PKC 2014
@Buenos Aires
3/26～3/28

Chosen Ciphertext Security via UCE

Takahiro Matsuda (RISEC, AIST)

Goichiro Hanaoka (RISEC, AIST)

t-matsuda@aist.go.jp

2014/3/26 Wed.

This Work

- **UCE**: Universal Computational Extractor [Bellare et al.@CRYPTO'13]
  - ＝Standard model security notion for a family of hash functions that "behave like a random oracle"
- We ask: **CPA PKE** **+** **UCE** **??** → **CCA PKE**
- **Our results:**

CPA PKE **+** **UCE**

Negative ☹ Fujisaki-Okamoto → **counterexample**
- **CPA PKE**
- **CCA1 PKE** (for random messages)

Positive ☺ Dolev-Dwork-Naor (DDN) →
- **CCA PKE** (via KEM)
- **CCA Deterministic PKE** (with some constraint)

Outline

- Background, Motivation, Results
- Definitions for UCE
- Negative Results
- Positive Results

Random Oracles and Their Problems

- Random Oracle (RO) Model [Bellare-Rogaway@CCS'93]
  ≒ View a <u>cryptographic hash function</u> as a random function

  SHA1, Keccak, etc.

- Using ROs, many efficient and simple constructions are possible ☺
  - PKE (OAEP, etc.), Signature (FDH, PSS, etc.), more

- However, ROs have several problems ☹
  - [CGH98] : a scheme secure in RO model, insecure in the std. model
  - [Nielsen02]: a primitive that is only achievable using a RO

➔**In general, constructions and security proofs w/o ROs are desirable**

Universal Computational Extractor (UCE) [Bellare et al. @CRYPTO'13]

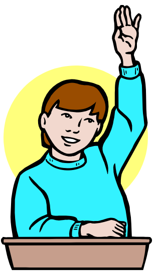
- ＝Standard model security notion for a family of (hash) functions that "behave like random oracle"
  - Purpose：To instantiate ROs in RO-based constructions

- [Bellare et al.@CRYPTO'13] showed simple (and potentially efficient) constructions of cryptographic primitives whose (efficient) constructions were only known in the RO model

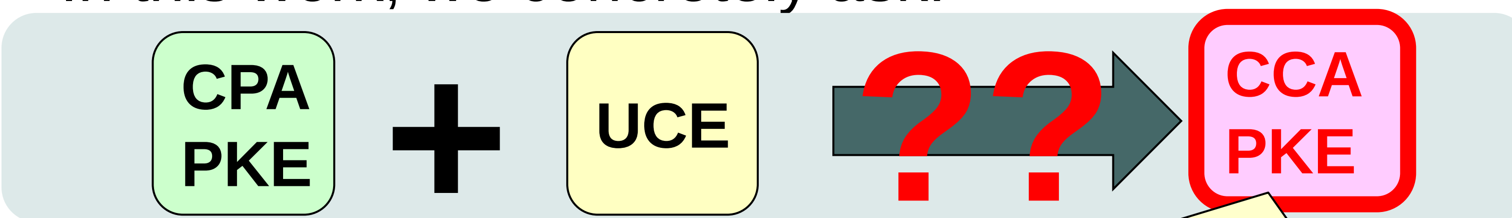
  - PRIV-secure deterministic PKE
  - Related-key secure & KDM secure SKE
  - Point function obfuscation
  - Message-Locked Encryption
  - CPA secure instantiation of OAEP
  - Adaptively secure garbling schemes
  - etc.

UCE is quite powerful!!

Our Motivation



- UCE is new, and have not been understood well
- **Q. Is UCE useful for constructing <u>other primitives</u>?**
- In this work, we concretely ask:



One of the most important cryptographic primitives

- CCA security = de-facto standard security of PKE used in practice
  - implies NM, UC, security against Bleichenbacher's attack

A number of practical constructions using ROs are known:

- OAEP, Fujisaki-Okamoto, SAEP, REACT, OAEP+, etc.

Our Results

- We ask: CPA PKE + UCE ?? → CCA PKE

- **Our results:**

CPA PKE + UCE

Negative ☹ Fujisaki-Okamoto → counterexample

- CPA PKE
- CCA1 PKE (for random messages)

Positive ☺ Dolev-Dwork-Naor (DDN) →

- CCA PKE (via KEM)
- CCA Deterministic PKE (with some constraint)

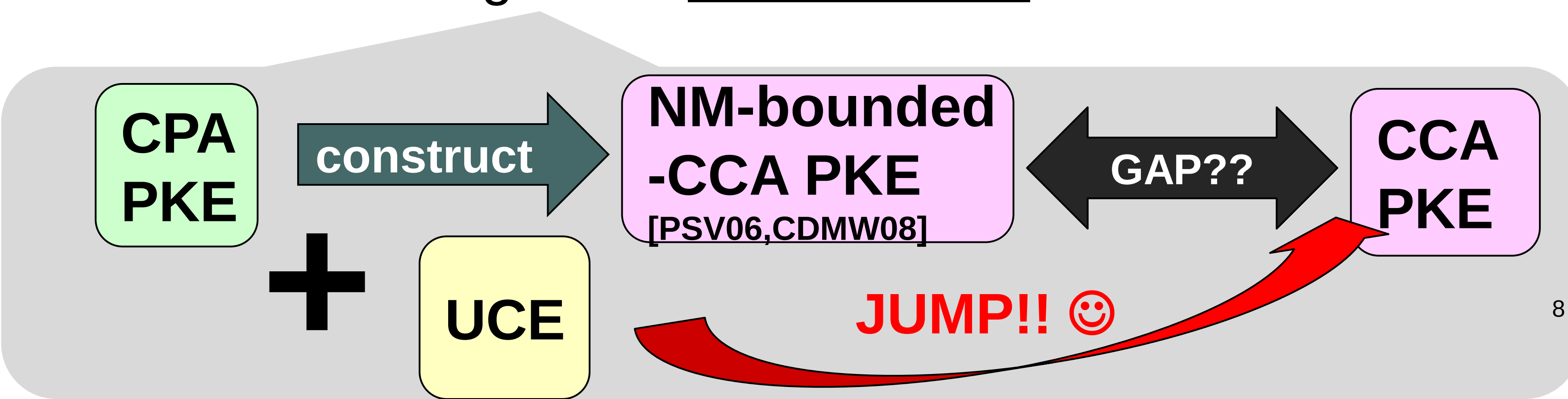
We also do some abstraction of the "core" of the DDN construction as tag-based encryption (TBE)

