Innovative R&D by NTT

Public-Key Cryptosystems Resilient to Continuous Tampering and Leakage of Arbitrary Functions

Eiichiro Fujisaki (藤崎 英一郎)   Keita Xagawa (草川 恵太)

NTT Secure Platform Laboratories

ASIACRYPT 2016

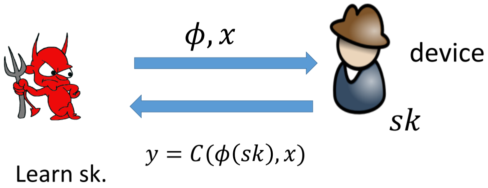
First,

A part of this talk is closely related to Antonio's talk (the previous talk).

- We also analyze Qin-Liu PKE scheme in the tampering attacks with a different setting.
  - bounded tampering vs. continual tampering.
  - standard PKE vs. PKE w/ self-destruction mechanism.
- Our impossible result to signature complements their result on signature.

Agenda

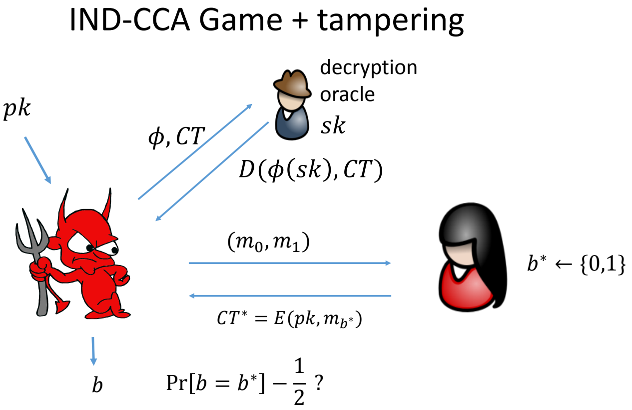
Tampering Attacks



ϕ: *tampering function*, or RKD function.

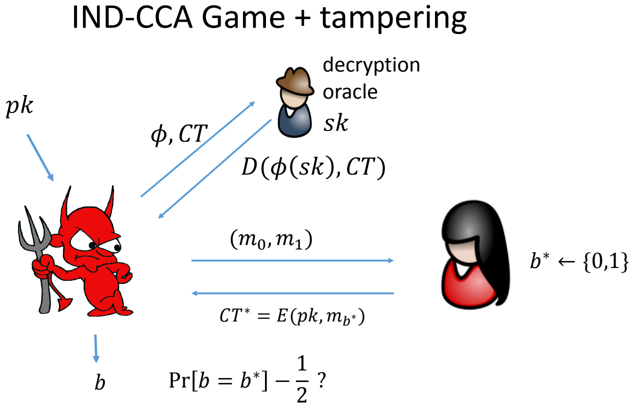
The tampering attacks allow an adversary to modify the secret of a target cryptographic device and observe the effect of the changes at the output (Gennaro etal [GLM+04] and Bellare and Kohno [BK03]).

Mount tampering on the IND-CCA Game.



We focus on tampering attacks with *arbitrary* function ϕ. Then, some restrictions are required.

Mount tampering on the IND-CCA Game.



We focus on tampering attacks with *arbitrary* function ϕ. Then, some restrictions are required.

Impossible Result [GLM$^+$04]

Theorem

There is no IND-CCA *secure (standard) PKE or* EUF-CMA *secure (standard) signature resilient to unbounded polynomial many tamperings of arbitrary function (even in the CRS model or a stronger model (= the ATP model [GLM$^+$04])).*

Proof.

Choose the following $\phi_1, \ldots, \phi_{|sk|}$:

$$\phi_i(sk) = \begin{cases} sk & \text{if the } i\text{-th bit of } sk \text{ is } 0. \\ \bot & \text{otherwise.} \end{cases}$$

By querying with $\phi_1, \ldots, \phi_{|sk|}$, the adversary can retrieve sk from the decyrption or signing oracle. □

To circumvent the impossibility result of [GLM+04]

1. Only allow a bounded number of tampering queries (**Bounded tampering attacks** [DFMV13, FV16]).
   - [FV16]: The previous talk.
2. Allow an unbounded number of tampering queries, but allow a device to self-destruct when it detects tampering (**Continuous tampering w/ self-destruction mechanism** [KKS11]).
   - This talk.
3. Allow an unbounded number of tampering queries, but allow a device to update its secret key (**Continuous tampering w/ key-update mechanism** [KKS11]).
   - This talk.

NTT

To circumvent the impossibility result of [GLM+04]

1. Only allow a bounded number of tampering queries (**Bounded tampering attacks** [DFMV13, FV16]).
   - [FV16]: The previous talk.
2. Allow an unbounded number of tampering queries, but allow a device to self-destruct when it detects tampering (**Continuous tampering w/ self-destruction mechanism** [KKS11]).
   - This talk.
3. Allow an unbounded number of tampering queries, but allow a device to update its secret key (**Continuous tampering w/ key-update mechanism** [KKS11]).
   - This talk.

