Toward Practical Group Encryption

Laila El Aimani and Marc Joye

Technicolor, Security & Content Protection Labs
1 avenue de Belle Fontaine, 35576 Cesson-Sevigné Cedex, France

Abstract. A group encryption scheme allows anyone to form a ciphertext for a given group member while keeping the receiver's identity private. At the same time, the encryptor is capable of proving that some (anonymous) group member is able to decrypt the ciphertext and, optionally, that the corresponding plaintext satisfies some *a priori* relation (to prevent sending bogus messages). Finally, in case of a dispute, the identity of the intended receiver can be recovered by a designated authority. In this paper, we abstract a generic approach to construct group encryption schemes. We also introduce several new implementation tricks. As a result, we obtain group encryption schemes that significantly improve the state of the art. Both interactive and non-interactive constructions are considered.

Key words: Group encryption, Canetti-Halevi-Katz paradigm, homomorphic encryption, structure-preserving signatures, (non)-interactive zero-knowledge.

1 Introduction

Basically, group signature schemes [13] allow a registered group member to conceal her identity when issuing digital signatures. However, any group signature can be opened by a designated group authority to reveal the signature's originator. In a dual way, group encryption schemes [23] provide revocable anonymity to the ciphertext's receiver. More specifically, a group encryption scheme is a public-key encryption scheme augmented with special properties: (1) the receiver's identity is hidden among the set of group members, (2) an opening authority is able to uncover the receiver's identity if need be, and (3) the ciphertext's originator is able to convince a verifier that (3-a) the ciphertext can be decrypted by a group member, (3-b) the opening authority can open the ciphertext and revoke the anonymity, and (3-c) the corresponding plaintext satisfies some *a priori* relation.

The additional features enjoyed by group encryption schemes make them suitable for a number of privacy-aware applications. One of them resides in secure oblivious retriever storage where anonymous credentials may move between computing elements (computer, mobile unit, etc...). Asynchronous transfer, which does not require the presence of all devices (subject to the transfer) at the same time, may resort to an untrusted server for storing temporarily the encrypted credentials. Group encryption can be employed in implementing such a storage server where it is guaranteed that (1) the server stores well formed encrypted credentials; (2) the credentials have a legitimate anonymous retriever; (3) if necessary, an authority is able to pin down the identity of the retriever.

Further scenarios where group encryption can be utilized are described in [23, 12]. It is worth noting that when the opening functionality is not needed in these applications, one can thrive on a closely related primitive titled ring encryption [25].

Related work The concept of group encryption was first formalized by Kiayias, Tsiounis, and Yung [23]. They also provide a modular design to build such schemes along with a concrete instantiation. Their realization achieves a ciphertext size of 2.4 kB and a well-formedness proof of approximately 70 kB for an 80-bit security level and a 2^{-50} soundness error. The main criticism to the proposal lies in entailing interaction with the verifier in order to prove the validity of the ciphertext. In fact, interaction can be cumbersome in situations where the encryptor needs to run the proof several times with different verifiers, as this would require remembering all the random coins used to form the ciphertext.

This shortcoming was addressed in subsequent works. First, Qin *et al.* [27] suggested a closely related primitive with non-interactive proofs of well-formedness of the ciphertext using the random oracle idealization. Then, Cathalo, Libert, and Yung [12] provided the first non-interactive realization of group encryption in the standard model. Their ciphertext and proof are also significantly shorter than those of [23] (the ciphertext size is 1.25 kB and the proof size is 16.125 kB for a 128-bit security level). However, the dark side of this non-interactive proposal resides in the expensive cost of the proof verification (several thousands of pairing evaluations) due to the recourse to the Groth-Sahai [21] proof system.

To summarize the state of the art in group encryption, there is on the one hand an interactive proposal with a rather consequent size of the ciphertext and its proof of well-formedness, but which has the merit of having an efficient verification of this proof, and on the other hand, there is a non-interactive realization which significantly reduces the size of the ciphertext and its validity proof, but which is characterized by its computationally demanding proof verification that renders it fairly impractical.

