Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions

Seiko Arita and Koji Tsurudome

Institute of Information Security,
Yokohama, Kanagawa, Japan
{arita,mgs068101}@iisec.ac.jp

Abstract. In this paper, we propose a notion of threshold tag-based encryption schemes that simplifies the notion of threshold identity-based encryption schemes, and we show a conversion from any stag-CCA-secure threshold tag-based encryption schemes to CCA-secure threshold public-key encryption schemes. Moreover, we give two concrete constructions of stag-CCA-secure threshold tag-based encryption schemes, under the decisional bilinear Diffie-Hellman assumption and the decisional linear assumption, respectively. Thus, we obtain two concrete constructions of threshold public-key encryption schemes, both of which are non-interactive, robust and can be proved secure without random oracle model. Our threshold public-key encryption schemes are conceptually more simple and shown to be more efficient than those of Boneh, Boyen and Halevi.

Keywords: threshold public-key encryption schemes, tag-based encryption schemes, the decisional bilinear Diffie-Hellman assumption, the decisional linear assumption.

1 Introduction

A threshold public-key encryption scheme is a public-key encryption scheme where a private key is distributed and shared among several decryption servers and some number of those decryption servers must cooperate to decrypt any ciphertext [2,4,9]. In a model of k-out-of-n threshold public-key encryption scheme, an entity, called *combiner*, has a ciphertext C that it wishes to decrypt. The combiner sends C to the decryption servers, and receives partial decryption shares from at least k out of the n decryption servers. It then combines these k partial decryptions into a complete decryption of C. Ideally, it is desirable that there is no other interaction in the system, namely the servers need not talk to each other during decryption. Such threshold systems are called non-interactive. Often one requires that threshold decryption be robust, namely if threshold decryption of a valid ciphertext fails, the combiner can identify the decryption servers that supplied invalid partial decryptions.

M. Abdalla et al. (Eds.): ACNS 2009, LNCS 5536, pp. 186–200, 2009.

In 2006, Boneh, Boyen and Halevi [2] gave a construction of CCA-secure threshold public-key encryption scheme by converting a sID-CPA-secure threshold *identity-based* encryption scheme into it, which in turn is constructed based on the decisional bilinear Diffie-Hellman assumption. Their CCA-secure threshold public-key encryption scheme is the first one that is non-interactive, robust and can be proved secure without random oracle model.

On the other hand, in 2006, Kiltz [6] proposed a notion of tag-based encryption scheme through simplifying the notion of identity-based encryption schemes [3,7], and gave a transformation from any stag-CCA-secure tag-based encryption schemes to CCA-secure public-key encryption schemes.

In this paper, we propose a notion of threshold tag-based encryption schemes that simplifies the notion of threshold identity-based encryption schemes in a similar way as [6], and then we show a conversion from any stag-CCA-secure threshold tag-based encryption schemes to CCA-secure threshold public-key encryption schemes, that is an adaption of the CHK transform [5] to the setting of threshold encryption. Moreover, we give two concrete constructions of stag-CCA-secure threshold tag-based encryption schemes, that are non-interactive, robust and can be proved without random oracle model, under the decisional bilinear Diffie-Hellman assumption and the decisional linear assumption, respectively. Thus, we obtain two concrete constructions of threshold public-key encryption schemes, through applying the conversion to the two threshold tag-based encryption schemes, both of which are non-interactive, robust and can be proved secure without random oracle model.

In the threshold identity-based encryption scheme of [2], a decryption share is regarded as a ciphertext of private key share corresponding to decrypter's ID. But in our threshold tag-based encryption scheme, a decryption share can be regarded naturally as a partial decrypted ciphertext. As a result, our threshold public-key encryption schemes, obtained through the conversion, are conceptually more simple and shown to be more efficient than those of [2].

2 Threshold Tag-Based Encryptions and Their Conversion to Threshold Public-Key Encryptions

In this section, after reviewing the definition of threshold public-key encryptions and their security following [2], we propose a notion of threshold tag-based encryptions and show a conversion from any stag-CCA-secure threshold tag-based encryption schemes to CCA-secure threshold public-key encryption schemes.

2.1 Threshold Public-Key Encryption

Scheme. A threshold public-key encryption scheme TPKE consists of five algorithms:

$$\mathsf{TPKE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDec}, \mathsf{ShareVf}, \mathsf{Combine}).$$

Setup takes as input the number of decryption servers n, a threshold k ($1 \leq k \leq n$), and a security parameter Λ. It outputs a triple (PK, VK, SK) where PK

is a public key, VK is a verification key and $SK = (SK_1, \cdots, SK_n)$ is a vector of n private key shares. $\mathsf{Encrypt}$ takes as input a public key PK and a message M, and it outputs a ciphertext C. $\mathsf{ShareDec}$ takes as input a public key PK, a ciphertext C, and the i-th private key share (i, SK_i). It outputs a decryption share μ_i of the encrypted message, or a special symbol $(i, \perp)$. $\mathsf{ShareVf}$ takes as input a public key PK, verification key VK, ciphertext C and a decryption share μ_i. It outputs valid or $\mathsf{invalid}$. When the output is valid, we say that μ_i is a valid decryption share of C. $\mathsf{Combine}$ takes as input PK, VK, ciphertext C, and k decryption shares $\{\mu_1, \cdots, \mu_k\}$. It outputs a message M or $\perp$.

For any output (PK, VK, SK) of $\mathsf{Setup}(n, k, \Lambda)$, we require the two consistency properties:

1. For any valid ciphertext C, if $\mu_i \leftarrow \mathsf{ShareDec}(PK, i, SK_i, C)$, then $\mathsf{ShareVf}(PK, VK, C, \mu_i)$ is valid.
2. If C is the output of $\mathsf{Encrypt}(PK, M)$ and $S = \{\mu_1, \cdots, \mu_k\}$ is a set of decryption shares $\mu_i \leftarrow \mathsf{ShareDec}(PK, i, SK_i, C)$ for k distinct private key shares in SK, then $\mathsf{Combine}(PK, VK, C, S) = M$.

Security. Security of threshold public-key encryption scheme TPKE is defined in terms of chosen ciphertext security and decryption consistency. *Chosen ciphertext security* is defined using the following game between a challenger and an adversary. Both are given n, k, Λ as input.

1. Init. The adversary outputs a set $S \subset \{1, \cdots, n\}$ of $k-1$ decryption servers to corrupt.
2. Setup. The challenger runs $\mathsf{Setup}(n, k, \Lambda)$ to obtain a random instance (PK, VK, SK). It gives the adversary PK, VK, and all (j, SK_j) for $j \in S$.
3. Query phase 1. The adversary adaptively issues decryption queries (C, i) where $C \in \{0,1\}^*$ and $i \in \{1, \cdots, n\}$. The challenger responds with $\mathsf{ShareDec}(PK, i, SK_i, C)$.
4. Challenge. The adversary outputs two messages M_0, M_1 of equal length. The challenger picks a random $b \in \{0,1\}$ and lets $C^* \leftarrow \mathsf{Encrypt}(PK, M_b)$. It gives C^* to the adversary.
5. Query phase 2. The adversary issues further decryption queries (C, i), under the constraint that $C \neq C^*$. The challenger responds as in Query Phase 1.
6. Guess. The adversary outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We define an advantage of adversary $\mathcal{A}$ for threshold public-key encryption scheme TPKE with respect to chosen ciphertext security as $\mathsf{Adv}^{cca}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda) = |\Pr[b = b'] - 1/2|$.

