Mediated Traceable Anonymous Encryption

Malika Izabachène[1], David Pointcheval[2], and Damien Vergnaud[2]

[1] UVSQ, France
[2] ENS/CNRS/INRIA, Paris, France

Abstract. The notion of *key privacy* for asymmetric encryption schemes was formally defined by Bellare, Boldyreva, Desai and Pointcheval in 2001: it states that an eavesdropper in possession of a ciphertext is not able to tell which specific key, out of a set of known public keys, is the one under which the ciphertext was created. Since anonymity can be misused by dishonest users, some situations could require a tracing authority capable of revoking key privacy when illegal behavior is detected. Prior works on *traceable anonymous encryption* miss a critical point: an encryption scheme may produce a covert channel which malicious users can use to communicate illegally using ciphertexts that trace back to nobody or, even worse, to some honest user.

In this paper, we examine subliminal channels in the context of traceable anonymous encryption and we introduce a new primitive termed *mediated traceable anonymous encryption* that provides confidentiality and anonymity while preventing malicious users to embed subliminal messages in ciphertexts. In our model, all ciphertexts pass through a *mediator* (or possibly several successive mediators) and our goal is to design protocols where the absence of covert channels is guaranteed as long as the mediator is honest, while semantic security and key privacy hold even if the mediator is dishonest.

We give security definitions for this new primitive and constructions meeting the formalized requirements. Our generic construction is fairly efficient, with ciphertexts that have logarithmic size in the number of group members, while preventing collusions. The security analysis requires classical complexity assumptions in the standard model.

1 Introduction

Motivation. The notion of *key privacy* for (asymmetric) encryption schemes was formally defined by Bellare, Boldyreva, Desai and Pointcheval [2]. The motivation for this security notion is anonymous communication, where eavesdroppers are prevented from learning the identities of the communicating parties, and namely the target recipient of a ciphertext. Since people are more and more concerned about all their actions being linkable to each other (or even worse, to their identity), key privacy is obviously a very attractive notion from the user's point of view. However, some organizations and governments are concerned about how anonymity can be abused by criminals, and the key privacy property can potentially be dangerous against public safety. Therefore anonymity revocation should be available when illegal behavior is detected, as provided by group signatures [9].

This motivates the notion of *traceable anonymous encryption* in which an adversary cannot determine which user's public key has been used to generate the ciphertext that it sees while a trusted third party (given some trapdoor information) is able to revoke anonymity and thus to trace back to the intended recipient. Some work dealt with such a property, but to the best of our knowledge, a critical point was missed: an encryption scheme may contain a steganographic channel (or a covert channel) which malicious users can use to communicate illegally using ciphertexts that trace back to nobody, or even worse to some honest user. More precisely, still using the official scheme, it may be possible to encrypt a message to a user A (the official target recipient for the tracing authority) so that the randomness is used for transmitting some extra information to a user B, that would not be traced as a possible recipient.

For instance, in 2007, Kiayias, Tsiounis and Yung [17] presented *group encryption*, a cryptographic primitive which can be seen as the encryption analogue of a *group signature* [9]. It provides semantic security, anonymity and a way for the *group manager* to revoke anonymity of ciphertexts. However, it makes use of zero-knowledge proofs to determine whether a ciphertext is valid or not. As a consequence, an invalid ciphertext can be used to transmit some information. Above all, subliminal channels (available in the randomness) can be exploited to send some information in addition to a clean message, or even to frame an honest user.

Let us consider the following example: in 2000, Sako [20] proposed a novel approach to achieve bid secrecy in auction protocols. Her technique consists in expressing each bid as an encryption of a *known* message, with a key corresponding to the value of the bid. Therefore, what needs to be hidden in the ciphertext is not the message, but the key itself; the use of traceable anonymous encryption (e.g. group encryption) seems very promising for such applications (the bid itself being identified using the tracing procedure). However, one major concern in auction protocols is the problem of collusion between bidders and it is highly desirable to prevent bidders from engaging in such collaborative bidding strategies. Unfortunately, no known construction of traceable anonymous encryption is free of covert channels and the main purpose of the present paper is precisely to propose a new cryptographic primitive addressing this issue. Recently, Abdalla, Bellare and Neven [1] introduced the concept of *robust encryption* in which it is hard to produce a ciphertext that is valid for two different users. They showed that if the anonymous encryption scheme used in Sako's protocol is not robust then the auction protocol does not achieve *fairness* (*i.e.* a cheating bidder can place a bid that he can open later to an arbitrary value). The primitive we propose in this paper permits to achieve simultaneously fairness and collusion-freeness in Sako's protocol.

Contributions. We introduce a new primitive which we call *mediated traceable anonymous encryption* and that provides confidentiality and anonymity while preventing malicious users to embed subliminal messages in ciphertexts, which would allow to transmit information to a recipient that would remain perfectly anonymous, or even worse to frame an honest user.

In order to provide semantic security, asymmetric encryption has to be probabilistic, and randomness can be used as a covert channel. We will thus have to avoid the users to control the randomness used in the ciphertext. We introduce a *mediator* that is not provided with any secret information, but whose role —similar to the warden model introduced by Simmons [21]— is to add more randomness to each ciphertext so that any hidden message is smothered. It is worth noting that —contrary to the group encryption primitive— the mediator only checks the syntactic validity of the ciphertext but it does not verify the soundness of any complex proof. We can even iterate the re-randomization process by several *mediators* to be sure that any ciphertext has been re-randomized at least once.

Another point for traceability, it is impossible to get it without assuming that users (recipients) are registered in the system, since registered users only can be traced. As for (dynamic) group signatures [3], we also introduce a trusted party termed the *issuer* that registers new users. One may wonder whether the issuer has to be trusted or not with respect to the semantic security property, since he might know all the decryption keys of the system: one does not really care about that since one can use its own additional encryption scheme to encrypt the plaintext before re-encrypting using our traceable encryption primitive.

It is relatively easy to design an anonymous encryption scheme that provides traceability or the absence of steganographic channel, but the task is more challenging if we want to achieve both simultaneously. Indeed, the existence of the tracing procedure implies that a ciphertext contains (at least implicitly) some information about the recipient, but this value can be used to transmit one bit of covert information. For instance, a Boneh-Franklin [6] identity-based variant of the ElGamal universal re-encryption scheme [15] cannot be used to achieve covert-free traceable anonymous encryption, since it turns out that the re-randomization preserves some predicates that can be used to transfer some information.

In this paper we propose a formal definition of *Mediated Traceable Anonymous Encryption Scheme* (𝓜TAES) and a security model capturing the (seemingly contradictory) following security notions:

1. *subliminal channel freeness:* after re-randomization, no information at all can be extracted from a ciphertext that does not trace back to a user registered to the issuer. We thus assume that the communication network guarantees that any ciphertext will go through a mediator. In practice, we can assume that in a specific network, all the routers implement the mediator re-randomization, which enforces the use of our scheme by all the users, and which guarantees that all the ciphertexts are re-randomized at least once. Then, packets that do not follow the encryption rules are made random;
2. the tracing trapdoor information does not allow the *opener* to violate the semantic security of ciphertexts nor to issue valid decryption keys (*i.e.* to play the role of the issuer);

3. the ciphertext remains confidential and anonymous even with respect to the mediator (*i.e.* we only rely on the mediator to guarantee the absence of covert channels in ciphertexts, by properly re-randomizing the ciphertexts), or any collusion of users and the mediator. As already said, multiple re-randomizations are even possible by various independent mediators.

We propose efficient constructions of $\mathcal{M}$TAES in the standard model, which security relies on DDH-like assumptions. The first one does not split the roles of the issuer and the opener and can be applied in any group where the DDH assumption holds while the second scheme makes use of (Type-2 or Type-3 [13]) pairing-friendly groups, with asymmetric pairing where the XDH and the (asymmetric[1]) DBDH assumptions hold, to separate the two authority roles (and then achieve the second above property). The two first schemes lead to ciphertexts that are linear in the number of registered users. This is not very practical, but they are fully-collusion secure. Using public collusion-secure codes (e.g. IPP codes [16]), better efficiency can be achieved: we provide a generic construction, with (almost) logarithmic ciphertexts.

2 Security Model

2.1 Syntactic Definitions

In this section, we give the formal definitions of our new concept: the *Mediated Traceable Anonymous Encryption*, which involves two trusted authorities, an *issuer* for adding new members to the system, and an *opener* for revoking anonymity; and a non-trusted authority, the *mediator* that systematically re-randomizes all its inputs, without any private information. There even can be several independent mediators. Mediators will be assumed to be honest (but possibly curious, and thus definitely not trusted): for the subliminal-channel freeness security notion we will define later, it is clear that in case of collusion with the last mediator, this strong security level cannot be achieved.

Definition 1 (Mediated Traceable Anonymous Encryption Schemes). A $\mathcal{M}$TAES is a tuple of efficient algorithms or protocols (GSetup, Join, Encrypt, ReRand, Decrypt, Trace, Judge) such that:

- GSetup(λ) → (mpk, msk, sk$_O$): this is a randomized algorithm run by a trusted party that, on input of a security parameter λ, produces three keys consisting of a group public key mpk, a manager's secret key msk and an opening key sk$_O$; It also generates a data structure $\mathcal{L}$, called a *registration list* which is initially empty.
- Join(id, mpk, msk) → (pk$_{id}$, sk$_{id}$) takes a bit-string identity id, the group public key mpk and the manager's secret key msk as inputs. It outputs a pair of member keys (pk$_{id}$, sk$_{id}$) associated to id, and updates the registration list $\mathcal{L}$ with the pair (id, pk$_{id}$).
- Encrypt(mpk, pk$_{id}$, m) → C takes as input the group public key mpk, a user public key pk$_{id}$ and a message m, and outputs a *pre-ciphertext* C.
- ReRand(mpk, C) → C' takes as input the group public key mpk and a *pre-ciphertext* C, and outputs a randomized ciphertext C'. Note that it may be applied again on a re-randomized ciphertext.
- Decrypt(mpk, sk$_{id}$, C) → m takes as input the group public key mpk, a member secret key sk$_{id}$, as well as a *ciphertext* C, then it outputs a message, or ⊥ in case of invalid ciphertext.
- Trace(mpk, sk$_O$, $\mathcal{L}$, C) → (id, Π): This is a deterministic algorithm that on input the group public key mpk, the opening key sk$_O$, the registration list $\mathcal{L}$ and a ciphertext C, outputs a user identity id (equivalently, the public key pk) and a proof Π for the judge, otherwise ⊥ in case of failure.
- Judge(mpk, $\mathcal{L}$, C, id, Π): This is a deterministic algorithm that on input the group public key mpk, the registration list $\mathcal{L}$, a ciphertext in C, a user identity id and a proof Π checks whether Π indeed proves that id is the target recipient of the ciphertext C (or one of them, in case of collusion).

Random tapes have been omitted in the notations, for the sake of clarity, but in some cases, they may be explicit: Join(id, mpk, msk; r), Encrypt(mpk, pk$_{id}$, m; r) or ReRand(mpk, C; r).

---

[1] The asymmetric DBDH assumption [11] reduces to the classical DBDH assumption [6] in the case of Type-3 bilinear structures [13].

2.2 Security Notions

In any group protocol, where each user owns a private key, collusions have to be dealt with: a collusion of users may help them to manufacture a new pair $(\mathsf{pk}, \mathsf{sk})$ associated to no user, or to an honest user. The latter case should be unlikely. The tracing algorithm should thus (in case the ciphertext "contains" some information) either output the identity of the target receiver if this is a well-formed ciphertext to this user, or the identity of one of the colluders who helped to manufacture the target public key. If the pre-ciphertext does not target any public key (nobody can be traced) then we want the re-randomization to cancel any information. Collusion will be modeled by corrupt queries, that will provide the secret keys of the corrupted users to the adversary. Then, from all these keys, the adversary will be allowed to do anything it wants to transfer some information, in an untraceable way.

As a consequence, in the security model we provide the adversary with two oracles: a restricted Join oracle (denoted Join^*) that just outputs the public keys, and a $\mathsf{Corrupt}$ oracle that outputs the user's secret key. Corrupted users are then registered in the *corruption list* $\mathcal{C}$, initially empty:

- $\mathsf{Join}^*(\mathsf{id}) \to \mathsf{pk}_{\mathsf{id}}$, takes as input an identity, and then runs $\mathsf{Join}(\mathsf{id}, \mathsf{mpk}, \mathsf{msk})$ to get $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$, but outputs $\mathsf{pk}_{\mathsf{id}}$ only. Note that the registration list $\mathcal{L}$ has been updated by the Join procedure;
- $\mathsf{Corrupt}(\mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$ takes as input a registered identity, and outputs the secret key corresponding to $\mathsf{pk}_{\mathsf{id}}$. It also updates the corruption list $\mathcal{C}$ with $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$.