Interpretation of Our Results

- Negative results:
  - UCE is not as powerful as ROs
  - Our positive results are non-trivial

c.f.)
・[MH@TCC'14]
・[Dachman-Soled@PKC'14]

- Positive results
  - Imply that the DDN construction is quite powerful
  - Give us insights for CPA vs. CCA



Outline

- Background, Motivation, Results
- Definitions for UCE
- Negative Results
- Positive Results

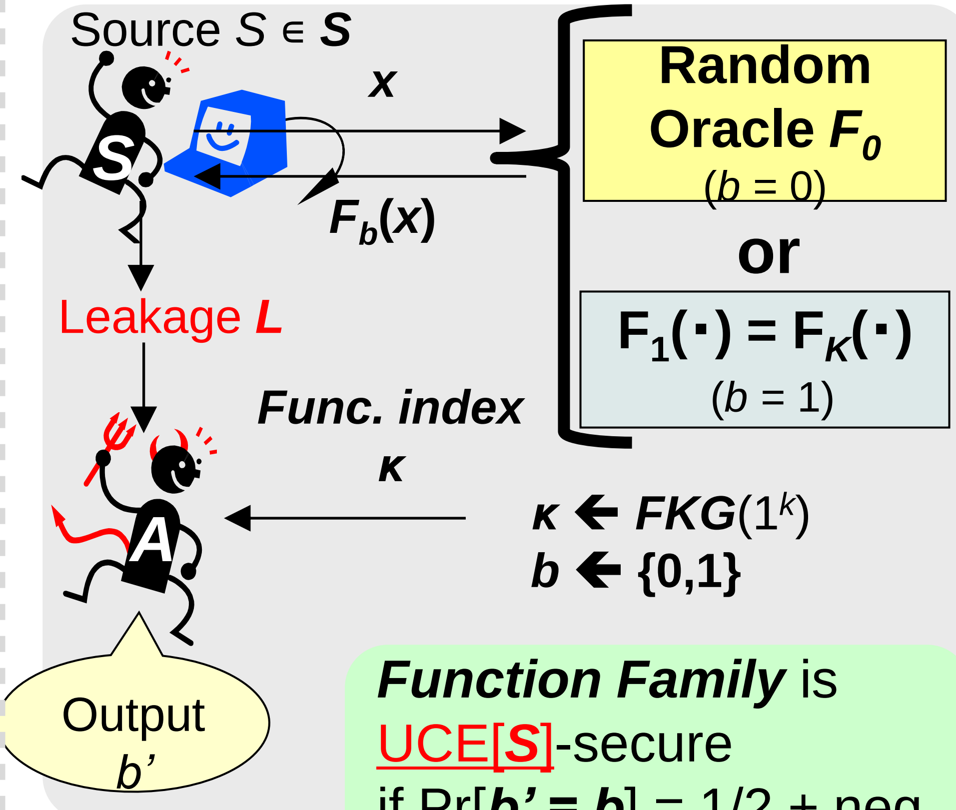
Family of Functions and UCE Security

- A family of functions (function family) consists of (***FKG***, ***F***)

Key Generation	κ ← ***FKG***(1^k)
Evaluation	y ← $F_\kappa(x)$

κ : function index

- UCE security for source class ***S*** (UCE[***S***] security)



Function Family is UCE[***S***]-secure if Pr[***b' = b***] = 1/2 + neg. for ∀ S ∊ ***S*** and ∀ PPT ***A***

Family of Functions and UCE Security

- A family of functions (function family) consists of (***FKG***, ***F***)

