This paper appears in the proceedings of
the 33rd International Colloquium on Automata, Languages and Programming (ICALP)
(9 – 16 july 2006, Venice, Italy)
B. Preneel Ed. Springer-Verlag, LNCS 4052, pages 264–275.

Generic Construction of Hybrid Public Key Traitor Tracing with Full-Public-Traceability

Duong Hieu Phan[1], Reihaneh Safavi-Naini[2], and Dongvu Tonien[2]

[1] University College London
Adastral Park Postgraduate Research Campus,
Ross Building, Adastral Park, IP5 3RE, Ipswich, United Kingdom.
h.phan@adastral.ucl.ac.uk
[2] University of Wollongong,Australia
School of Information Technology and Computer Science,
Wollongong, NSW 2522, Australia.
{rei,dong}@uow.edu.au

Abstract. In Eurocrypt 2005, Chabanne, Phan and Pointcheval introduced an interesting property for traitor tracing schemes called *public traceability*, which makes tracing a black-box public operation. However, their proposed scheme only worked for two users and an open question proposed by authors was to provide this property for multi-user systems.
In this paper, we give a comprehensive solution to this problem by giving a *generic construction for a hybrid traitor tracing scheme that provides full-public-traceability*. We follow the Tag KEM/DEM paradigm of hybrid encryption systems and extend it to multi-receiver scenario. We define Tag-BroadcastKEM/DEM and construct a secure Tag-BroadcastKEM from a CCA secure PKE and target-collision resistant hash function. We will then use this Tag-BroadcastKEM together with a semantically secure DEM to give a generic construction for Hybrid Public Key Broadcast Encryption. The scheme has a black box tracing algorithm that *always* correctly identifies a traitor. The hybrid structure makes the system very efficient, both in terms of computation and communication cost. Finally we show a method of reducing the communication cost by using codes with identifiable parent property.

1 Introduction

Broadcast encryption and traitor tracing systems are the main cryptographic primitives for secure distribution of copyrighted digital content. In broadcast encryption systems the user group is dynamic and changes over time and access control is by distributing a new session key to authorised users in each session. The session key is used to securely encrypt (e.g. using AES) the content. The separation of content encryption and session key encryption provides flexibility in choosing encryption algorithms that are suitable for specific content (e.g. MPEG2 streams).

Traitor tracing systems however, aim at providing traceability against colluding users who have constructed a *pirate* decoder that can illegally decrypt the content. Public key traitor tracing schemes allow anyone to send content to members of an authorised group. In the model of tracing proposed in [CFN94], tracing is performed by a trusted authority who has access to the secret key of all users. The tracer is able to identify one of the c traitors who have colluded to construct a pirate decoder. The tracing algorithm may always correctly identify a traitor, or it may have ϵ error in which case in some cases the tracing algorithm either fails to identify a colluder, or it outputs an innocent user as a traitor.

Full Public Traceability. In Eurocrypt 2005, Chabanne, Phan and Pointcheval (CPP05) introduced the notion of *public traceability* where tracing is a black-box and publicly computable procedure. This is an interesting property that strengthens the overall security of the system as it separates the two tasks of key generation and tracing and allows all users to perform tracing on a pirate decoder. However their construction of fully public traceability system only worked for two users (it also required a new strong computational assumptions). The restriction to two user systems was due to a synthesis method that was inspired by a construction in [KY02] that used c-secure codes. However in using this approach to multi-user case, public traceability will be lost because tracing in c-secure codes is a private operation performed by a trusted centre. Authors raised the construction of a multi-receiver traitor tracing scheme with full public traceability as an interesting open problem.

Efficiency. Efficiency of broadcast encryption systems is often measured in terms of (i) *ciphertext rate* that captures the extra bandwidth that is required for transmission of the ciphertext, measured as the ratio of ciphertext length to plaintext length and, (ii) *computational efficiency* of performing encryption and decryption operations. The two most efficient systems from ciphertext rate view points are KY02 and CPP05 scheme, both with constant ciphertext rate. However both schemes require exponentiation of group elements followed by hashing for encryption and decryption of messages and so compared to symmetric key encryption systems, are inefficient.

In two party systems, Hybrid Tag-KEM/DEM provides a secure, efficient, and flexible method of encrypting messages using public key encryption systems for delivering the key information and using symmetric key systems for the encryption of the actual data. In this approach, Key Encapsulation Mechanism (KEM) encrypts a short random key in a header, and a Data Encapsulation Mechanism (DEM) uses this key to encrypt the message into a ciphertext using a symmetric encryption scheme. It is shown [AGKS05] that strong security for the ciphertext can be guaranteed with a semantically secure DEM and CCA security of KEM.

Our contributions In this paper, we answer the open question of CCP05 by constructing a generic Hybrid Public Key Traitor Tracing that provides full public traceability and very efficient.

Our approach can be summarised as follows. We first extend the Tag KEM/DEM paradigm of hybrid encryption systems to multi-receiver scenario and define *Hybrid Tag-BroadcastKEM/DEM (Hybrid-PKBE)*. In Hybrid-PKBE the random key that is used for the encryption of a message is encrypted by *TBKEM*. This key is only extractable by authorized receivers. We define security of Hybrid-PKBE and prove a result similar to Hybrid Tag-KEM/DEM. We show that a Replayable CCA secure Hybrid-PKBE can be obtained from a Replayable CCA secure TBKEM and a semantically secure DEM. Replayable CCA (RCCA) security was introduced in [CKN03] to capture a security notion that is strictly weaker than CCA but sufficient for many practical applications.

