new stronger security model for DTABS.

2. A new generic construction for DTABS with much more efficient traceability.

3. More efficient instantiations in the standard model in Type-3 bilinear groups.

Shortcomings in existing models

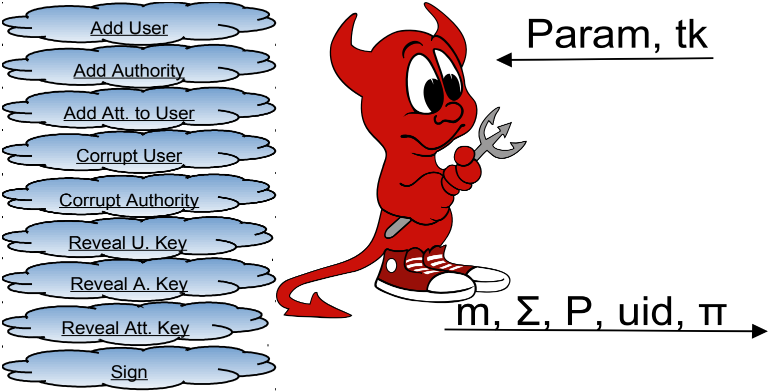
- **Non-Frameability:**
  - **Issue:** Knowledge of the secret key for any attribute allows framing an honest user ⇒In existing models:
    - All attribute authorities are trusted not to frame users.
    - Attribute keys must be delivered securely to users.
  - **Solution:** Assign users a personal key pair ⇒Even attribute authorities cannot frame a user without knowledge of her personal secret key.

To simplify the definitions, we separate Non-frameability from Unforgeability.

- **Non-Frameability:**
  - **Issue:** Knowledge of the secret key for any attribute allows framing an honest user ⇒In existing models:
    - All attribute authorities are trusted not to frame users.
    - Attribute keys must be delivered securely to users.
  - **Solution:** Assign users a personal key pair ⇒Even attribute authorities cannot frame a user without knowledge of her personal secret key.

To simplify the definitions, we separate Non-frameability from Unforgeability.

- **Non-Frameability:** If all users, all attribute authorities and the tracing authority collude, they cannot frame an honest user.



Adversary wins if:

1. uid is honest, Σ is valid and π accepted by Judge.
2. $(\mathsf{uid}, \cdot, m, \Sigma, P)$ was not obtained from the Sign oracle.

- **Lack of Tracing Soundness:**
  Similar to Group Signatures [Sakai et al. 2012], existing models do not prevent a signature being opened differently.

  Example Scenarios:

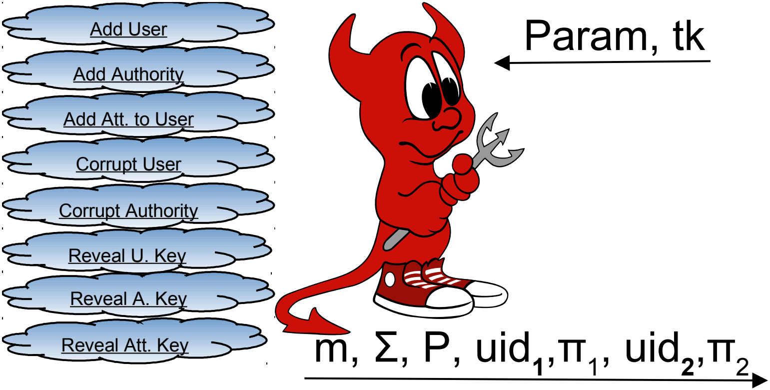
  - Claiming authorship of a signature by another (honest) user.
  - A signature opens to two different users.

  Example applications where this is needed:

  - Signatures used as evidence in court.
  - Users are rewarded for producing signatures.

Security of DTABS

- **Tracing Soundness:** A signature cannot trace to two different users.



Adversary wins if:

1. Σ is valid and π_i is a valid proof for user uid_i for all $i \in \{1, 2\}$.
2. $\mathsf{uid}_1 \neq \mathsf{uid}_2$.

How our construction differs from [El Kaafarani et al. 2014]:

1. Users have a personal key pair.
2. Dispense with the *pseudo-attribute technique* (Prove you satisfy $\mathbb{P}$ or have signature w.r.t. some public verification key on the message and $\mathbb{P}$).
3. Replace the IND-wCCA Tag-based Encryption (used to encrypt the signer's identity) with a Robust Non-Interactive Distributed/Threshold IND-wCCA Tag-Based Encryption.
   - $\Rightarrow$ We do without the *expensive zero-knowledge proofs* in the opening.

Tools used:

- A NIZK proof system $\mathcal{NIZK}$.
- A tagged signature scheme $\mathcal{TS}$: a signature scheme that signs a tag and a message.
- An existentially unforgeable (against weak chosen-message attack) signature scheme $\mathcal{WDS}$.
- An ST-IND-wCCA robust distributed/threshold tag-based encryption scheme $\mathcal{DTBE}$.
- A strongly unforgeable one-time signature scheme $\mathcal{OTS}$.

- **Setup:**
  - Generate $(\mathsf{epk}, \mathsf{esk})$ for $\mathcal{DTBE}$ and crs for $\mathcal{NIZK}$.
  - Choose CR hash functions $\hat{\mathcal{H}} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathcal{DTBE}}$ & $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathcal{OTS}}$.
  - Set $\mathsf{tk} := \mathsf{esk}$ and $\mathsf{param} := (\mathsf{crs}, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$.

- **User Key Generation:**
  Generate a key pair $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ for $\mathcal{WDS}$.

- **Attribute Authority Join:**
  Generate a key pair$(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{assk}_{\mathsf{aid}})$ for $\mathcal{TS}$.

- **Attribute Key Generation:**
  To generate a key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for attribute α for signer uid, compute $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathcal{TS}.\mathsf{Sign}(\mathsf{assk}_{\mathsf{aid}(\alpha)}, \mathbf{uvk}[\mathsf{uid}], \alpha)$.

- **Setup:**
  - Generate $(\mathsf{epk}, \mathsf{esk})$ for $\mathcal{DTBE}$ and crs for $\mathcal{NIZK}$.
  - Choose CR hash functions $\hat{\mathcal{H}} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathcal{DTBE}}$ & $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathcal{OTS}}$.
  - Set $\mathsf{tk} := \mathsf{esk}$ and $\mathsf{param} := (\mathsf{crs}, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$.

- **User Key Generation:**
  Generate a key pair $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ for $\mathcal{WDS}$.

- **Attribute Authority Join:**
  Generate a key pair$(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{assk}_{\mathsf{aid}})$ for $\mathcal{TS}$.

- **Attribute Key Generation:**
  To generate a key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for attribute α for signer uid, compute $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathcal{TS}.\mathsf{Sign}(\mathsf{assk}_{\mathsf{aid}(\alpha)}, \mathbf{uvk}[\mathsf{uid}], \alpha)$.

- **Setup:**
  - Generate (epk, esk) for $\mathcal{DTBE}$ and crs for $\mathcal{NIZK}$.
  - Choose CR hash functions $\hat{\mathcal{H}} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathcal{DTBE}}$ & $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathcal{OTS}}$.
  - Set $\mathsf{tk} := \mathsf{esk}$ and $\mathsf{param} := (\mathsf{crs}, \mathsf{epk}, \hat{\mathcal{H}}, \mathcal{H})$.