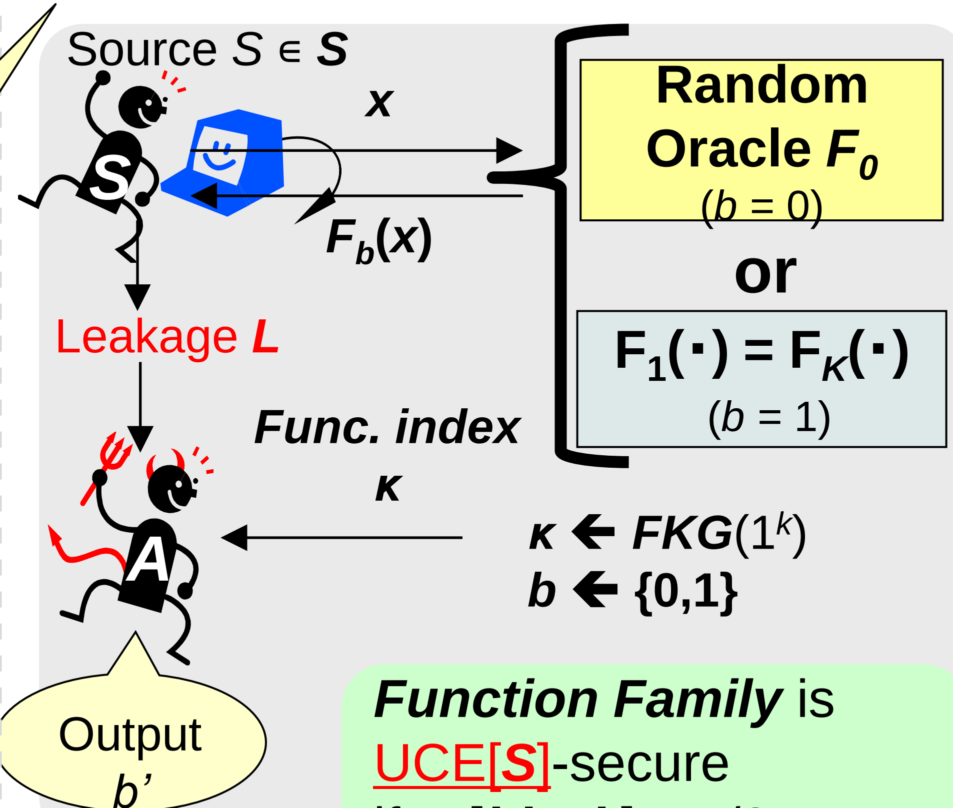
Key Generation	$\kappa \leftarrow \boldsymbol{FKG}(1^k)$
Evaluation	$y \leftarrow \boldsymbol{F}_\kappa(x)$

κ : function index

Actual strength of UCE security depends on what restrictions we put on the class of sources

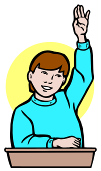
Class ***S*** is larger
➔UCE[***S***] security is stronger

- UCE security for source class ***S*** (UCE[***S***] security)



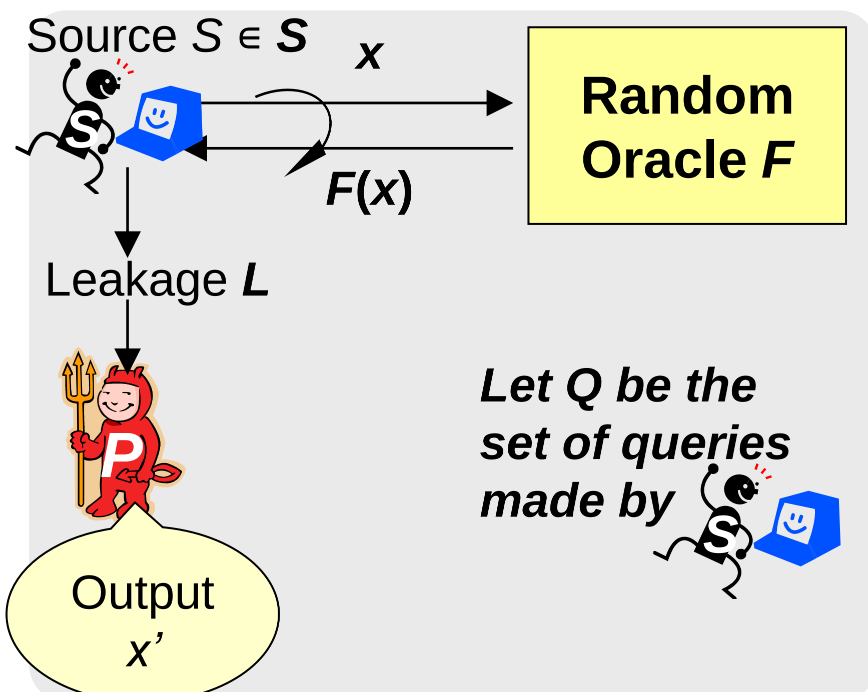
Function Family is UCE[***S***]-secure
if Pr[***b' = b***] = 1/2 + neg.
for ∀ $S \in \boldsymbol{S}$ and ∀ PPT ***A***

Restrictions on Sources (1/2)



Q. Why not consider all PPT algo. for sources?
(i.e. Why not set ***S*** = {PPT algo.} ?)

A. UCE[**PPT algo.**] security is unachievable.
Sources have to be at least (computationally) unpredictable:



$S \in \boldsymbol{S}^{cup}$

Source S is
computationally unpredictable
if Pr[***x' ∈ Q***] = neg
for any PPT ***P***

$S \in \boldsymbol{S}^{sup}$

Source S is
statistically unpredictable
if Pr[***x' ∈ Q***] = neg
for any comp. unbounded ***P***

Restrictions on Sources (2/2)

- Very recently, Brzsuka, Farshim, Mittelbach (BFM) attacked UCE[$\boldsymbol{S}^{\boldsymbol{cup}}$] security using indistinguishability obfuscation (**iO**)
  - eprint 2014/099

Appeared on Feb. 10.
However, we had known an "overview" of the attack by personal communication

- To avoid BFM's attack, we have to put further restrictions on the class of sources (… or disbelieve **iO**…)
  - $\boldsymbol{S}^{\boldsymbol{cup}}_{t,q}$: the class of sources that are comp. unpredictable, run at most t steps, and make at most q queries
  - $\boldsymbol{S}^{\boldsymbol{sup}}_{t,q}$: (similar)