NTT

To circumvent the impossibility result of [GLM+04]

1. Only allow a bounded number of tampering queries (**Bounded tampering attacks** [DFMV13, FV16]).
   - [FV16]: The previous talk.
2. Allow an unbounded number of tampering queries, but allow a device to self-destruct when it detects tampering (**Continuous tampering w/ self-destruction mechanism** [KKS11]).
   - This talk.
3. Allow an unbounded number of tampering queries, but allow a device to update its secret key (**Continuous tampering w/ key-update mechanism** [KKS11]).
   - This talk.

NTT

To circumvent the impossibility result of [GLM+04]

1. Only allow a bounded number of tampering queries (**Bounded tampering attacks** [DFMV13, FV16]).
   - [FV16]: The previous talk.
2. Allow an unbounded number of tampering queries, but allow a device to self-destruct when it detects tampering (**Continuous tampering w/ self-destruction mechanism** [KKS11]).
   - This talk.
3. Allow an unbounded number of tampering queries, but allow a device to update its secret key (**Continuous tampering w/ key-update mechanism** [KKS11]).
   - This talk.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks $>$ persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks > persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks $>$ persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks > persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks $>$ persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Option: Persistent or Non-Persistent [JW15]

- **Persistent tampering attacks:** A tampering is applied to the current version of the secret overwritten by the previous tampering function.
  - For queries (ϕ_1, x_1) and (ϕ_2, x_2) to device $G(sk, \cdot)$ in this order, receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(\phi_1(sk)), x_2)$.
- **Non-persistent tampering attacks:** A tampering is always applied to the original secret.
  - For the same series of queries above, instead receives $G(\phi_1(sk), x_1)$ and $G(\phi_2(sk), x_2)$.

Remarks.

- **non-key-update:** non-persistent attacks $>$ persistent attacks.
  because one can simulate persistent query $\phi' = \phi_2 \circ \phi_1$ in the non-persistent attack.
- **key-update:** unknown which is stronger.

Another Impossible Result to PKE

Theorem ([DFMV13])

There is no $\mathrm{IND\text{-}CCA}$ secure PKE scheme resilient to even one post-challenge tampering query of arbitrary function.

Proof.

Choose the following ϕ:

$$\phi(sk) = \begin{cases} sk & \text{if } \mathbf{D}(sk, \mathsf{CT}^*) = m_0. \\ \perp & \text{otherwise.} \end{cases}$$

□

This attack is unavoidable even with self-destruction, key-updating, and bounded persistent/non-persistent tampering in the ATP model [GLM+04] (i.e., in the strongest compromised model).

Another Impossible Result to PKE

Theorem ([DFMV13])

There is no $\mathrm{IND\text{-}CCA}$ *secure PKE scheme resilient to even one post-challenge tampering query of arbitrary function.*

Proof.

Choose the following ϕ:

$$\phi(sk) = \begin{cases} sk & \text{if } \mathbf{D}(sk, \mathsf{CT}^*) = m_0. \\ \perp & \text{otherwise.} \end{cases}$$

□

This attack is unavoidable even with self-destruction, key-updating, and bounded persistent/non-persistent tampering in the ATP model [GLM+04] (i.e., in the strongest compromised model).

NTT

Public Parameter: CRS vs Others

We concentrate on the CRS model, because we treat tampering of *arbitrary* functions.

- The CRS model.
  - The CRS model is popular. The CRS ρ is common among all users and is not tampered.
- ATP (algorithmic tamper-proof) Model [GLM$^+$04] (stronger than the CRS model) .
  - The CRS ρ is the verification key of a trusted party. Unlike the CRS model, the trusted party actively signs on secret of each device after publishing ρ.
- Non-CRS models.
  - Possible only for tampering of *a restricted class of* functions (split-state, linear function, etc) .

NTT

Summary of Previous work

Table: Tampering-Resilient Primitives against *arbitrary* tampering functions (in the CRS model).

Prim.	Self-Dest.	Key Update	Tamp.	Security	Model	Notes
PKE			c-tamp	CCA	even in ATP	Impossible [GLM+04]
PKE	✓	✓	b-tamp	CCA	post-challenge. tampering	Impossible [DFMV13]
PKE			b-tamp	CCA	per./n-per.	[DFMV13]
PKE			b-tamp	CCA	per./n-per.	[FV16] (This conference)
PKE		✓	c-tamp	CPA	persist	[KKS11]
PKE	✓		c-tamp	CCA	persist	?
PKE	✓		c-tamp	CCA	n-persist	?
PKE		✓	c-tamp	CCA	persist	?
PKE		✓	c-tamp	CCA	n-persist	?
Sig			c-tamp	CMA	per./n-per.	Impossible [GLM+04]
Sig	✓		c-tamp		persist	KKS [KKS11]
Sig		✓	c-tamp^{-}	CMA	persist	KKS [KKS11]
Sig	✓		c-tamp		n-persist	?
Sig		✓	c-tamp		n-persist	?

b-tamp: bounded tampering. c-tamp: continuous tampering. c-tamp^{-}: somewhat weak continuous tampering.
In non-key-update, n-persist $>$ persist.