Next we give a construction of a RCCA secure TBKEM (in above sense) from a CCA secure public key encryption (PKE) system and a *target collision free hash function*. Combining this TBKEM and a semantically secure DEM gives a secure Hybrid-PKBE. Moreover, we will show that this construction of Hybrid-PKBE support a tracing algorithm and hence it is a *Hybrid Public Key Traitor Tracing (Hybrid-PKTT)*. The tracing algorithm is black-box and only uses the public key of the system and so the system has *full public traceability*. This provides an elegant solution to the open problem of CPP05.

The hybrid construction makes the system very efficient. The ciphertext rate for long messages approaches one, and computational efficiency is obtained because of the decoupling of key encapsulation mechanism and data encryption module. The RCCA security of the system makes it *the first* construction of traitor tracing systems with this level of security (compared to previous constructions with constant ciphertext rate).

In the final section of the paper, we focus on increasing efficiency of the system. The communication overhead in the above system is a linear function of the size of the receiver group . Although for long messages and fixed size groups this gives a ciphertext rate of 1 (asymptotically), but it is desirable to reduce the size of the ciphertext overhead to make it more applicable for large groups. We use an approach similar to KY02 and CPP05, replacing collusion-secure codes with IPP codes. This reduces the length of the header and makes it *logarithmic in the number of users*. Interestingly, the composition preserves full public traceability as, unlike collusion-secure codes, IPP codes have public tracing algorithm.

2 Preliminaries

2.1 Public-key Broadcast Encryption (PKBE)

A public-key broadcast encryption (without revocation, without traceability) consists of the following algorithms:

- *Key generation algorithm* $\mathsf{PKBE.Gen}(1^\lambda, n) \rightarrow (pk, sk_1, \ldots, sk_n)$:
  An algorithm that generates a public-key pk and n private-keys sk_1, sk_2, $\ldots$, sk_n.
- *Encryption algorithm* $\mathsf{PKBE.Enc}_{pk}(m) \rightarrow c$:
  An algorithm that encrypts a message m into c by a public key pk.
- *Decryption algorithm* $\mathsf{PKBE.Dec}_{sk_i}(c) \rightarrow m$:
  An algorithm that decrypts a ciphertext c to m by a secret key sk_i.

The security against replayable adaptive chosen ciphertext attack (RCCA) is defined as follows. Let A_{pkbe} be a polytime oracle machine that plays the following game.

[GAME.PKBE]
Step 1. $(pk, sk_1, \ldots, sk_n) \leftarrow \mathsf{PKBE.Gen}(1^\lambda, n)$
Step 2. $(m_0, m_1) \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(pk)$
Step 3. $b \leftarrow \{0,1\}$, $c^* \leftarrow \mathsf{PKBE.Enc}_{pk}(m_b)$
Step 4. $\hat{b} \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(c^*)$

Here $\mathcal{O}_{\text{pkbe}}$ denotes the decryption oracle. A decryption query is of the form (i, c) where i is an integer $\in [1, n]$ and c is a ciphertext with a constraint that in Step 4 c must be different from c^*. To answer this query, the oracle calculates $\mathsf{PKBE.Dec}_{sk_i}(c) = m$. In Step 2 the oracle outputs m. Moreover, in Step 4, it checks: if $m = m_0$ or $m = m_1$ then it outputs $\perp$, otherwise it outputs m. We define $\epsilon_{\text{pkbe-rcca},A_{\text{pkbe}}} = |Pr[\hat{b} = b] - 1/2|$, and $\epsilon_{\text{pkbe-rcca}} = \max(\epsilon_{\text{pkbe-rcca},A_{\text{pkbe}}})$, where the maximum is taken over all polytime machines A_{pkbe}. We say that a PKBE is RCCA secure if $\epsilon_{\text{pkbe-rcca}}$ is negligible in λ.

Public Key Traitor Tracing Scheme with Public Traceability. A public key traitor tracing (PKTT) scheme is a public key broadcast encryption with an extra algorithm, the *tracing algorithm*. This tracing algorithm takes as input the tracing information *trace-infor* and a pirate decoder $\mathcal{D}$. It outputs at least one of the users (called *traitors*) who have collaborated in producing the pirate decoder $\mathcal{D}$. In a c-traitor tracing scheme, we assume that there are at most c traitors who created the pirate decoder $\mathcal{D}$. The other obvious assumption on $\mathcal{D}$ is that $\mathcal{D}$ can effectively reverse the encryption, *i.e.* $\mathcal{D}(\mathsf{PKBE.Enc}_{pk}(m)) = m$, with high probability.

In general, tracing information *trace-infor* consists of some public information and some system secret parameters (for example, users' secret keys). Chabanne *et. al* introduced [CPP05] an interesting property of *public traceability* for traitor tracing schemes in which *trace-infor* consists of only public key of the system. Thus, it makes it possible for anyone to execute the tracing algorithm.

2.2 Data Encapsulation Mechanism (DEM)

A data encapsulation mechanism consists of the following algorithms:

- *Setup algorithm* $\mathsf{DEM.Setup}(1^\lambda) \rightarrow \mathcal{K}_D$:
  An algorithm that specifies the symmetric key space $\mathcal{K}_D$.
- *Encryption algorithm* $\mathsf{DEM.Enc}_{dk}(m) \rightarrow c$:
  An algorithm that encrypts m into c using a symmetric-key $dk \in \mathcal{K}_D$.
- *Decryption algorithm* $\mathsf{DEM.Dec}_{dk}(c) \rightarrow m$:
  An algorithm that decrypts c to m using a symmetric-key $dk \in \mathcal{K}_D$.