Decryption consistency is defined using the following game. The game starts with the Init, Setup, and Query phase 1 steps as in the game above. The adversary then outputs a ciphertext C and two sets of decryption shares $S = \{\mu_1, \cdots, \mu_k\}$ and $S' = \{\mu'_1, \cdots, \mu'_k\}$ each of size k. The adversary wins if:

- The shares in S and S' are valid decryption shares for C under VK.
- S and S' each contain decryption shares from k distinct servers.
- $\mathsf{Combine}(PK, VK, C, S) \neq \mathsf{Combine}(PK, VK, C, S')$, with either side not equal to $\perp$.

We let $\mathsf{Adv}^{dc}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda)$ denote the probability that the adversary $\mathcal{A}$ wins this game.

Definition 1. *We say that a threshold public-key encryption scheme* TPKE *is* CCA-secure *if for any n, k ($1 \leq k \leq n$) and any probabilistic polynomial time algorithm $\mathcal{A}$, both of the functions $\mathsf{Adv}^{cca}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda)$ and $\mathsf{Adv}^{dc}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda)$ are negligible.*

2.2 Threshold Tag-Based Encryption

A notion of threshold tag-based encryptions is obtained by simplifying threshold identity-based encryption schemes in a similar way as [6] in the non-threshold setting. Threshold tag-based encryptions simply needs a *tag* as input in addition to ordinary inputs of threshold encryptions.

Scheme. A threshold tag-based encryption scheme TTBE consists of five algorithms:

$$\mathsf{TTBE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDec}, \mathsf{ShareVf}, \mathsf{Combine}).$$

Setup takes as input the number of decryption servers n, a threshold k ($1 \leq k \leq n$) and a security parameter Λ. It outputs a triple (PK, VK, SK) where PK is a public key, VK is a verification key, and $SK = (SK_1, \cdots, SK_n)$ is a vector of n private key shares. Encrypt takes as input a public key PK, a *tag* t and a message M, and it outputs a ciphertext C. ShareDec takes as input a public key PK, a ciphertext C, a *tag* t, and a i-th private key share (i, SK_i). It outputs a decryption share μ_i of the encrypted message, or a special symbol $(i, \perp)$. ShareVf takes as input PK, VK, a ciphertext C, a *tag* t and a decryption share μ_i. It outputs valid or invalid. When the output is valid, we say that μ_i is a valid decryption share of C. Combine takes as input PK, VK, a ciphertext C, a *tag* t and k decryption shares $\{\mu_1, \cdots, \mu_k\}$. It outputs a message M or $\perp$.

As in the threshold public-key encryption scheme, we require the following two consistency properties. Let (PK, VK, SK) be the output of $\mathsf{Setup}(n, k, \Lambda)$.

1. For any tuple (C, t) of a valid ciphertext and a tag, if $\mu_i \leftarrow \mathsf{ShareDec}(PK, i, SK_i, C, t)$, then $\mathsf{ShareVf}(PK, VK, C, t, \mu_i) = \mathsf{valid}$.
2. If C is the output of $\mathsf{Encrypt}(PK, t, M)$ and $S = \{\mu_1, \cdots, \mu_k\}$ is a set of decryption shares $\mu_i \leftarrow \mathsf{ShareDec}(PK, i, SK_i, C, t)$ for k distinct private key shares in SK, then $\mathsf{Combine}(PK, VK, C, t, S) = M$.

Security. Security of threshold tag-based encryption scheme TTBE is defined in terms of stag-chosen-ciphertext security and stag decryption consistency. *Stag chosen ciphertext security* is defined using the following game between a challenger and an adversary. Both are given n, k, Λ as input.

1. Init. The adversary outputs a target tag t^* that it wants to attack and a set of $k-1$ decryption servers $S \subset \{1, \cdots, n\}$ that it wants to corrupt.
2. Setup. The challenger runs $\mathsf{Setup}(n, k, \Lambda)$ to obtain a random instance (PK, VK, SK). It gives the adversary PK, VK, and all (j, SK_j) for $j \in S$.
3. Query phase 1. The adversary adaptively issues decryption share queries $((C, t), i)$ with $i \in \{1, \cdots, n\}$, under the constraint that $t \neq t^*$. The challenger responds with $\mathsf{ShareDec}(PK, i, SK_i, C, t)$.
4. Challenge. The adversary outputs two messages M_0, M_1 of equal length. The challenger picks a random $b \in \{0, 1\}$ and lets $C^* \leftarrow \mathsf{Encrypt}(PK, t^*, M_b)$. It gives C^* to the adversary.
5. Query phase 2. The adversary adaptively issues decryption share queries $((C, t), i)$ with $i \in \{1, \cdots, n\}$, under the constraint that $t \neq t^*$. The challenger responds as in phase 1.
6. Guess. The adversary outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We define an advantage of adversary $\mathcal{A}$ for threshold tag-based encryption scheme TTBE with respect to stag-chosen-ciphertext security as $\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE},n,k}(\Lambda) = |\Pr[b = b'] - 1/2|$.

Stag decryption consistency is defined using the following game. The game starts with the Init, Setup and Query phase 1 steps as in the game above. The adversary then outputs a tag t, a ciphertext C and two sets of decryption shares $S = \{\mu_1, \cdots, \mu_k\}$ and $S' = \{\mu'_1, \cdots, \mu'_k\}$ each of size k. The adversary wins if:

1. The shares in S and S' are valid decryption shares for (C, t) under VK.
2. S and S' each contain decryption shares from k distinct servers.
3. $\mathsf{Combine}(PK, VK, C, t, S) \neq \mathsf{Combine}(PK, VK, C, t, S')$, with either side not equal to $\perp$.

We let $\mathsf{Adv}^{stag-dc}_{\mathcal{A},\mathsf{TTBE},n,k}(\Lambda)$ denote the probability that the adversary $\mathcal{A}$ wins this game.

Definition 2. *We say that a threshold tag-based encryption scheme* TTBE *is* stag-CCA-secure *if for any* n, k $(1 \leq k \leq n)$ *and any probabilistic polynomial time algorithm* $\mathcal{A}$, *both of the functions* $\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE},n,k}(\Lambda)$ *and* $\mathsf{Adv}^{stag-dc}_{\mathcal{A},\mathsf{TTBE},n,k}(\Lambda)$ *are negligible.*

2.3 Conversion from Threshold Tag-Based Encryption Schemes into Threshold Public-Key Encryption Schemes

In this section we show a conversion from any stag-CCA-secure threshold tag-based encryption scheme to CCA-secure threshold public-key encryption scheme. The conversion is a direct adjustment of the conversions of [5,6] into the threshold setting.

We convert a given threshold tag-based encryption scheme

$$\mathsf{TTBE} = (\mathsf{Setup}_{\mathsf{ttbe}}, \mathsf{Encrypt}_{\mathsf{ttbe}}, \mathsf{ShareDec}_{\mathsf{ttbe}}, \mathsf{ShareVf}_{\mathsf{ttbe}}, \mathsf{Combine}_{\mathsf{ttbe}})$$

into a threshold public-key encryption scheme

$\mathsf{TPKE} = \mathrm{TT2TP}(\mathsf{TTBE}, \mathsf{S}) = (\mathsf{Setup}_{\mathsf{tpke}}, \mathsf{Encrypt}_{\mathsf{tpke}}, \mathsf{ShareDec}_{\mathsf{tpke}}, \mathsf{ShareVf}_{\mathsf{tpke}}, \mathsf{Combine}_{\mathsf{tpke}})$

using a strong one-time signature $\mathsf{S} = (\mathsf{KG}, \mathsf{SGN}, \mathsf{VF})$ as in Figure 1.