Summary of Previous work

Table: Tampering-Resilient Primitives against *arbitrary* tampering functions (in the CRS model).

Prim.	Self-Dest.	Key Update	Tamp.	Security	Model	Notes
PKE			c-tamp	CCA	even in ATP	Impossible [GLM+04]
PKE	✓	✓	b-tamp	CCA	post-challenge. tampering	Impossible [DFMV13]
PKE			b-tamp	CCA	per./n-per.	[DFMV13]
PKE			b-tamp	CCA	per./n-per.	[FV16] (This conference)
PKE		✓	c-tamp	CPA	persist	[KKS11]
PKE	✓		c-tamp	CCA	persist	This work
PKE	✓		c-tamp	CCA	n-persist	This work
PKE		✓	c-tamp	CCA	persist	?
PKE		✓	c-tamp	CCA	n-persist	This work
Sig			c-tamp	CMA	per./n-per.	Impossible [GLM+04]
Sig	✓		c-tamp	CMA	persist	KKS [KKS11]
Sig		✓	c-tamp^{-}		persist	KKS [KKS11]
Sig	✓		c-tamp		n-persist	Impossible (This work)
Sig		✓*	c-tamp		n-persist	Impossible (This work)

b-tamp: bounded tampering. c-tamp: continuous tampering. c-tamp^{-}: somewhat weak continuous tampering.
In non-key-update, n-persist $>$ persist. *: remark (see the next slide).

Our Result

- **[PKE]** The first CCA-secure PKE schemes resilient to continuous (pre-challenge) tampering of *arbitrary* functions.
    - Qin-Liu PKE scheme at ASIACRYPT 13 [QL13] w/ self-destructive mechanism is resilient to *continuous tampering and bounded memory leakage* ($\mathrm{CTBL\text{-}CCA}$ secure).
    - A variant of Agrawal et al.PKE scheme [ADVW13] w/ a key-updating mechanism is resilient to *continuous tampering and continuous memory leakage* ($\mathrm{CTL\text{-}CCA}$ secure).
- **[Sig]** Impossible result: There is no signature scheme resilient to continuous non-persistent tampering even with a self-destructive mechanism.
    - (*) If a key-update mechanism works only when a tampering is detected, then no signature scheme even with a key-update mechanism.

Agenda

Definition: CTBL-CCA Game

Let $\Pi = (\mathsf{Setup}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ be PKE.

- Adversary A is given (ρ, pk) generated by Setup and $\mathbf{K}$, respectively.
- A may submit tampering queries (ϕ, CT) to the decryption oracle D, where D self-destructs if $\mathbf{D}(\phi(sk), \mathsf{CT}) = \perp$; otherwise, returns $\mathbf{D}(\phi(sk), \mathsf{CT})$.
- A may submit leakage queries L to the leakage oracle Leak, and Leak returns $L(sk)$ (if the total leakage bits $\leq \lambda$).
- A makes (m_0, m_1) and receives $\mathsf{CT}^* = \mathbf{E}_{pk}(m_{b^*})$ where $b^* \leftarrow \{0, 1\}$.
- A may submit decryption queries CT ($\neq \mathsf{CT}^*$) to the decryption oracle D, where D self-destructs if $\mathbf{D}(sk, \mathsf{CT}) = \perp$; otherwise, returns $\mathbf{D}(sk, \mathsf{CT})$.
- A returns b.

Π is $\mathrm{CTBL\text{-}CCA}$ secure if $\mathsf{Adv}_{\Pi}^{\text{ctbl-cca}}(\kappa) = |2\Pr[b = b^*] - 1| = \mathsf{negl}(\kappa)$.

To begin with

Consider a $\mathrm{BL\text{-}CCA}$ secure PKE, where $\mathrm{BL\text{-}CCA}$ security $:=$ $\mathrm{IND\text{-}CCA}$ security plus resilience to bounded memory leakage.

To begin with

Consider a $\mathrm{BL\text{-}CCA}$ secure PKE, where $\mathrm{BL\text{-}CCA}$ security := $\mathrm{IND\text{-}CCA}$ security plus resilience to bounded memory leakage.

(Fact) If $|m|$ is smaller than the limit of bounded leakage, then any $\mathrm{BL\text{-}CCA}$ secure PKE is resilient to (at least one) bounded number of tampering.

To begin with

Consider a $\mathrm{BL\text{-}CCA}$ secure PKE, where $\mathrm{BL\text{-}CCA}$ security := $\mathrm{IND\text{-}CCA}$ security plus resilience to bounded memory leakage.