The IND (indistinguishable against passive attack) security of DEM is defined as follows. Let A_{dem} be a poly-time oracle machine that plays the following game.

[GAME.DEM]
Step 1. $\mathcal{K}_D \leftarrow \mathsf{DEM.Setup}(1^\lambda)$
Step 2. $(m_0, m_1) \leftarrow A_{\text{dem}}$
Step 3. $b \leftarrow \{0,1\}$, $dk \leftarrow \mathcal{K}_D$, $c \leftarrow \mathsf{DEM.Enc}_{dk}(m_b)$
Step 4. $\hat{b} \leftarrow A_{\text{dem}}(c)$

We define $\epsilon_{\text{dem},A_{\text{dem}}} = |Pr[\hat{b} = b] - 1/2|$, and $\epsilon_{\text{dem}} = \max(\epsilon_{\text{dem},A_{\text{dem}}})$, where the maximum is taken over all polytime machines A_{dem}. We say that a DEM is IND secure if ϵ_{dem} is negligible in λ.

2.3 Target Collision Resistant Hash Functions

A family $\mathcal{H} = \{H_k : \mathcal{A} \to \mathcal{B}\}_{k \in K}$ of keyed hash functions is *target collision resistant* if given a random $\tau \in \mathcal{A}$ and a random $H_k \in \mathcal{H}$, it is computationally infeasible to find $\tau' \in \mathcal{A}$ such that $\tau' \neq \tau$ and $H_k(\tau') = H_k(\tau)$. A random function H_k of $\mathcal{H}$ is called a *target collision-free* hash function. Associated with H_k, we define the quantity ϵ_{tch} as $\epsilon_{\text{tch}} = \max Pr[\tau' \in \mathcal{A}, \tau' \neq \tau, H_k(\tau') = H_k(\tau) : \tau \leftarrow \mathcal{A}, \tau' \leftarrow A_{\text{tch}}(\tau)]$ where the maximum is taken over all poly-time machines A_{tch}.

3 Generic Construction of Hybrid PKBE

In this section, we generalize Abe *et. al*'s [AGKS05] Tag-KEM/DEM construction of Hybrid-PKE. We show how to construct a hybrid public key broadcast encryption scheme (without revocation and traitor tracing) using two components: a Tag-BroadcastKEM and a DEM, and with this construction, we prove the following composition theorem,
(relaxed) CCA Tag-BKEM + semantic secure DEM → (relaxed) CCA Hybrid-PKBE.

3.1 Tag-Broadcast Key Encapsulation Mechanism (TBKEM)

A tag-broadcast key encapsulation mechanism consists of the following algorithms:

- *Key generation algorithm* $\mathsf{TBKEM.Gen}(1^\lambda, n) \to (pk, sk_1, \ldots, sk_n)$:
  An algorithm that generates a public-key pk and n private-keys sk_1, sk_2, …, sk_n. It also specifies tag space $\mathcal{T}$ and encapsulated key space $\mathcal{K}_D$.
- *Key derivation algorithm* $\mathsf{TBKEM.Key}(\mathsf{pk}) \to (\omega, dk)$:
  An algorithm that generates a one-time key dk and internal state information ω.
- *Encryption algorithm* $\mathsf{TBKEM.Enc}(\omega, \tau) \to \psi$:
  An algorithm that encrypts dk (embedded in ω) into ψ using a tag τ.
- *Decryption algorithm* $\mathsf{TBKEM.Dec}_{sk_i}(\psi, \tau) \to dk$:
  An algorithm that recovers dk from ψ and τ using one of the private-key sk_i. It may output a special symbol $\perp \notin \mathcal{K}_D$.

The RCCA security of TBKEM is defined as follows. Let A_{tbkem} be a poly-time oracle machine that plays the following game.

[GAME.TBKEM]
Step 1. $(pk, sk_1, \ldots, sk_n) \leftarrow \mathsf{TBKEM.Gen}(1^\lambda, n)$
Step 2. $(w, dk_1) \leftarrow \mathsf{TBKEM.Key}(pk), dk_0 \leftarrow \mathcal{K}_D, \delta \leftarrow \{0, 1\}$
Step 3. $\tau^* \leftarrow A^{\mathcal{O}_{\text{tbkem}}}_{\text{tbkem}}(pk, dk_\delta)$
Step 4. $\psi^* \leftarrow \mathsf{TBKEM.Enc}(w, \tau^*)$
Step 5. $\hat{\delta} \leftarrow A^{\mathcal{O}_{\text{tbkem}}}_{\text{tbkem}}(\psi^*)$

Here $\mathcal{O}_{\text{tbkem}}$ denotes the decryption oracle. A decryption query is of the form (i, ψ, τ) where i is an integer $\in [1, n]$, τ is a tag and ψ is a ciphertext with a constraint that in Step 5 (ψ, τ) must be different from (ψ^*, τ^*). To answer this query, the oracle calculates $\mathsf{TBKEM.Dec}_{sk_i}(\psi, \tau) = dk$. In Step 3 the oracle outputs dk. However, in Step 5, for a decryption query (i, ψ, τ), the oracle first checks if $\tau = \tau*$ and if $dk = dk_\delta$, then outputs the symbol $\perp \notin \mathcal{K}_D$, otherwise outputs dk. In other words, the sole difference between CCA security and RCCA security in the TBKEM model is that the adversary in the RCCA security is forbidden to ask (ψ, τ) where $\tau = \tau*$ and $dk = dk_\delta$. We remark that in TBKEM game, the adversary only knows dk_δ but not both dk_0 and dk_1.