We then denote by $\mathcal{LU}$ the list of registered identities, and by $\mathcal{CU}$ the list of the corrupted users, whereas $\mathcal{L}$ and $\mathcal{C}$ also contain keys. We additionally define a predicate $\mathsf{Traceable}(\mathcal{U}, C)$, where $\mathcal{U}$ is a list of identities and C a pre-ciphertext, that tests whether the tracing algorithm, run on the global parameters and a re-randomization of C, outputs an identity $\mathsf{id} \in \mathcal{U}$ with a convincing proof Π.

Since we are dealing with anonymity and traceability, we should address full-anonymity and chosen-ciphertext security, providing access to the opening oracle and to the decryption oracle respectively. But due to the inherent malleability of such a re-randomizable encryption scheme, some constraints are required. This is discussed in the Appendix A. In this section, we focus on the basic anonymity (without opening oracle access) and the chosen-plaintext security (without decryption oracle access).

In order to avoid the opener to frame an honest user, as the recipient of a ciphertext that is not aimed to him, we can rely on an additional Public Key Infrastructure (PKI), in which each user owns a pair of public/private key, and signs $\mathsf{pk}_{\mathsf{id}}$ when it receives the associated secret key $\mathsf{sk}_{\mathsf{id}}$. This signature can be added to the registration list $\mathcal{L}$. We thus trust the link between an identity id and the associated public key. And we will use both in an indifferentiable way.

SEMANTIC SECURITY. The main security notion for an encryption scheme is of course the semantic security of the ciphertext (after re-randomization), that is formalized by the indistinguishability of the two experiments (for $b = 0, 1$) presented Fig. 1(a). As usual, the adversary has to guess which plaintext has been encrypted in the challenge ciphertext. Note that we provide the opening key sk_O to the adversary, which models a collusion with the opener. We restrict the adversary to use a valid (registered) public key, since otherwise no one can decrypt the ciphertext. We are thus only interested in the privacy of real ciphertexts. We define the advantage of $\mathcal{A}$ in breaking the semantic security by its advantage:

$$\mathsf{Adv}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{rind}}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{rind}-1}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{rind}-0}(\lambda) = 1].$$

We define a weaker version, where the opening key is not provided to the adversary (which is useful when we cannot separate the issuing and opening/tracing roles, as in our first simple scheme below):

$$\mathsf{Adv}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{weak-rind}}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{weak-rind}-1}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{M}\text{TAES}, \mathcal{A}}^{\mathsf{weak-rind}-0}(\lambda) = 1].$$

As usual, we denote by $\mathsf{Adv}_{\mathcal{M}\text{TAES}}(\lambda, \tau)$, for any security notion, the best advantage any adversary can get within time τ. Furthermore, we may add an extra parameter ω, when we limit the number of $\mathsf{Corrupt}$-queries. This will be useful for schemes that are not fully-collusion secure (see Section 4), but ω-resilient: the security holds if the number of $\mathsf{Corrupt}$-queries is less than ω.

Experiment $\mathsf{Exp}^{\mathsf{rind}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk},\mathsf{sk}_O)$
      $\rightarrow (\mathsf{pk}, m_0, m_1, s)$
  $C \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk},m_b)$
  $C^* \leftarrow \mathsf{ReRand}(\mathsf{mpk},C)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{pk} \notin \mathcal{LU}\backslash\mathcal{CU}$ RETURN 0
  ELSE RETURN b'

Experiment $\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk},\mathsf{sk}_O)$
      $\rightarrow (\mathsf{pk}, m_0, m_1, s)$
  $C^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk},m_b)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{pk} \notin \mathcal{LU}\backslash\mathcal{CU}$ RETURN 0
  ELSE RETURN b'

(a) Semantic Security

Experiment $\mathsf{Exp}^{\mathsf{ranon}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (\mathsf{pk}_0, \mathsf{pk}_1, m, s)$
  $C \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk}_b,m)$
  $C^* \leftarrow \mathsf{ReRand}(\mathsf{mpk},C)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{pk}_0 \notin \mathcal{LU}\backslash\mathcal{CU}$
    OR $\mathsf{pk}_1 \notin \mathcal{LU}\backslash\mathcal{CU}$ RETURN 0
  ELSE RETURN b'

Experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (\mathsf{pk}_0, \mathsf{pk}_1, m, s)$
  $C^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk}_b,m)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{pk}_0 \notin \mathcal{LU}\backslash\mathcal{CU}$
    OR $\mathsf{pk}_1 \notin \mathcal{LU}\backslash\mathcal{CU}$ RETURN 0
  ELSE RETURN b'

(b) Anonymity

Experiment $\mathsf{Exp}^{\mathsf{correct}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (\mathsf{pk}, m, r, r')$
  $C \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk},m;r)$
  $C^* \leftarrow \mathsf{ReRand}(\mathsf{mpk},C;r')$
  IF $\mathsf{pk} \notin \mathcal{LU}$ RETURN 0
  IF $\mathsf{pk} \neq \mathsf{Trace}(\mathsf{mpk},\mathsf{sk}_O,\mathcal{L},C^*)$
    RETURN 1
  IF $\mathsf{Decrypt}(\mathsf{mpk},\mathsf{sk},C^*) \neq m$
    (where sk associated to pk) RETURN 1
  ELSE RETURN 0

(c) Correctness

Experiment $\mathsf{Exp}^{\mathsf{subF}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (C_0, C_1, s)$
  $C^* \leftarrow \mathsf{ReRand}(\mathsf{mpk},C_b)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{Traceable}(\mathcal{CU},C_0)$
    AND $\mathsf{Traceable}(\mathcal{CU},C_1)$ RETURN 0
  ELSE RETURN b'

(d) Subliminal-Channel Freeness

Experiment $\mathsf{Exp}^{\mathsf{priv}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (\mathsf{pk}_0, \mathsf{pk}_1, m_0, m_1, s)$
  $C^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk},\mathsf{pk}_d,m_d)$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{pk}_0 \notin \mathcal{LU}\backslash\mathcal{CU}$
    OR $\mathsf{pk}_1 \notin \mathcal{LU}\backslash\mathcal{CU}$ RETURN 0
  ELSE RETURN b'

(e) Privacy

Experiment $\mathsf{Exp}^{\mathsf{unlink}-b}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$
  $(\mathsf{mpk},\mathsf{msk},\mathsf{sk}_O) \leftarrow \mathsf{GSetup}(\lambda)$
  $\mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{mpk})$
      $\rightarrow (C, s)$
  $C'_0 \leftarrow \mathsf{ReRand}(\mathsf{mpk},C)$
  $C'_1 \xleftarrow{R} \mathcal{C}, \; C^* \leftarrow C'_b$
  $b' \leftarrow \mathcal{A}^{\mathsf{Join}^*(\cdot),\mathsf{Corrupt}(\cdot)}(s,C^*)$
  IF $\mathsf{Traceable}(\mathcal{CU},C'_0)$ RETURN 0
  ELSE RETURN b'

(f) Ciphertext Unlinkability

Fig. 1. Experiments for Security Notions

The above security notion is about the re-randomized ciphertext, and thus for an adversary that does not have access to the pre-ciphertext before re-randomization. We thus define the same security notion about the pre-ciphertext characterized by the advantages $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{weak-ind}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)$. It is clear that the latter security notions (before re-randomization) are stronger than the former, and address the case where the mediator is honest but curious:

Theorem 2. *For any scheme* $\mathcal{M}\mathrm{TAES}$ *and any time bound* τ,

$$\mathsf{Adv}^{\mathsf{rind}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau) \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau) \qquad \mathsf{Adv}^{\mathsf{weak-rind}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau) \leq \mathsf{Adv}^{\mathsf{weak-ind}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau)$$

ANONYMITY. Our goal in this work is anonymity of the recipient (a.k.a. key privacy [2]), we formalize it as usual by the indistinguishability of the two experiments presented Fig. 1(b): the adversary has to guess which public key has been used to generate the challenge ciphertext. Again, we restrict the adversary to use valid (registered) public keys. We define the advantage of $\mathcal{A}$ in breaking the anonymity (after re-randomization or before) by:

$$\mathsf{Adv}^{(\mathsf{r})\mathsf{anon}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{(\mathsf{r})\mathsf{anon}-1}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{(\mathsf{r})\mathsf{anon}-0}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1].$$

Theorem 3. *For any scheme* $\mathcal{M}\mathrm{TAES}$ *and any time bound* τ, $\mathsf{Adv}^{\mathsf{ranon}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau) \leq \mathsf{Adv}^{\mathsf{anon}}_{\mathcal{M}\mathrm{TAES}}(\lambda,\tau)$.

CORRECTNESS. Of course, an encryption scheme that is non-decryptable could be secure (semantically secure and anonymous). We thus need the scheme to be both decryptable and traceable: a well-formed ciphertext should be decryptable by the target recipient (using sk associated to the target pk, both generated by the Join oracle), and should be traced (with high probability) to this recipient by the opener. No adversary should be able to win the correctness game (see Fig. 1(c)) with significant advantage:

$$\mathsf{Adv}^{\mathsf{correct}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{correct}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1].$$

SUBLIMINAL-CHANNEL FREENESS. Let us remind that our ultimate goal is that either the ciphertext can be traced to a corrupted user (under the control of the adversary), or the adversary cannot transfer any information. This is modeled by the *subliminal-channel freeness property* (see Fig. 1(d)): the adversary generates two pre-ciphertexts C_0 and C_1, with which it tries to transmit some information. If they are well-formed, and really trace to corrupted users, then this is normal that the information is transferred and so the adversary does not win (hence the two tests with the predicate Traceable). The challenger provides a re-randomized version of one of them to the adversary, and the latter has to guess which one: it has no bias unless some information leaks in the re-randomized ciphertext. As a consequence, subliminal-channel freeness is quantified by

$$\mathsf{Adv}^{\mathsf{subF}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{subF}-1}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{subF}-0}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1].$$

We note that for this security notion, the adversary generates the ciphertext itself, and is thus allowed to generate ill-formed ciphertexts, contrarily to the previous security notions. In such a case, only, it wins if it manages to transmit some information. We now provide two additional one that will imply the above ones.

PRIVACY. This privacy notion (see Fig. 1(e)) encompasses both semantic security (plaintext-privacy) and anonymity (key-privacy) before re-randomization, and thus even with respect to the mediator:

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{priv}-1}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{priv}-0}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1].$$

CIPHERTEXT-UNLINKABILITY. Then, we want that either the pre-ciphertext sent to the mediator has a well identified target recipient, or the re-randomization cancel any information: unless the ciphertext traces back to a user controlled by the adversary, or the re-randomized ciphertext is unlinkable to the input pre-ciphertext, and thus indistinguishable with a truly random ciphertext (in the Real-or-Random sense) – (see Fig. 1(f)):

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{unlink}-1}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{unlink}-0}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) = 1].$$

2.3 Relations between the Security Notions

In this section, we state that the later security notions imply the former, and the proofs can be found in the Appendix A.

Theorem 4. *For any scheme $\mathcal{M}$TAES and any time bound τ,*

$$\mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{weak-ind}}(\lambda, \tau) \leq \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{priv}}(\lambda, \tau) \qquad \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{anon}}(\lambda, \tau) \leq \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{priv}}(\lambda, \tau)$$
$$\mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{priv}}(\lambda, \tau) \leq \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{weak-ind}}(\lambda, \tau) + \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{anon}}(\lambda, \tau).$$

Theorem 5. *For any scheme $\mathcal{M}$TAES and any time bound τ,*

$$\mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{subF}}(\lambda, \tau) \leq \frac{1}{2} \times \mathsf{Adv}_{\mathcal{M}\text{TAES}}^{\mathsf{unlink}}(\lambda, \tau).$$

3 Our scheme

First, we present a simple scheme achieving the above security requirements. This scheme inherits the properties of some combination of broadcast encryption and re-randomizable techniques. It fulfills the strongest properties of privacy and unlinkabillity under the sole **DDH** assumption, but with the restriction that the same trapdoor is used for both decrypting and tracing ciphertexts. We then show how we can separate these capabilities, under the **XDH** [5] and the (asymmetric) **DBDH** [11] assumptions. These classical assumptions are reviewed in the Appendix B. We here use classical advantage notations for all the decisional problems.