(Fact) If $|m|$ is smaller than the limit of bounded leakage, then any $\mathrm{BL\text{-}CCA}$ secure PKE is resilient to (at least one) bounded number of tampering.

Because one can simulate tampering oracle by using leakage oracle, as $L(\cdot) := \mathbf{D}_{\phi(\cdot)}(\mathsf{CT})$.

To begin with

Consider a $\mathrm{BL\text{-}CCA}$ secure PKE, where $\mathrm{BL\text{-}CCA}$ security := $\mathrm{IND\text{-}CCA}$ security plus resilience to bounded memory leakage.

(Fact) If $|m|$ is smaller than the limit of bounded leakage, then any $\mathrm{BL\text{-}CCA}$ secure PKE is resilient to (at least one) bounded number of tampering.

Because one can simulate tampering oracle by using leakage oracle, as $L(\cdot) := \mathbf{D}_{\phi(\cdot)}(\mathsf{CT})$.

However, this does not work for continuous tampering.

To begin with

Consider a $\mathrm{BL\text{-}CCA}$ secure PKE, where $\mathrm{BL\text{-}CCA}$ security := $\mathrm{IND\text{-}CCA}$ security plus resilience to bounded memory leakage.

(Fact) If $|m|$ is smaller than the limit of bounded leakage, then any $\mathrm{BL\text{-}CCA}$ secure PKE is resilient to (at least one) bounded number of tampering.

Because one can simulate tampering oracle by using leakage oracle, as $L(\cdot) := \mathbf{D}_{\phi(\cdot)}(\mathsf{CT})$.

However, this does not work for continuous tampering.

Even for bounded tampering, this black-box usage of leakage oracle gives very bad bound.

(Reminder) Hash Proof System [CS02]

HPS = (HPS.param, HPS.pub, HPS.priv) is a hash proof system if

- HPS.param(1^κ) outputs params $= (\Lambda, \mathcal{C}, \mathcal{V}, \mathcal{SK}, \mathcal{PK}, \mu)$, where
  - $\mu : \mathcal{SK} \to \mathcal{PK}$.
  - $\mathcal{V}$ is a subset of $\mathcal{C}$
  - hash Λ is projective and γ-entropic.
  - $\{C \mid C \overset{\mathsf{U}}{\leftarrow} \mathcal{V}\}_{\kappa\in\mathbb{N}} \overset{\mathsf{c}}{\approx} \{C' \mid C' \overset{\mathsf{U}}{\leftarrow} \mathcal{C}\backslash\mathcal{V}\}_{\kappa\in\mathbb{N}}$.
- HPS.pub$(pk, C, w) = \Lambda_{sk}(C)$ for $pk = \mu(sk)$ and w is witness of C that belongs to $\mathcal{V}$.
- HPS.priv$(sk, C) = \Lambda_{sk}(C)$ for $C \in \mathcal{C}$.

$\Lambda : \mathcal{SK} \times \mathcal{C} \to \mathcal{K}$: projective and γ-*entropic* if

- projective: For all sk, sk' s.t. $\mu(sk) = \mu(sk')$ and all $C \in \mathcal{V}(\subset \mathcal{C})$, $\Lambda_{sk}(C) = \Lambda_{sk'}(C)$.
- γ-*entropic*: For all $pk \in \mathcal{PK}$, $C \in \mathcal{C}\backslash\mathcal{V}$, and all $K \in \mathcal{K}$,

$$\Pr[K = \Lambda_{sk}(C) | (pk, C)] \leq 2^{-\gamma}.$$

All-But-One Injective (ABO) Fuction

ABO function (called one-time lossy filter in [QL13]) is a weaker version of all-but-one trapdoor function [PW08], where a trapdoor function is replaced by an injective function.

Let A be an ABO function. For only one tag t (called the lossy branch), $A(t, \cdot)$ is lossy, while for all-but-one tags $t'(\neq t)$, $A(t', \cdot)$ is injective.

One cannot distinguish lossy branch t from injective branch t'.

Qin-Liu PKE at ASIACRYPT 2013

Qin-Liu PKE scheme [QL13] is an $\mathrm{IND\text{-}CCA}$ secure and resilient to bounded memory leakage ($\mathrm{BL\text{-}CCA}$ secure).

Qin-Liu PKE: (construction) hash proof system (HPS) + all-but-one injective (ABO) function.

Encryption of m: $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ where $K = \Lambda_{sk}(C)$, and σ is a one-time signature on $(C, m \oplus K, A(vk, K), vk)$ w.r.t. vk.

(Our claim) Put the HPS parameter and ABO public-key A in the CRS. Then, Qin-Liu scheme is $\mathrm{CTBL\text{-}CCA}$ secure, with a self-destruction mechanism.

Qin-Liu PKE at ASIACRYPT 2013

Qin-Liu PKE scheme [QL13] is an $\mathrm{IND\text{-}CCA}$ secure and resilient to bounded memory leakage ($\mathrm{BL\text{-}CCA}$ secure).