$\mathbf{Setup}_{\mathbf{tpke}}(n, k, \Lambda)$: $(PK, VK, SK) \leftarrow \mathsf{Setup}_{\mathsf{ttbe}}(n, k, \Lambda)$, output (PK, VK, SK).
$\mathbf{Encrypt}_{\mathbf{tpke}}(PK, M)$: $(sigk, verk) \leftarrow \mathsf{KG}(\Lambda)$, $C_{ttbe} \leftarrow \mathsf{Encrypt}_{\mathsf{ttbe}}(PK, verk, M)$, $\sigma \leftarrow \mathsf{SGN}(sigk, C_{ttbe})$; output $C_{tpke} = (C_{ttbe}, verk, \sigma)$.
$\mathbf{ShareDec}_{\mathbf{tpke}}(PK, i, SK_i, C_{tpke} = (C_{ttbe}, verk, \sigma))$: If $\mathsf{VF}(verk, C_{ttbe}, \sigma) = \mathsf{invalid}$ then output $\mu_i = (i, \perp)$, else output $\mathsf{ShareDec}_{\mathsf{ttbe}}(PK, i, SK_i, C_{ttbe}, verk)$.
$\mathbf{ShareVf}_{\mathbf{tpke}}(PK, VK, C_{tpke} = (C_{ttbe}, verk, \sigma), \mu_i)$: If $\mathsf{VF}(verk, C_{ttbe}, \sigma) = \mathsf{invalid}$ then output $\mathsf{invalid}$, else output $\mathsf{ShareVf}_{\mathsf{ttbe}}(PK, VK, C_{ttbe}, verk, \mu_i)$.
$\mathbf{Combine}_{\mathbf{tpke}}(PK, VK, C_{tpke} = (C_{ttbe}, verk, \sigma), \{\mu_1, \cdots, \mu_k\})$: If $\exists i$, $\mu_i = (i, \perp)$ or $\mathsf{ShareVf}_{\mathsf{tpke}}(PK, VK, C_{tpke}, \mu_i) = \mathsf{invalid}$ then output $\perp$, else output $\mathsf{Combine}_{\mathsf{ttbe}}(PK, VK, C_{ttbe}, verk, \{\mu_1, \cdots, \mu_k\})$.

Fig. 1. TT2TP: Conversion from threshold tag-based encryption schemes to threshold public-key encryption schemes

Theorem 1. *If a threshold tag-based encryption scheme* TTBE *is stag-CCA-secure and* S *is a strong one-time signature, then the threshold public-key encryption scheme* $\mathsf{TPKE} = \mathrm{TT2TP}(\mathsf{TTBE}, \mathsf{S})$ *is CCA-secure.*

More precisely, for an arbitrary efficient adversary $\mathcal{A}$ *against chosen ciphertext security of* TPKE*, there exists an efficient algorithm* $\mathcal{B}$ *against stag-chosen-ciphertext security of the underlying* TTBE *and a forger* $\mathcal{F}$ *of the underlying* S *that satisfy*

$$\mathsf{Adv}^{cca}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda) \leq \mathsf{Adv}^{stag-cca}_{\mathcal{B},\mathsf{TTBE},n,k}(\Lambda) + \mathsf{Adv}^{ot-cma}_{\mathcal{F},\mathsf{S}}(\Lambda).$$

(Here, $\mathsf{Adv}^{ot-cma}_{\mathcal{F},\mathsf{S}}$ *denotes the advantage of forger* $\mathcal{F}$ *against one-time signature* S *in the usual game of strong chosen-message attack with at most one signing query.) Similarly, for an arbitrary efficient adversary* $\mathcal{A}'$ *against decryption consistency of* TPKE*, there exists an efficient algorithm* $\mathcal{B}'$ *against stag decryption consistency of the underlying* TTBE *and a forger* $\mathcal{F}'$ *of the underlying* S *that satisfy*

$$\mathsf{Adv}^{dc}_{\mathcal{A}',\mathsf{TPKE},n,k}(\Lambda) \leq \mathsf{Adv}^{stag-dc}_{\mathcal{B}',\mathsf{TTBE},n,k}(\Lambda) + \mathsf{Adv}^{ot-cma}_{\mathcal{F}',\mathsf{S}}(\Lambda).$$

Proof. First, we consider chosen ciphertext security of TPKE. Let $\mathcal{A}$ be an arbitrary efficient adversary against chosen ciphertext security of TPKE. Using adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that attacks stag-chosen-ciphertext security of the underlying TTBE.

Algorithm $\mathcal{B}$ proceeds as follows:

1. Initialization. Given input (n, k, Λ) algorithm $\mathcal{B}$ runs $\mathcal{A}$ on the same input to obtain a list S ($\subset \{1, \cdots, n\}$) of the $k-1$ servers that $\mathcal{A}$ wishes to corrupt. Next, $\mathcal{B}$ runs KG on Λ to obtain a signing key $sigk^*$ and a verification key $verk^*$. It outputs the set S and the target tag $t^* = verk^*$ to the TTBE challenger.
2. Setup. The TTBE challenger runs $\mathsf{Setup}_{\mathsf{ttbe}}(n, k, \Lambda)$ to obtain (PK, VK, SK). It gives $\mathcal{B}$ the values PK, VK, and all (j, SK_j) for $j \in S$. Algorithm $\mathcal{B}$ forwards these values to $\mathcal{A}$.
3. Query Phase 1. Adversary $\mathcal{A}$ adaptively issues decryption queries of the form (C_{tpke}, i) where $C_{tpke} = (C_{ttbe}, verk, \sigma)$ and $i \in \{1, \cdots, n\}$. For each such a query (C_{tpke}, i), $\mathcal{B}$ proceeds as follows:
   (a) If $\mathsf{VF}(verk, C_{ttbe}, \sigma) = \mathsf{invalid}$ then $\mathcal{B}$ gives $\mu_i = (i, \perp)$ to $\mathcal{A}$.
   (b) Else if $verk = t^*$ then $\mathcal{B}$ outputs $b \xleftarrow{\$} \{0, 1\}$ and aborts.
   (c) Else $\mathcal{B}$ issues a decryption query $((C_{ttbe}, verk), i)$ to own TTBE decryption oracle and obtains a decryption share μ_i in return. It gives the decryption share μ_i to $\mathcal{A}$.
4. Challenge. Adversary $\mathcal{A}$ outputs two equal-length messages M_0 and M_1. $\mathcal{B}$ forwards these M_0 and M_1 to its own TTBE challenger. The TTBE challenger responds with the encryption C^*_{ttbe} of M_b under t^* for some $b \in \{0, 1\}$. $\mathcal{B}$ then runs SGN on $(sigk^*, C^*_{ttbe})$ to obtain a signature σ^*, and it gives $C^*_{tpke} = (C^*_{ttbe}, t^*, \sigma^*)$ to $\mathcal{A}$ as challenge ciphertext.
5. Query Phase 2. $\mathcal{A}$ continues to issue decryption queries $(C_{tpke} \ (\neq C^*_{tpke}), i)$. $\mathcal{B}$ responds as in Query Phase 1.
6. Guess. Eventually, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b. $\mathcal{B}$ forwards b' to the TTBE challenger and wins the game if $b = b'$.