It is would be nice to combine the best of the two works and come up with a scheme with short ciphertexts and proofs, and where both the interactive and non-interactive setting are efficiently supported. This is the main contribution of this paper.

Contributions and underlying ideas We propose a new design strategy for group encryption which significantly improves the performance. Two main ideas underlay our conception.

First, instead of assembling highly secure components, we start with weaker — and so more efficient — primitives to get a group encryption scheme secure in a weak sense. The so-obtained scheme is next converted with a generic transform into a fully-secure group encryption scheme. In addition to efficiency, starting with weaker components also brings diversity and permits to develop further schemes, under various security assumptions. As a by-product, we show that the transform used to upgrade the security in group encryption applies to tag-based encryption and allows also to uplift the security in this primitive while preserving the verifiability properties.

Second, we encrypt only an alias of the receiver's public key in order to realize the opening functionality, leading consequently to important extra savings in both size and computation. In fact, the prior works [23, 12] include in the ciphertext an encryption (using the opening authority's public key) of the receiver's public key in order to implement the opening function. Since a public key often consists of a vector of group elements, [23, 12] use a CCA secure encryption to encrypt each component of the key. We remark that such an operation is unnecessary as the public keys are all maintained in a public database. Therefore, encrypting only an alias of the key (which will be recorded along with the key in the database) is enough for this functionality. The opening authority needs then to execute the extra step of looking up the database for the key corresponding to the alias, however we note that the recourse to the opening function is done only in case of disputes and occurs thus very rarely.

Our new generic construction accepts many practical instantiations which support both interactive and non-interactive validity proofs. For instance, we get for a 128-bit security level, a concrete realization in the standard model with a ciphertext size of 0.4 kB, an interactive proof of 1 kB, a non-interactive proof of 2 kB which requires 325 pairing evaluations (vs 3895 in [12]) for the verification.

Outline of the paper The rest of this paper is organized as follows. The next section formally defines the syntax and the model of a secure group encryption scheme. Section 3 presents a generic construction to build group encryption schemes. It also provides a transform to turn a weakly-secure group encryption scheme into a fully-secure one. Subsequently, sufficient conditions for practical realizations and a detailed instantiation are described in Section 4. Finally, the paper concludes in Section 5.

2 Group Encryption: Syntax and Security Model

In this section, we review the formal definition of group encryption, as introduced in [23]. We also present the corresponding security notions.

It is useful to introduce some notation. For a two-party protocol between A and B, we represent its execution as $\langle output_A \mid output_B \rangle \leftarrow \langle A(input_A), B(input_B) \rangle (common\text{-}input)$. The security properties are described through experiments where the adversary is given access to oracles. We write $\mathcal{A}^{\texttt{oracle}(\cdot)}$ to denote that adversary $\mathcal{A}$ has access to oracle $\texttt{oracle}(\cdot)$. When a query is not allowed, we use the symbol $\neg$: $\mathcal{A}^{\texttt{oracle}^{\neg(\text{some query})}(\cdot)}$.

2.1 Syntax

A *group encryption scheme* consists of the following algorithms/protocols:

$\texttt{setup}(1^\kappa)$ On input a security parameter κ, this probabilistic algorithm generates the public parameters *param* of the scheme. Although not always explicitly mentioned, *param* will serve as an input to all the algorithms/protocols that follow.

$(\mathcal{G}_r, \mathcal{R}, \texttt{sample}_\mathcal{R})$ This tuple of algorithms is part of the $\texttt{setup}$ procedure and is needed for verifiability; *i.e.*, proving that the decryption of a certain ciphertext satisfies a given relation. In this sense, $\mathcal{G}_r$ generates the key pair $(pk_\mathcal{R}, sk_\mathcal{R})$ of the relation $\mathcal{R}$ from a security parameter. Similarly to [23, 12], $sk_\mathcal{R}$ can be empty if the relation $\mathcal{R}$ is publicly sampleable (*e.g.*, the Diffie-Hellman relation in bilinear groups). On input the key pair of the relation $\mathcal{R}$, algorithm $\texttt{sample}_\mathcal{R}$ produces a pair (x, w) consisting of an instance x and a witness w for the relation $\mathcal{R}$. The polynomial-time testing procedure $\mathcal{R}(x, w)$ returns 1 iff (x, w) belongs to the relation based on the public parameter $pk_\mathcal{R}$.