We define $\epsilon_{\text{tbkem-rcca},A_{\text{tbkem}}} = |Pr[\hat{\delta} = \delta] - 1/2|$, and $\epsilon_{\text{tbkem-rcca}} = \max(\epsilon_{\text{tbkem-rcca},A_{\text{tbkem}}})$, where the maximum is taken over all polytime machines A_{tbkem}. We say that a TBKEM is RCCA secure if $\epsilon_{\text{tbkem-rcca}}$ is negligible in λ. We will see later that TBKEM with RCCA security and can be constructed from CCA-secure PKE and target collision-free hash function.

3.2 Hybrid Public Key Broadcast Encryption Scheme (Hybrid-PKBE)

The description of the hybrid public key broadcast encryption scheme is as follows.

- *Algorithm* $\mathsf{Hybrid\text{-}PKBE.Gen}(1^\lambda, n) \rightarrow (pk, sk_1, \ldots, sk_n)$:
  Call $\mathsf{TBKEM.Gen}(1^\lambda, n) \rightarrow (pk, sk_1, \ldots, sk_n)$.
- *Algorithm* $\mathsf{Hybrid\text{-}PKBE.Enc}_{pk}(m) \rightarrow c$: call $\mathsf{TBKEM.Key}(pk) \rightarrow (\omega, dk)$,
  $\mathsf{DEM.Enc}_{dk}(m) \rightarrow \tau$, $\mathsf{TBKEM.Enc}(\omega, \tau) \rightarrow \psi$ and set $c = (\psi, \tau)$.
- *Algorithm* $\mathsf{Hybrid\text{-}PKBE.Dec}_{sk_i}(c) \rightarrow m$: suppose $c = (\psi, \tau)$.
  Call $\mathsf{TBKEM.Dec}_{sk_i}(\psi, \tau) \rightarrow dk$ and $\mathsf{DEM.Dec}_{dk}(\tau) \rightarrow m$.

Theorem 1. *Hybrid-PKBE is RCCA secure under the assumptions that TBKEM is RCCA secure and DEM is IND secure:* $\epsilon_{\textit{pkbe-rcca}} \leq 2\,\epsilon_{\textit{tbkem-rcca}} + \epsilon_{\textit{dem}}$.

Proof. We prove the Hybrid-PKBE is RCCA secure using a sequence of games.

Game 0. Let A_{pkbe} be an adversary that plays the following attack game in the definition of RCCA security (section 2.1).

$[\mathsf{GAME.PKBE_0}]$
Step 1. $(pk, sk_1, \ldots, sk_n) \leftarrow \mathsf{TBKEM.Gen}(1^\lambda, n)$
Step 2. $(m_0, m_1) \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(pk)$
Step 3. $b \leftarrow \{0,1\}$, $(\omega, dk^*) \leftarrow \mathsf{TBKEM.Key}(pk)$,
$\tau^* \leftarrow \mathsf{DEM.Enc}_{dk^*}(m_b)$, $\psi^* \leftarrow \mathsf{TBKEM.Enc}(\omega, \tau^*)$
Step 4. $\hat{b} \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(\psi^*, \tau^*)$

Let X_0 be the event that $b = \hat{b}$ in the above game then $\epsilon_{\text{pkbe-rcca},A_{\text{pkbe}}} = |Pr[X_0] - 1/2|$.

In Step 4, a decryption query is of the form (i, ψ, τ) where $(\psi, \tau) \neq (\psi^*, \tau^*)$. To answer this query, the oracle executes

$\mathsf{Hybrid\text{-}PKBE.Dec}_{sk_i}(\psi, \tau)$:
1. $\mathsf{TBKEM.Dec}_{sk_i}(\psi, \tau) = dk$;
2. $\mathsf{DEM.Dec}_{dk}(\tau) = m$.

If $m = m_0$ or $m = m_1$ then $\mathcal{O}_{\text{pkbe}}$ outputs $\perp$, otherwise, it outputs m.

Game 1. We modify Game 0 in Step 3, instead of encrypting m_b using a key produced by $\mathsf{TBKEM.Key}$, we encrypt m_b using a random key.

$[\mathsf{GAME.PKBE_1}]$
Step 1. $(pk, sk_1, \ldots, sk_n) \leftarrow \mathsf{TBKEM.Gen}(1^\lambda, n)$
Step 2. $(m_0, m_1) \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(pk)$
Step 3. $b \leftarrow \{0,1\}$, $(\omega, dk_1) \leftarrow \mathsf{TBKEM.Key}(pk)$,
$dk_0 \leftarrow \mathcal{K}_D$, $\tau^* \leftarrow \mathsf{DEM.Enc}_{dk_0}(m_b)$, $\psi^* \leftarrow \mathsf{TBKEM.Enc}(\omega, \tau^*)$
Step 4. $\hat{b} \leftarrow A_{\text{pkbe}}^{\mathcal{O}_{\text{pkbe}}}(\psi^*, \tau^*)$

Let X_1 be the event that $b = \hat{b}$ in Game 1.

Claim 1. $|Pr[X_0] - Pr[X_1]| \leq 2\epsilon_{\text{tbkem-rcca}}$.
The proof of this claim can be found in the full version [PST06].