Restrictions on Sources (2/2)

- Very recently, Brzsuka, Farshim, Mittelbach (BFM) attacked UCE[$\boldsymbol{S}^{cup}$] security using indistinguishability obfuscation (**iO**)
  - eprint 2014/099

Appeared on Feb. 10.
However, we had known an "overview" of the attack
by personal communication

- To avoid BFM's attack, we have to put further restrictions on the class of sources (… or disbelieve **iO**…)
  - $\boldsymbol{S}^{cup}{}_{t,q}$: the class of sources that are comp. unpredictable,
    run at most t steps, and make at most q queries
  - $\boldsymbol{S}^{sup}{}_{t,q}$: (similar)
- Later, it turned out that BFM's attack can be mounted by a comp. unpredictable source with q = 1 (much stronger than we expected ☹)
- To avoid it, t has to be smaller than their **iO**-based source…
  - Exactly how small t has to be depends on the running time of **iO**
    - So far, **iO** is very impractical, so that our results seem to survice
  - We can also restrict the "leakage size" of sources to avoid BFM's attack

Outline

- Background, Motivation, Results
- Definitions for UCE
- Negative Results
- Positive Results

Fujisaki-Okamoto (FO) Construction (PKC'99 ver.)

- Is a very important and useful result in public key crypto.

$$\text{CPA PKE} + \text{RO} \xrightarrow{\text{FO}} \text{CCA PKE (in RO model)}$$

$\boldsymbol{PKG}_{\mathbf{FO}}(1^k)$

- $(pk, sk) \leftarrow \boldsymbol{PKG}(1^k)$
- Output (pk, sk)

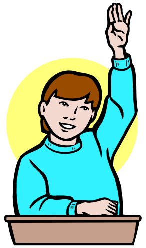
$\boldsymbol{Enc}_{\mathbf{FO}}(pk, m; r)$

- $C_{\mathbf{FO}} \leftarrow \boldsymbol{Enc}(pk, (r||m) ; \boldsymbol{H}(r||m))$
- Output $C_{\mathbf{FO}}$

$\boldsymbol{Dec}_{\mathbf{FO}}(sk, C_{\mathbf{FO}})$

- $(r||m) \leftarrow \boldsymbol{Dec}(sk, C_{\mathbf{FO}})$
- Check $C_{\mathbf{FO}} = \boldsymbol{Enc}(pk, (r||m) ; \boldsymbol{H}(r||m))$
- Output m

Natural Question



Q. Can we instantiate RO in the FO construction with UCE?

CPA PKE + **UCE** $\xrightarrow{\text{FO}}$ **?** **CCA PKE (in std. model)**

$\boldsymbol{PKG}_{\mathbf{FO}}(1^k)$

- $(pk, sk) \leftarrow \boldsymbol{PKG}(1^k)$
- $\kappa \leftarrow \boldsymbol{FKG}(1^k)$
- Output $((pk, \kappa), sk)$

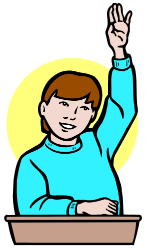
$\boldsymbol{Enc}_{\mathbf{FO}}(pk, m; r)$

- $C_{\mathbf{FO}} \leftarrow \boldsymbol{Enc}(pk, (r||m) ; \boldsymbol{F}_{\kappa}(r||m))$
- Output $C_{\mathbf{FO}}$

$\boldsymbol{Dec}_{\mathbf{FO}}(sk, C_{\mathbf{FO}})$

- $(r||m) \leftarrow \boldsymbol{Dec}(sk, C_{\mathbf{FO}})$
- Check
  $C_{\mathbf{FO}} = \boldsymbol{Enc}(pk, (r||m) ; \boldsymbol{F}_{\kappa}(r||m))$
- Output m

Natural Question



Q. Can we instantiate RO in the FO construction with UCE?

CPA PKE + UCE —FO→ ? CCA PKE (in std. model)

(Unfortunately) **NO!**

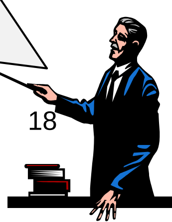
- **counterexample 1**

CPA PKE + UCE —FO→ ⃠ CPA PKE

- **counterexample 2**

CPA PKE + UCE —FO→ ⃠ CCA1 PKE (for random messages)

$r||m$))



Design Counterexample Pair PKE π' and UCE F'

- Suppose we are given CPA secure PKE π and function family ***F***

- Modify PKE π into π'
  - ***PKG'*** = ***PKG***
  - ***Enc'***$(pk, m; r)$
    - If $r = 0^k$, then $z = 1$ else $z = 0$
    - Return $c = (z \| \mathbf{Enc}(pk, m; r))$
  - ***Dec'*** ignores the first bit of c

- Modify the function family ***F*** into ***F'*:**
  - ***FKG'***(1^k)
    - $\kappa \leftarrow \mathbf{FKG}(1^k)$
    - Pick a "weak input" $v^* \leftarrow \{0,1\}^k$
    - Return $\kappa' = (\kappa, v^*)$
  - $\mathbf{F'}_{\kappa'}(x)$
    - If last k-bit of x is v^* then return $y = 0^k$
    - Return $y = \mathbf{F}_\kappa(x)$

Design Counterexample Pair PKE π' and UCE F'