- **User Key Generation:**
  Generate a key pair $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ for $\mathcal{WDS}$.

- **Attribute Authority Join:**
  Generate a key pair$(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{assk}_{\mathsf{aid}})$ for $\mathcal{TS}$.

- **Attribute Key Generation:**
  To generate a key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for attribute α for signer uid, compute $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathcal{TS}.\mathsf{Sign}(\mathsf{assk}_{\mathsf{aid}(\alpha)}, \mathbf{uvk}[\mathsf{uid}], \alpha)$.

- **Setup:**
  - Generate (epk, esk) for $\mathcal{DTBE}$ and crs for $\mathcal{NIZK}$.
  - Choose CR hash functions $\hat{\mathcal{H}} : \{0,1\}^* \to \mathcal{T}_{\mathcal{DTBE}}$ & $\mathcal{H} : \{0,1\}^* \to \mathcal{M}_{\mathcal{OTS}}$.
  - Set tk := esk and param := (crs, epk, $\hat{\mathcal{H}}, \mathcal{H}$).

- **User Key Generation:**
  Generate a key pair $(\mathbf{uvk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ for $\mathcal{WDS}$.

- **Attribute Authority Join:**
  Generate a key pair$(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{assk}_{\mathsf{aid}})$ for $\mathcal{TS}$.

- **Attribute Key Generation:**
  To generate a key $\mathsf{sk}_{\mathsf{uid},\alpha}$ for attribute α for signer uid, compute $\mathsf{sk}_{\mathsf{uid},\alpha} \leftarrow \mathcal{TS}.\mathsf{Sign}(\mathsf{assk}_{\mathsf{aid}(\alpha)}, \mathbf{uvk}[\mathsf{uid}], \alpha)$.

- **Signing:** To sign m w.r.t. $\mathbb{P}$:
  1. Choose a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for $\mathcal{OTS}$.
  2. $C_{\text{dtbe}} \leftarrow \mathcal{DTBE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathbf{uvk}[\mathsf{uid}])$.
  3. $\sigma \leftarrow \mathcal{WDS}.\mathsf{Sign}(\mathbf{usk}[\mathsf{uid}], \hat{\mathcal{H}}(\mathsf{otsvk}))$.
  4. Produce a proof π of $(\mathcal{A}, \sigma, \mathbf{uvk}[\mathsf{uid}])$ that:
     1. C_{dtbe} is formed correctly.
     2. σ is valid.
     3. Has attributes $\mathcal{A}$ s.t. $\mathbb{P}(\mathcal{A}) = 1$
        $\Rightarrow$ Has a valid tagged signature on $(\mathbf{uvk}[\mathsf{uid}], \alpha)$ for each $\alpha \in \mathcal{A}$.
  5. Compute $\sigma_{\text{ots}} \leftarrow \mathcal{OTS}.\mathsf{Sign}(\mathsf{otssk}, (\mathcal{H}(m, \mathbb{P}), \pi, C_{\text{dtbe}}, \mathsf{otsvk}))$.

  The signature is $\Sigma := (\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$.

- **Tracing:**
  - Use esk to produce a decryption share ν of C_{dtbe} and recover $\mathsf{vk}_{\mathsf{uid}}$.
  - Return (uid, ν) if it matches any $\mathbf{uvk}[\mathsf{uid}]$ or $(0, \nu)$ otherwise.

- **Signing:** To sign m w.r.t. $\mathbb{P}$:
  1. Choose a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for $\mathcal{OTS}$.
  2. $C_{\text{dtbe}} \leftarrow \mathcal{DTBE}.\mathsf{Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathbf{uvk}[\mathsf{uid}])$.
  3. $\sigma \leftarrow \mathcal{WDS}.\mathsf{Sign}(\mathbf{usk}[\mathsf{uid}], \hat{\mathcal{H}}(\mathsf{otsvk}))$.
  4. Produce a proof π of $(\mathcal{A}, \sigma, \mathbf{uvk}[\mathsf{uid}])$ that:
     1. C_{dtbe} is formed correctly.
     2. σ is valid.
     3. Has attributes $\mathcal{A}$ s.t. $\mathbb{P}(\mathcal{A}) = 1$
        $\Rightarrow$ Has a valid tagged signature on $(\mathbf{uvk}[\mathsf{uid}], \alpha)$ for each $\alpha \in \mathcal{A}$.
  5. Compute $\sigma_{\text{ots}} \leftarrow \mathcal{OTS}.\mathsf{Sign}(\mathsf{otssk}, (\mathcal{H}(m, \mathbb{P}), \pi, C_{\text{dtbe}}, \mathsf{otsvk}))$.

  The signature is $\Sigma := (\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$.

- **Tracing:**
  - Use esk to produce a decryption share ν of C_{dtbe} and recover $\mathsf{vk}_{\mathsf{uid}}$.
  - Return (uid, ν) if it matches any $\mathbf{uvk}[\mathsf{uid}]$ or $(0, \nu)$ otherwise.

Anonymity:

- Zero-Knowledge of $\mathcal{NIZK}$.
- ST-IND-wCCA of $\mathcal{DTBE}$.
- Unforgeability of $\mathcal{OTS}$.
- Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

Unforgeability:

- Soundness of $\mathcal{NIZK}$.
- Unforgeability of $\mathcal{TS}$ and $\mathcal{OTS}$.
- Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

Non-Frameability:

- Soundness of $\mathcal{NIZK}$.
- Unforgeability of $\mathcal{WDS}$ and $\mathcal{OTS}$.
- Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Anonymity:**
  - Zero-Knowledge of $\mathcal{NIZK}$.
  - ST-IND-wCCA of $\mathcal{DTBE}$.
  - Unforgeability of $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Unforgeability:**
  - Soundness of $\mathcal{NIZK}$.
  - Unforgeability of $\mathcal{TS}$ and $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Non-Frameability:**
  - Soundness of $\mathcal{NIZK}$.
  - Unforgeability of $\mathcal{WDS}$ and $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Anonymity:**
  - Zero-Knowledge of $\mathcal{NIZK}$.
  - ST-IND-wCCA of $\mathcal{DTBE}$.
  - Unforgeability of $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Unforgeability:**
  - Soundness of $\mathcal{NIZK}$.
  - Unforgeability of $\mathcal{TS}$ and $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

- **Non-Frameability:**
  - Soundness of $\mathcal{NIZK}$.
  - Unforgeability of $\mathcal{WDS}$ and $\mathcal{OTS}$.
  - Collision-Resistance of $\mathcal{H}$ and $\hat{\mathcal{H}}$.

Traceability:

- Soundness of $\mathcal{NIZK}$.
- Unforgeability of $\mathcal{TS}$.

Tracing Soundness:

- Decryption Consistency of $\mathcal{DTBE}$.

- **Traceability:**
  - Soundness of $\mathcal{NIZK}$.
  - Unforgeability of $\mathcal{TS}$.

- **Tracing Soundness:**
  - Decryption Consistency of $\mathcal{DTBE}$.