Claim 2. $|Pr[X_1] - 1/2| \leq \epsilon_{\text{dem}}$.
The proof of this claim can be found in the full version [PST06].

Finally, we have

$$\begin{aligned}\epsilon_{\text{pkbe-rcca},A_{\text{pkbe}}} &= |Pr[X_0] - 1/2| \\ &\leq |Pr[X_0] - Pr[X_1]| + |Pr[X_1] - 1/2| \\ &\leq 2\,\epsilon_{\text{tbkem-rcca}} + \epsilon_{\text{dem}},\end{aligned}$$

thus, $\epsilon_{\text{pkbe-rcca}} \leq 2\,\epsilon_{\text{tbkem-rcca}} + \epsilon_{\text{dem}}$, where $\epsilon_{\text{tbkem-rcca}}$ and ϵ_{dem} are assumed to be negligible. ■

Remark. We can also prove that

CCA Tag-BKEM + semantic secure DEM $\rightarrow$ CCA Hybrid-PKBE.

This is a more natural generalization of the result of Abe *et. al* [AGKS05]. However, we don't know how to construct, in a simple manner, a CCA TBKEM from PKE only. This is the reason we introduce the notion of RCCA TBKEM which will be constructed from CCA PKE.

4 Construction of a Basic Hybrid-PKTT

4.1 How to construct TBKEM from PKE

In this section, we show a generic construction of a TBKEM that is RCCA secure from a CCA secure PKE and a target collision-free hash function H. The construction is as follows.

- *Algorithm* $\mathsf{TBKEM.Gen}(1^\lambda, n) \rightarrow (pk, sk_1, \ldots, sk_n)$:
  For each $i = 1, \ldots, n$, call $\mathsf{PKE.Gen}(1^\lambda) \rightarrow (pk_i, sk_i)$. Set $pk = (pk_1, \ldots, pk_n)$.
- *Algorithm* $\mathsf{TBKEM.Key}(pk) \rightarrow (\omega, dk)$:
  Choose a random dk and set $\omega = pk||dk$.
- *Algorithm* $\mathsf{TBKEM.Enc}(\omega, \tau) \rightarrow \psi$ (where $\omega = pk||dk$):
  Compute $h = H(\tau)$. Call $\mathsf{PKE.Enc}_{pk_i}(dk||h) \rightarrow \sigma_i$ for $i = 1, \ldots, n$. Output $\psi = (\sigma_1, \ldots, \sigma_n)$.
- *Algorithm* $\mathsf{TBKEM.Dec}_{sk_i}(\psi, \tau) \rightarrow dk$ or $\perp$ (where $\psi = (\sigma_1, \ldots, \sigma_n)$):
  Call $\mathsf{PKE.Dec}_{sk_i}(\sigma_i) \rightarrow dk||h$. If $h = H(\tau)$, return dk. Otherwise, return $\perp$.

Theorem 2. TBKEM *is RCCA secure under the assumptions that* PKE *is CCA secure and H is target collision-free:* $\epsilon_{tbkem\text{-}rcca} \leq \epsilon_{tch} + n\,\epsilon_{pke\text{-}cca}$.

We use the same technique as used for proof of Theorem 1, namely Game approach, to prove this theorem. The proof can be found in the full version [PST06].

4.2 Basic Hybrid-PKTT

In section 3, we show how to construct a Hybrid-PKBE using a TBKEM and a DEM. In section 4, we show how to construct a TBKEM using a PKE and a target collision-free hash function. In this section, we combine the above two constructions. Thus, from a DEM, a PKE and a target collision-free hash function, we can construct a Hybrid-PKBE. We show that this Hybrid-PKBE is very special: it is a Hybrid-PKTT with full-public-traceability. Therefore, we have the following composition:

CCA PKE + semantic secure DEM + target collision-free hash function
$\rightarrow$ RCCA Hybrid-PKTT with full-public-traceability.

The description of the Hybrid-PKTT is as follows.

Algorithm $\mathsf{Hybrid\text{-}PKTT.Gen}(1^\lambda, n) \rightarrow (pk, sk_1, \ldots, sk_n)$:
For each $i = 1, \ldots, n$, call $\mathsf{PKE.Gen}(1^\lambda) \rightarrow (pk_i, sk_i)$. Set $pk = (pk_1, \ldots, pk_n)$.

Algorithm $\mathsf{Hybrid\text{-}PKTT.Enc}_{pk}(m) \rightarrow c$:
Choose a random dk and call $\mathsf{DEM.Enc}_{dk}(m) \rightarrow \tau$. Compute $h = H(\tau)$ and for each $i = 1, \ldots, n$, call $\mathsf{PKE.Enc}_{pk_i}(dk||h) \rightarrow \sigma_i$. Output $c = (\sigma_1, \ldots, \sigma_n, \tau)$.

Algorithm $\mathsf{Hybrid\text{-}PKTT.Dec}_{sk_i}(c) \rightarrow m$ or $\perp$, where $c = (\sigma_1, \ldots, \sigma_n, \tau)$:
Call $\mathsf{PKE.Dec}_{sk_i}(\sigma_i) \rightarrow dk||h$. If $h \neq H(\tau)$, return $\perp$. Otherwise, call $\mathsf{DEM.Dec}_{dk}(\tau) \rightarrow m$ and output m.

Algorithm $\mathsf{Hybrid\text{-}PKTT.Public\text{-}Trace}(pk, \mathcal{D})$: A black-box traitor tracing algorithm that can be executed by anyone using the public-key to find a traitor who had created pirate decoder.