Qin-Liu PKE: (construction) hash proof system (HPS) + all-but-one injective (ABO) function.

Encryption of m: $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ where $K = \Lambda_{sk}(C)$, and σ is a one-time signature on $(C, m \oplus K, A(vk, K), vk)$ w.r.t. vk.

(Our claim) Put the HPS parameter and ABO public-key A in the CRS. Then, Qin-Liu scheme is $\mathrm{CTBL\text{-}CCA}$ secure, with a self-destruction mechanism.

Qin-Liu PKE at ASIACRYPT 2013

Qin-Liu PKE scheme [QL13] is an $\mathrm{IND\text{-}CCA}$ secure and resilient to bounded memory leakage ($\mathrm{BL\text{-}CCA}$ secure).

Qin-Liu PKE: (construction) hash proof system (HPS) + all-but-one injective (ABO) function.

Encryption of m: $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ where $K = \Lambda_{sk}(C)$, and σ is a one-time signature on $(C, m \oplus K, A(vk, K), vk)$ w.r.t. vk.

(Our claim) Put the HPS parameter and ABO public-key A in the CRS. Then, Qin-Liu scheme is $\mathrm{CTBL\text{-}CCA}$ secure, with a self-destruction mechanism.

Useful Lemma

Lemma

For any random variables, X and Z,

$$\mathsf{H}_\infty(X|Z=z) \geq \mathsf{H}_\infty(X) - \log\left(\frac{1}{\Pr[Z=z]}\right).$$

Useful Lemma

Lemma

For any random variables, X and Z,

$$\mathrm{H}_\infty(X|Z=z) \geq \mathrm{H}_\infty(X) - \log\left(\frac{1}{\Pr[Z=z]}\right).$$

Proof.

For any $z \in Z$,

$$-\log\left(\max_x\left(\Pr[X=x|Z=z]\right)\right) = -\log\left(\max_x\left(\frac{\Pr[X=x \wedge Z=z]}{\Pr[Z=z]}\right)\right)$$

$$\geq -\log\left(\max_x\left(\Pr[X=x]\right)\right) - \log\left(\frac{1}{\Pr[Z=z]}\right).$$

□

Observation

Let $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ be a query ciphertext of Qin-Liu PKE and $K^* = \Lambda_{sk}(C^*)$ be the challenge hash in CT^* (in the simulation: $C^* \notin \mathcal{V}$).

- (1) When $\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) = \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_0),$$

  where $p_0 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp]$.
- (2) When $\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_1)$$

  where $p_1 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp]$.

Observation

Let $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ be a query ciphertext of Qin-Liu PKE and $K^* = \Lambda_{sk}(C^*)$ be the challenge hash in CT^* (in the simulation: $C^* \notin \mathcal{V}$).

- (1) When $\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) = \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_0),$$

  where $p_0 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp]$.
- (2) When $\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_1)$$

  where $p_1 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp]$.

(1) immediately follows from the useful lemma.

Observation

Let $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ be a query ciphertext of Qin-Liu PKE and $K^* = \Lambda_{sk}(C^*)$ be the challenge hash in CT^* (in the simulation: $C^* \notin \mathcal{V}$).

- (1) When $\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) = \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_0),$$

  where $p_0 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp]$.
- (2) When $\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_1)$$

  where $p_1 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp]$.

(1) immediately follows from the useful lemma.
But, how about (2)?

Observation

Let $\mathsf{CT} = (C, m \oplus K, A(vk, K), vk, \sigma)$ be a query ciphertext of Qin-Liu PKE and $K^* = \Lambda_{sk}(C^*)$ be the challenge hash in CT^* (in the simulation: $C^* \notin \mathcal{V}$).

- (1) When $\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) = \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_0),$$

  where $p_0 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) = \perp]$.
- (2) When $\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp$,

$$\mathrm{H}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp) \geq \mathrm{H}_\infty(K^*) - \log(1/p_1)$$

  where $p_1 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp]$.

(1) immediately follows from the useful lemma.
But, how about (2)? Except for revealing the fact $\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp$, it apparently reveals message $\mathbf{D}(\phi(SK), \mathsf{CT})$...

Observation, Cont.

However, the entropy of $\mathbf{D}(\phi(SK), \mathsf{CT})$ is zero, given CT, because of injective $A(vk, K)$. Therefore,

$$\begin{aligned} \widetilde{\mathsf{H}}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp) &\geq \widetilde{\mathsf{H}}_\infty(K^* | \mathbf{D}(\phi(SK), \mathsf{CT})) - \log(1/p_1) \\ &= \widetilde{\mathsf{H}}_\infty(K^* | \Lambda_{\phi(SK)}(C)) - \log(1/p_1) \\ &= \widetilde{\mathsf{H}}_\infty(K^* | K) - \log(1/p_1) \\ &= \mathsf{H}_\infty(K^*) - \log(1/p_1) \end{aligned}$$

where $p_1 = \Pr[\mathbf{D}(\phi(SK), \mathsf{CT}) \neq \perp]$.