- Suppose we are given CPA secure PKE π and function family ***F***

- Modify PKE π into π'
  - ***PKG'*** = ***PKG***
  - ***Enc'***(*pk*, *m*; *r*)
    - If $r = 0^k$, then $z = 1$ else $z = 0$
    - Return $c = (z \,||\, \textbf{\textit{Enc}}(pk, m; r))$
  - ***Dec'*** ignores the first bit of *c*

The MSB of a ciphertext *c* reveals whether $r = 0^k$

- If the PKE π is CPA secure ➔ So is the PKE π'

- Modify the function family ***F*** into ***F'***:
  - ***FKG'***(1^k)
    - $\kappa \leftarrow$ ***FKG***(1^k)
    - Pick a "weak input" $v^* \leftarrow \{0,1\}^k$
    - Return $\kappa' = (\kappa, v^*)$
  - $F'_{\kappa'}(x)$
    - If last *k*-bit of *x* is v^* then return $y = 0^k$
    - Return $y = F_\kappa(x)$

F' reveals whether the last *k*-bit of input *x* is v^*

- For any $S \subseteq S^{cup}$:
  If ***F*** is UCE[***S***] secure ➔ So is ***F'***

Use π' and F' in the FO Construction

- $PK_{\mathbf{FO}} = (\ pk,\ \kappa' = (\kappa,\ v^*)\ \ )$

- If we encrypt the weak input v^* by $\boldsymbol{Enc}_{\mathbf{FO}}(PK_{\mathbf{FO}},\ \cdot)$,
  ➔ ***The MSB of the ciphertext C_{FO} is always 1,*** because…
  - $C_{\mathbf{FO}} = \boldsymbol{Enc'}(pk,\ (r||v^*),\ \boldsymbol{F'}_{\kappa'}(r||v^*)\ )$
    $= \boldsymbol{Enc'}(\mathrm{pk},\ (\mathrm{r}||\mathrm{v}^*),\ \mathbf{0^k})$
    $= (\mathbf{1}\ ||\ \mathrm{c'})$ for some c'

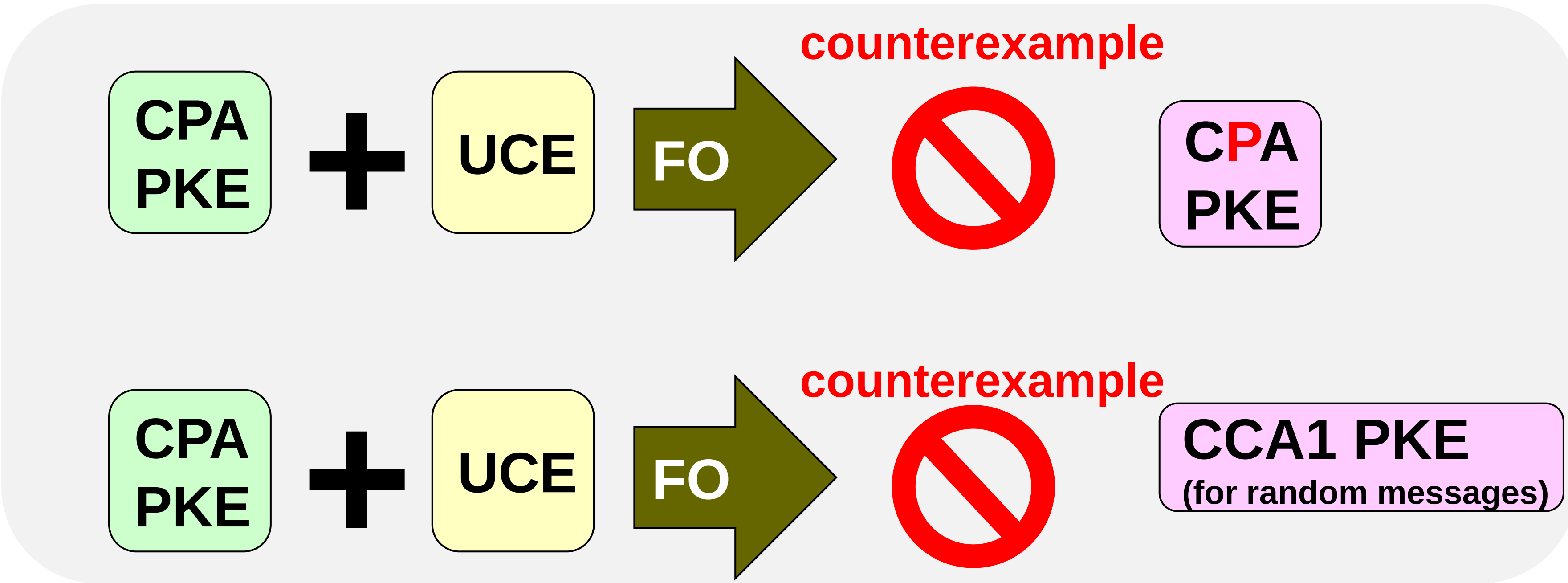
Because $\boldsymbol{F'}_{\kappa'}(r||v^*) = 0^k$