$\texttt{keygen}_\mathsf{E}(param)$ This probabilistic algorithm outputs the key pair $(pk_\mathsf{E}, sk_\mathsf{E})$ of the entity E in the system; E can either be the group manager GM who manages the set of receivers (group members), or the opening authority OA that recovers the receiver's identity from a given ciphertext, or a group member User who receives ciphertexts.

$\texttt{join} = \langle \mathsf{J}_\mathsf{User}(param), \mathsf{GM}(sk_\mathsf{GM}) \rangle (pk_\mathsf{GM})$ This is an interactive protocol between GM and the potential joining group member J_User. The latter sends her public key pk to GM and prospectively proves the correctness of her key, whereas GM issues (at the end of the protocol) a certificate $cert_{pk}$ that marks the effectiveness of the user's membership. GM stores additionally the pair $(pk, cert_{pk})$ in a public directory *database*.

$\texttt{encrypt}(pk_\mathsf{GM}, pk_\mathsf{OA}, pk, w, L)$ On input the respective public keys pk_GM and pk_OA of GM and OA, the (certified) public key pk of the receiver, this algorithm encrypts the witness w to produce a ciphertext ψ for a certain label L (which specifies the "context" of the encryption).

$\texttt{prove} = \langle \mathcal{P}(w, coins_\psi), \mathcal{V}(param) \rangle (pk_\mathsf{GM}, pk_\mathsf{OA}, pk_\mathcal{R}, x, \psi, L)$ This is an interactive protocol between a sender $\mathcal{P}$ (acting as the prover) who has generated the ciphertext ψ and any verifier $\mathcal{V}$; in this protocol, the sender uses the random coins used to produce ψ in order to prove that there is a group member whose key is registered in *database* and who is capable of decrypting ψ, under label L, and recovering a witness w such that $(x, w) \in \mathcal{R}$. At the end of the protocol, the verifier outputs 1 if the proof is accepted, and 0 otherwise.

$\texttt{decrypt}(sk, \psi, L)$ On input the private key sk of the group user, this algorithm decrypts the ciphertext ψ, under label L, and outputs the witness w (or a rejection symbol $\perp$).

$\texttt{open}(sk_\mathsf{OA}, \psi, L)$ On input the private key sk_OA of OA and a ciphertext ψ with corresponding label L, this algorithm outputs the public key pk under which ψ was created.

Remark 1. As aforementioned, the verifiability of encryption is optional. If verifiability is not desired, the relation $\mathcal{R}$ can be set to the trivial relation that includes any string of fixed size as a witness.

We require the scheme to be *correct*, that is, to properly offer the desired features for group encryption. More formally, we require for a group encryption scheme that the following "correctness game" returns 1 with overwhelming probability.

$\boxed{\text{Experiment } \mathbf{Exp}^{\text{correctness}}(\kappa)}$

1. $param \leftarrow \texttt{setup}(1^\kappa)$; $(pk_\mathcal{R}, sk_\mathcal{R}) \leftarrow \mathcal{G}_r(1^\kappa)$; $(x, w) \leftarrow \texttt{sample}_\mathcal{R}(pk_R, sk_R)$;
2. $(pk_\mathsf{GM}, sk_\mathsf{GM}) \leftarrow \texttt{keygen}_\mathsf{GM}(1^\kappa, param)$; $(pk_\mathsf{OA}, sk_\mathsf{OA}) \leftarrow \texttt{keygen}_\mathsf{OA}(1^\kappa, param)$;

3. $\langle pk, sk, cert_{pk} \mid pk, cert_{pk}\rangle \leftarrow \langle J_{\mathsf{User}}(param), \mathsf{GM}(sk_{\mathsf{GM}})\rangle(pk_{\mathsf{GM}})$;
4. $(\psi, coins_\psi) \leftarrow \mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, w, L)$;
5. $\langle done \mid out_1\rangle \leftarrow \langle \mathcal{P}(w, coins_\psi), \mathcal{V}(param)\rangle(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, \psi, L)$;
6. $out_2 \leftarrow (pk = \mathtt{open}(sk_{\mathsf{OA}}, \psi, L))$;
7. $out_3 \leftarrow (w = \mathtt{decrypt}(sk, \psi, L))$;
8. If $(out_1 = out_2 = out_3 = 1)$ return 1 else return 0.