Now,

Let p_i $(1 \leq i < \ell)$ be the probability that **D** does not reject i-th query ciphertext. Let p_ℓ be the probability that **D** rejects ℓ-th query ciphertext.

Note that there is a trade-off between leakage bit $\log(1/p)$ and probability p, i.e., If $\log(1/p)$ is big, then p is small, and vice versa.

Now,

Let p_i $(1 \leq i < \ell)$ be the probability that $\mathbf{D}$ does not reject i-th query ciphertext. Let p_ℓ be the probability that $\mathbf{D}$ rejects ℓ-th query ciphertext.

Note that there is a trade-off between leakage bit $\log(1/p)$ and probability p, i.e., If $\log(1/p)$ is big, then p is small, and vice versa.

If the total leakage bits from all tampering queries $\sum_{i=1}^{\ell} \log(1/p_i) \geq \omega(\log \kappa)$, then the probability that occurs is

$$\prod_{i=1}^{\ell} p_i = 2^{-\omega(\log \kappa)} = \mathsf{negl}(\kappa).$$

Now,

Let p_i $(1 \leq i < \ell)$ be the probability that $\mathbf{D}$ does not reject i-th query ciphertext. Let p_ℓ be the probability that $\mathbf{D}$ rejects ℓ-th query ciphertext.

Note that there is a trade-off between leakage bit $\log(1/p)$ and probability p, i.e., If $\log(1/p)$ is big, then p is small, and vice versa.

If the total leakage bits from all tampering queries $\sum_{i=1}^{\ell} \log(1/p_i) \geq \omega(\log \kappa)$, then the probability that occurs is

$$\prod_{i=1}^{\ell} p_i = 2^{-\omega(\log \kappa)} = \mathsf{negl}(\kappa).$$

So, Qin-Liu PKE reveals at most $\omega(\log \kappa)$ bits against tampering attacks w/ overwhelming prob.

To sum up,

Qin-Liu PKE reveals at most $\omega(\log \kappa)$ bits against tampering attacks.

Qin-Liu PKE is $\mathrm{BL\text{-}CCA}$ secure and can afford $O(\kappa)$ bit memory leakage.

- Instantiations: $(1 - o(1))|SK|$ from DCR. $\frac{1}{4}(1 - o(1))|SK|$ from DDH, where $|SK| = O(\kappa)$.

Therefore, Qin-Liu PKE is $\mathrm{CTBL\text{-}CCA}$ secure.

Agenda

Remark

The $\mathrm{CTBL\text{-}CCA}$ security notion does not imply the $\mathrm{IND\text{-}CCA}$ security notion, because the decryption oracle self-destructs even when it receives an invalid ciphertext under the original secret sk – it cannot distinguish a tampering query from a normal decryption query.

The $\mathrm{CTL\text{-}CCA}$ security notion implies the $\mathrm{IND\text{-}CCA}$ security notion.

Definition: PKE with a Key-Update mechanism [BKKV10]

$\Pi = (\mathsf{Setup}, \mathsf{Update}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ is PKE with a key-update mechanism if

- $(\mathsf{Setup}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ is a standard PKE and
- Update takes sk and updates it to sk' (with fresh randomness) without changing pk.

Definition: CTL-CCA Game

Let $\Pi = (\mathsf{Setup}, \mathsf{Update}, \mathbf{K}, \mathbf{E}, \mathbf{D})$ be PKE with key-update.

- Adversary A is given (ρ, pk) generated by Setup and $\mathbf{K}$, respectively.
- A may submit tampering queries (ϕ, CT) to the decryption oracle D, and D returns $\mathbf{D}(\phi(sk), \mathsf{CT})$. If $\mathbf{D}(\phi(sk), \mathsf{CT}) = \perp$, then D updates sk to sk'.
- A may submit leakage queries L to the leak oracle Leak, and Leak returns $L(sk)$ (if the total leak bits $\leq \lambda$ for the same sk).
- A makes (m_0, m_1) and receives $\mathsf{CT}^* = \mathbf{E}_{pk}(m_{b^*})$ where $b^* \leftarrow \{0, 1\}$.
- A may submit decryption queries CT $(\neq \mathsf{CT}^*)$ to the decryption oracle D and D returns $\mathbf{D}(sk, \mathsf{CT})$. If $\mathbf{D}(sk, \mathsf{CT}) = \perp$, then D updates sk to sk'.
- A returns b.

Π is $\mathrm{CTL\text{-}CCA}$ secure if $\mathsf{Adv}_{\Pi}^{\mathsf{ctl\text{-}cca}}(\kappa) = |2\Pr[b = b^*] - 1| = \mathsf{negl}(\kappa)$.

Reminder: Why is Qin-Liu PKE CTBL-CCA secure ?

Remember Qin-Liu PKE (= HPS+ABO).