Because of how ***Enc'*** is designed

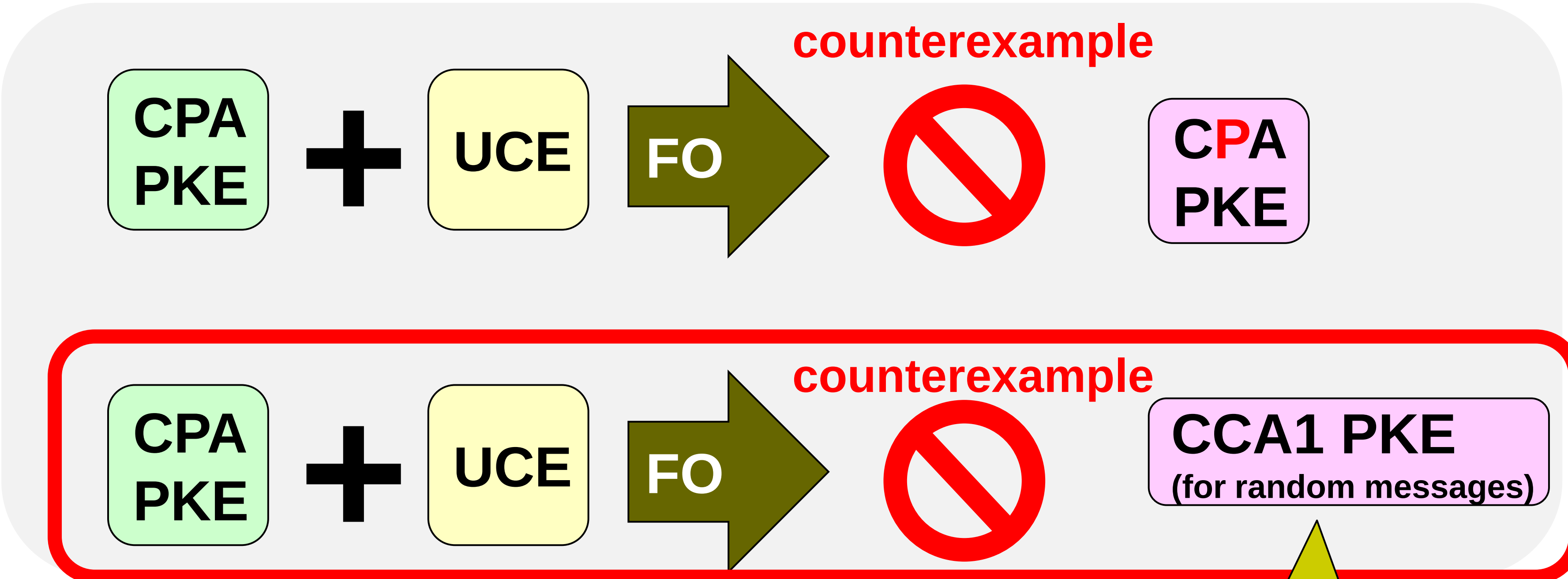
- If we encrypt a random message by $\boldsymbol{Enc}_{\mathbf{FO}}(PK_{\mathbf{FO}},\ \cdot)$,
  ➔ $\Pr[\mathrm{MSB}(C_{\mathbf{FO}}) = 1]$ is neg., due to UCE[***S***] security of ***F'***

➔ Adversary using challenge plaintexts $(M_0, M_1) = (v^*, \text{random})$ can break CPA security

Negative Results: Summary



Negative Results: Summary



Not explained in this slide.
The counterexample pair is slightly more complicated to bypass the “re-encryption” validity check of ciphertexts in ***Dec***$_{\textbf{FO}}$

PKE secure for random messages may be used as a secure KEM

Outline

- Background, Motivation, Results
- Definitions for UCE
- Negative Results
- Positive Results

Key Encapsulation Mechanisms (KEM)

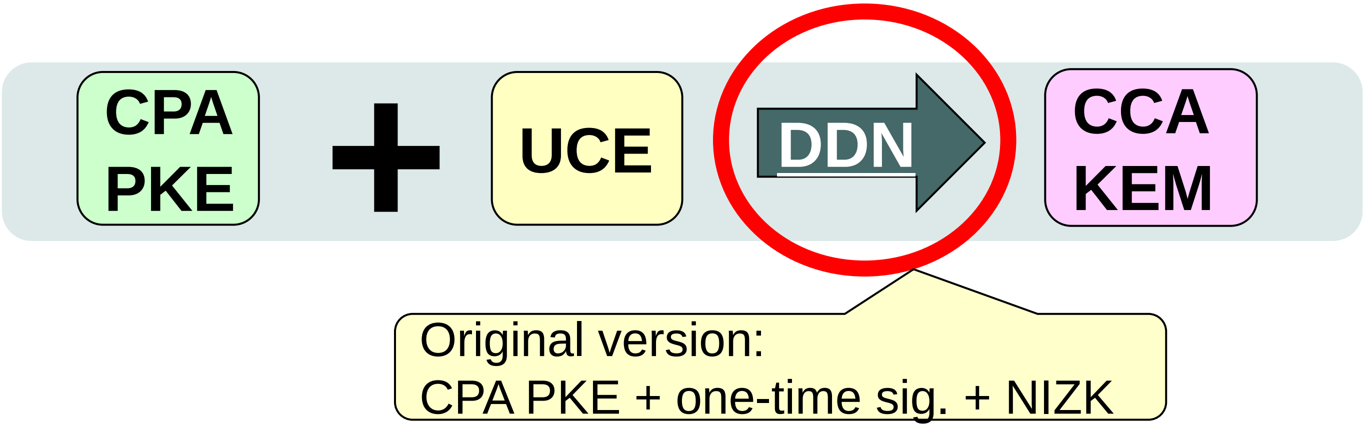
= "Public Key" part of hybrid encryption