3.1 Description of the Scheme $\mathcal{M}$TAES$_1$

– **GSetup**(λ): The GSetup algorithm takes as input a security parameter λ. It defines a cyclic group $\mathbb{G}$ of prime order q, in which the **DDH** assumption holds, in some basis g. We denote $\mathbb{G}^* = \mathbb{G}\backslash\{1\}$, the subset of elements of order exactly q. It chooses random scalars $x_1, \ldots, x_t \in \mathbb{Z}_q^*$ with t the maximum number of registered users. It sets the master secret key and the opening key as $\mathsf{msk} = \mathsf{sk}_O = (x_1, x_2, \ldots, x_t)$ and the group public key $\mathsf{mpk} = (y_1 = g^{x_1}, \ldots, y_t = g^{x_t})$. It also sets the registration list $\mathcal{L}$ to empty.
– **Join**$(\mathsf{id}, \mathsf{mpk}, \mathsf{msk})$: The id is assumed to be an integer between 1 and t. This algorithm thus outputs $\mathsf{sk}_{\mathsf{id}} = x_{\mathsf{id}}$ as the secret key of user id, whereas the public key is y_{id}. It adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}} = y_{\mathsf{id}})$ in the list $\mathcal{L}$.
– **Encrypt**(mpk, i, m): To encrypt a message $m \in \mathbb{G}$ under a public key $\mathsf{pk}_i = y_i$, we choose two random scalars $r, s \in \mathbb{Z}_q^*$ and compute $(A_1, A_2, B_1, B_2) = (g^r, m \cdot y_i^r, g^s, y_i^s)$.
– **ReRand**(mpk, C): To re-randomize a pre-ciphertext in $(\mathbb{G}^*)^4$, we first choose $4t$ random scalars r_j, r_j' and s_j, s_j' in $\mathbb{Z}_q^*$, for $j = 1, \cdots, t$ and compute t sub-ciphertexts as follows:

$$C_{1,j} \leftarrow A_1 \cdot B_1^{r_j'} \cdot g^{r_j} \qquad\qquad C_{2,j} \leftarrow A_2 \cdot B_2^{r_j'} \cdot y_j^{r_j}$$
$$D_{1,j} \leftarrow B_1^{s_j'} \cdot g^{s_j} \qquad\qquad D_{2,j} \leftarrow B_2^{s_j'} \cdot y_j^{s_j}.$$

Then, we obtain a sequence C of t tuples $(C_{1,j}, C_{2,j}, D_{1,j}, D_{2,j})$ for $j = 1, \ldots, t$. Note that the second halves of the tuples allow to re-randomize again the ciphertext, with $4t$ random scalars a_j, a_j' and b_j, b_j':

$$C_{1,j}' \leftarrow C_{1,j} \cdot D_{1,j}^{a_j'} \cdot g^{a_j} \qquad\qquad C_{2,j}' \leftarrow C_{2,j} \cdot D_{2,j}^{a_j'} \cdot y_j^{a_j}$$
$$D_{1,j}' \leftarrow D_{1,j}^{b_j'} \cdot g^{b_j} \qquad\qquad D_{2,j}' \leftarrow D_{2,j}^{b_j'} \cdot y_j^{b_j}.$$

– **Decrypt**(mpk, i, C): To decrypt a ciphertext for user i, using the secret key x_i, we compute $m \leftarrow C_{2,i} \cdot C_{1,i}^{-x_i}$. Note that user i can check whether he really is the target recipient: $D_{2,i} \stackrel{?}{=} D_{1,i}^{x_i}$.

– Trace($\mathsf{msk}, \mathcal{L}, \mathsf{sk}_O, C$): To trace the recipient of a ciphertext C, parse C as $(C_{1,i}, C_{2,i}, D_{1,i}, D_{2,i})_{i=1,\ldots,t}$ and check whether $g, y_i, D_{1,i}, D_{2,i}$ is a DDH-tuple for one of the index i (which would correspond to a registered user in the list $\mathcal{L}$). If such an index is found, we provide a non-interactive zero-knowledge proof of validity Π showing the existence of $x_i \in \mathbb{Z}_q$ such that $y_i = g^{x_i}$ and $D_{2,i} = D_{1,i}^{x_i}$; otherwise we output an error symbol $\perp$. Note that the same can be done on (g, y_i, B_1, B_2) from the pre-ciphertext.

– Judge($\mathsf{mpk}, \mathcal{L}, C, \mathsf{id}, \Pi$): This algorithm checks whether the proof Π is valid *wrt* $\mathcal{L}, C, \mathsf{id}$.

3.2 Security Analysis

CORRECTNESS. First, one should note that after re-randomization, a ciphertext for $\mathsf{pk}_i = y_i$ looks like

$$C_{1,j} \leftarrow A_1 \cdot B_1^{r'_j} \cdot g^{r_j} = g^{r+s\cdot r'_j + r_j} \qquad C_{2,j} \leftarrow A_2 \cdot B_2^{r'_j} \cdot y_j^{r_j} = m \cdot y_i^{r+s\cdot r'_j} \cdot y_j^{r_j}$$

$$D_{1,j} \leftarrow B_1^{s'_j} \cdot g^{s_j} = g^{s\cdot s'_j + s_j} \qquad D_{2,j} \leftarrow B_2^{s'_j} \cdot y_j^{s_j} = y_i^{s\cdot s'_j} \cdot y_j^{s_j},$$

where r, s are the random values chosen by the sender, whereas r'_j, s'_j and the r_j, s_j are chosen by the re-randomizer for $j = 1, \ldots, t$. If $y_i \in \mathcal{L}$, and thus $y_i = g^{x_i}$, then

$$C_{1,i} = g^{r+s\cdot r'_i + r_i} \qquad C_{2,i} = m \cdot y_i^{r+s\cdot r'_i + r_i} = m \cdot C_{1,i}^{x_i}$$

$$D_{1,i} = g^{s\cdot s'_i + s_i} \qquad D_{2,i} = y_i^{s\cdot s'_i + s_i} = D_{1,i}^{x_i}.$$

Using the secret key $\mathsf{sk}_i = x_i$, we immediately get m by computing $C_{2,i} \cdot C_{1,i}^{-x_i}$. A similar computation on $D_{1,i}$ and $D_{2,i}$ traces back user i. For any index $j \neq i$ (or if $y_i = g^{x_i} \notin \mathcal{L}$), with $\delta_j = x_j - x_i \neq 0$,

$$C_{1,j} = g^{r+s\cdot r'_j + r_j} \qquad C_{2,j} = m \cdot g^{x_i(r+s\cdot r'_j) + x_j r_j} = m \cdot C_{1,j}^{x_j} \cdot g^{(r+s\cdot r'_j)\delta_j}$$

$$D_{1,j} = g^{s\cdot s'_j + s_j} \qquad D_{2,j} = g^{x_i \cdot s\cdot s'_j + x_j s_j} = D_{1,j}^{x_j} \cdot g^{\delta_j s s'_j}.$$

Then, the tuple $(C_{1,j}, C_{2,j}, D_{1,j}, D_{2,j})$ follows a distribution statistically close to random in $\mathbb{G}^4$, since s and s'_j are non-zero. It thus contains no information on m and i. We will formally justify that in the following.

PRIVACY. The privacy property implies both semantic security and anonymity of pre-ciphertexts. We obtain the following result, whose proof can be found in the Appendix C:

Theorem 6. *Our scheme* $\mathcal{M}\text{TAES}_1$ *proposed in Section 3.1 fulfills the privacy property under the* DDH *assumption:*

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}\text{TAES}_1}(\lambda, \tau) \leq 2t \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau + 2\tau_{\mathsf{exp}}),$$

where τ_{exp} *denotes an upper bound on the time computation for one exponentiation.*

UNLINKABILITY. In order to get the subliminal-channel freeness, we will prove the unlinkability, which holds under the DDH assumption too (the proof can be found in the Appendix C):

Theorem 7. *Our scheme* $\mathcal{M}\text{TAES}_1$ *proposed in Section 3.1 is unlinkable under the* DDH *assumption:*

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{M}\text{TAES}_1}(\lambda, \tau) \leq 2(t+1) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\tau + 2\tau_{\mathsf{exp}}),$$

where τ_{exp} *denotes an upper bound on the time computation for one exponentiation.*

3.3 Two-Level Scheme $\mathcal{M}\text{TAES}_2$

In the previous scheme $\mathcal{M}\text{TAES}_1$, the same key is used for both the join and the tracing procedures, hence it achieves the weak security level only. In this section, we separate these two capabilities in $\mathcal{M}\text{TAES}_2$, making use of a so-called Type-2 or Type-3 pairing-friendly structure [13]. It consists in a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$ where e is an admissible bilinear map [6], g_1, g_2 and $G = e(g_1, g_2)$ are generators of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively (and additionally there exists an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ in the case of Type-2 structure). We consider structures in which the XDH (*i.e.* the DDH assumption in $\mathbb{G}_1$) and the (asymmetric) DBDH[2] [11] assumptions can be made (*e.g.*, using Weil or Tate pairings on certain MNT curves as defined in [18], these assumptions seem reasonable).

Description.

- $\mathsf{GSetup}(\lambda)$: The GSetup algorithm takes as input a security parameter λ and generates parameters for a bilinear structure $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$ As before, the master secret key is set to be a sequence of t random scalars in $\mathbb{Z}_q^*$, $\mathsf{msk} = (x_1, x_2, \ldots, x_t)$, and the group public key is defined by

$$\mathsf{mpk} = \begin{pmatrix} g_1, g_2, G, (y_1 = g_1^{x_1}, \ldots, y_t = g_1^{x_t}), \\ (Y_1 = e(g_1^{x_1}, g_2), \ldots, Y_t = e(g_1^{x_t}, g_2)) \end{pmatrix}.$$

  The opening key consists of the sequence $\mathsf{sk}_O = (h_1 = g_2^{x_1}, \cdots, h_t = g_2^{x_t})$. This algorithm also sets the registration list $\mathcal{L}$ to empty.
- $\mathsf{Join}(\mathsf{id}, \mathsf{mpk}, \mathsf{msk})$: As before, this algorithm outputs x_{id} as the secret key $\mathsf{sk}_{\mathsf{id}}$ of user id, whereas the public key is $(y_{\mathsf{id}}, Y_{\mathsf{id}})$. It also adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}} = (y_{\mathsf{id}}, Y_{\mathsf{id}}))$ in the list $\mathcal{L}$. (Note that y_{id} would be enough in the public key, and in the master public key, but the use of Y_{id} will simplify the notations).
- $\mathsf{Encrypt}(\mathsf{mpk}, i, m)$: To encrypt a message $m \in \mathbb{G}_T$ under a public key $\mathsf{pk}_i = (y_i, Y_i)$, one first chooses two random scalars $r, s \in \mathbb{Z}_q^*$ and then computes $(A_1, A_2, B_1, B_2) = (G^r, m \cdot Y_i^r, g_1^s, y_i^s)$.
- $\mathsf{ReRand}(\mathsf{mpk}, C)$: To re-randomize a pre-ciphertext, one first chooses $4t$ random scalars r_j, r_j' and s_j, s_j' in $\mathbb{Z}_q^*$ for $j = 1, \cdots, t$ and computes t sub-ciphertexts as follows:

$$C_{1,j} \leftarrow A_1 \cdot e(B_1, g_2)^{r_j'} \cdot G^{r_j} \qquad\qquad C_{2,j} \leftarrow A_2 \cdot e(B_2, g_2)^{r_j'} \cdot Y_j^{r_j}$$
$$D_{1,j} \leftarrow B_1^{s_j'} \cdot g_1^{s_j} \qquad\qquad\qquad\qquad D_{2,j} \leftarrow B_2^{s_j'} \cdot y_j^{s_j}.$$

  Then, we obtain a sequence C of t tuples $(C_{1,j}, C_{2,j}, D_{1,j}, D_{2,j})$ for $j = 1, \ldots, t$. Note that the second halves of the tuples allow to re-randomize again the ciphertext, as in $\mathcal{M}\text{TAES}_1$.
- $\mathsf{Decrypt}(\mathsf{mpk}, i, C)$: To decrypt a ciphertext for user i, using the secret key x_i, we compute $m \leftarrow C_{2,i} \cdot C_{1,i}^{-x_i}$.
- $\mathsf{Trace}(\mathsf{msk}, \mathcal{L}, \mathsf{sk}_O, C)$: To trace the recipient of a given ciphertext C, check whether $(g_1, y_i, D_{1,i}, D_{2,i})$ is a DDH-tuple in $\mathbb{G}_1$ for some registered user i in the list $\mathcal{L}$, by testing whether $e(D_{2,i}, g_2) = e(D_{1,i}, h_i)$. If no such one is found, we output $\perp$ otherwise, we provide a non-interactive zero-knowledge proof of validity, Π showing the existence of $x_i \in \mathbb{Z}_q$ such that $h_i = g_2^{x_i}$ and $D_{2,i} = D_{1,i}^{x_i}$.
- $\mathsf{Judge}(\mathsf{mpk}, \mathcal{L}, C, \mathsf{id}, \Pi)$: This algorithm checks whether the proof Π is valid *wrt* $\mathcal{L}, C, \mathsf{id}$.

Security Properties. Granted the richer structure, and both the XDH and the DBDH assumptions, this scheme achieves all the expected security properties: anonymity, indistinguishability (even given access to the opening/tracing key) and unlinkability. The proof can be found in the Appendix D.