- Choose random dk, m, and call $\mathsf{Hybrid\text{-}PKTT.Enc}_{pk}(m) \rightarrow (\sigma_1, \ldots, \sigma_n, \tau)$.
- For each $i = 1, \ldots, n$, choose random $d_i' \neq dk||h$ so that d_i' has the same length as $dk||h$ and call $\mathsf{PKE.Enc}_{pk_i}(d_i') \rightarrow \sigma_i'$; modify the ciphertext and give them to $\mathcal{D}$ and check if $\mathcal{D}(\sigma_1, \sigma_2, \ldots, \sigma_n, \tau) \stackrel{?}{=} m$, $\mathcal{D}(\sigma_1', \sigma_2, \ldots, \sigma_n, \tau) \stackrel{?}{=} m$, $\mathcal{D}(\sigma_1', \sigma_2', \ldots, \sigma_n, \tau) \stackrel{?}{=} m, \ldots, \mathcal{D}(\sigma_1', \sigma_2', \ldots, \sigma_n', \tau) \stackrel{?}{=} m$.

– Calculate the following probabilities
$p_0 = Pr[\mathcal{D}(\sigma_1, \sigma_2, \ldots, \sigma_n, \tau) = m], p_1 = Pr[\mathcal{D}(\sigma'_1, \sigma_2, \ldots, \sigma_n, \tau) = m]$,
$p_2 = Pr[\mathcal{D}(\sigma'_1, \sigma'_2, \ldots, \sigma_n, \tau) = m], \ldots, p_n = Pr[\mathcal{D}(\sigma'_1, \sigma'_2, \ldots, \sigma'_n, \tau) = m]$.
We assume that $\mathcal{D}$ is a usable decoder so p_0 is not negligible (indeed $p_0 \approx 1$), and obviously, $p_n \approx 0$.
So there must exist i such that $|p_i - p_{i-1}|$ is not negligible, in this case, output i as a traitor.

The above scheme, without tracing algorithm, is a Hybrid-PKBE. The security of encryption of above scheme, denoted by $\epsilon_{\text{pktt-rcca}}$, is evidently independent of the tracing algorithm. Therefore, following theorem is a corollary of Theorem 1 and Theorem 2.

Theorem 3. *Hybrid-PKTT is RCCA secure under the assumptions that PKE is CCA secure, H is target collision-free, and DEM is IND secure:* $\epsilon_{\textit{pktt-rcca}} \leq 2(\epsilon_{\textit{tch}} + n\,\epsilon_{\textit{pke-cca}}) + \epsilon_{\textit{dem}}$.

We obtain thus a generic construction of RCCA Hybrid-PKTT with public traceability. One could doubt about the RCCA model. This is, theoretically, a weaker model than the standard CCA. However, it seem to be sufficiently secure for most practical purposes, as showed in [CKN03]. Moreover, in [CKN03], the authors distinguish two types of RCCA: secretly detectable RCCA (sd-RCCA) and publicly detectable RCCA (pd-RCCA). The former means that the detection of a ciphertext, whose underlying plaintext is identical to the underlying plaintext of the challenge, requires secret information and the latter, which is much less restricted, means that such a detection can be done from the public information only. The RCCA used in our proof is publicly detectable RCCA. In fact, if $c = (\sigma_1^*, \ldots, \sigma_n^*, \tau^*)$ is a valid ciphertext outputed by $\mathsf{Hybrid\text{-}PKTT.Enc}_{pk}(m)$, then anyone can choose random $\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n$ to construct a new ciphertext $c' = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_i^*, \sigma_{i+1}, \ldots, \sigma_n, \tau^*)$ so that the decryption of c' under the key sk_i gives back the original message m, *i.e.* $\mathsf{Hybrid\text{-}PKTT.Enc}_{sk_i}(c) = m$.

We call the above scheme the *basic* Hybrid-PKTT scheme. In this scheme, a ciphertext c consists of a ciphertext body τ and a ciphertext header $(\sigma_1, \ldots, \sigma_n)$. The ciphertext body τ has approximately the same size as the message m. We can say somewhat that the transmission rate of the above scheme is asymptotically 1 because the length of the message to be encrypted could be arbitrary.

The two practically inconveniences of the above scheme is that the size of the ciphertext header is linear in the number of users and that the cost of reduction in the security proof is linear on the number of user.

We can overcome both these problems by using the method in [KY02,CPP05] of using a convenient code, namely the IPP codes. Remark that if we use the collusion secure code [BS98], the scheme does not, unfortunately, support the public traceability anymore. The reason is that, in collusion secure code, for tracing back a traitor from a codeword, one has use the secret permutation in the construction of the code and therefore, only the center can do it. Why we don't use the IPP code with the schemes in [CPP05,KY02]? The obstacle is that the basic scheme in [CPP05,KY02] supports only 2 users and there does not exist binary IPP codes. Fortunately, we can combine our basic scheme above with a q-ary IPP code for any $q \geq 3$. As the tracing procedure in IPP code does not require any secret information, the combined scheme supports full public traceability.

We will present the hybrid scheme $\mathsf{Hybrid}_{\text{IPP}}\text{-}\mathsf{PKTT}$ based on IPP codes in the next section. We remark that in order to use q-ary code, the parameter n in the above basic scheme must be set to $n = q$. Since we can choose q as small as $q = 3$, the number of users in each basic scheme is small ($n = q = 3$), the ciphertext header in the new scheme become small, the cost of reduction in the security proof become constant, and this make the new scheme $\mathsf{Hybrid}_{\text{IPP}}\text{-}\mathsf{PKTT}$ become a very efficient scheme.