- $\mathcal{NIZK}$ $\Rightarrow$Groth-Sahai proofs [GS08] secure under SXDH.
- $\mathcal{TS}$ $\Rightarrow$The re-randomizable structure-preserving scheme [Abe et al. 2011] (interactive assumption) or the strongly unforgeable [Abe et al. 2011] scheme (secure under q-AGHO).
- $\mathcal{DTBE}$ $\Rightarrow$[Ghadafi 2014] (secure under XDLIN in $\mathbb{G}_1$ or $\mathbb{G}_2$).
- $\mathcal{WDS}$ $\Rightarrow$The Weak Boneh-Boyen scheme (secure under q-SDH).
- $\mathcal{OTS}$ $\Rightarrow$The full Boneh-Boyen scheme (secure under q-SDH).

Scheme	Signature Size	Model	Setting
[EHM11]	$\mathbb{G}^{\lvert\mathbb{P}\rvert+\beta+7}$	ROM	Composite
[EGK14]	$\mathbb{G}_1^{34\cdot\lvert\mathbb{P}\rvert+28} + \mathbb{G}_2^{32\cdot\lvert\mathbb{P}\rvert+32} + \mathbb{Z}_p^{\beta+1}$	STD	Prime
Inst. I	$\mathbb{G}_1^{27\cdot\lvert\mathbb{P}\rvert+19} + \mathbb{G}_2^{22\cdot\lvert\mathbb{P}\rvert+15} + \mathbb{Z}_p^{\beta+3}$	STD	Prime
Inst. II	$\mathbb{G}_1^{30\cdot\lvert\mathbb{P}\rvert+18} + \mathbb{G}_2^{30\cdot\lvert\mathbb{P}\rvert+16} + \mathbb{Z}_p^{\beta+3}$	STD	Prime

TABLE: Signature Size

Scheme	Model	Setting	Tracing		
			Size	Compute	Verify
[EHM11]	ROM	Composite	N/A	N/A	N/A
[EGK14]	STD	Prime	$\mathbb{G}_1^3 \times \mathbb{G}_2^4$	$4E_{\mathbb{G}_1} + 6E_{\mathbb{G}_2}$	$34P$
Inst. I	STD	Prime	$\mathbb{G}_2^2$	$2E_{\mathbb{G}_2}$	$4P$
Inst. II	STD	Prime	$\mathbb{G}_1^2$	$2E_{\mathbb{G}_1}$	$4P$

TABLE: Tracing

Thank you for your attention!
Questions?

RSAConference2015

San Francisco | April 20-24 | Moscone Center

CHANGE

Challenge today's security thinking

SESSION ID: CRYP-F01

Re-encryption Verifiability: How to Detect Malicious Activities of a Proxy in Proxy Re-encryption

Satsuya Ohata[1,3], Yutaka Kawai[2], Takahiro Matsuda[3], Goichiro Hanaoka[3], Kanta Matsuura[1]

1. The University of Tokyo, Japan
2. Mitsubishi Electric, Japan
3. National Institute of Advanced Industrial Science and Technology, Japan

(Cryptology ePrint Archive: Report 2015/112)

Our Result

- We introduce a new functionality called “re-encryption verifiability” in proxy re-encryption (PRE).
  - To check whether a proxy works correctly or not

- We show a new CCA security definition of PRE.
  - Stronger definition than previous works

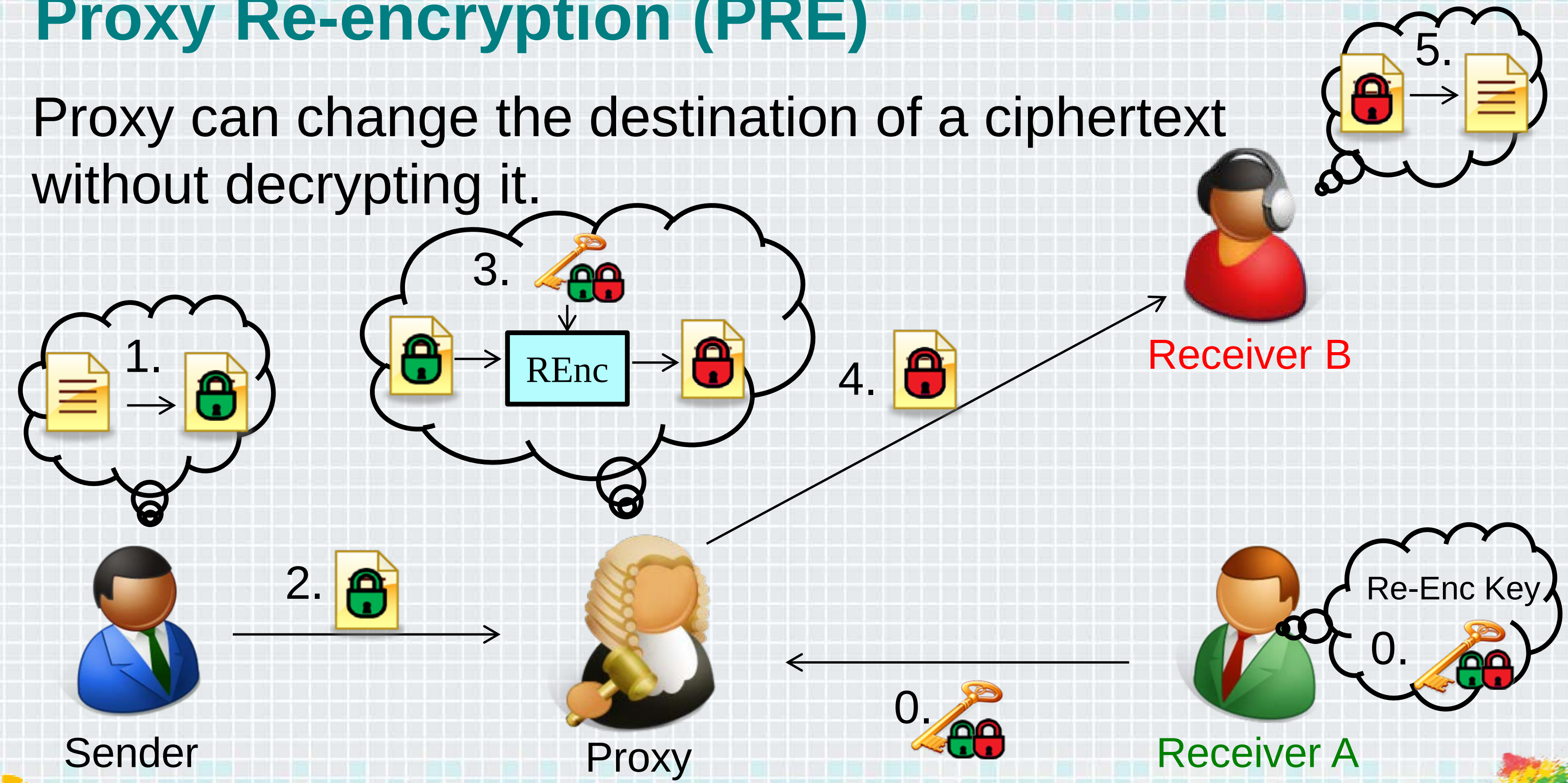
- We prove that previous generic construction[HKK+12] of a PRE satisfies our new stronger security definition.