---

[2] Our scheme actually relies on a weaker *mixed* assumption which states that given g_1^a, g_2^a, $e(g_1, g_2)^b$ and $e(g_1, g_2)^c$, it is intractable to decide whether $c \stackrel{?}{=} ab \bmod q$.

Theorem 8. *Our scheme $\mathcal{M}$TAES$_2$ proposed in Section 3.3 is anonymous, indistinguishable and unlinkable under the* XDH *assumption (the* DDH *assumption in $\mathbb{G}_1$), and the* DBDH *assumption:*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{M}\mathrm{TAES}_2}(\lambda, \tau), \mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{M}\mathrm{TAES}_2}(\lambda, \tau) \leq 2t \cdot \mathsf{Adv}^{\mathsf{xdh}}_{\mathcal{BS}}(\tau + 2\tau_{\mathsf{exp}})$$

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\mathrm{TAES}_2}(\lambda, \tau) \leq t \cdot \mathsf{Adv}^{\mathsf{dbdh}}_{\mathcal{BS}}(\tau),$$

where τ_{exp} denotes an upper bound on the time computation for one exponentiation.

3.4 Protection Against the Issuer

As the Private Key Generator (PKG) in Identity Based-Encryption, the issuer can be involved in malicious activities since it knows all users' registered secret keys. We can easily prevent the issuer from breaking the semantic security of a ciphertext sent to a specific user by first encrypting the message with an appropriate encryption scheme (for which the issuer does not know the decryption keys) and then re-encrypt (component by component) the ciphertext with our *mediated traceable anonymous encryption scheme*: If the global ciphertext is not traceable, then the underlying ciphertext will be totally random, and thus, no information will be transmitted.

More precisely, for such a technique to be applied with our schemes, we need an underlying ElGamal [12] encryption scheme in $\mathbb{G}$ for $\mathcal{M}$TAES$_1$ (or $\mathbb{G} = \mathbb{G}_T$ for $\mathcal{M}$TAES$_2$). For encrypting a message $m \in \mathbb{G}$, one gets the recipient's ElGamal public key, and applies the ElGamal encryption of m: it gets $(c_1, c_2) \in \mathbb{G}^2$. Thereafter, c_1 and c_2 are independently re-encrypted under our $\mathcal{M}$TAES$_1$ scheme, using the recipient registered key. One could use another encryption scheme, different or stronger than ElGamal. The unique constraint is that the underlying encryption scheme should produce a ciphertext in $\mathbb{G}^k$, for some k, so that it can thereafter be re-encrypted by our $\mathcal{M}$TAES's.

4 An ω-Resilient Generic Construction

In this section, we propose a generic scheme that is ω-*resilient*, which means that the malicious adversary can corrupt up to a maximum of ω users adaptively and thus possess the ω corresponding private keys. However, it cannot obtain any information relevant to ciphertexts that are encrypted for public keys not within the corrupt users. For a fixed ω, our construction is fairly efficient, with ciphertexts that have logarithmic size in the number of group members (instead of linear with the two previous proposals $\mathcal{M}$TAES$_1$ and $\mathcal{M}$TAES$_2$). However, tracing might fail (with small probability, depending on the code). Our construction relies on well-known tools, namely *collusion-secure codes* and *homomorphic encryption schemes* that generalize the protocols from the previous section.

4.1 Collusion-secure codes

Our construction makes use of ω-*traceable* codes [22], in the same vein as the collusion-secure codes proposed by Boneh and Shaw [7] as a method of digital fingerprinting while preventing a collusion of a specified size ω from framing a user not in the coalition, but furthermore allowing the traceability of a traitor from a word generated by the coalition. We consider a code $\mathcal{C}$ of length ℓ on an alphabet T, with $\#T = t$ (*i.e.* $\mathcal{C} \subseteq T^\ell$) and we call it an (n, ℓ, t)-code if $\#\mathcal{C} = n$. The elements of $\mathcal{C}$ are called *codewords*.

For any subset of codewords $\mathcal{C}_0 \subset \mathcal{C}$, we define the set of descendants of $\mathcal{C}_0$ (*a.k.a.* the feasible set), denoted $\mathsf{Desc}(\mathcal{C}_0) = \{\boldsymbol{x} \in T^\ell : x_i \in \{a_i : \boldsymbol{a} \in \mathcal{C}_0\}, 1 \leq i \leq \ell\}$. We now recall the following definitions concerning non-frameability and traceability of codes. Let $\mathcal{C}$ be an (n, ℓ, t)-code and ω any integer such that $n > t \geq \omega \geq 1$.

Definition 9. $\mathcal{C}$ *is an ω-frameproof code if for any subset $\mathcal{C}_0 \subset \mathcal{C}$ such that $\#\mathcal{C}_0 \leq \omega$, $\mathsf{Desc}(\mathcal{C}_0) \cap \mathcal{C} = \mathcal{C}_0$.*

Optimal explicit constructions of ω-frameproof codes are known for small coalitions [4]. Wang and Xing [24] provided explicit constructions of ω-frameproof codes based on algebraic curves over finite fields; they obtain infinite classes of such (n, ℓ, t)-codes with $\ell = O(\log n)$ for fixed t and ω.

However, a frameproof-code just guarantees that no coalition (not too large) can produce a codeword of a user not in the coalition. But in the fingerprinting setting and for traitor tracing [10], we furthermore want a tracing algorithm $\mathsf{Trace}_{\mathcal{C}}$ which, on input a word $\boldsymbol{x}$ generated by the coalition, outputs a member of the coalition $\mathcal{C}_0$:

Definition 10. Let $\varepsilon > 0$. An $(n, \ell, \omega, t, \varepsilon)$-collusion-secure code is an (n, ℓ, t)-code for which there exists a (probabilistic) *tracing algorithm* $\mathsf{Trace}_{\mathcal{C}}$ satisfying the following condition: for any $\mathcal{C}_0 \subset \mathcal{C}$ such that $\#\mathcal{C}_0 \leq \omega$, and any $\boldsymbol{x} \in \mathsf{Desc}(\mathcal{C}_0)$, $\Pr[\mathsf{Trace}_{\mathcal{C}}(\boldsymbol{x}) \in \mathcal{C}_0] > 1 - \varepsilon$.

Such codes [23] with efficient tracing algorithms have been proposed with $\ell = O(\log n)$.

4.2 Homomorphic encryption

Homomorphic encryption is a form of encryption where one can perform a group operation on the plaintexts by performing a (possibly different) algebraic operation on the ciphertexts. More formally, a $\mathbb{H}$-homomorphic encryption scheme is a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}, \odot, \mathsf{Decrypt})$ such that $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is an encryption scheme with message space a group $\mathbb{H}$ and $\odot$ is an algorithm (written infix style) that takes two elements in $\mathsf{Encrypt}$'s codomain and outputs an element of $\mathsf{Encrypt}$'s codomain such that for all messages $m, m' \in \mathbb{H}$ and any matching key pair $(\mathsf{pk}, \mathsf{sk})$, we have:

$$\mathsf{Decrypt}(\mathsf{Encrypt}(m, \mathsf{pk}) \odot \mathsf{Encrypt}(m', \mathsf{pk}), \mathsf{sk}) = m \cdot m' \in \mathbb{H}. \tag{1}$$

Examples of homomorphic cryptosystems are due to ElGamal [12], Golwasser-Micali [14] and Paillier [19]. Note that (1) further implies the existence of an efficient function that allows exponentiation of ciphertexts (using a *square-and-multiply* algorithm): $\mathsf{Encrypt}(m, \mathsf{pk})^{\odot^r} = \mathsf{Encrypt}(m, \mathsf{pk}) \odot \cdots \odot \mathsf{Encrypt}(m, \mathsf{pk})$ (r times). Our construction relies on a pair of encryption schemes that are compatible for two algebraic operations $\odot$ and $\otimes$ in the following way:

Definition 11. Let $\mathbb{H}_1$ and $\mathbb{H}_2$ be two abelian groups of prime order q and let $\varphi : \mathbb{H}_2 \to \mathbb{H}_1$ be a group homomorphism. A $(\mathbb{H}_1, \mathbb{H}_2, \varphi)$-*compatible encryption scheme* is a tuple of (probabilistic) polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}_1, \mathsf{Encrypt}_2, \odot, \otimes, \mathsf{Decrypt}_1, \mathsf{Decrypt}_2)$ such that:

- $\mathsf{Setup}(\lambda) \to \mathsf{params}$: this algorithm is run by a (trusted) party that, on input of a security parameter λ, produces a set params of common public parameters.
- $\mathsf{Kg}(\mathsf{params}) \to (\mathsf{pk}, \mathsf{sk}^{(1)}, \mathsf{sk}^{(2)})$: on input of public parameters params, all parties use this randomized algorithm to generate a private/public key pair $(\mathsf{pk}, \mathsf{sk}^{(1)}, \mathsf{sk}^{(2)})$. We denote Kg_i for $i \in \{1, 2\}$ the algorithm that executes Kg but only returns $(\mathsf{pk}, \mathsf{sk}^{(i)})$.
- $(\mathsf{Setup}, \mathsf{Kg}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$ is an encryption scheme with message space $\mathbb{H}_1$.
- $(\mathsf{Setup}, \mathsf{Kg}_2, \mathsf{Encrypt}_2, \odot, \mathsf{Decrypt}_2)$ is a $\mathbb{H}_2$-homomorphic encryption scheme, but for which we ask for the decryption algorithm to output $\varphi(m) \in \mathbb{H}_1$ only (and not $m \in \mathbb{H}_2$ itself).
- $\otimes$ is an algorithm (written infix style) that on input an $\mathsf{Encrypt}_1$-ciphertext and an $\mathsf{Encrypt}_2$-ciphertext outputs an $\mathsf{Encrypt}_1$-ciphertext such that for all messages $m_1 \in \mathbb{H}_1$ and $m_2 \in \mathbb{H}_2$ and any matching key pair $(\mathsf{pk}, \mathsf{sk}^{(1)}, \mathsf{sk}^{(2)})$, we have: $\mathsf{Decrypt}_1(\mathsf{Encrypt}_1(m_1, \mathsf{pk}) \otimes \mathsf{Encrypt}_2(m_2, \mathsf{pk}), \mathsf{sk}^{(1)}) = m_1 \cdot \varphi(m_2) \in \mathbb{H}_1$.

Remark 12. Any $\mathbb{G}$-homomorphic encryption gives rise to a $(\mathbb{H}_1 = \mathbb{G}, \mathbb{H}_2 = \mathbb{G}, \mathsf{id})$-compatible encryption scheme in the trivial way (*i.e.* with $\otimes = \odot$). In a bilinear structure $\mathcal{BS} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$, a non-straightforward $(\mathbb{H}_1 = \mathbb{G}_T, \mathbb{H}_2 = \mathbb{G}_1, \varphi : y \mapsto e(y, g_2))$-compatible encryption scheme is the one we implicitly use in the construction of the scheme $\mathcal{M}\textsc{taes}_2$ (see the Appendix D):

- Setup generates the parameters for an appropriate bilinear structure $\mathcal{BS} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$ where g_1, g_2 and $G = e(g_1, g_2)$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively;

- Kg picks at random a scalar $x \in \mathbb{Z}_q^*$ and outputs a triple $(\mathsf{pk}, \mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}) = (y = g_1^x, x, h = g_2^x)$;
- $(\mathsf{Setup}, \mathsf{Kg}_2, \mathsf{Encrypt}_2, \odot, \mathsf{Decrypt}_2)$ is the ElGamal encryption scheme in the group $\mathbb{G}_1$: it performs the encryption of $m \in \mathbb{G}_1$ as $C = (c_1, c_2) = (m \cdot \mathsf{pk}^r, g_1^r) \in \mathbb{G}_1^2$, for a random $r \xleftarrow{R} \mathbb{Z}_q^*$, and the knowledge of $\mathsf{sk}^{(2)}$ allows to recover $\varphi(m) \in \mathbb{G}_T$ from C as: $\varphi(m) = e(c_1, g_2)/e\left(c_2, \mathsf{sk}^{(2)}\right)$. The operation $\odot$ is the component-wise product in $\mathbb{G}_1^2$.
- $(\mathsf{Setup}, \mathsf{Kg}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$ is the ElGamal encryption scheme in the group $\mathbb{G}_T$ with public key $Y = e(\mathsf{pk}, g_2) = \varphi(\mathsf{pk})$;
- The operation $\otimes : \mathbb{G}_T^2 \times \mathbb{G}_1^2 \longrightarrow \mathbb{G}_T^2$ is: $(Y_1, Y_2) \otimes (z_1, z_2) = (Y_1 \cdot e(z_1, g_2), Y_2 \cdot e(z_2, g_2))$.

Notation. Let $\mathbb{H}$ be a finite group.