5 Hybrid-PKTT based on IPP codes

This is the most interesting section of our paper. We will show how to combine the basic scheme in the previous section with a q-ary c-IPP code to construct an efficient hybrid traitor tracing scheme with full-public-traceability.

5.1 IPP codes

Let $\mathcal{Q}$ be an alphabet set containing q symbols. If $C = \{w_1, w_2, \ldots, w_N\} \subset \mathcal{Q}^\ell$, then C is called a q-ary code of size N and length ℓ. Each $w_i \in C$ is called a codeword and we write $w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,\ell})$ where $w_{i,j} \in \mathcal{Q}$ is called the j^{th} component of the codeword w_i.

We define *descendants* of a subset of codewords as follows. Let $X \subset C$ and $u = (u_1, u_2, \ldots, u_\ell) \in \mathcal{Q}^\ell$. The word u is called a descendant of X if for any $1 \leq j \leq \ell$, the j^{th} component u_j of u is equal to a j^{th} component of a codeword in X. In this case, codewords in X are called *parent codewords* of u. For example, $(3, 2, 1, 3)$ is a descendant of three codewords $(3, 1, 1, 2)$, $(1, 2, 1, 3)$ and $(2, 2, 2, 2)$. We denote by $\mathsf{Desc}(X)$ the set of all descendants of X. For a positive integer c, denote by $\mathsf{Desc}_c(C)$ the set of all descendants of subsets of up to c codewords. Codes with identifiable parent property (IPP codes) are defined as follows.

Definition 4. A code C is called c-IPP if, for any $u \in \mathsf{Desc}_c(C)$, there exists $w \in C$ such that for any $X \subset C$, if $|X| \leq c$ and $u \in \mathsf{Desc}(X)$ then $w \in X$.

In a c-IPP code, given a descendant $u \in \mathsf{Desc}_c(C)$, we can always identify at least one of its parent codewords. Binary c-IPP codes (with more than two codewords) do not exists, thus in any c-IPP code, the alphabet size $q \geq 3$. There are many constructions [SW98,SSW01,TM05,TS06] of c-IPP codes. We remark that even both c-IPP codes and c-collusion secure codes can be constructed with large number of codewords of similar length, there are three major differences between them:

- In collusion secure codes, there is an error parameter that specifies the probability that the tracing algorithm fails to output the correct parent codeword, however, in IPP code, there is not such error, thus the tracing is *error-free* and a correct traitor is always identified.
- Collusion secure codes use secret permutation and thus the tracing algorithm cannot be made public, whereas, in IPP codes, everything is public.
- Known collusion secure codes are binary codes, whereas, nontrivial IPP codes have alphabet size at least 3. (Binary IPP code has at most two codewords).

5.2 Description of $\mathsf{Hybrid}_{\text{IPP}}$-PKTT

If a q-ary c-IPP code of size N and length ℓ is used, then constructing ℓ instances of Hybrid-PKTT, in each instance of Hybrid-PKTT set the parameter $n = q$, we have a new public key traitor tracing called $\mathsf{Hybrid}_{\text{IPP}}$-PKTT. In this new scheme, there are $q\ell$ public keys and N users, each user holds ℓ secret keys. The formal construction follows:

Let $C = \{\omega_1, \ldots, \omega_N\}$ be a q-ary c-IPP code that allows collusion of up to c users. The N-user $\mathsf{Hybrid}_{\text{IPP}}$-PKTT scheme is a combination of ℓ basic Hybrid-PKTT schemes $S_1, S_2, \ldots, S_\ell$, each scheme S_i is for q users:

Setup: Given the security parameters λ and c:
For each $j = 1, \ldots, \ell$, call the algorithm $\mathsf{Hybrid\text{-}PKBE.Gen}(1^\lambda, q)$ to generate an encryption key pk_j and q decryption keys $sk_{j,1}, \ldots, sk_{j,q}$ for the q-user system S_j.

Public key: the tuple $(pk_i)_{i=1,\ldots,\ell}$ and the code C.

Private key of each user: user i (for $i = 1, \ldots, N$) is associated to a codeword w_i in C and the corresponding ℓ-tuple key $sk_{1,w_{i,1}}$, $sk_{2,w_{i,2}}$, $\ldots$, $sk_{\ell,w_{i,\ell}}$ where $w_{i,j} \in \mathcal{Q} = \{1, 2, \ldots, q\}$ is the symbol at the j^{th} position of the codeword w_i.

Encryption algorithm: The plaintext space of the ℓ-key system is $\mathcal{M}^\ell$. On input $(m_1, m_2, \ldots, m_\ell)$, the encryption algorithm outputs the ciphertext $(c_1, c_2, \ldots, c_\ell)$, where $c_j = \mathsf{Hybrid\text{-}PKBE.Enc}_{pk_j}(m_j) = (\sigma_{j,1}, \ldots, \sigma_{j,q}, \tau_j)$.

Decryption Algorithm: On the ciphertext $(c_1, c_2, \ldots, c_\ell)$, user i uses his secret key to compute $m_j = \mathsf{Hybrid\text{-}PKBE.Dec}_{sk_{j,w_{i,j}}}(c_j)$.