#RSAC

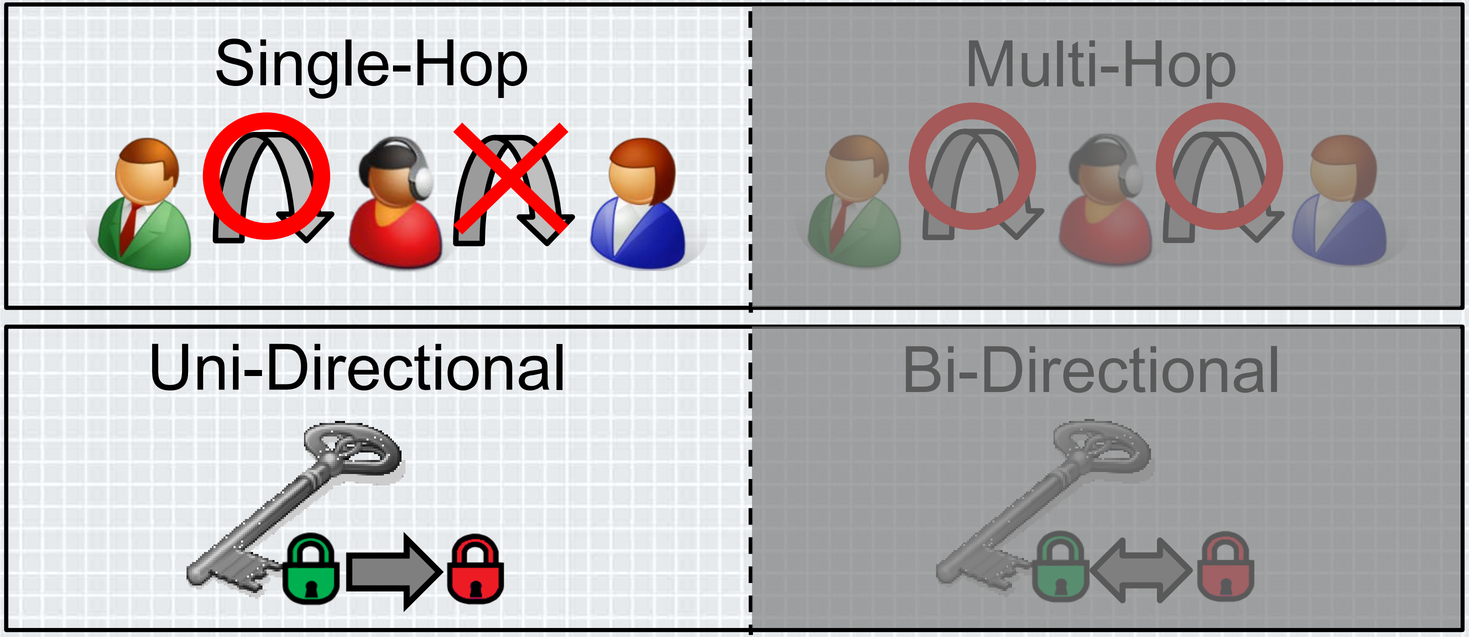
Background and Motivation

Proxy Re-encryption (PRE)

Proxy can change the destination of a ciphertext without decrypting it.

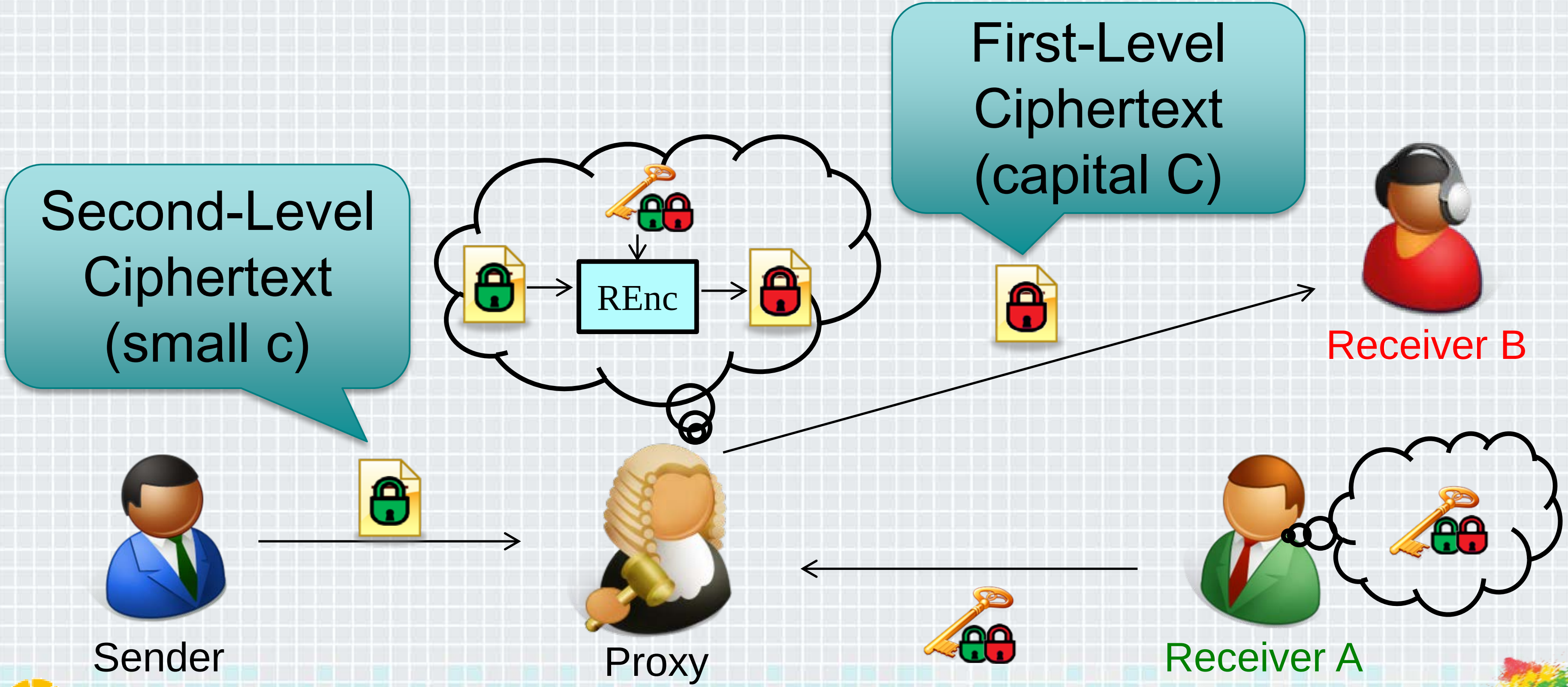


Types of PRE



In this work, we consider a **S**ingle-hop **U**ni-directional PRE.

Second-Level and First-Level Ciphertext



Problem of Previous Works

[LV08] Libert et al. “Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption”, (PKC’08).
[HKK+12] Hanaoka et al. “Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption”, (CT-RSA’12).