- HPS makes $\mathrm{BL\text{-}CPA}$ secure PKE.
- ABO transforms $\mathrm{BL\text{-}CPA}$ secure PKE to $\mathrm{BL\text{-}CCA}$ secure one (proven by Qin and Liu), and also keeps it small to reveal secret key *sk* by answering *one* tampering query.

NTT

Reminder: Why is Qin-Liu PKE CTBL-CCA secure ?

Remember Qin-Liu PKE (= HPS+ABO).

- HPS makes $\mathrm{BL\text{-}CPA}$ secure PKE.
- ABO transforms $\mathrm{BL\text{-}CPA}$ secure PKE to $\mathrm{BL\text{-}CCA}$ secure one (proven by Qin and Liu), and also keeps it small to reveal secret key *sk* by answering *one* tampering query.

Although the leakage is small for one tampering, it is leaked step by step. So, the self-destruction is needed. The decryption algorithm can detect tampering before it reveals too much.

Reminder: Why is Qin-Liu PKE CTBL-CCA secure ?

Remember Qin-Liu PKE (= HPS+ABO).

- HPS makes $\mathrm{BL\text{-}CPA}$ secure PKE.
- ABO transforms $\mathrm{BL\text{-}CPA}$ secure PKE to $\mathrm{BL\text{-}CCA}$ secure one (proven by Qin and Liu), and also keeps it small to reveal secret key *sk* by answering *one* tampering query.

Although the leakage is small for one tampering, it is leaked step by step. So, the self-destruction is needed. The decryption algorithm can detect tampering before it reveals too much.

(Observation) If there is HPS with a key-update mechanism, then, by combining it with ABO, we can construct $\mathrm{CTL\text{-}CCA}$ secure PKE.

ADVW PKE Scheme at ASIACRYPT 2013

Agrawal et al. [ADVW13] PKE scheme is hash proof system based and $\mathrm{IND\text{-}CPA}$ secure and resilient to continuous leakage in the floppy disk model.

The floppy disk model: There are two secret-keys, sk and usk, for a user.

- sk is used for decryption, which is the target of leakage.
- usk is not revealed and is used to update sk to sk' (with fresh randomness), i.e., $sk' \leftarrow \mathsf{Update}(usk, sk)$.

(Goal) Modify the key-update algorithm in the floppy disk model to one in the key-update model [BKKV10], such as $sk' \leftarrow \mathsf{Update}(sk)$.

Proof Idea (CTL-CCA)

There are two steps.

- A hash proof system in Agrawal et al. [ADVW13] is defined on an ordinary prime order group. We translate it in bilinear groups, which makes it possible to key-update without other secret.
- For security proof, we modify the random subspace lemma in [ADVW13].

Proof Idea (CTL-CCA)

The Agrawal et al.version of Random subspace lemma [ADVW13].

Lemma

Let $2 \le d < t \le n$ *and* $\lambda < (d-1)\log(q)$. *Let* $\mathcal{W} \subset \mathbb{F}_q^n$ *be an arbitrary vector subspace in* $\mathbb{F}_q^n$ *of dimension* t. *Let* $L : \{0,1\}^* \to \{0,1\}^\lambda$ *be an arbitrary function. Then, we have*

$$\mathrm{Dist}\left(\left(\mathbf{A}, L(\mathbf{A}\vec{v})\right), \left(\mathbf{A}, L(\vec{u})\right)\right) = \mathrm{negl}(\kappa)$$

where $\mathbf{A} := (\vec{a_1}, \ldots, \vec{a_d}) \leftarrow \mathcal{W}^d$ *(seen as a* $n \times d$ *matrix),* $\vec{v} \leftarrow \mathbb{F}_q^d$*, and* $\vec{u} \leftarrow \mathcal{W}$.

Proof Idea (CTL-CCA), Ctd.

We instead use the random sub subspace lemma in this work.

Lemma

Let $2 \le d \le t' < t \le n$ and $\lambda < (d-1)\log(q)$. Let $\mathcal{W} \subset \mathbb{F}_q^n$ be an arbitrary vector subspace in $\mathbb{F}_q^n$ of dimension t. Let $L : \{0,1\}^* \to \{0,1\}^\lambda$ be an arbitrary function. Then, we have

$$\mathrm{Dist}\left(\left(\mathbf{A}, L(\mathbf{A}\vec{v})\right), \left(\mathbf{A}, L(\vec{u})\right)\right) = \mathrm{negl}(\kappa),$$

where $\mathcal{W}'$ is a random vector subspace in $\mathcal{W}$ of dimension t' (independent of function L), $\mathbf{A} := (\vec{a}_1, \ldots, \vec{a}_d) \leftarrow \mathcal{W}'^d$ (seen as a $n \times d$ matrix), $\vec{v} \leftarrow \mathbb{F}_q^d$, and $\vec{u} \leftarrow \mathcal{W}$.

Then, we succeed in constructing a $\mathrm{CTL\text{-}CCA}$ secure PKE scheme.