Key Generation	$(pk, sk) \leftarrow$ ***KKG***(1^k)	
Encapsulation	$(C, K) \leftarrow$ ***Encap***(pk)	K: session-key used by SKE
Decapsulation	$K \,/\, \perp \leftarrow$ ***Decap***(sk, C)	

- Cramer-Shoup'03

CCA KEM + **CCA SKE** → **CCA PKE**

Our CCA Secure KEM: Overview



- In the original DDN, a plaintext is encrypted multiple times under independently generated *pk*'s
  - Extension from Naor-Yung's double encryption
- Its "core" structure can be understood as a special kind of tag-based encryption (TBE)
- We formalize it as a stand-alone cryptographic primitive: "***Puncturable TBE***" to reduce "description complexity"

Puncturable TBE (PTBE)

The name "puncturable" is inspired by "puncturable PRF" of [Sahai-Waters@eprint 2013/454]

- = TBE with two decryption modes

Key Generation	$(pk, sk) \leftarrow \boldsymbol{TKG}(1^k)$
Encryption	$c \leftarrow \boldsymbol{TEnc}(tpk, \boldsymbol{tag}, m)$
Decryption	$m / \perp \leftarrow \boldsymbol{TDec}(tsk, \boldsymbol{tag}, c)$
Puncturing	$psk_{tag^*} \leftarrow \boldsymbol{Punc}(sk, \boldsymbol{tag^*})$
Punctured Decryption	$m / \perp \leftarrow \boldsymbol{PTDec}(psk_{tag^*}, \boldsymbol{tag}, c)$

- Correctness: $\forall\, \boldsymbol{tag} \neq \boldsymbol{tag^*}$, $\forall\, c \leftarrow \boldsymbol{TEnc}(pk, \boldsymbol{tag}, m)$:
  - $\boldsymbol{TDec}(sk, \boldsymbol{tag}, c) = \boldsymbol{PTDec}(psk_{tag^*}, \boldsymbol{tag}, c) = m$
- Security : Extended CPA security

≒CPA security in the presence of psk_{tag^*}

Concrete instantiations from…
- ･CPA PKE
  (i.e. DDN's building block itself)
- ･Broadcast encryption
- ･Multi-recipient PKE/KEM

PTBE based on CPA PKE (Core Structure of Original DDN)

- $pk = \begin{pmatrix} pk^0_1 & pk^0_2 & \dots & pk^0_k \\ pk^1_1 & pk^1_2 & \dots & pk^1_k \end{pmatrix}$, $sk = \begin{pmatrix} sk^0_1 & sk^0_2 & \dots & sk^0_k \\ sk^1_1 & sk^1_2 & \dots & sk^1_k \end{pmatrix}$

- ***TEnc***(PK, tag, m) :
  - *Let t_i be the i-th bit of tag*
  - $\forall i = 1,2,\dots,k : c_i \leftarrow$ ***Enc***$(pk^{t_i}_i, m)$
  - $C = \{c_i\}_{i=1,2,\dots,k}$

- ***Punc***(sk, tag^*) :
  - *Let t^*_i be the i-th bit of tag^**
  - $psk_{tag^*} = \{sk^{(1-t^*_i)}_i\}_{i=1,2,\dots,k}$

- ***TDec*** (SK, tag, C):
  - *Let t_1 be the first bit of tag*
  - $m \leftarrow$ ***Dec***$(sk^{t_1}_1, c_1)$

- ***PTDec*** (psk_{tag^*}, tag, C):
  - *If $tag^* = tag$ then abort*
  - *Let t_i be the i-th bit of tag*
  - $\ell \leftarrow \min\{ i \mid t_i \neq t^*_i\}$
  - $m \leftarrow$ ***Dec***$(sk^{(1-t^*_\ell)}_\ell, c_\ell)$

Our CCA Secure KEM

- $PK = (pk, ck, \kappa)$
- $SK = sk$

(pk, sk): PTBE key pair
ck: commitment key
κ: UCE's function index

- ***Encap***(PK)

1. $\alpha \leftarrow$ random
2. $(r \,||\, r' \,||\, K) \leftarrow \boldsymbol{UCE}_{\kappa}(\alpha)$
3. $\boldsymbol{tag} \leftarrow \boldsymbol{Com}(ck, \alpha; r')$
4. $c \leftarrow \boldsymbol{TEnc}(pk, \boldsymbol{tag}, \alpha; r)$
5. $C \leftarrow (\boldsymbol{tag}, c)$
6. Output (C, K)

- ***Decap***$(SK, C = (\boldsymbol{tag}, c))$

1. $\alpha \leftarrow \boldsymbol{TDec}(sk, \boldsymbol{tag}, c)$
2. $(r \,||\, r' \,||\, K) \leftarrow \boldsymbol{UCE}_{\kappa}(\alpha)$
3. *Check*
   $c = \boldsymbol{TEnc}(pk, \boldsymbol{tag}, \alpha; r)$
   $\wedge\ \boldsymbol{tag} = \boldsymbol{Com}(ck, \alpha: r')$
4. Output K

Our CCA Secure KEM

- $PK = (pk, ck, \kappa)$
- $SK = sk$

(pk, sk): PTBE key pair
ck: commitment key
κ: UCE's function index

- ***Encap***(*PK*)