Adversary

(pk_j, C) →

Dec1

← m/⊥/test

Challenger

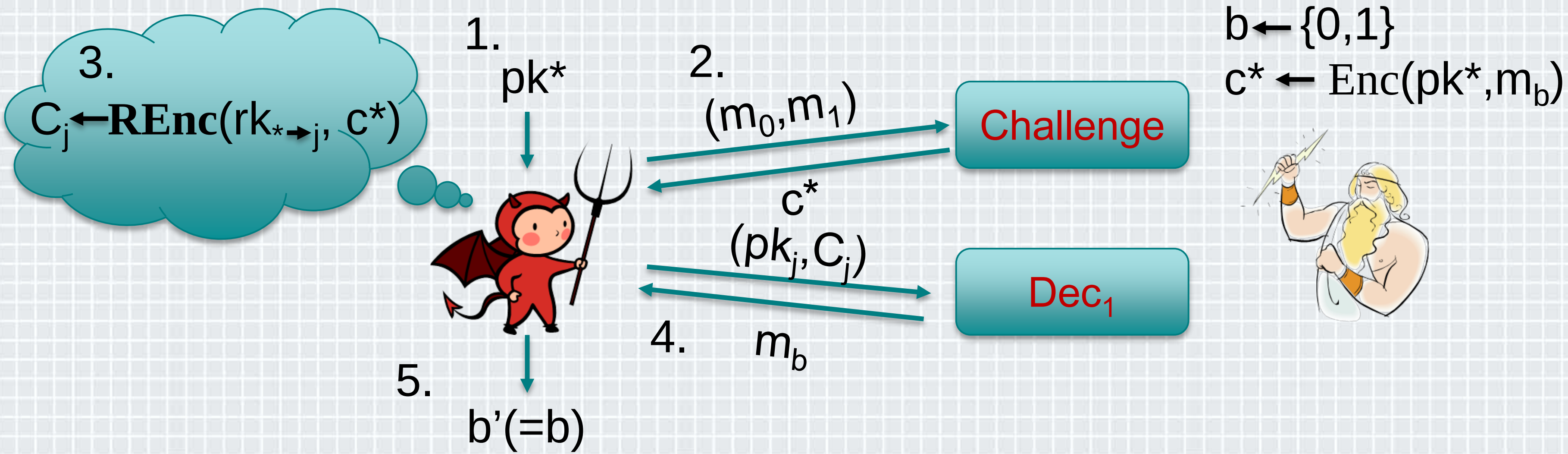
If $\mathrm{Dec}_1(sk_j, C) = \{m_0, m_1\}$,

then return *test*.

Can we avoid this Replayable CCA-like security definition?

Why RCCA?

- There exists an inevitable attack in PRE.



However, the adversary might succeed in generating C_j which satisfies $\mathrm{Dec}_1(sk_j,C_j) \in \{m_0,m_1\}$ without following the above procedure.

Verifiability of Re-encryption

- To solve this problem, we introduce a new functionality that we call "re-encryption verifiability".

(pk_i, pk_j, sk_j, c, C)

↓

REncVer

↓

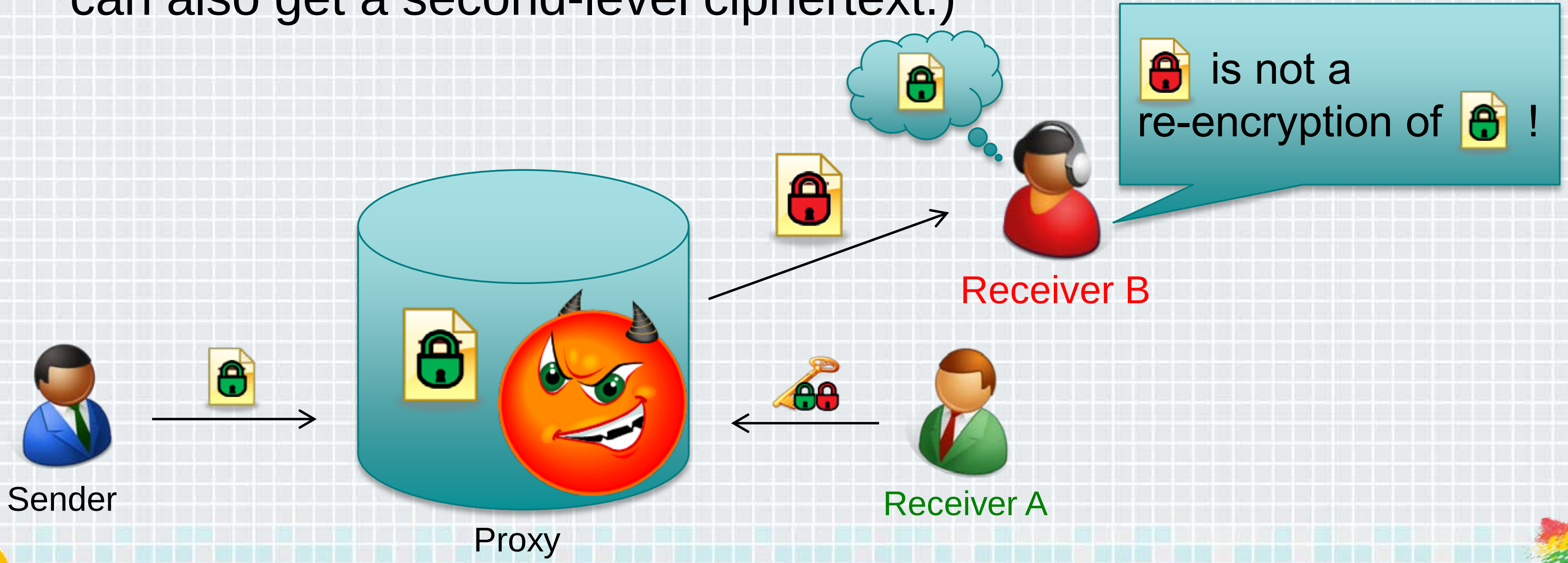
1/0

If C is a re-encrypted ciphertext of c, this REncVer algorithm outputs 1. Otherwise, it outputs 0.

- By using this algorithm, the challenger can distinguish whether the first-level ciphertext which is queried to Dec_1 oracle is a re-encryption of the challenge ciphertext or not.

Practical Merit of Re-encryption Verifiability

- A user who receives a first-level ciphertext can detect malicious activities of a proxy. (We assume the situation in which receiver B can also get a second-level ciphertext.)



Note

- In CT-RSA2013, Isshiki et al. showed new CCA security definitions of PRE and argued that their definitions are stronger than those of Hanaoka et al.'s definitions in CT-RSA2012.
  However, this is not correct.
  - They used the "knowledge of secret key assumption".
  - They showed a counterexample scheme. They argued that their counterexample scheme is secure under the Hanaoka et al.'s definitions, but not secure under their definitions. However, this counterexample scheme is also not secure under the Hanaoka et al.'s definitions.
- These two definitions are incomparable.