This completes the description of algorithm $\mathcal{B}$.

Let Abort be the event that $\mathcal{B}$ aborts in Query Phase 1 or 2 during the simulation. As easily seen, as long as Abort does not happen, $\mathcal{B}$'s simulation of TPKE challenger is perfect. Therefore, we have $|\mathsf{Adv}^{stag-cca}_{\mathcal{B},\mathsf{TTBE},n,k}(\Lambda) - \mathsf{Adv}^{cca}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda)| < \Pr[\mathsf{Abort}]$. By definition, Abort means $\mathcal{A}$'s forgery of valid signature σ under verification key $verk^*$, and it leads to a forger $\mathcal{F}$ of S satisfying $\Pr[\mathsf{Abort}] \leq \mathsf{Adv}^{ot-cma}_{\mathcal{F},\mathsf{S}}$. Thus, $\mathsf{Adv}^{cca}_{\mathcal{A},\mathsf{TPKE},n,k}(\Lambda) \leq \mathsf{Adv}^{stag-cca}_{\mathcal{B},\mathsf{TTBE},n,k}(\Lambda) + \mathsf{Adv}^{ot-cma}_{\mathcal{F},\mathsf{S}}(\Lambda)$.

Second, we see decryption consistency of TPKE. Let $\mathcal{A}'$ be an arbitrary efficient adversary against decryption consistency of TPKE. Using adversary $\mathcal{A}'$, we build an algorithm $\mathcal{B}'$ that attacks stag decryption consistency of the underlying TTBE.

Algorithm $\mathcal{B}'$ proceeds exactly as algorithm $\mathcal{B}$, until $\mathcal{A}'$ outputs the challenge $(\hat{C_{tpke}} = (\hat{C_{ttbe}}, \hat{verk}, \hat{\sigma}), S, S')$, and then $\mathcal{B}'$ outputs $(\hat{verk}, \hat{C_{ttbe}}, S, S')$ after verifying validity of $\hat{\sigma}$ under $\hat{verk}$.

Just as in the case of chosen ciphertext security, let Abort be the event that $\mathcal{B}'$ aborts in Query Phase 1 during the simulation. Then, as above, $|\mathsf{Adv}^{stag-dc}_{\mathcal{B}',\mathsf{TPKE},n,k}(\Lambda) - \mathsf{Adv}^{dc}_{\mathcal{A}',\mathsf{TTBE},n,k}(\Lambda)| < \Pr[\mathsf{Abort}]$. Again, Abort leads to a forger $\mathcal{F}'$ of S, and we have $\mathsf{Adv}^{dc}_{\mathcal{A}',\mathsf{TPKE},n,k}(\Lambda) \leq \mathsf{Adv}^{stag-dc}_{\mathcal{B}',\mathsf{TTBE},n,k}(\Lambda) + \mathsf{Adv}^{ot-cma}_{\mathcal{F}',\mathsf{S}}(\Lambda)$. □

3 Construction of Threshold Tag-Based Encryption Schemes

In this section, we construct two concrete stag-CCA-secure threshold tag-based encryption schemes based on the decisional bilinear Diffie-Hellman assumption and on the decisional linear assumption, respectively.

3.1 Preliminaries

We recall necessary primitives around bilinear maps.

Bilinear Maps. Let $\mathbb{G}$ be a group of prime order p with generator g. Let $\mathbb{G}_1$ be another group of prime order p. A *bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a map with the properties:

1. For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, it holds $e(u^a, v^b) = e(u, v)^{ab}$.
2. $e(g, g) \neq 1$.
3. For all u, v, $e(u, v)$ is efficiently computable.

Decisional Bilinear Diffie-Hellman Assumption. If a bilinear Diffie-Hellman tuple $(g, g^a, g^b, g^c, e(g, g)^{abc})$ is indistinguishable from a bilinear random tuple $(g, g^a, g^b, g^c, e(g, g)^d)$, we say the decisional bilinear Diffie-Hellman assumption holds. More formally, as for algorithm G_{DBDH} that takes a security parameter Λ and outputs order p, generator g, and descriptions of groups $\mathbb{G}$ and $\mathbb{G}_1$ with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, the following two experiments are defined. $\mathbf{Exp}^{\mathsf{bdh-1}}_{G_{DBDH},\mathcal{A}}$ on input Λ generates $param = (p, g, \mathbb{G}, \mathbb{G}_1, e)$ by $G_{DBDH}(\Lambda)$ and chooses three random elements a, b, c from $\mathbb{Z}_p$. Then it invokes $\mathcal{A}$ on $(param, g, g^a, g^b, g^c, e(g, g)^{abc})$ and returns its output. On a while, $\mathbf{Exp}^{\mathsf{bdh-2}}_{G_{DBDH},\mathcal{A}}$ chooses four random elements a, b, c, d from $\mathbb{Z}_p$ and returns $\mathcal{A}(param, g, g^a, g^b, g^c, e(g, g)^d)$.

We say that the *decisional bilinear Diffie-Hellman (DBDH) assumption* holds for G_{DBDH} if for any probabilistic polynomial time algorithm $\mathcal{A}$, $\mathsf{Adv}^{dbdh}_{\mathcal{A},G_{DBDH}}(\Lambda) \stackrel{def}{=} \left| \Pr[\mathbf{Exp}^{\mathsf{bdh-1}}_{G_{DBDH},\mathcal{A}}(\Lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{bdh-2}}_{G_{DBDH},\mathcal{A}}(\Lambda) = 1] \right|$ is a negligible function of Λ.

Decisional Linear Assumption. If a linear tuple $(g_1, g_2, z, g_1^{r_1}, g_2^{r_2}, z^{r_1+r_2})$ is indistinguishable from a random tuple $(g_1, g_2, z, g_1^{r_1}, g_2^{r_2}, z^s)$, we say the decisional linear assumption holds. More formally, as for algorithm G_{DLIN} that takes a security parameter Λ and outputs order p, generator g, and descriptions of groups $\mathbb{G}$ and $\mathbb{G}_1$ with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, the following two experiments are defined. $\mathbf{Exp}^{\mathsf{lin-1}}_{G_{DLIN},\mathcal{A}}$ on input Λ generates $param = (p, g, \mathbb{G}, \mathbb{G}_1, e)$ by $\mathcal{G}_{DLIN}(\Lambda)$ and chooses four random elements u, v, r_1, r_2 from $\mathbb{Z}_p$. Then it invokes $\mathcal{A}$ on $(param, g, g^u, g^v, g^{r_1}, g^{ur_2}, g^{v(r_1+r_2)})$ and returns its output. On a while, $\mathbf{Exp}^{\mathsf{lin-2}}_{G_{DLIN},\mathcal{A}}$ chooses five random elements u, v, r_1, r_2, s from $\mathbb{Z}_p$ and returns $\mathcal{A}(param, g, g^u, g^v, g^{r_1}, g^{ur_2}, g^{vs})$.