- We denote $\mathsf{Split}_{\mathbb{H}}$ the probabilistic algorithm that on input $m \in \mathbb{H}$ and $\ell \geq 1$, picks uniformly at random $m_1, \ldots, m_{\ell-1} \in \mathbb{H}$, sets $m_\ell = m/(m_1 \ldots m_{\ell-1})$ and outputs the vector $\boldsymbol{m} = (m_1, \ldots, m_\ell) = \mathsf{Split}_{\mathbb{H}}(m, \ell)$.
- For an encryption scheme $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with domain $\mathbb{H}$, we denote $\mathsf{Encrypt}^{(\ell)}$ the algorithm that takes as input $\boldsymbol{m} = (m_1, \ldots, m_\ell) \in \mathbb{H}^\ell$ and $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and returns the vector $\boldsymbol{c} = \mathsf{Encrypt}(\boldsymbol{m}, \mathsf{pk})$ defined as the coordinate-wise encryption of $\boldsymbol{m}$ under the public-key $\mathbf{pk}$. Similarly, we denote $\mathsf{Decrypt}^{(\ell)}$ the algorithm that given a vector of ciphertexts $\boldsymbol{c}$ and a vector of secret keys $\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$, parses $\boldsymbol{c}$ as $\boldsymbol{c} = (c_1, \ldots, c_\ell)$ (where each c_i is a ciphertext), outputs $\perp$ if $\mathsf{Decrypt}(c_i, \mathsf{sk}_i) = \perp$ for some $i \in \{1, \ldots, \ell\}$ and $\prod_{i=1}^\ell \mathsf{Decrypt}(c_i, \mathsf{sk}_i) \in \mathbb{H}$, otherwise.
  In particular, for any vector of matching key pairs $(\mathbf{pk}, \mathbf{sk})$, for any $m \in \mathbb{H}$, and any integer $\ell \geq 1$, we have:
  $$\mathsf{Decrypt}^{(\ell)}\left(\mathsf{Encrypt}^{(\ell)}(\mathsf{Split}_{\mathbb{H}}(m, \ell), \mathbf{pk}), \mathbf{sk}\right) = m.$$
- If $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a $\mathbb{H}$-homomorphic encryption scheme, we denote $\odot$ the coordinate-wise product defined on the codomain of $\mathsf{Encrypt}^{(\ell)}$.
  If $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}_1, \mathsf{Encrypt}_2, \odot, \otimes, \mathsf{Decrypt}_1, \mathsf{Decrypt}_2)$ is a $(\mathbb{H}_1, \mathbb{H}_2, \varphi)$-compatible encryption scheme, we also denote $\otimes$ the coordinate-wise operation defined on the cartesian product of the $\mathsf{Encrypt}_1^{(\ell)}$-ciphertexts and the $\mathsf{Encrypt}_2^{(\ell)}$-ciphertexts.

4.3 Description of the Scheme $\mathcal{M}\textsc{taes}_3^\ell$

Let $\ell \geq 1$ be an integer, let $\mathbb{H}_1$ and $\mathbb{H}_2$ be two abelian groups of prime order q and let the map $\varphi : \mathbb{H}_2 \to \mathbb{H}_1$ be a group homomorphism. Let the system $(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Encrypt}_1, \mathsf{Encrypt}_2, \odot, \otimes, \mathsf{Decrypt}_1, \mathsf{Decrypt}_2)$ be a $(\mathbb{H}_1, \mathbb{H}_2, \varphi)$-compatible encryption scheme and $\mathcal{C}$ a $(n, \ell, \omega, t, \varepsilon)$-collusion-secure code of length ℓ on the alphabet $T = \{1, \ldots, t\}$. The following construction describes the generic scheme $\mathcal{M}\textsc{taes}_3^\ell$:

- $\mathsf{GSetup}(\lambda)$: The GSetup algorithm takes as input a security parameter λ. It executes $\mathsf{Setup}(\lambda)$ and given the common public parameters params, it runs $t\ell$ times $\mathsf{Kg}(\mathsf{params})$ and gets $t\ell$ triples $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j}^{(1)}, \mathsf{sk}_{i,j}^{(2)})$ for $(i, j) \in \{1, \ldots, \ell\} \times \{1, \ldots, t\}$. It sets the group public key, the opening key and the master secret key:

$$\mathsf{mpk} = \begin{pmatrix} \mathsf{pk}_{1,1} & \mathsf{pk}_{1,2} & \ldots & \mathsf{pk}_{1,t} \\ \mathsf{pk}_{2,1} & \mathsf{pk}_{2,2} & \ldots & \mathsf{pk}_{2,t} \\ \vdots & \vdots & \vdots & \\ \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\ell,2} & \ldots & \mathsf{pk}_{\ell,t} \end{pmatrix}, \qquad \mathsf{sk}_O = \begin{pmatrix} \mathsf{sk}_{1,1}^{(2)} & \mathsf{sk}_{1,2}^{(2)} & \ldots & \mathsf{sk}_{1,t}^{(2)} \\ \mathsf{sk}_{2,1}^{(2)} & \mathsf{sk}_{2,2}^{(2)} & \ldots & \mathsf{sk}_{2,t}^{(2)} \\ \vdots & \vdots & \vdots & \\ \mathsf{sk}_{\ell,1}^{(2)} & \mathsf{sk}_{\ell,2}^{(2)} & \ldots & \mathsf{sk}_{\ell,t}^{(2)} \end{pmatrix}$$

$$\mathsf{msk} = \begin{pmatrix} \mathsf{sk}_{1,1}^{(1)} & \mathsf{sk}_{1,2}^{(1)} & \ldots & \mathsf{sk}_{1,t}^{(1)} \\ \mathsf{sk}_{2,1}^{(1)} & \mathsf{sk}_{2,2}^{(1)} & \ldots & \mathsf{sk}_{2,t}^{(1)} \\ \vdots & \vdots & \vdots & \\ \mathsf{sk}_{\ell,1}^{(1)} & \mathsf{sk}_{\ell,2}^{(1)} & \ldots & \mathsf{sk}_{\ell,t}^{(1)} \end{pmatrix}$$

It also outputs the registration list $\mathcal{L}$ that is initially empty.

- Join($\mathsf{id}, \mathsf{mpk}, \mathsf{msk}$): This algorithm encodes the id into the public key, $\mathsf{pk}_{\mathsf{id}} = \boldsymbol{c} = (c_1 c_2 \ldots c_\ell) \in \mathcal{C}$, where for each i, $c_i \in \{1, \ldots, t\}$, and outputs $\mathsf{sk}_{\boldsymbol{c}}^{(1)} = (\mathsf{sk}_{1,c_1}^{(1)}, \ldots, \mathsf{sk}_{\ell,c_\ell}^{(1)})$ the secret key of user id. It furthermore adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ in the list $\mathcal{L}$.
- Encrypt($\mathsf{mpk}, \boldsymbol{c}, m$): To encrypt a message $m \in \mathbb{H}_1$ under a public key $\mathsf{pk}_{\mathsf{id}} = \boldsymbol{c}$, the algorithm computes $\boldsymbol{A} = \mathsf{Encrypt}_1^{(\ell)}(\mathsf{Split}_{\mathbb{H}_1}(m, \ell), \mathbf{pk}_{\boldsymbol{c}})$ and $\boldsymbol{B} = \mathsf{Encrypt}_2^{(\ell)}((1_{\mathbb{H}_2}, \ldots, 1_{\mathbb{H}_2}), \mathbf{pk}_{\boldsymbol{c}})$ where $\mathbf{pk}_{\boldsymbol{c}} = (\mathsf{pk}_{1,c_1}, \ldots, \mathsf{pk}_{\ell,c_\ell})$ and outputs $C = (\boldsymbol{A}, \boldsymbol{B})$.
- ReRand(mpk, C): To re-randomize a pre-ciphertext $C = (\boldsymbol{A}, \boldsymbol{B})$, the algorithm picks at random t vectors $\boldsymbol{r_j}, \boldsymbol{s_j} \in (\mathbb{Z}_q^*)^\ell$ for $j \in \{1, \ldots, t\}$, generates $\boldsymbol{u} = \mathsf{Split}_{\mathbb{H}_2}(1, \ell)$ and $\boldsymbol{v} = \mathsf{Split}_{\mathbb{H}_2}(1, \ell)$ and computes t vectors of ciphertexts as follows:

$$\boldsymbol{C}_j \leftarrow \boldsymbol{A} \otimes (\boldsymbol{B}^{\odot \boldsymbol{r_j}} \odot \mathsf{Encrypt}_2^{(\ell)}(\boldsymbol{u}, \mathbf{pk}_{\mathbb{j}})) \quad \boldsymbol{D}_j \leftarrow \boldsymbol{B}^{\odot \boldsymbol{s_j}} \odot \mathsf{Encrypt}_2^{(\ell)}(\boldsymbol{v}, \mathbf{pk}_{\mathbb{j}}),$$

  for $j \in \{1, \ldots, t\}$ and $\mathbf{pk}_{\mathbb{j}} = (\mathsf{pk}_{1,j}, \ldots, \mathsf{pk}_{\ell,j})$. It outputs the $2t\ell$ vector $C' = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_t)$.
- Decrypt(mpk, i, C): To decrypt a ciphertext $C = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_t)$ for the public key $\mathbf{pk}_{\boldsymbol{c}}$, using the secret key $\mathbf{sk}_{\boldsymbol{c}}^{(1)}$, the algorithm computes $\mathsf{Decrypt}_1^{(\ell)}((C_{1,c_1}, \ldots, C_{\ell,c_\ell}), \mathbf{sk}_{\boldsymbol{c}}^{(1)})$.
- Trace($\mathsf{msk}, \mathcal{L}, \mathsf{sk}_O, C$): To trace a user *wrt* a given ciphertext C, parse C as $(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_t)$ and decrypts $\boldsymbol{D}_{\boldsymbol{c}} = (D_{1,c_1}, \ldots, D_{\ell,c_\ell})$ with the decryption algorithm $\mathsf{Decrypt}_2^{(\ell)}$ for all the secret keys $\mathbf{sk}_{\boldsymbol{c}}^{(2)}$ (where $\boldsymbol{c}$ draws all the domain $\{1, \ldots, t\}^\ell$). If for one of the vector $\boldsymbol{c}$, the decryption leads to $\varphi(1_{\mathbb{H}_2}) = 1_{\mathbb{H}_1}$, then the algorithm executes $\mathsf{Trace}_{\mathcal{C}}(\boldsymbol{c})$ and if it returns a codeword $\boldsymbol{v}$, it provides a non-interactive zero-knowledge proof of validity Π showing that $\mathsf{Decrypt}_2^{(\ell)}(\boldsymbol{D}_{\boldsymbol{c}}, \mathbf{sk}_{\boldsymbol{c}}^{(2)}) = 1_{\mathbb{H}_1}$ and that $\mathsf{Trace}_{\mathcal{C}}(\boldsymbol{c}) = \boldsymbol{v}$. Otherwise, it outputs an error symbol $\perp$.
- Judge($\mathsf{mpk}, \mathcal{L}, C, \mathsf{id}, \Pi$): This algorithm checks whether the proof Π is valid *wrt* $\mathcal{L}, C, \mathsf{id}$.

Remark 13. If we instantiate the scheme $\mathcal{M}\mathrm{TAES}_3^1$ with the ElGamal homomorphic encryption scheme in a group $\mathbb{G}$ (*resp.* with the $(\mathbb{G}_T, \mathbb{G}_1, \varphi : y \mapsto e(y, g_2))$-compatible encryption scheme in a bilinear structure $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$, described in Remark 12) and the code $\mathcal{C} = \{1, \ldots, t\} = T$, we obtain the scheme $\mathcal{M}\mathrm{TAES}_1$ (*resp.* the scheme $\mathcal{M}\mathrm{TAES}_2$) described in the Section 3.

4.4 Security Analysis

Let us explain why this scheme works, and which security level it provides.

Correctness. Since $\boldsymbol{B} = \mathsf{Encrypt}_2^{(\ell)}((1_{\mathbb{H}_2}, \ldots, 1_{\mathbb{H}_2}), \mathbf{pk}_{\boldsymbol{c}})$, for any random $\boldsymbol{r} \in (\mathbb{Z}_q^*)^\ell$, $\mathsf{Decrypt}_1^{(\ell)}(\boldsymbol{B}^{\odot \boldsymbol{r}}, \mathbf{pk}_{\boldsymbol{c}}) = 1_{\mathbb{H}_2}$ and therefore

$$\mathsf{Decrypt}_1^{(\ell)}(\boldsymbol{B}^{\odot \boldsymbol{r}} \odot \mathsf{Encrypt}^{(\ell)}(\boldsymbol{u}, \mathbf{pk}_{\boldsymbol{c}}), \mathbf{sk}_{\boldsymbol{c}}) = 1_{\mathbb{H}_2}.$$

The extracted ciphertext $(C_{1,c_1}, \ldots, C_{\ell,c_\ell})$ is $\boldsymbol{A} \otimes (\boldsymbol{B}^{\odot \boldsymbol{r}} \odot \mathsf{Encrypt}^{(\ell)}(\boldsymbol{u}, \mathbf{pk}_{\boldsymbol{c}}))$ with $\boldsymbol{r} = (r_{c_1}, \ldots, r_{c_\ell})$ and thus $\mathsf{Decrypt}_1^{(\ell)}\left((C_{1,c_1}, \ldots, C_{\ell,c_\ell}), \mathbf{sk}_{\boldsymbol{c}}^{(1)}\right) = m \in \mathbb{H}_1$. Note that because of the code, tracing may fail with some small probability ε.