RSAConference2015

San Francisco | April 20-24 | Moscone Center

#RSAC

Verifiable Proxy Re-encryption

Syntax of Verifiable PRE (VPRE)

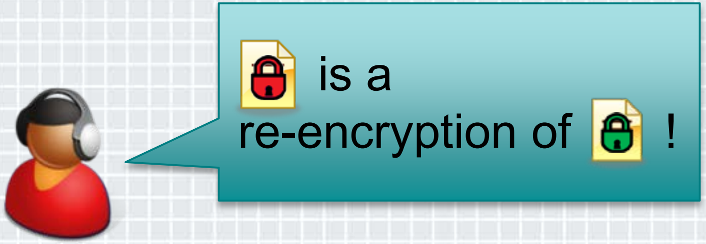
- 1^k → **KG** → (pk, sk)
- (pk, m) → **Enc** → c
- (sk_i, pk_j) → **RKG** → $Rk_{i\rightarrow j}$
- $(rk_{i\rightarrow j}, c_i)$ → **REnc** → C_j
- $(pk_i, pk_j, sk_j, c_i, c_j)$ → **REncVer** → 1/0 New!!
- (sk_j, C) → **Dec1** → $m/\perp$
- (sk_i, c) → **Dec2** → $m/\perp$

Security

- We consider three security definitions.

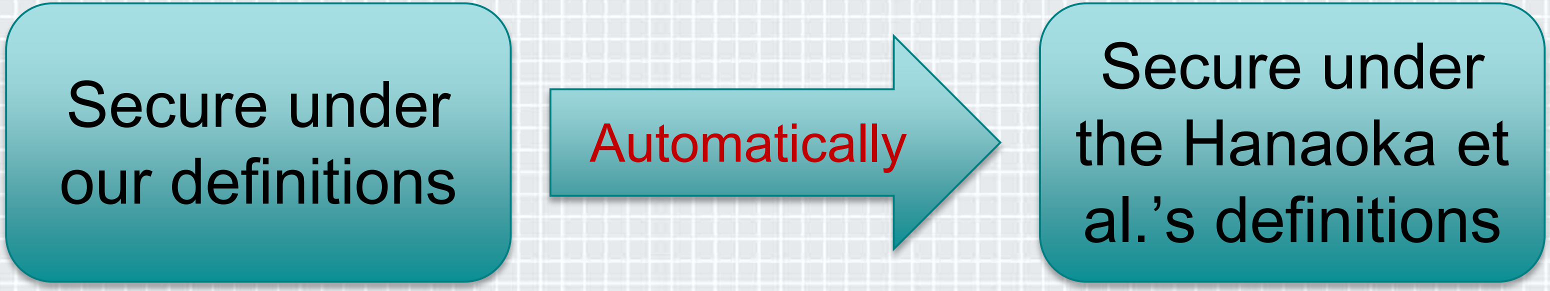
1. CCA security of second-level ciphertexts
2. CCA security of first-level ciphertexts
3. Soundness of Re-encryption Verification
   - If $\mathrm{REncVer}(\cdot,\cdot,\cdot,c,C)$ outputs 1, C is guaranteed to be a re-encryption of c.

$1 \leftarrow \mathrm{REncVer}(pk_A, pk_A, sk_B,$  ,  $)$



Our Contribution 1

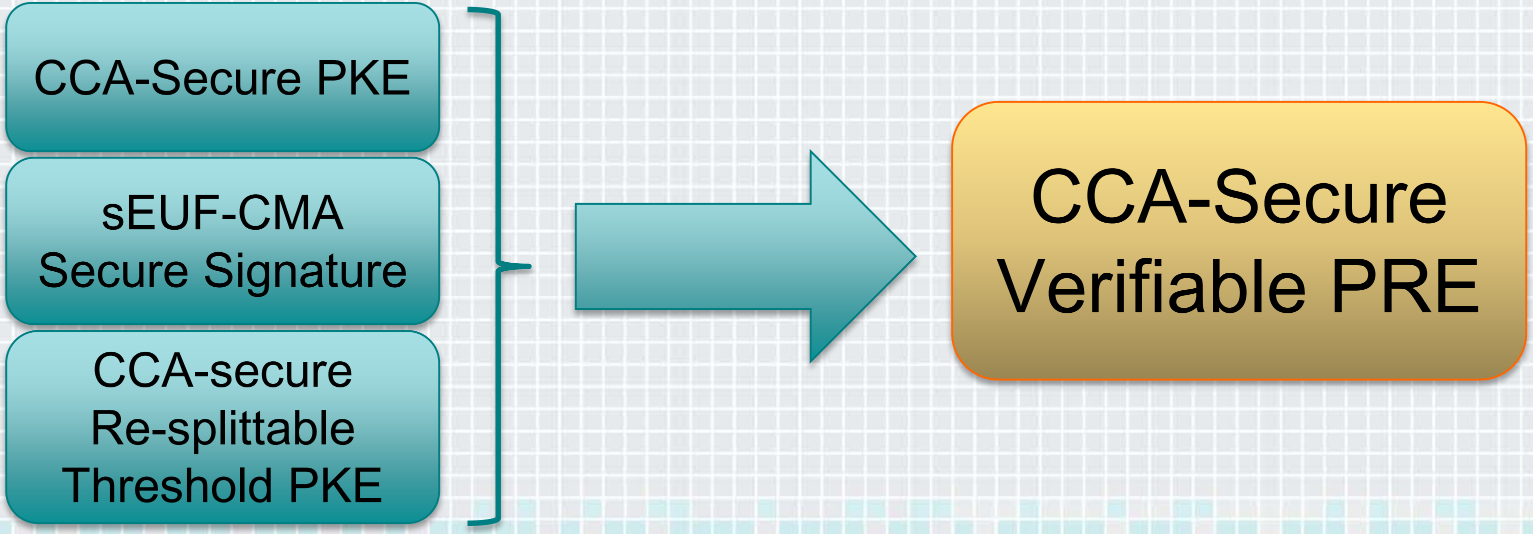
- We prove that a PRE scheme secure under our security definitions of VPRE is secure under the Hanaoka et al.'s security definitions of standard PRE.
  - That is, our new definitions are stronger than Hanaoka et al.'s definitions.