We say that the *decisional linear (DLIN) assumption* holds for $\mathcal{G}_{DLIN}$ if for any probabilistic polynomial time algorithm $\mathcal{A}$, $\mathsf{Adv}^{dlin}_{\mathcal{A},G_{DLIN}}(\Lambda) \stackrel{def}{=} \left|\Pr[\mathbf{Exp}^{\mathsf{lin-1}}_{G_{DLIN},\mathcal{A}}(\Lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{lin-2}}_{G_{DLIN},\mathcal{A}}(\Lambda) = 1]\right|$ is a negligible function of Λ.

3.2 A Construction TTBE1 of Threshold Tag-Based Encryption Scheme Based on the DBDH Assumption

Our first construction TTBE1 of threshold tag-based encryption scheme is obtained through a simplification and "thresholding" of the identity-based encryption scheme of Boneh and Boyen [1].

As easily seen, the identity-based encryption scheme of [1] can be simplified into a following tag-based encryption scheme. A public-key is randomly selected elements $g_1(=g^x), g_2, h_1$ on a bilinear group $\mathbb{G}$ (with generator g). The corresponding secret key is x. A message M is encrypted with respect to tag t as $(C, D, E) = (g^r, (g_1^t h_1)^r, M \cdot e(g_1, g_2)^r)$. Ciphertext (C, D, E) is decrypted with respect to tag t as $M = E/e(C, g_2)^x$ if it holds $e(C, g_1^t h_1) = e(D, g)$, otherwise $M = \perp$.

In the threshold identity-based encryption scheme of [2], which is also based on the identity-based scheme of [1], a decryption share is regarded as a ciphertext of private key share corresponding to decrypter's ID. On a while, in order to convert the above tag-based encryption scheme into a threshold version, thanks to the simple setting of tag-based encryption scheme, we can naturally distribute the secret key x into shares $\{f(i)\}_i$ using Shamir's secret sharing scheme [8] and make the i-th decryption share to be $(C^{f(i)}, E)$ as the usual threshold version of ElGamal encryption. More precisely TTBE1 is described in Figure 2.

Setup(n, k, Λ): $(p, g, \mathbb{G}, \mathbb{G}_1, e) \leftarrow G_{DBDH}(\Lambda)$; $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ f \stackrel{\$}{\leftarrow} \mathbb{Z}_p[X]$ satisfying $\deg(f) = k-1$ and $f(0) = x$; $y, z \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ g_1 \leftarrow g^x,\ g_2 \leftarrow g^y,\ h_1 \leftarrow g^z$; $PK = (p, \mathbb{G}, \mathbb{G}_1, e, g, g_1, g_2, h_1),\ SK = (f(1), \cdots, f(n)),\ VK = (g^{f(1)}, \cdots, g^{f(n)})$; return (PK, VK, SK).
Encrypt(PK, t, M): $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ C \leftarrow g^r,\ D \leftarrow ({g_1}^t h_1)^r,\ E \leftarrow M \cdot e(g_1, g_2)^r$, return $C_{tbe} = (C, D, E)$.
ShareDec$(PK, i, SK_i = f(i), C_{tbe} = (C, D, \cdot), t)$: If $e(C, {g_1}^t h_1) \neq e(D, g)$ then return $\mu_i = (i, \perp)$ else return $\mu_i = (i, C^{f(i)})$.
ShareVf$(PK, VK = (g^{f(i)}), C_{tbe} = (C, \cdot, \cdot), t, \mu_i = (i, C_i))$: If $e(C_i, g) \neq e(C, g^{f(i)})$ then return invalid else return valid.
Combine$(PK, VK, C_{tbe} = (\cdot, \cdot, E), t, \{\mu_1 = (1, C_1), \cdots, \mu_k = (k, C_k)\})$: If $\exists i$, **ShareVf**$(PK, VK_i, C_{tbe}, t, \mu_i) =$ invalid then return $\perp$, else return $E/e(\prod_{i=1}^k C_i^{\lambda_i}, g_2)$ using Lagrange coefficients $\lambda_1, \cdots, \lambda_k$ satisfying $f(0) = \sum_{i=1}^k \lambda_i f(i)$.

Fig. 2. Threshold Tag-Based Encryption Scheme TTBE1

Theorem 2. *Under the DBDH assumption for G_{DBDH}, the threshold tag-based encryption scheme* TTBE1 *is stag-CCA-secure.*

More precisely, for an arbitrary adversary $\mathcal{A}$ against stag-chosen-ciphertext security of TTBE1 *that runs in time at most τ and makes at most Q decryption queries, there exists an algorithm $\mathcal{B}$ for the DBDH problem on G_{DBDH} that runs in time at most τ plus the time to perform $O(Q+n)$ exponentiations and $O(Q)$ pairing computations, and satisfies*

$$\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE1},n,k}(\Lambda) = \mathsf{Adv}^{dbdh}_{\mathcal{B},G_{DBDH}}(\Lambda).$$

For an arbitrary adversary $\mathcal{A}'$ against stag decryption consistency of TTBE1*, it holds that*

$$\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE1},n,k}(\Lambda) = 0.$$

Proof. First, we consider stag-chosen-ciphertext security of TTBE1. Let $\mathcal{A}$ be an arbitrary adversary that runs in time at most τ, makes at most Q decryption queries, and has advantage $\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE1},n,k}(\Lambda)$ in attacking TTBE1 in the game of stag-chosen-ciphertext security. Using the adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the DBDH problem on $G_{DBDH}(\Lambda)$.

Given $(\Lambda, p, \mathbb{G}, \mathbb{G}_1, e, g, g^a, g^b, g^c, W)$ as input, algorithm $\mathcal{B}$ proceeds as follows. (The aim of $\mathcal{B}$ is to distinguish two cases between $W = e(g,g)^{abc}$ or random.)

1. Initialization. Algorithm $\mathcal{B}$ invokes adversary $\mathcal{A}$ on input (n, k, Λ). Adversary $\mathcal{A}$ outputs a target tag t^* and a list $S = \{s_1, \cdots, s_{k-1}\} (\subset \{1, \cdots, n\})$ of the $k-1$ servers that it wishes to corrupt.
2. Setup. Then, $\mathcal{B}$ does the following:
   (a) $\mathcal{B}$ sets $g_1 = g^a, g_2 = g^b$ and computes $h_1 = g_1^{-t^*} g^\gamma$ with a random γ ($\stackrel{\$}{\leftarrow} \mathbb{Z}_p$). (This defines implicitly as $x = a, y = b, z = -t^* x + \gamma$.) $\mathcal{B}$ sets $PK = (p, \mathbb{G}, \mathbb{G}_1, e, g, g_1, g_2, h_1)$.
   (b) Next, $\mathcal{B}$ picks $k-1$ random integers $\alpha_1, \cdots, \alpha_{k-1} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. (We let $f \in \mathbb{Z}_p[X]$ be a polynomial of degree $k-1$ defined by $f(0) = x$ and $f(s_i) = \alpha_i$ for $i = 1, \cdots, k-1$. $\mathcal{B}$ does not know f.) $\mathcal{B}$ sets $SK|_S = (\alpha_1, \cdots, \alpha_{k-1})$.
   (c) For $i \in S$, $\mathcal{B}$ lets $u_i = g^{\alpha_i}$. For $i \notin S$, it computes $u_i = g_1^{\lambda_0} (g^{\alpha_1})^{\lambda_1} \cdots (g^{\alpha_{k-1}})^{\lambda_{k-1}}$, where $\lambda_0, \cdots, \lambda_{k-1} (\in \mathbb{Z}_p)$ are the Lagrange coefficients satisfying $f(i) = \lambda_0 f(0) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$. (Note u_i satisfies $u_i = g^{f(i)}$.) $\mathcal{B}$ sets $VK = (u_1, \cdots, u_n)$.
   (d) $\mathcal{B}$ gives PK, VK and $SK|_S$ to $\mathcal{A}$.
3. Phase 1. $\mathcal{A}$ issues decryption share queries $((C_{tbe}, t), i)$ under the constraint that $t \neq t^*$ and $i \notin S$. First, $\mathcal{B}$ validates $e(C, g_1^t h_1) \stackrel{?}{=} e(D, g)$ to clarify the validity of ciphertext $C_{tbe} = (C, D, E)$. If validity test fails, $\mathcal{B}$ gives to $\mathcal{A}$ $(i, \perp)$. Otherwise, $\mathcal{B}$ computes the Lagrange coefficients $\lambda_1, \cdots, \lambda_{k-1}, \lambda_i \in \mathbb{Z}_p$ satisfying $f(0) = \lambda_i f(i) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$ and sets