1. $\alpha \leftarrow$ random
2. $(r \,||\, r' \,||\, K) \leftarrow \boldsymbol{UCE}_\kappa(\alpha)$
3. $\boldsymbol{tag} \leftarrow \boldsymbol{Com}(ck, \alpha; r')$
4. $\boxed{c \leftarrow \boldsymbol{TEnc}(pk, \boldsymbol{tag}, \alpha; r)}$
5. $C \leftarrow (\boldsymbol{tag}, c)$
6. Output (C, K)

- ***Decap***(SK, C = (***tag***, c))

1. $\alpha \leftarrow \boldsymbol{TDec}(sk, \boldsymbol{tag}, c)$
2. $(r \,||\, r' \,||\, K) \leftarrow \boldsymbol{UCE}_\kappa(\alpha)$
3. Check
   $c = \boldsymbol{TEnc}(pk, \boldsymbol{tag}, \alpha; r)$
   $\wedge\ \boldsymbol{tag} = \boldsymbol{Com}(ck, \alpha: r')$
4. Output K

By using a commitment of α as a "tag", we do not need one-time signature in DDN

Due to validity check of c and ***tag***, we do not need NIZK in DDN

Our CCA Secure KEM

(t_M: running time of algorithm M)

There is a circularity between α and (r, r'), but it can be overcome by UCE[$\mathbf{S}^{cup}_{t,1}$] security of the function family with $t = O(t_{TKG}+t_{ComKG}+t_{Enc}+t_{Com}+t_{Punc})$

- SK = sk

Use ***PTDec***(psk$_{tag^*}$, ·) to answer dec. queries

- ***Encap***(PK)
  1. α ← random
  2. (r || r' || K) ← ***UCE***$_\kappa$(α)
  3. ***tag*** ← ***Com***(ck, α; r')
  4. c ← ***TEnc***(pk, ***tag***, α; r)
  5. C ← (***tag***, c)
  6. Output (C, K)

By using a commitment of α as a "tag", we do not need one-time signature in DDN

- ***Decap***(SK, C = (***tag***, c))
  1. α ← ***TDec***(sk, ***tag***, c)
  2. (r || r' || K) ← ***UCE***$_\kappa$(α)
  3. Check
     c = ***TEnc***(pk, ***tag***, α; r)
     ∧ ***tag*** = ***Com***(ck, α: r')
  4. Output K

Due to validity check of c and ***tag***, we do not need NIZK in DDN

If PTBE is extended-CPA secure, COM is hiding and binding, ***F*** is UCE[$\boldsymbol{S}^{cup}_{t,1}$] secure (with *t* below),
➔ Our KEM is CCA secure

($\mathbf{t}_M$: running time of algorithm ***M***)

There is a circularity between α and (*r*, *r'*), but it can be overcome by UCE[$\boldsymbol{S}^{cup}_{t,1}$] security of the function family with $t = \boldsymbol{O}(\mathbf{t}_{TKG}+\mathbf{t}_{ComKG}+\mathbf{t}_{Enc}+\mathbf{t}_{Com}+\mathbf{t}_{Punc})$

- *SK* = *sk*

Use ***PTDec***(psk_{tag^*}, ·) to answer dec. queries

- ***Encap***(*PK*)
  1. α ← random
  2. (*r* || *r'* || *K*) ← $\boldsymbol{UCE}_\kappa$(α)
  3. ***tag*** ← ***Com***(*ck*, α; *r'*)
  4. *c* ← ***TEnc***(*pk*, ***tag***, α; *r*)
  5. *C* ← (***tag***, *c*)
  6. Output (*C*, *K*)

- ***Decap***(*SK*, *C* = (***tag***, *c*))
  1. α ← ***TDec***(*sk*, ***tag***, *c*)
  2. (*r* || *r'* || *K*) ← $\boldsymbol{UCE}_\kappa$(α)
  3. Check
     c = ***TEnc***(*pk*, ***tag***, α; *r*)
     ∧ ***tag*** = ***Com***(*ck*, α: *r'*)
  4. Output *K*

By using a commitment of *α* as a "tag", we do not need one-time signature in DDN

Due to validity check of *c* and ***tag***, we do not need NIZK in DDN

Extensions

- <u>Deterministic PKE</u>
  - Slight modification from our KEM
    - Derive (r, r') for ***TEnc*** and ***Com*** from a high min-entropy plaintext
  - Achieve CCA security for block sources [BFO08] **with bounded running time**
    - Restriction is due to the BFM's **iO**-based attack
    - It is weaker than security for ordinary block sources, but still a meaningful security notion in practice
- <u>Weakening the UCE assumption</u>
  - If we replace CPA PKE with Lossy PKE [BHY09], then we can weaken the assumption on the function family from UCE[$\boldsymbol{S}^{\boldsymbol{cup}}_{t,1}$] security to UCE[$\boldsymbol{S}^{\boldsymbol{sup}}_{t,1}$] security
  - BFM's **iO**-based attack does not apply to UCE[$\boldsymbol{S}^{\boldsymbol{sup}}$] security ☺

Summary

- We ask: CPA PKE + UCE ??→ CCA PKE

- **Our results:**

CPA PKE + UCE

We can use Lossy PKE for weakening the UCE assumption

Negative ☹ **Fujisaki-Okamoto** → **counterexample**
- **CPA PKE**
- **CCA1 PKE (for random messages)**

Positive ☺ **Dolev-Dwork-Naor (DDN)** →
- **CCA PKE (via KEM)**
- **CCA Deterministic PKE (for block sources with bounded running time)**

Abstraction by **Puncturable TBE**