Public Tracing Algorithm: For each $j = 1, \ldots, \ell$, fix $\ell - 1$ valid ciphertexts $c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_\ell$ and use the tracing algorithm of the instance S_j to trace a traitor $u_j \in \mathcal{Q} = \{1, 2, \ldots, q\}$ in this instance. From the descendant word $(u_1, \ldots, u_\ell) \in \mathcal{Q}^\ell$, identify one of its parent codewords. The user associated with this codeword is a traitor.

Efficiency: The ciphertext contains two parts: ciphertext body $(\tau_1, \ldots, \tau_\ell)$ and ciphertext header $(\sigma_{j,1}, \ldots, \sigma_{j,q})_{j=1,\ldots,\ell}$. The ciphertext body has approximately the same size as the message and the ciphertext header is proportional to $q\ell$. We can choose IPP code with small alphabet size such as $q = 3$ and the code length ℓ is a logarithmic function of the code size N. Thus, the ciphertext header has fixed size and very small compared to the message size.

For the security analysis, one could use the following assumption, from [KY02]: the *threshold assumption* says that a pirate-decoder that just returns correctly a fraction p of a plaintext of length λ where $1 - p$ is a non-negligible function in λ, is useless. However, as already mentioned in [KY02], by employing an all-or-nothing transform [Riv97,CDHKS00], this assumption is not necessary.

Proposition 5. *The leak of the secret keys in the $(\ell - 1)$ q-user systems of ℓ q-user systems does not affect the security of the remained q-user system.*

The proof of this proposition is quite similar to the corresponding ones in [KY02,CPP05] and can be found in the full version [PST06].

This proposition combines with the fact that C is a c-IPP code, leads to following corollary:

Corollary 6. *The above scheme is a N-user, c-traitor tracing scheme with full-public traceability.*

We give an example of a concrete scheme in the full version [PST06]. This scheme is inspired from the Cramer-Shoup scheme [CS03].

6 Conclusion

Motivated by an open problem proposed in CPP05, we extended Tag-KEM/DEM paradigm of hybrid encryption to multi-receiver scenario and constructed a hybrid traitor tracing scheme that has the following properties

- full public traceability, and thus is a comprehensive solution to the open problem of CPP05;
- blackbox traitor tracing algorithm that is *error-free* and always can identify correctly at least one traitor. This is an important advantage of our scheme over [KY02,CPP05], because these schemes use collusion secure code and the tracing algorithm of collusion secure code has error;
- it is a generic construction and provides significant improvement in terms of security and efficiency and this is without resorting to new computational assumptions. In fact security is based on the assumptions underlying security of the public key and symmetric key encryption systems used in KEM and DEM, respectively. In comparison the scheme in [CPP05] (which supports local public traceability against passive attack only) is based on new assumptions which are all stronger than the standard Bilinear DDH assumption;
- the generic construction provides the following powerful composition

IPP code + CCA PKE + semantic secure DEM + target collision-free hash function
$\rightarrow$ (relaxed) CCA Hybrid$_{IPP}$-PKTT with full-public-traceability.

Our security proofs are in replayable CCA model. Although all previous schemes with constant transmission rate achieve only semantic security against passive attack and so our scheme has much stronger security level, it is a quite interesting question if similar results can be derived if CCA model is assumed. Finally, combining revocation and public traceability to the Hybrid Traitor tracing scheme is an important open problem.

References

[AGKS05] M. Abe, R. Gennaro, K. Kurosawa and V. Shoup, Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM, EUROCRYPT'05, LNCS 3494, 128–146, 2005.

[BS98] D. Boneh and J. Shaw, Collusion secure fingerprinting for digital data, *IEEE Transactions on Information Theory* **44** (1998), 1897–1905.

[CDHKS00] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai, Exposure-resilient functions and all-or-nothing transforms, EUROCRYPT'00, LNCS 1807, 453–469, 2000.

[CKN03] R. Canetti, H. Krawczyk and J.B. Nielsen, Relaxing chosen ciphertext security, CRYPTO'03, LNCS 2729, 565–582, 2003.

[CPP05] H. Chabanne, D.H. Phan and D. Pointcheval, Public traceability in traitor tracing schemes, EUROCRYPT'05, LNCS 3494, 542–558, 2005.

[CFN94] B. Chor, A. Fiat and M. Naor, Tracing traitor, CRYPTO'94, LNCS 839, 257–270, 1994.

[CS03] R. Cramer and V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, *SIAM J. of Computing* **33** (2003), 167–226.

[KY02] A. Kiayias and M. Yung, Traitor tracing with constant transmission rate, EUROCRYPT'02, LNCS 2332, 450–465, 2002.

[PST06] D.H. Phan, R. Safavi-Naini and D. Tonien, Generic construction of hybrid public key traitor tracing with full-public-traceability. Full version available from `http://www.di.ens.fr/users/phan/`.

[Riv97] R. Rivest, All-or-nothing encryption and the package transform, FSE'97, LNCS 1267, 210–218, 1997.

[SSW01] J.N. Staddon, D.R. Stinson and R. Wei, Combinatorial properties of frameproof and traceability codes, *IEEE Transactions on Information Theory* **47** (2001), 1042–1049.

[SW98] D.R. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Mathematics* **11** (1998), 41–53.

[TM05] Tran van Trung and S. Martinosyan, New constructions for IPP codes, *Designs, Codes and Cryptography* **35** (2005), 227–239.

[TS06] D. Tonien and R. Safavi-Naini, Recursive constructions of secure codes and hash families using difference function families, *J. of Combinatorial Theory* A **113** (4)(2006), 664–674.