$$C_i = \left\{ \frac{(\frac{D}{C^\gamma})^{\frac{1}{t-t^*}}}{C^{\sum_{j=1}^{k-1} \lambda_j \alpha_j}} \right\}^{\frac{1}{\lambda_i}}$$

and then gives to $\mathcal{A}$ (i, C_i) as decryption share. (If the ciphertext is valid, it must be the case that $C^{tx+z} = D$. Then, we have $D = C^{tx+z} = C^{tx+(-t^*x+\gamma)} = (C^x)^{t-t^*}C^\gamma$. Since $t \neq t^*$, we get $C^x = (D/C^\gamma)^{\frac{1}{t-t^*}}$. Then, substituting $f(0) = \sum_{j=1}^{k-1} \lambda_j f(s_j) + \lambda_i f(i)$ for x, and noting $C_i = C^{f(i)}$ we obtain the above expression of C_i.)

4. Challenge. $\mathcal{A}$ outputs two same-length messages M_0 and M_1. $\mathcal{B}$ flips a fair coin $b \in \{0, 1\}$, and responds with the challenge ciphertext $C^*_{tbe} = (g^c, (g^c)^\gamma, M_b W)$. (As $\mathcal{B}$ sets $h_1 = g_1^{-t^*} g^\gamma$, it holds that $(g^c)^\gamma = (h_1 g_1^{t^*})^c$. Moreover, if $W = e(g, g)^{abc}$, then we have $M_b \cdot W = M_b \cdot e(g_1, g_2)^c$ and C^*_{tbe} is a valid ciphertext of M_b under PK with tag t^*.)
5. Phase 2. $\mathcal{A}$ issues additional queries as in Phase 1, to which $\mathcal{B}$ responds as before.
6. Guess. Eventually, $\mathcal{A}$ outputs a guess b'. $\mathcal{B}$ outputs 1 if $b = b'$, or outputs 0 otherwise.

This completes the description of algorithm $\mathcal{B}$, that runs in time at most τ plus the time to perform $O(Q + n)$ exponentiations and $O(Q)$ pairing computations. By the comments in the description, it is immediate that $\mathcal{B}$ perfectly simulates a stag-CCA game for $\mathcal{A}$ if $W = e(g, g)^{abc}$. When W is a random element, the view of $\mathcal{B}$ is independent of the choice of b. So, $\mathsf{Adv}^{dbdh}_{\mathcal{B},G_{DBDH}}(\Lambda) = |\Pr[b = b'] - 1/2| = |(1/2 + \mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE1},n,k}(\Lambda)) - 1/2| = \mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE1},n,k}(\Lambda)$.

Second, we consider stag decryption consistency of TTBE1. Let $\mathcal{A}'$ be an arbitrary adversary with advantage $\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE1},n,k}(\Lambda)$ in attacking TTBE1 in the game of stag decryption consistency. Suppose adversary $\mathcal{A}'$ outputs $t, C_{tbe}, S = (\mu_1 = (1, C_1), \cdots, \mu_k = (1, C_k)), S' = (\mu'_1 = (1, C'_1), \cdots, \mu'_k = (k, C'_k))$. If those shares μ_i in S are valid, they must satisfy $e(C_i, g) = e(C, g^{f(i)})$, so $C_i = C^{f(i)}$. Then, it holds that $\prod_{i=1}^k C_i^{\lambda_i} = C^{\sum_{i=1}^k \lambda_i f(i)} = C^x$. Similarly, if the shares μ'_i in S' are valid, we have $\prod_{i=1}^k {C'}_i^{\lambda_i} = C^x$. This means $\mathsf{Combine}(PK, VK, C_{tbe}, t, S) = \mathsf{Combine}(PK, VK, C_{tbe}, t, S')$. Thus, $\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE1},n,k}(\Lambda) = 0$. □

3.3 A Construction TTBE2 of Threshold Tag-Based Encryption Scheme Based on the DLIN Assumption

Our second construction TTBE2 of threshold tag-based encryption scheme naturally expands the Kiltz's tag-based encryption scheme [6] to the threshold setting. A secret key x_1, x_2 of the Kiltz's scheme is distributed among n secret key shares $(f_1(1), f_2(1)), \cdots, (f_1(n), f_2(n))$ with polynomials f_1, f_2 satisfying $x_1 = f_1(0), x_2 = f_2(0)$. Decryption shares are of the form $(C_1^{f_1(i)}, C_2^{f_2(i)})$.

More precisely TTBE2 is described in Figure 3.

Theorem 3. *Under the DLIN assumption for G_{DLIN}, the threshold tag-based encryption scheme* TTBE2 *is stag-CCA-secure.*

More precisely, for an arbitrary adversary $\mathcal{A}$ against stag-chosen-ciphertext security of TTBE2 *that runs in time at most τ and makes at most Q decryption*

Setup(n, k, Λ):
$(p, g, \mathbb{G}, \mathbb{G}_1, e) \leftarrow G_{DLIN}(\Lambda)$; $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$;
$f_1, f_2 \xleftarrow{\$} \mathbb{Z}_p[X]$ satisfying $\deg(f_1) = \deg(f_2) = k-1$ and $f_1(0) = x_1, f_2(0) = x_2$;
$z \leftarrow g_1^{x_1},\ g_2 \leftarrow z^{\frac{1}{x_2}},\ y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p,\ u_1 \leftarrow g_1^{y_1}, u_2 \leftarrow g_2^{y_2}$;
$PK = (p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, z, u_1, u_2),\ SK = ((f_1(1), f_2(1)), \cdots, (f_1(n), f_2(n)))$;
$VK = ((v_{11} = g_1^{f_1(1)}, v_{12} = g_2^{f_2(1)}), \cdots, (v_{n1} = g_1^{f_1(n)}, v_{n2} = g_2^{f_2(n)}))$;
return (PK, VK, SK).

Encrypt(PK, t, M):
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p,\ C_1 \leftarrow g_1^{r_1},\ C_2 \leftarrow g_2^{r_2},\ D_1 \leftarrow (z^t u_1)^{r_1},\ D_2 \leftarrow (z^t u_2)^{r_2},\ E \leftarrow M z^{r_1+r_2}$;
return $C_{tbe} = (C_1, C_2, D_1, D_2, E)$.