Semantic Security and Privacy. We now prove that if the basic construction $\mathcal{M}\mathrm{TAES}_3^1$ is semantically secure (*resp.* anonymous or private) then for any integer $\ell \geq 1$ and any ω-collusion secure code $\mathcal{C}$, the scheme $\mathcal{M}\mathrm{TAES}_3^\ell$ is ω-resilient semantically secure (*resp.* ω-resilient anonymous or ω-resilient private).

Theorem 14. *Let $\ell \geq 1$ be an integer and let $\mathcal{C}$ be an $(n, \ell, \omega, t, \varepsilon)$-secure code. Our scheme $\mathcal{M}\mathrm{TAES}_3^\ell$ fulfills the ω-resilient (strong) semantic security property if and only if our scheme $\mathcal{M}\mathrm{TAES}_3^1$ fulfills the (strong) semantic security property:*

$$\mathsf{Adv}_{\mathcal{M}\mathrm{TAES}_3^\ell}^{\mathsf{ind}}(\lambda, \tau, \omega) \leq n \times \mathsf{Adv}_{\mathcal{M}\mathrm{TAES}_3^1}^{\mathsf{ind}}(\lambda, \tau + t\ell \tau_{\mathsf{Kg}}),$$

where τ_{Kg} denotes an upper bound on the time computation for execution of the Kg algorithm.

Proof. Let us consider an adversary $\mathcal{A}$ against the ω-resilient (strong) semantic security of $\mathcal{M}\textsc{taes}_3^\ell$. We construct an adversary $\mathcal{B}$ against the (strong) semantic security of $\mathcal{M}\textsc{taes}_3^1$.

KEY GENERATION. The adversary $\mathcal{B}$ simulates the GSetup algorithm by using params, the common parameters received from its own challenger as the common parameters given to $\mathcal{A}$. In addition, $\mathcal{B}$ receives from its challenger a list of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ as well as the second-level matching secret keys $(\mathsf{sk}_1^{(2)}, \ldots, \mathsf{sk}_\ell^{(2)})$ for the scheme $\mathcal{M}\textsc{taes}_3^1$. The algorithm then uniformly picks at random a vector $(a_1, \ldots, a_\ell) \in \mathcal{C} \subset \{1, \ldots, t\}^\ell$ and sets $(\mathsf{pk}_{i,a_i}, \mathsf{sk}_{i,a_i}^{(1)}, \mathsf{sk}_{i,a_i}^{(2)}) \leftarrow (\mathsf{pk}_i, \bot, \mathsf{sk}_i^{(2)})$ for $i \in \{1, \ldots, \ell\}$. It then executes $(t-1)\ell$ times the key generation algorithm $\mathsf{Kg}(\lambda)$ and gets $(t-1)\ell$ triples $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j}^{(1)}, \mathsf{sk}_{i,j}^{(2)})$, for $(i,j) \in \{1, \ldots, \ell\} \times \{1, \ldots, t\}$ but $j \neq a_i$. It sets the group public key, the opening key and the master secret key as in the real scheme. The algorithm $\mathcal{B}$ runs $\mathcal{A}$ on input params, mpk and sk_O.

Join* AND Corrupt QUERIES. When $\mathcal{A}$ asks a Join* query for user j, $\mathcal{B}$ encodes the id into the public key, $\mathsf{pk}_{\mathsf{id}} = \boldsymbol{c} = (c_1 c_2 \ldots c_\ell) \in \mathcal{C}$, where $c_i \in \{1, \ldots, t\}$. It furthermore adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ in the list $\mathcal{L}$.

When $\mathcal{A}$ asks a Corrupt query for user j with identity $\mathsf{id} = \boldsymbol{c}$, $\mathcal{B}$ outputs $(\mathsf{sk}_1^{(1)}, \ldots, \mathsf{sk}_\ell^{(1)})$ possibly by asking to its own Corrupt oracle the keys $\mathsf{sk}_{i,a_i}^{(1)}$ it does not know. It furthermore adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ in the list $\mathcal{C}$. If the adversary $\mathcal{A}$ wants to corrupt player $(a_1, \ldots, a_\ell)$, then $\mathcal{B}$ aborts. Otherwise, thanks to the ω-collusion security of the code $\mathcal{C}$, we know the ω corruptions cannot involve all the coordinates of the challenge code $(a_1, \ldots, a_\ell)$, otherwise the latter would be in the feasible set, which would contradict the frameproof-property: $\mathcal{B}$ will query its Corrupt oracle at most $\ell - 1$ times. Then, pk_I has not been corrupted, for some index I.

CHALLENGE CIPHERTEXT. Eventually, $\mathcal{A}$ outputs a public key $\mathsf{pk}_{\mathsf{id}}$ and two messages m_0 and m_1 (and possibly some state information s). With probability greater than $1/n$ (which automatically includes the fact that $(a_1, \ldots, a_\ell)$ has not been asked as a Corrupt-query), we have $\mathsf{id} = (a_1, \ldots, a_\ell)$. If this is not the case, $\mathcal{B}$ aborts. As said above, we know that $\mathsf{sk}_{I,a_I}^{(1)}$ has not been queried to $\mathcal{B}$'s Corrupt oracle. The algorithm $\mathcal{B}$ then computes $\boldsymbol{u} = (u_1, \ldots, u_\ell) = \mathsf{Split}_{\mathbb{H}_1}(1, \ell)$ and for $j \in \{1, \ldots, \ell\} \setminus \{I\}$, $c_j^\star = \mathsf{Encrypt}_1(u_j, \mathsf{pk}_{a_j})$. It then sends to its own challenger $(\mathsf{pk}_I = \mathsf{pk}_{a_I}, u_I \cdot m_0, u_I \cdot m_1)$ and receives the encryption $c_I^\star$ of $u_I m_b$ for a random $b \in \{0, 1\}$ under the public key pk_I. The algorithm $\mathcal{B}$ sends $\boldsymbol{c}^\star = (c_1^\star, \ldots, c_\ell^\star)$ to $\mathcal{A}$. This is a valid encryption of m_b: $\mathcal{A}$ outputs a bit $b^\star$ that $\mathcal{B}$ forwards to its own challenger.

CONCLUSION. Globally, the running time of $\mathcal{B}$ is the same as $\mathcal{A}$ (and the real challenger) plus the time to execute $(t-1)\ell$ times Kg and its success probability is identical to the one of $\mathcal{A}$ when the guess for $(a_1, \ldots, a_\ell)$ has been correct (which happens with probability greater than $1/n$). This concludes the proof of (strong) semantic security. $\square$

Theorem 15. *Let $\ell \geq 1$ be an integer and let $\mathcal{C}$ be an $(n, \ell, \omega, t, \varepsilon)$-secure code. Our scheme $\mathcal{M}\textsc{taes}_3^\ell$ fulfills the ω-resilient privacy property if and only if our scheme $\mathcal{M}\textsc{taes}_3^1$ fulfills the privacy property*

$$\mathsf{Adv}_{\mathcal{M}\textsc{taes}_3^\ell}^{\mathsf{priv}}(\lambda, t, \omega) \leq n \times \mathsf{Adv}_{\mathcal{M}\textsc{taes}_3^1}^{\mathsf{priv}}(\lambda, \tau + t\ell\tau_{\mathsf{Kg}}),$$

where τ_{Kg} denotes an upper bound on the time computation for execution of the Kg algorithm.

Proof (Sketch). The proof is very close to the previous one. In the *Challenge ciphertext* phase, the adversary $\mathcal{A}$ outputs a tuple $(\mathsf{pk}_0, \mathsf{pk}_1, m_0, m_1, s)$, and we know that the two (which are possibly the same) public keys are not corrupted. Again, still with probability greater than $1/n$, the above simulation worked and there is I such that pk_{a_I} has not been queried to $\mathcal{B}$'s Corrupt oracle. $\square$

This theorem implies the anonymity property.

Unlinkability. Finally, we also state that if the basic construction $\mathcal{M}\textsc{taes}_3^1$ is unlinkable then for any integer $\ell \geq 1$ and any ω-secure code $\mathcal{C}$ the scheme $\mathcal{M}\textsc{taes}_3^\ell$ is ω-resilient unlinkable. The proof can be found in the Appendix E.

Theorem 16. *Let $\ell \geq 1$ be an integer and let $\mathcal{C}$ be an $(n, \ell, \omega, t, \varepsilon)$-secure code. Our scheme $\mathcal{M}\text{TAES}_3^\ell$ is ω-resilient unlinkable if and only if our scheme $\mathcal{M}\text{TAES}_3^1$ is unlinkable.*

References

1. M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, Feb. 2010.
2. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, Dec. 2001.
3. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, Feb. 2005.
4. S. R. Blackburn. Perfect hash families: Probabilistic methods and explicit constructions. *J. Comb. Theory, Ser. A*, 92(1):54–60, 2000.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, Aug. 2004.
6. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Aug. 2001.
7. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465. Springer, Aug. 1995.
8. R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, Aug. 2003.
9. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, Apr. 1991.
10. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, Aug. 1994.
11. L. Ducas. Anonymity from asymmetry: New constructions for anonymous HIBE. In J. Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 148–164. Springer, Mar. 2010.
12. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, Aug. 1985.
13. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
14. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
15. P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178. Springer, Feb. 2004.
16. H. D. L. Hollmann, J. H. van Lint, J.-P. Linnartz, and L. M. G. M. Tolhuizen. On codes with identifiable parent property. *J. Comb. Theory, Ser. A*, 82:121–133, 1998.
17. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In K. Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 181–199. Springer, Dec. 2007.
18. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, 2001.
19. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999.
20. K. Sako. An auction protocol which hides bids of losers. In H. Imai and Y. Zheng, editors, *PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 422–432. Springer, Jan. 2000.
21. G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1984.
22. J. Staddon, D. R. Stinson, and R. Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Transactions on Information Theory*, 47(3):1042–1049, 2001.
23. T. van Trung and S. Martirosyan. New constructions for IPP codes. *Des. Codes Cryptography*, 35(2):227–239, 2005.
24. H. Wang and C. Xing. Explicit constructions of perfect hash families from algebraic curves over finite fields. *J. Comb. Theory, Ser. A*, 93(1), 2001.

A On Security Notions

A.1 Enhanced Security Notions

CHOSEN-CIPHERTEXT. To address chosen-ciphertext attacks, we have to provide the adversary with a decryption oracle. But due to the inherent malleability of such a re-randomizable encryption scheme, some constraints will be added, in the same vein as RCCA definition [8], with some forbidden requests (answered by Test):

- Decrypt$^*(\mathcal{P}, (\mathsf{id}, C)) \to \{m, \perp\}$, takes as input a list $\mathcal{P} = \{(\mathsf{id}_i, C_i)\}$ of critical pairs, a registered identity id and a ciphertext C, it runs $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, C)$ to get either m, or $\perp$ in case of invalid ciphertext. In the latter case, $\perp$ is returned. Otherwise, it furthermore runs $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}_i}, C_i)$ for all the pairs in $\mathcal{P}$ to get either m_i or $\perp$. If for some index i, $m_i = m$, then it outputs Test, otherwise m is the output.

The restrictions, without which attacks cannot be avoided, are the following ones:

- for the semantic security, we set $\mathcal{P} = \{(pk, C^*)\}$;
- for the anonymity, there is not restriction;
- for privacy, we set $\mathcal{P} = \{(pk_0, C^*), (pk_1, C^*)\}$;
- for both unlinkability and subliminal-channel freeness, we have $\mathcal{P} = \mathcal{LU} \times \{C^*\}$.

FULL ANONYMITY. To address full-anonymity, we have to provide the adversary with an opening oracle. But against, because of the inherent malleability, some constraints will be added, with some forbidden requests (answered by Test):

- Open$^*(\mathcal{P}, C) \to \{pk, \perp\}$, takes as input a list $\mathcal{P} = \{C_i\}$ of critical ciphertexts and a ciphertext C, it runs $\mathsf{Trace}(\mathsf{mpk}, \mathsf{sk}_O, \mathcal{L}, C)$ to get either pk, or $\perp$ in case of invalid ciphertext. In the latter case, $\perp$ is returned. Otherwise, it furthermore runs $\mathsf{Trace}(\mathsf{mpk}, \mathsf{sk}_O, \mathcal{L}, C_i)$ for all the ciphertexts in $\mathcal{P}$ to get either pk_i or $\perp$. If for some index i, $pk_i = pk$, then it outputs Test, otherwise pk is the output.