Construction

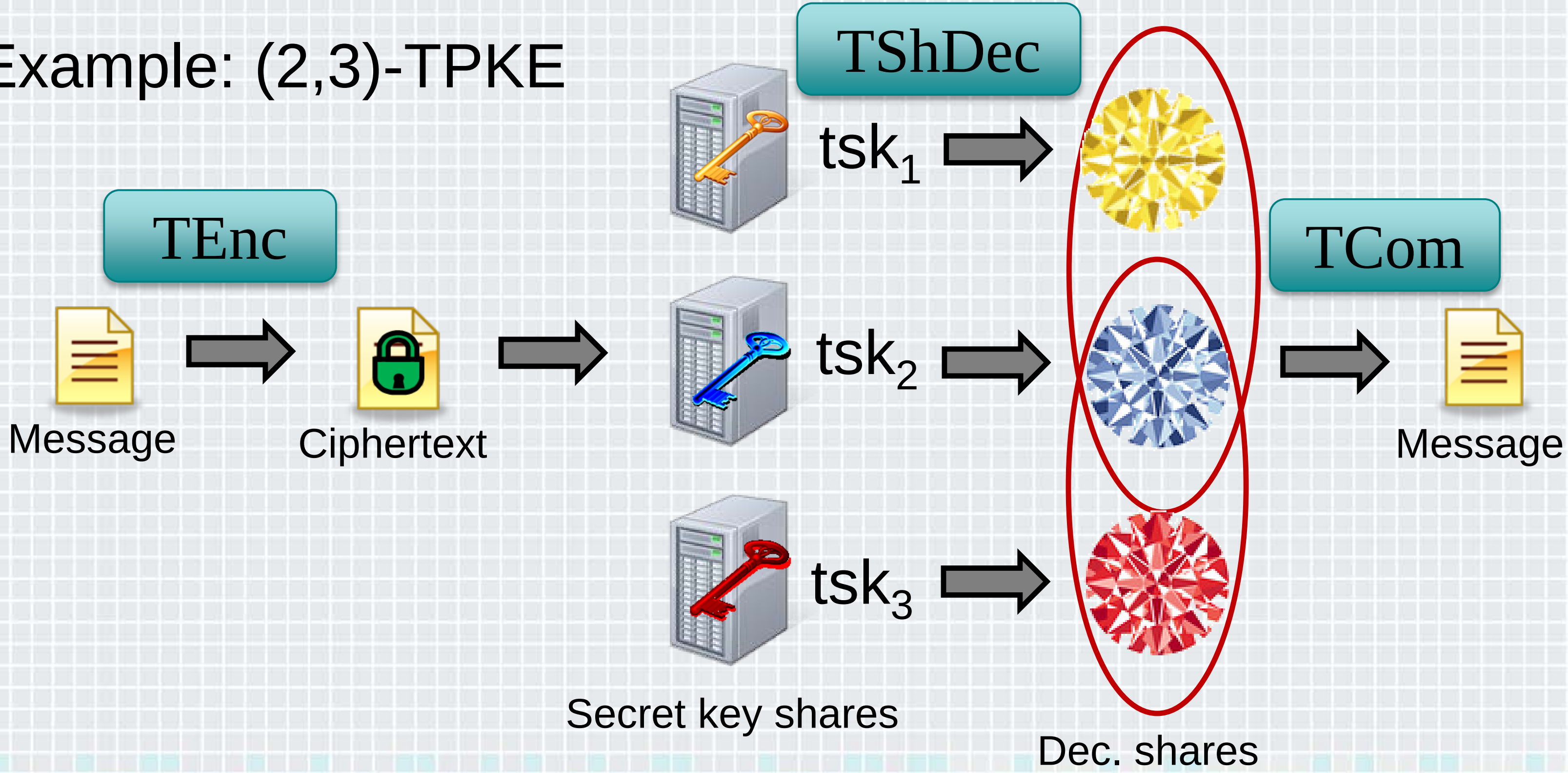
Our Contribution 2

- We extend Hanaoka et al.'s PRE scheme and prove that it satisfies our new definitions of VPRE.
  - The algorithms other than $\mathrm{REncVer}$ is exactly the same as Hanaoka et al.'s generic construction of a standard PRE scheme.



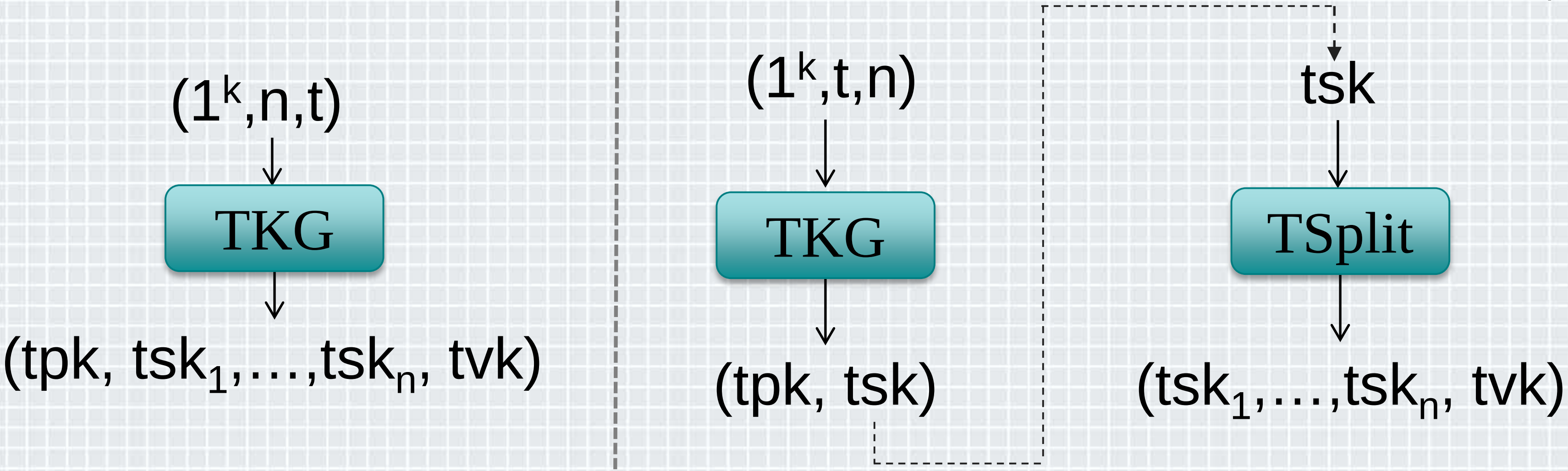
Preliminaries: Threshold PKE (TPKE)

Example: (2,3)-TPKE



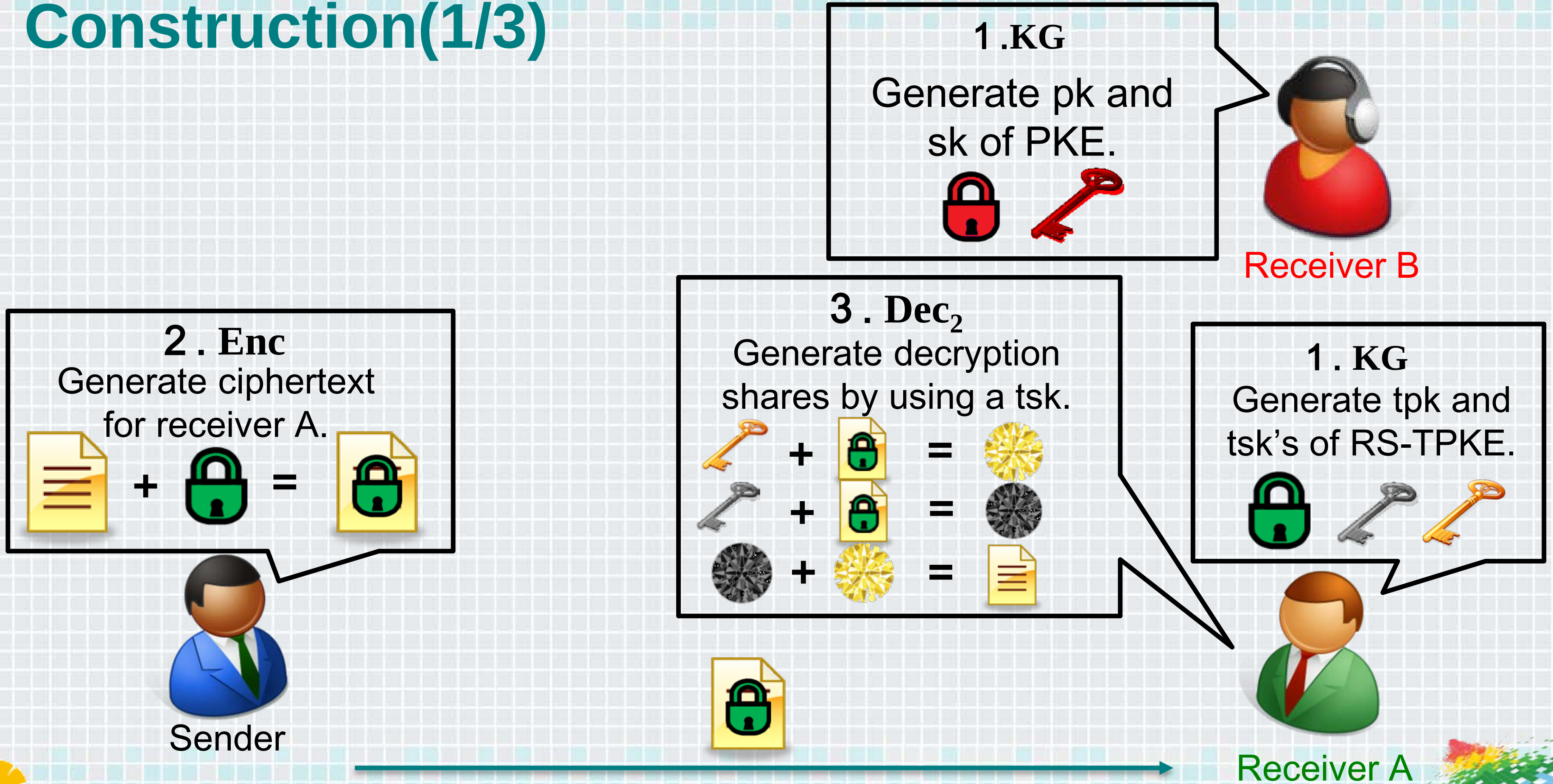
Preliminaries: Re-splittable TPKE (RS-TPKE)