ShareDec$(PK, i, SK_i = (f_1(i), f_2(i)), C_{tbe} = (C_1, C_2, D_1, D_2, \cdot), t)$:
If $e(C_1, z^t u_1) \neq e(D_1, g_1)$ or $e(C_2, z^t u_2) \neq e(D_2, g_2)$ then return $\mu_i = (i, \perp)$,
else return $\mu_i = (i, (C_1^{f_1(i)}, C_2^{f_2(i)}))$.

ShareVf$(PK, VK = ((v_{i1}, v_{i2})), C_{tbe} = (C_1, C_2, \cdot, \cdot, \cdot), t, \mu_i = (i, (C_{i1}, C_{i2})))$:
If $e(C_{i1}, g_1) \neq e(C_1, v_{i1})$ or $e(C_{i2}, g_2) \neq e(C_2, v_{i2})$ then return invalid, else return valid.

Combine$(PK, VK, C_{tbe} = (\cdot, \cdot, \cdot, \cdot, E), t, \{\mu_1 = (1, (C_{11}, C_{12})), \cdots, \mu_k = (k, (C_{k1}, C_{k2}))\})$:
If $\exists i$, **ShareVf**$(PK, VK, C_{tbe}, t, \mu_i) =$ invalid then return $\perp$,
else return $E / \prod_{i=1}^{k} (C_{i1} C_{i2})^{\lambda_i}$ using Lagrange coefficients $\lambda_1, \cdots, \lambda_k$
satisfying $f_1(0) = \sum_{i=1}^{k} \lambda_i f_1(i)$.

Fig. 3. Threshold Tag-Based Encryption Scheme TTBE2

queries, there exists an algorithm $\mathcal{B}$ for the DLIN problem on G_{DLIN} that runs in time at most τ plus the time to perform $O(Q+n)$ exponentiations and $O(Q)$ pairing computations and satisfies

$$\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE2},n,k}(\Lambda) = \mathsf{Adv}^{dbdh}_{\mathcal{B},G_{DLIN}}(\Lambda).$$

For an arbitrary adversary $\mathcal{A}'$ against stag decryption consistency of TTBE2*, it holds that*

$$\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE2},n,k}(\Lambda) = 0.$$

Proof. First, we consider stag chosen ciphertext security of TTBE2. Let $\mathcal{A}$ be an arbitrary adversary that runs in time at most τ, makes at most Q decryption queries, and has advantage $\mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE2},n,k}(\Lambda)$ in attacking TTBE2 in the game of stag-chosen-ciphertext security. Using the adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the DLIN problem of $G_{DLIN}(\Lambda)$.

Given $(\Lambda, p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, z, g_1^{r_1}, g_2^{r_2}, W)$ as input, algorithm $\mathcal{B}$ proceeds as follows. (The aim of $\mathcal{B}$ is to distinguish two cases between $W = z^{r_1+r_2}$ or random.)

1. Initialization. Algorithm $\mathcal{B}$ invokes adversary $\mathcal{A}$ on input (n, k, Λ). Adversary $\mathcal{A}$ outputs a target tag t^* and a list $S = \{s_1, \cdots, s_{k-1}\} (\subset \{1, \cdots, n\})$ of the $k-1$ servers that it wishes to corrupt.
2. Setup. Then, $\mathcal{B}$ does the following:
   (a) $\mathcal{B}$ picks random integers $c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $u_1 = z^{-t^*} g_1^{c_1},\ u_2 = z^{-t^*} g_2^{c_2}$. (This defines implicitly as $y_1 = -t^* x_1 + c_1,\ y_2 = -t^* x_2 + c_2$.) $\mathcal{B}$ sets $PK = (p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, z, u_1, u_2)$.

(b) Next, $\mathcal{B}$ picks $2k-2$ random integers $\alpha_1, \cdots, \alpha_{k-1}, \beta_1, \cdots, \beta_{k-1} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. (We let $f_1, f_2 \in \mathbb{Z}_p[X]$ be two polynomials of degree $k-1$ defined by $f_1(0) = x_1, f_1(s_i) = \alpha_i$ $(i = 1, \cdots, k-1)$ and $f_2(0) = x_2, f_2(s_i) = \beta_i$ $(i = 1, \cdots, k-1)$. $\mathcal{B}$ does not know f_1, f_2.) $\mathcal{B}$ sets $SK|_S = (SK_{s_1} = (\alpha_1, \beta_1), \cdots, SK_{s_{k-1}} = (\alpha_{k-1}, \beta_{k-1}))$.

(c) For $i \in S$, $\mathcal{B}$ lets $VK_i = (v_{i1}, v_{i2}) = (g_1^{\alpha_i}, g_2^{\beta_i})$. For $i \notin S$, it computes $v_{i1} = z^{\lambda_0}(g_1^{\alpha_1})^{\lambda_1} \cdots (g_1^{\alpha_{k-1}})^{\lambda_{k-1}}$ and $v_{i2} = z^{\lambda_0}(g_2^{\beta_1})^{\lambda_1} \cdots (g_2^{\beta_{k-1}})^{\lambda_{k-1}}$, where $\lambda_0, \cdots, \lambda_{k-1} (\in \mathbb{Z}_p)$ are the Lagrange coefficients satisfying $f(i) = \lambda_0 f(0) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$ for degree k polynomials f. (As easily seen, (v_{i1}, v_{i2}) satisfies $v_{i1} = g_1^{f_1(i)}, v_{i2} = g_2^{f_2(i)}$.) $\mathcal{B}$ sets $VK = (VK_1 = (v_{11}, v_{12}), \cdots, VK_n = (v_{n1}, v_{n2}))$.

(d) $\mathcal{B}$ gives PK, VK and $SK|_S$ to $\mathcal{A}$.

3. Phase 1. $\mathcal{A}$ issues decryption share queries $((C_{tbe}, t), i)$ under the constraint that $t \neq t^*$ and $i \notin S$. First, $\mathcal{B}$ validates $e(C_1, z^t u_1) \stackrel{?}{=} e(D_1, g_1)$ and $e(C_2, z^t u_2) \stackrel{?}{=} e(D_2, g_2)$ to clarify validity of the ciphertext $C_{tbe} = (C_1, C_2, D_1, D_2, E)$. If validity test fails, $\mathcal{B}$ gives to $\mathcal{A}$ $(i, \perp)$. Otherwise, $\mathcal{B}$ computes the Lagrange coefficients $\lambda_1, \cdots, \lambda_{k-1}, \lambda_i \in \mathbb{Z}_p$ satisfying $f(0) = \lambda_i f(i) + \sum_{j=1}^{k-1} \lambda_j f(s_j)$ for degree k polynomials f and sets

$$C_{i1} = \left\{ \frac{(\frac{D_1}{C_1^{c_1}})^{\frac{1}{t-t^*}}}{C_1^{\sum_{j=1}^{k-1} \lambda_j \alpha_j}} \right\}^{\frac{1}{\lambda_i}}, \quad C_{i2} = \left\{ \frac{(\frac{D_2}{C_2^{c_2}})^{\frac{1}{t-t^*}}}{C_2^{\sum_{j=1}^{k-1} \lambda_j \beta_j}} \right\}^{\frac{1}{\lambda_i}}$$

and then gives to $\mathcal{A}$ $(i, (C_{i1}, C_{i2}))$ as decryption share. (If the ciphertext is valid, it must be the case that $D_1 = C_1^{(t-t^*)x_1 + c_1} = (C_1^{x_1})^{t-t^*} C_1^{c_1}$. Then, since $t \neq t^*$, we get $C_1^{x_1} = (D_1 / C_1^{c_1})^{\frac{1}{t-t^*}}$. Substituting $f_1(0) = \sum_{j=1}^{k-1} \lambda_j f_1(s_j) + \lambda_i f_1(i)$ for x_1, and noting $C_{1i} = C_1^{f_1(i)}$, we obtain the above expression of C_{i1}. Similar for C_{i2}.)