The restrictions, without which attacks cannot be avoided, are the following ones:

- for the semantic security, there is no restriction;
- for the anonymity, we set $\mathcal{P} = \{C^*\}$;
- for privacy, we set $\mathcal{P} = \{C^*\}$;
- for both unlinkability and subliminal-channel freeness, we have $\mathcal{P} = \{C^*\}$.

A.2 Proofs of the Relations between Security Notions

Proof (of Theorem 4). Let us first show that the privacy implies both weak semantic security (without the opening key) and anonymity:

- let $\mathcal{A}$ be an adversary with advantage ε against weak semantic security. We describe an adversary $\mathcal{B}$ against privacy. Upon receiving the master key mpk, $\mathcal{B}$ runs $\mathcal{A}$ and gets (pk, m_0, m_1, s). It then outputs (pk, pk, m_0, m_1, s) to the challenger. The latter chooses a random bit b, and encrypts m_b under pk (since the two public keys are the same). $\mathcal{B}$ then runs $\mathcal{A}$ on the challenge ciphertext, and forwards the answer b':

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}_{\mathrm{TAES}}, \mathcal{B}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{weak-ind}-1}_{\mathcal{M}_{\mathrm{TAES}}, \mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{weak-ind}-0}_{\mathcal{M}_{\mathrm{TAES}}, \mathcal{A}}(\lambda) = 1] = \varepsilon$$

Since $\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}_{\mathrm{TAES}}}(\lambda, \tau) \geq \mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}_{\mathrm{TAES}}, \mathcal{B}}(\lambda)$, and the above equality holds for any adversary $\mathcal{A}$ with running time bounded by τ, we get the expected result.

– let $\mathcal{A}$ be an adversary with advantage ε against anonymity. We describe an adversary $\mathcal{B}$ against privacy, the same way as above, but duplicating the plaintext instead of the public key.

For the other direction, we use the hybrid technique, for any adversary $\mathcal{A}$ against the privacy security notion, with running time bounded by τ. But we could extend this privacy game with two random bits b and c, where b specifies the message to be encrypted (as in the semantic security game) and c the target public key (as in the anonymity game):

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda) &= \Pr[d' = 1 | b = 1 \wedge c = 1] - \Pr[d' = 1 | b = 0 \wedge c = 0] \\
&= \Pr[d' = 1 | b = 1 \wedge c = 1] - \Pr[d' = 1 | b = 0 \wedge c = 1] \\
&\quad + \Pr[d' = 1 | b = 0 \wedge c = 1] - \Pr[d' = 1 \wedge b = 0 \wedge c = 0] \\
&\leq \mathsf{Adv}^{\mathsf{weak-ind}}_{\mathcal{M}\mathrm{TAES}}(\lambda, \tau) + \mathsf{Adv}^{\mathsf{anon}}_{\mathcal{M}\mathrm{TAES}}(\lambda, \tau)
\end{aligned}
$$

$\square$

Proof (of Theorem 5). Let $\mathcal{A}$ be an adversary with advantage ε against the subliminal-channel freeness property. We describe an adversary $\mathcal{B}$ against ciphertext unlinkability. Upon receiving the master key mpk, $\mathcal{B}$ runs $\mathcal{A}$ and gets (C_0, C_1, s). It then chooses a random bit c, and outputs $C = C_c$. The challenger chooses a random bit b, and then outputs C^* (that is either the re-randomization of C_c if $b = 0$ or a truly random ciphertext if $b = 1$, in which case no information is leaked about c). It is forwarded to $\mathcal{A}$ that answers its guess b'. If $b' = c$, we make $\mathcal{B}$ output 0, otherwise it outputs 1:

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{M}\mathrm{TAES},\mathcal{B}}(\lambda) &= \Pr[b' \neq c | b = 1] - \Pr[b' \neq c | b = 0] \\
&= 1/2 - \Pr[b' = 1 \wedge c = 0 | b = 0] - \Pr[b' = 0 \wedge c = 1 | b = 0] \\
&= \frac{1}{2} \times \big(1 - \Pr[b' = 1 | b = 0 \wedge c = 0] - \Pr[b' = 0 | b = 0 \wedge c = 1]\big) \\
&= \frac{1}{2} \times \big(\Pr[b' = 1 | b = 0 \wedge c = 1] - \Pr[b' = 1 | b = 1 \wedge c = 0]\big) \\
&= \frac{1}{2} \times \mathsf{Adv}^{\mathsf{subF}}_{\mathcal{M}\mathrm{TAES},\mathcal{A}}(\lambda)
\end{aligned}
$$

$\square$

B Complexity Assumptions

Our first proposal relies on the classical DDH assumption only. Our second proposal $\mathcal{M}\mathrm{TAES}_2$ requires the XDH and a new assumption weaker than the widely used (asymmetric) DBDH assumptions.

Definition 17 (Decisional Diffie-Hellman Assumption DDH). Let $\mathbb{G}$ be a cyclic group of prime order q. The DDH assumption in basis g states that the distributions of the tuples (g^x, g^y, g^{xy}) and (g^x, g^y, g^z) for random scalars x, y, z in $\mathbb{Z}_q$ are computationally indistinguishable.

Definition 18 (External Diffie-Hellman Assumption XDH). Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order q, with an admissible (which means non-degenerate and efficiently computable) bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$. We denote $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$, a bilinear structure $\mathcal{BS}$. The XDH assumption in $\mathcal{BS}$ assumes that the DDH assumptions holds in $\mathbb{G}_1$.

Definition 19 ((Asymmetric) Decisional Bilinear Diffie-Hellman Assumption DBDH). Let $\mathcal{BS}$ be a bilinear setting as above. The asymmetric DBDH assumption in bases g_1 and g_2 states that the distributions of the tuples $(g_1^x, g_2^x, g_1^y, g_2^z, G^{xyz})$ and $(g_1^x, g_2^x, g_1^y, g_2^z, G^w)$ for random scalars x, y, z, w in $\mathbb{Z}_q$ are computationally indistinguishable.

Definition 20 (Decisional Mixed Bilinear Diffie-Hellman Assumption DMBDH). Let $\mathcal{BS}$ be a bilinear setting as above. The DMBDH assumption in bases g_1 and g_2 states that the distributions of the tuples $(g_1^x, g_2^x, G^y, G^{xy})$ and (g_1^x, g_2^x, G^y, G^w) for random scalars x, y, z, w in $\mathbb{Z}_q$ are computationally indistinguishable.

C   Proofs of Security of our Scheme $\mathcal{M}$TAES$_1$

C.1   Privacy of $\mathcal{M}$TAES$_1$

Proof (of Theorem 6). Consider an adversary $\mathcal{A}$ against the privacy of our first scheme, we will construct an adversary $\mathcal{B}$ against the 2-DDH problem: This is a simple variant of the DDH-problem in basis g, where an instance is a 5-tuple $(g^x, g^y, g^z, g^{\bar{y}}, g^{\bar{z}})$, and one has to decide whether both $z = xy$ and $\bar{z} = x\bar{y}$, or both z and $\bar{z}$ are random. Using an hybrid argument, one easily gets that $\mathsf{Adv}_{\mathbb{G}}^{\text{2-ddh}}(\tau) \leq 2 \times \mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(\tau + 2\tau_{\text{exp}})$, where τ_{exp} is the time to compute an exponentiation in $\mathbb{G}$.

Let g be any generator of $\mathbb{G}$, and we have to guess at least one of the players that will not be corrupted: we choose an index I at random: we are given a 2-DDH instance $(U = g^u, V = g^v, Z = g^z, \bar{V} = g^{\bar{v}}, \bar{Z} = g^{\bar{z}})$, and set: $\mathsf{mpk} = (y_1 = g^{x_1}, y_2 = g^{x_2}, \cdots, y_I \leftarrow U, \cdots, y_t = g^{x_t})$ for random scalars $x_i \xleftarrow{R} \mathbb{Z}_q$ for $i \neq I$. When $\mathcal{A}$ asks a Join^* query for user j, we output the public key $\mathsf{pk}_j = y_j$. When $\mathcal{A}$ asks a $\mathsf{Corrupt}$ query for user j, we output $\mathsf{sk} = x_j$. With probability greater than $1/t$, there is no problem here, otherwise, we abort (if the adversary wants to corrupt player I).

Then, the adversary outputs a tuple $(\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}, m_0, m_1, s)$, and we know that the two (which are possibly the same) public keys are not corrupted. Again, still with probability greater than $1/t$, the above simulation worked (no corruption of player I) and there is d such that $\mathsf{pk}_{i_d} = y_I = U$. The random choice of d is equivalent to the above random choice of I, if $i_0 \neq i_1$, we thus either choose an additional random bit d (if $i_0 = i_1$), or set d so that $\mathsf{pk}_{i_d} = y_I = U$, and define the challenge ciphertext as described below:

$$A_1 = V \quad A_2 = Z \times m_d \quad B_1 = \bar{V} \quad B_2 = \bar{Z}$$

Then, $\mathcal{A}$ outputs his guess d' and we output the bit $\beta \leftarrow (d = d')$ as $\mathcal{B}$'s response. Let γ be the bit that indicates whether the 2-DDH-tuple follows a real ($\gamma = 0$) or random ($\gamma = 1$) distribution.

Note that if $\gamma = 0$, then the above challenge ciphertext really corresponds to the encryption of m_d under pk_{i_d}, and thus $\Pr[d' = d | \gamma = 0] = (\mathsf{Adv}_{\mathcal{M}\text{TAES}_1, \mathcal{A}}^{\text{priv}}(\lambda) + 1)/2$. On the other hand, if $\gamma = 1$, the challenge ciphertext is a truly random tuple in $\mathbb{G}^4$, and thus $\Pr[d' = d | \gamma = 1] = 1/2$. As a consequence,

$$\mathsf{Adv}_{\mathbb{G}}^{\text{2-ddh}}(\mathcal{B}) = \Pr[\beta = 1 | \gamma = 0] - \Pr[\beta = 1 | \gamma = 1]$$
$$= (\mathsf{Adv}_{\mathcal{M}\text{TAES}_1, \mathcal{A}}^{\text{priv}}(\lambda) + 1)/2 - 1/2 = \mathsf{Adv}_{\mathcal{M}\text{TAES}_1, \mathcal{A}}^{\text{priv}}(\lambda)/2.$$

Globally, the running time of $\mathcal{B}$ is the same as $\mathcal{A}$ and the real challenger, when the 2-DDH instance is given, but when the guess for I has been correct only, which happens with probability greater than $1/t$. This concludes the proof of privacy.   □

C.2   Unlinkability of $\mathcal{M}$TAES$_1$

Proof (of Theorem 7). We consider an adversary that tries to transfer some information in a ciphertext that does not trace back to any corrupted key. We will show that after re-randomization, any pre-ciphertext (or even a ciphertext already re-randomized) that does not trace back to any corrupted y_i leads to a randomly looking tuple in $\mathbb{G}^{4t}$.

Two cases can appear: the ciphertext provided by the adversary traces back to none of the t keys, or it traces back to an honest user. We are given a 2-DDH tuple $(U = g^u, V = g^v, Z = g^z, \bar{V} = g^{\bar{v}}, \bar{Z} = g^{\bar{z}})$, and we choose a random index I between 1 and $t + 1$, in which to inject the DDH tuple. The case $t + 1$ means that we do not inject it, and thus we bet that the ciphertext will trace back to nobody.