Standard TPKE | Re-splittable TPKE (defined by Hanaoka et al.)

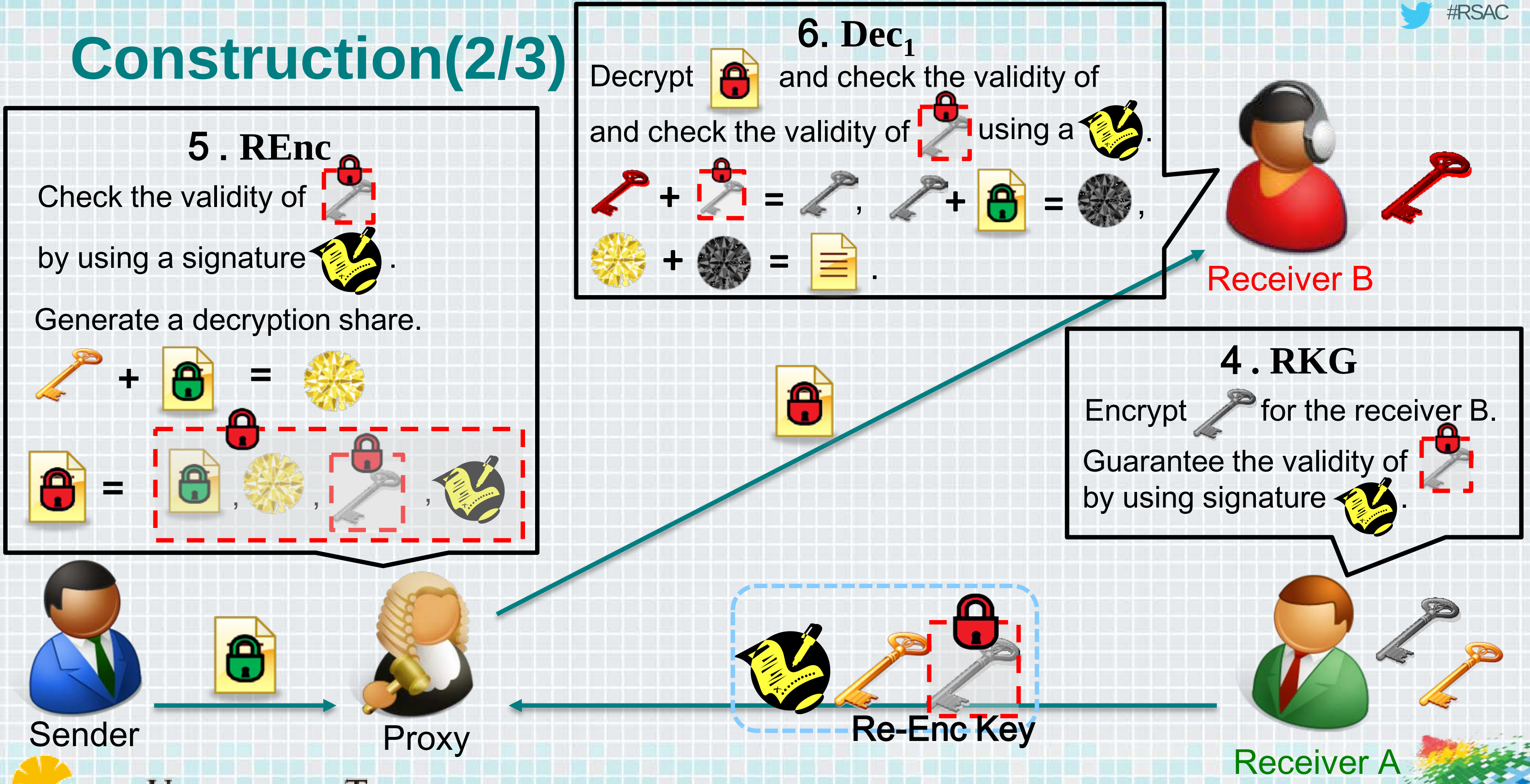


- In a re-splittable TPKE, a secret key can be split many times under the same public key. This time, we need (2,2)-RS-TPKE.

Construction(1/3)



Construction(2/3)



Construction(3/3)

7. REncVer

Decrypt and obtain etc.

If all decryption shares , and are valid, and if is the same ciphertext as the input of REncVer algorithm, then output 1. Otherwise, output 0.

Receiver B

- In our construction, receiver B can recover the second-level ciphertext. Therefore, the functionality of re-encryption verifiability can be achieved by checking the equality.

Conclusion and Future Work

Conclusion and Future Work

- We introduced a new functionality called “re-encryption verifiability” in proxy re-encryption (PRE).
  - enable to avoid the RCCA-like security definitions
  - enable to detect malicious activities of a proxy
- We showed security definitions and a construction.
  - stronger definitions than previous works
  - generic construction based on re-splittable threshold PKE
- Constructing an efficient concrete construction is a future work.

RSAConference2015
San Francisco | April 20-24 | Moscone Center

CHANGE
Challenge today's security thinking

SESSION ID: CRYP-F01

Thank you for your attention!

Re-encryption Verifiability:
How to Detect Malicious Activities of a Proxy in Proxy Re-encryption

(Cryptology ePrint Archive: Report 2015/112)

Satsuya Ohata, Yutaka Kawai, Takahiro Matsuda,
Goichiro Hanaoka, Kanta Matsuura