4. Challenge. $\mathcal{A}$ outputs two same-length messages M_0 and M_1. $\mathcal{B}$ flips a fair coin $b \in \{0, 1\}$, and responds with the challenge ciphertext $C^*_{tbe} = (g_1^{r_1}, g_2^{r_2}, (g_1^{r_1})^{c_1}, (g_2^{r_2})^{c_2}, M_b W)$. (As $\mathcal{B}$ sets $u_1 = z^{-t^*} g_1^{c_1}$ and $u_2 = z^{-t^*} g_2^{c_2}$, it holds that $(g_1^{r_1})^{c_1} = (u_1 z^{t^*})^{r_1}, (g_2^{r_2})^{c_2} = (u_2 z^{t^*})^{r_2}$. Moreover, if $W = z^{r_1 + r_2}$, then we have $M_b W = M_b z^{r_1 + r_2}$ and C^*_{tbe} is a valid ciphertext of M_b under PK with tag t^*.)

5. Phase 2. $\mathcal{A}$ issues additional queries as in Phase 1, to which $\mathcal{B}$ responds as before.

6. Guess. Eventually, $\mathcal{A}$ outputs a guess b'. $\mathcal{B}$ outputs 1 if $b = b'$, or outputs 0 otherwise.

This completes the description of algorithm $\mathcal{B}$, that runs in time at most τ plus the time to perform $O(Q + n)$ exponentiations and $O(Q)$ pairing computations. By the comments in the description, it is immediate that $\mathcal{B}$ perfectly simulates a stag-CCA game for $\mathcal{A}$ if $W = z^{r_1 + r_2}$. When W is a random element, the view

of $\mathcal{B}$ is independent of the choice b. So, $\mathsf{Adv}^{dlin}_{\mathcal{B},G_{DLIN}}(\Lambda) = |\Pr[b = b'] - 1/2| = |(1/2 + \mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE2},n,k}(\Lambda)) - 1/2| = \mathsf{Adv}^{stag-cca}_{\mathcal{A},\mathsf{TTBE2},n,k}(\Lambda)$.

Second, we consider stag decryption consistency of TTBE2. Let $\mathcal{A}'$ be an arbitrary adversary with advantage $\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE2},n,k}(\Lambda)$ in attacking TTBE2 in the game of stag decryption consistency. Suppose adversary $\mathcal{A}'$ outputs $t,\ C_{tbe},\ S = (\mu_1 = (1,(C_{11},C_{12})),\cdots,\mu_k = (k,(C_{k1},C_{k2}))),\ S' = (\mu'_1 = (1,(C'_{11},C'_{12})),\cdots,\mu'_k = (k,(C'_{k1},C'_{k2})))$. If those shares μ_i in S are valid, they must satisfy $e(C_{i1},g_1) = e(C_1,g_1^{f_1(i)}),\ e(C_{i2},g_2) = e(C_2,g_2^{f_2(i)})$, so $C_{i1} = C_1^{f_1(i)},\ C_{i2} = C_2^{f_2(i)}$. Then, it holds that $\prod_{i=1}^k C_{i1}^{\lambda_i} = C_1^{\sum_{i=1}^k \lambda_i f_1(i)} = C_1^{x_1}$ and $\prod_{i=1}^k C_{i2}^{\lambda_i} = C_2^{\sum_{i=1}^k \lambda_i f_2(i)} = C_2^{x_2}$. Similarly, if the shares μ'_i in S' are valid, we have $\prod_{i=1}^k C'^{\lambda_i}_{i1} = C_1^{x_1}$ and $\prod_{i=1}^k C'^{\lambda_i}_{i2} = C_2^{x_2}$. This means $\mathsf{Combine}(PK,VK,C_{tbe},t,S) = \mathsf{Combine}(PK,VK,C_{tbe},t,S')$. Thus, $\mathsf{Adv}^{stag-dc}_{\mathcal{A}',\mathsf{TTBE2},n,k}(\Lambda) = 0$. □

4 Construction of Threshold Public Key Encryption Schemes

By applying the conversion TT2TP in Section 2 to the two threshold tag-based encryption schemes TTBE1, TTBE2 in Section 3, we obtain two threshold public key encryption schemes TPKE1, TPKE2 that are both CCA-secure by Theorem 1, Theorem 2 and Theorem 3:

Theorem 4. *Let* S *be a strong one-time signature.*

- *Under the DBDH assumption,* TPKE1 $= TT2TP(\mathsf{TTBE1},\mathsf{S})$ *is a CCA-secure threshold public key encryption scheme.*
- *Under the DLIN assumption,* TPKE2 $= TT2TP(\mathsf{TTBE2},\mathsf{S})$ *is a CCA-secure threshold public key encryption scheme.*

Our threshold public key encryption schemes TPKE1 and TPKE2 are more simple than the construction BBH given by Boneh, Boyen and Halevi [2]. It is because our constructions are based on the tag-based schemes which is more simple than the ID-scheme used by BBH.

Instantly, Table 1 shows a comparison of efficiency among TPKE1, TPKE2 and BBH. The table shows the number of pairings, multi-exponentiations and regular-exponentiations required in operations of those schemes. (The entries of

Table 1. Efficiency Comparison among TPKE1, TPKE2 and BBH

Scheme	Assumption	Encrypt ShareDec ShareVf Combine ♯pairings + [♯multi−exp., ♯reg−exp.]	reduction
BBH	DBDH	$1+[1,3]$ $2+[1,2]$ $2+[0,1]$ $2+[2,0]$	tight
TPKE1	DBDH	$1+[1,3]$ $2+[0,2]$ $2+[0,0]$ $1+[1,0]$	tight
TPKE2	DLIN	$0+[2,3]$ $4+[0,3]$ $4+[0,0]$ $0+[1,0]$	tight

Combine do not include costs for ShareVf.) As shown, TPKE1 is more efficient than BBH. TPKE1 requires less number of exponentiation both in ShareDec, ShareVf and Combine than BBH. On the other hand, TPKE2 has an advantage that it is most efficient both in Encrypt and Combine because it requires no computation of bilinear map in those operations.

5 Conclusion

In this paper, we proposed a notion of threshold tag-based encryption schemes and showed a conversion from any stag-CCA-secure threshold tag-based encryption schemes to CCA-secure threshold public-key encryption schemes. We gave two constructions of threshold tag-based encryption schemes and obtained two constructions of threshold public-key encryption schemes, using that conversion. Those schemes are non-interactive, robust, proved secure without random oracle model and are more efficient than the construction by Boneh, Boyen and Halevi [2].

References

1. Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
2. Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)
5. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
6. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
7. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairings. In: Proceedings of the Symposium on Cryptography and Information Security, SCIS 2000, Japan (2000)
8. Shamir, A.: How to share a secret. Communications of the ACM, 612–613 (1979)
9. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)