As in the previous proof, we set $\mathsf{mpk} = (y_1 \leftarrow g^{x_1}, y_2 = g^{x_2}, \cdots, y_I = U, \cdots, y_t = g^{x_t})$ for random scalars $x_i \xleftarrow{R} \mathbb{Z}_q$. When the adversary $\mathcal{A}$ asks a Join^* query for user j, we output $\mathsf{pk}_j \leftarrow y_j = g^{x_j}$. When $\mathcal{A}$ asks a $\mathsf{Corrupt}$ query for user j, we output $\mathsf{sk}_j \leftarrow x_j$. We recall that $\mathcal{L}$ is the list of all the registered keys/users, and $\mathcal{C}$ the list of the corrupted keys/users. When $\mathcal{A}$ outputs his target ciphertext $C = (A_1 = g^a, A_2 = g^b, B_1 = g^c, B_2 = g^d)$, we know (or assume) that it does not trace back to any corrupted user: $d = xc$, with $x \notin \mathcal{L} \backslash \mathcal{C}$. We now re-rerandomize it:

$$\forall i \neq I, C_{1,i} = A_1 \times B_1^{r'_i} \times g^{r_i} \quad C_{2,i} = A_2 \times B_2^{r'_i} \times y_i^{r_i} \quad D_{1,i} = B_1^{s'_i} \times g^{s_i} \quad D_{2,i} = B_2^{s'_i} \times y_i^{s_i}$$
$$= g^{a+c_i} \qquad\qquad = g^{b+xc_i+c'_i} \qquad = g^{d_i} \qquad = g^{xd_i+d'_i}$$
$$\text{if } I \leq t, C_{1,I} = A_1 \times B_1^{r'_I} \times V \quad C_{2,I} = A_2 \times B_2^{r'_I} \times Z \quad D_{1,I} = B_1^{s'_I} \times \bar{V} \quad D_{2,I} = B_2^{s'_I} \times \bar{Z}$$
$$= g^{a+cr'_I+v} \qquad = g^{b+cur'_I+z} \qquad = g^{cs'_I+\bar{v}} \qquad = g^{cus'_I+\bar{z}}$$

where $c_i = cr'_i + r_i$, $c'_i = (x_i - x)r_i$, $d_i = cs'_i + s_i$, $d'_i = (x_i - x)s_i$, which can be seen as four independent random variables, when $i \neq I$. As a consequence, for any $i \neq I$, the tuple $(C_{1,i}, C_{2,i}, D_{1,i}, D_{2,i})$ is randomly distributed in $\mathbb{G}^4$, independently of other tuples. Which means that if $I > t$, the ciphertext C is randomly distributed in $\mathbb{G}^{4t}$.

However, if $I \leq t$, which means that the adversary tries to transfer some information in a ciphertext that traces back an honest user: the re-randomized line I is

$$(C_{1,I} = g^{a+c_I}, C_{2,I} = g^{b+uc_I+(z-uv)}, D_{1,I} = g^{d_I}, D_{2,I} = g^{ud_I+(\bar{z}-u\bar{v})}),$$

where $c_I = cr'_I + v$ and $d_I = cs'_I + \bar{v}$. In the case of a real 2-DDH tuple, this is a real re-randomization, with $r_I = v$ and $s_i = \bar{v}$. But under the DDH assumption, this is indistinguishable to the situation where we have a random 2-DDH tuple, and in such a case,

$$(C_{1,I} = g^{a+c_I}, C_{2,I} = g^{b+uc_I+r}, D_{1,I} = g^{d_I}, D_{2,I} = g^{ud_I} + \bar{r}),$$

where $r = z - uv$ and $\bar{r} = \bar{z} - u\bar{v}$, which are two random scalars. This line is also a truly random tuple in $\mathbb{G}^4$.

With such a random 2-DDH instance as input, our re-randomization of C is perfectly indistinguishable from a truly random ciphertext in $\mathbb{G}^{4t}$. Hence the view of the adversary is perfectly independent of the bit b involved in the unlinkability game. Globally, the running time of our simulator is the same as $\mathcal{A}$ and the real challenger, when the 2-DDH instance is given, but when the guess for I has been correct only, which happens with probability greater than $1/(t+1)$. This concludes the proof of unlinkability. $\qquad\square$

D Indistinguishability of $\mathcal{M}\textsc{taes}_2$

Proof (of Theorem 8). All the previous results for $\mathcal{M}\textsc{taes}_1$ still hold for $\mathcal{M}\textsc{taes}_2$, since without the tracing key, everything can be done with elements in $\mathbb{G}_1$. We just have to prove that we can reveal the tracing keys for the indistinguishability. Let us thus show how we can initialize the simulation in this particular case, in a bilinear setting $\mathcal{BS} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, g_1, g_2)$. We consider an adversary $\mathcal{A}$ against indistinguishability, and we construct an adversary $\mathcal{B}$ against the DMBDH problem: given a tuple $(U_1 = g_1^u, U_2 = g_2^u, V = G^v, Z = G^z)$, we set:

$$\mathsf{mpk} = (g_1, g_2, G, (y_1 = g_1^{x_1}, y_2 = g_1^{x_2}, \cdots, y_I \leftarrow U_1, \cdots, y_t = g_1^{x_t}), (Y_j = e(y_j, g_2))_{j=1,\dots,t})$$

for random scalars $x_i \xleftarrow{R} \mathbb{Z}_q$ for $i \neq I$. We also set $\mathsf{sk}_O = (h_1 = g_2^{x_1}, h_2 = g_2^{x_2}, \cdots, h_I \leftarrow U_2, \cdots, h_t = g_2^{x_t})$.

When $\mathcal{A}$ asks a Join^* query for user j, we output the public key $\mathsf{pk}_j = (y_j, Y_j)$. When $\mathcal{A}$ asks a $\mathsf{Corrupt}$ query for user j, we output $\mathsf{sk} = x_j$. Then, the adversary outputs a tuple $(\mathsf{pk}, m_0, m_1, s)$, and we assume that $\mathsf{pk} = \mathsf{pk}_I$. We choose a random bit d, and define the challenge ciphertext $(A_1 = V, A_2 = Z \times m_d, B_1 = g_1^s, B_2 = y_I^s)$. Eventually, $\mathcal{A}$ outputs his guess d' and we output the bit $\beta \leftarrow (d = d')$ as $\mathcal{B}$'s response. Let γ

be the bit that indicates whether the DMBDH-tuple follows a real ($\gamma = 0$) or random ($\gamma = 1$) distribution: If $\gamma = 0$, then $A_1 = G^v$, which means that $r = v$, and $A_2 = G^{uv} = Y_I^r$. The above challenge ciphertext really corresponds to the encryption of m_d under pk_I, and thus $\Pr[d' = d | \gamma = 0] = (\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\text{TAES}_2, \mathcal{A}}(\lambda) + 1)/2$; On the other hand, if $\gamma = 1$, the challenge ciphertext is truly random and independent of d, and thus $\Pr[d' = d | \gamma = 1] = 1/2$. As a consequence,

$$\mathsf{Adv}^{\mathsf{dbdh}}_{\mathbb{G}}(\mathcal{B}) = \Pr[\beta = 1 | \gamma = 0] - \Pr[\beta = 1 | \gamma = 1] = (\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\text{TAES}_2, \mathcal{A}}(\lambda) + 1)/2 - 1/2 = \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}\text{TAES}_2, \mathcal{A}}(\lambda)/2.$$

Globally, the running time of $\mathcal{B}$ is the same as $\mathcal{A}$ and the real challenger, when the DMBDH instance is given, but when the guess for I has been correct only, which happens with probability greater than $1/t$. This concludes the proof of indistinguishability. $\qquad \square$

E Unlinkability of $\mathcal{M}$TAES$_3$

Proof (of Theorem 16). Consider an adversary $\mathcal{A}$ against the ω-resilient unlinkability of $\mathcal{M}\text{TAES}_3^\ell$, we construct an adversary $\mathcal{B}$ against the unlinkability of $\mathcal{M}\text{TAES}_3^1$.

KEY GENERATION. The adversary $\mathcal{B}$ simulates the GSetup algorithm by using params the common parameters received from its own challenger as the common parameters given to $\mathcal{A}$. In addition, $\mathcal{B}$ receives from its challenger a list of public key $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ for the scheme $\mathcal{M}\text{TAES}_3(1)$.

The algorithm picks then uniformly at random a vector $(a_1, \ldots, a_\ell) \in \mathcal{C} \subset \{1, \ldots, t\}^\ell$ and sets $(\mathsf{pk}_{i,a_i}, \mathsf{sk}^{(1)}_{i,a_i}, \mathsf{sk}^{(2)}_{i,a_i}) = (\mathsf{pk}_i, \bot, \bot)$ for $i \in \{1, \ldots, \ell\}$. It then executes $(t-1)\ell$ times $\mathsf{Kg}(\lambda)$ and gets $(t-1)\ell$ triples $(\mathsf{pk}_{i,j}, \mathsf{sk}^{(1)}_{i,j}, \mathsf{sk}^{(2)}_{i,j})$ for $(i, j) \in \{1, \ldots, \ell\} \times \{1, \ldots, t\}$ with $j \neq a_i$. It sets the group public key, the opening key and the master secret key as in the real scheme. The algorithm $\mathcal{B}$ runs $\mathcal{A}$ on input params and mpk.

Join* AND Corrupt QUERIES. When $\mathcal{A}$ asks a Join* query for user j, $\mathcal{B}$ encodes the id into the public key, $\mathsf{pk}_{\mathsf{id}} = \boldsymbol{c} = (c_1 c_2 \ldots c_\ell) \in \mathcal{C}$, where $c_i \in \{1, \ldots, t\}$. It furthermore adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ in the list $\mathcal{L}$.

When $\mathcal{A}$ asks a Corrupt query for user j with identity $\mathsf{id} = \boldsymbol{c}$, $\mathcal{B}$ outputs $(\mathsf{sk}^{(1)}_1, \ldots, \mathsf{sk}^{(1)}_t)$ possibly by asking to its own Corrupt oracle the keys $\mathsf{sk}^{(1)}_{i,a_i}$ it does not know. It furthermore adds the pair $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}})$ in the list $\mathcal{C}$.

If the adversary $\mathcal{A}$ wants to corrupt player $(a_1, \ldots, a_\ell)$, then $\mathcal{B}$ aborts. As in the previous proof, thanks to the ω-frameproof property of the code $\mathcal{C}$, we know that there is a pk_I, for some index I that has not been corrupted.

CHALLENGE CIPHERTEXT. Eventually, $\mathcal{A}$ outputs a ciphertext $C = (\boldsymbol{A}, \boldsymbol{B})$ (and possibly some state information s). As said above, we know that $\mathsf{sk}^{(1)}_{I,a_I}$ has not been queried to $\mathcal{B}$'s Corrupt oracle. The adversary $\mathcal{B}$ sends $C_I = (A_I, B_I)$ to its own challenger (which is therefore by definition a non-traceable ciphertext). It receives $C_I^\star = (A_1^\star, \ldots, A_t^\star, B_1^\star, \ldots, B_t^\star)$ that is either a re-randomization of C_I or a truly random ciphertext.

The adversary then applies the re-randomization procedure of $\mathcal{M}\text{TAES}_3^\ell$ to the ciphertext $C = (\boldsymbol{A}, \boldsymbol{B})$ in the following way: it picks at random t vectors $\boldsymbol{r_j}, \boldsymbol{s_j} \in (\mathbb{Z}_q^\star)^\ell$ for $j \in \{1, \ldots, t\}$, then generates $\boldsymbol{u} = \mathsf{Split}_{\mathbb{H}_2}(1, \ell)$ and $\boldsymbol{v} = \mathsf{Split}_{\mathbb{H}_2}(1, \ell)$. It computes t vectors of ciphertexts as follows:

$$\boldsymbol{C_j} \leftarrow \boldsymbol{A} \otimes (\boldsymbol{B}^{\odot^{\boldsymbol{r_j}}} \odot \mathsf{Encrypt}_2^{(\ell)}(\boldsymbol{u}, \mathsf{pk_j})) \quad \boldsymbol{D_j} \leftarrow \boldsymbol{B}^{\odot^{\boldsymbol{s_j}}} \odot \mathsf{Encrypt}_2^{(\ell)}(\boldsymbol{v}, \mathsf{pk_j}),$$

for $j \in \{1, \ldots, t\}$ and $\mathsf{pk_j} = (\mathsf{pk}_{1,j}, \ldots, \mathsf{pk}_{\ell,j})$. The algorithm $\mathcal{B}$ then replaces, for $j \in \{1, \ldots, t\}$, the value $C_{j,i}$ and $D_{j,i}$ by:

$$C_{I,j} \leftarrow A_I^\star \otimes \mathsf{Encrypt}_2(u_I, \mathsf{pk}_{I,j}) \quad D_{I,j} \leftarrow B_I^\star \odot \mathsf{Encrypt}_2(v_I, \mathsf{pk}_{I,j}).$$

It outputs the $2t\ell$ vector $C' = (\boldsymbol{C_1}, \ldots, \boldsymbol{C_t}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_t})$. It is readily seen that if $C^\star$ is a valid re-randomization of $C_i = (A_i, B_i)$, then C' is a valid re-randomization of C. However, if $C^\star$ is a random

ciphertext, then C' is a random ciphertext too. When eventually, the adversary $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ forwards to its own challenger.

CONCLUSION. Globally, the running time of $\mathcal{B}$ is the same as $\mathcal{A}$ (and the real challenger) plus the time to execute $(t-1)\ell$ times Kg and its success probability is identical to the one of $\mathcal{A}$ when the guess for $(a_1, \ldots, a_\ell)$ has been correct (which happens with probability greater than $1/n$). This concludes the proof of unlinkability. $\qquad\qquad\Box$