Agenda

Impossibility result to SIG

Theorem

There is no EUF-CMA *signature resilient to unbounded polynomial many non-persistent tamperings of arbitary function even with a key-destruction mechanisim.*

Proof.

The adversary runs the key-generation algorithm, Gen, and obtains two legitimate key pairs, (vk_0, sk_0) and (vk_1, sk_1). Then, it sets a set of functions $\{\phi^i_{(sk_0,sk_1)}\}$, such that

$$\phi^i_{(sk_0,sk_1)}(sk) = \begin{cases} sk_0 & \text{if the } i\text{-th bit of } sk \text{ is } 0, \\ sk_1 & \text{otherwise.} \end{cases}$$

For query $(\phi^i_{(sk_0,sk_1)}, m)$, the adversary can obtain i-th bit of sk while the signing oracle cannot detect tampering. □

Agenda

Summary

- **[PKE]** The first CCA-secure PKE schemes resilient to continuous (pre-challenge) tampering of *arbitrary* functions.
  - Qin-Liu PKE scheme at ASIACRYPT 13 [QL13] w/ self-destructive mechanism is resilient to *continuous tampering and bounded memory leakage* ($\mathrm{CTBL\text{-}CCA}$ secure).
  - A variant of Agrawal et al.PKE scheme [ADVW13] w/ a key-updating mechanism is resilient to *continuous tampering and continuous memory leakage* ($\mathrm{CTL\text{-}CCA}$ secure).
- **[Sig]** Impossible result: There is no signature scheme resilient to continuous non-persistent tampering even with a self-destructive mechanism.
  - (*) If a key-update mechanism works only when a tampering is detected, then no signature scheme even with a key-update mechanism.

Comparison

Table: Tampering-Resilient Primitives against arbitrary tampering functions.

Prim.	Self-Dest.	Key Update	Tamp.	Leak	Security	Model	Notes
PKE			c-tamp		CCA	even in ATP	Impossible [GLM+04]
PKE	✓	✓	b-tamp		CCA	post-cha. tampering	Impossible [DFMV13]
PKE			b-tamp	b-leak	CCA	per./n-per.	[DFMV13]
PKE		✓	c-tamp	c-leak^{-}	CCA	Floppy	[DFMV13]
PKE			b-tamp	b-leak	CCA	per./n-per.	[FV16]
PKE		✓	c-tamp	c-leak	CPA	persist	[KKS11]
PKE	✓		c-tamp	b-leak	CCA	per./n-per.	This work
PKE		✓	c-tamp	c-leak	CCA	persist	?
PKE		✓	c-tamp	c-leak	CCA	n-persist	This work
Sig			c-tamp		CMA	per./n-per.	Impossible [GLM+04]
Sig	✓		c-tamp	b-leak	?	persist	KKS [KKS11]
Sig		✓	c-tamp^{-}	c-leak	CMA	persist	KKS [KKS11]
Sig	✓		c-tamp		CMA	n-persist	Impossible
Sig		(✓*)	c-tamp		CMA	n-persist	Impossible (This work)

b-tamp: bounded tampering. c-tamp: continuous tampering.

References I

[ADVW13] Shweta Agrawal, Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs.
On continual leakage of discrete log representations.
In Sako and Sarkar [SS13], pages 401–420.

[BK03] Mihir Bellare and Tadayoshi Kohno.
A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications.
In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, Heidelberg, 2003.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan.
Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage.
In *FOCS 2010*, pages 501–510. IEEE Computer Society, 2010.

[CS02] Ronald Cramer and Victor Shoup.
Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption.
In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Heidelberg, 2002.

[DFMV13] Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi.
Bounded tamper resilience: How to go beyond the algebraic barrier.
In Sako and Sarkar [SS13], pages 140–160.
See also `http://eprint.iacr.org/2013/677` and `http://eprint.iacr.org/2013/124`.

[FV16] Antonio Faonio and Daniele Venturi.
Efficient public-key cryptography with bounded leakage and tamper resilience.
IACR Cryptology ePrint Archive, 2016:529, 2016.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin.
Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering.
In Moni Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, Heidelberg, 2004.

NTT

References II

[JW15] Zahra Jafargholi and Daniel Wichs.
Tamper detection and continuous non-malleable codes.
In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015 (1)*, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480. Springer, Heidelberg, 2015.
See also `http://eprint.iacr.org/2014/956`.

[KKS11] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai.
Cryptography with tamperable and leaky memory.
In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 373–390. Springer, Heidelberg, 2011.

[PW08] Chris Peikert and Brent Waters.
Lossy trapdoor functions and their applications.
In Richard E. Ladner and Cynthia Dwork, editors, *STOC 2008*, pages 187–196. ACM, 2008.

[QL13] Baodong Qin and Shengli Liu.
Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter.
In Sako and Sarkar [SS13], pages 381–400.

[SS13] Kazue Sako and Palash Sarkar, editors.
Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II, volume 8270 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 2013.

Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions

Thank you! （完）