2.2 Security model

Message security The message security captures the property that an adversary cannot learn any information whatsoever on a message from an encryption of it. Strong security guarantees require that this holds true even when the adversary has adaptive access to a decryption oracle. For group encryption, it is also assumed that the adversary may control the group manager and the opening authority, and that he has access to the prove oracle in the challenge phase. We let IND-CCA denote the corresponding security notion. There is a weaker notion, denoted IND-sl-wCCA, where the adversary commits to the target label beforehand (selective-label attacks) and is not allowed to issue decryption queries involving the target label (weak chosen-ciphertext attacks).

Formally, a group encryption scheme meets the IND-sl-wCCA notion if the success probability of any polynomial-time adversary $\mathcal{A}$ to distinguish among encryptions of a chosen message and of a random message is at most negligibly better (in security parameter κ) than 1/2 in the experiment that follows. In this experiment we use the following notation (similar to that in [23, 12].

– $\mathtt{decrypt}^{\neg(\cdot,L)}(sk, \cdot)$: is a stateless decryption oracle which is restricted not to decrypt ciphertexts w.r.t. the label L.
– $\mathsf{CH}^b_{\mathsf{ror}}(1^\kappa, pk, w, L)$: is a real-or random challenge oracle that is only queried once. It returns $\psi, coins_\psi$ such that $\psi \leftarrow \mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, cert_{pk}, w, L)$ if $b = 1$, and $\psi \leftarrow \mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, cert_{pk}, w', L)$ otherwise, where w' is a random plaintext chosen uniformy in the space of messages of length 1^κ. In both cases $coins_\psi$ denote the random coins used to produce ψ.
– $\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi)$: this a stateful oracle that the adversary can query on multiple occasions. If $b = 1$, it runs the real prover $\mathcal{P}$ (of the prove procedure) using the private inputs $w, coins_\psi, pk, cert_{pk}$ to produce a real proof (the common input being $pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi$). If $b = 0$, the oracle runs a simulator $\mathcal{P}'$ on the same common input $pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi$, but which is deprived from the private input $w, coins_\psi$ ($\mathcal{P}'$ may have access to $pk, cert_{pk}$), to generate a simulated proof. As pointed in [23, 12], designing an efficient simulator $\mathcal{P}'$ is part of proving the security property.

$\boxed{\text{Experiment } \mathbf{Exp}^{\mathsf{IND\text{-}sl\text{-}wCCA}}_{\mathcal{A}}(\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(aux, pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, L) \leftarrow \mathcal{A}(param)$;
3. $\langle pk, sk, cert_{pk} \mid aux, pk, cert_{pk}\rangle \leftarrow \langle \mathrm{J}_{\mathsf{User}}(param), \mathcal{A}(aux)\rangle(pk_{\mathsf{GM}})$;
4. $(aux, x, w, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(\cdot,L)}(sk,\cdot)}(aux)$; ▷ `Find stage`
5. If $(x, w) \notin \mathcal{R}$ then abort;
6. $b \xleftarrow{R} \{0, 1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}^b_{\mathsf{ror}}(1^\kappa, pk, w, L)$;
7. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi), \mathtt{decrypt}^{\neg(\cdot,L)}(sk,\cdot)}(aux, \psi)$; ▷ `Guess stage`
8. If $(b = b^\star)$ return 1 else return 0.

To get the full IND-CCA security level, the above experiment is modified in a way such that:

(i) the adversary is required to select the target label L only at the end of its `find` stage, and
(ii) the adversary is no longer restricted in its decryption queries (with the sole exception of the pair (ψ, L) in its `guess` stage) — in particular, the adversary is allowed to issue decryption queries including the target label L.

Anonymity The notion of anonymity is described in an analogous way and comes with similar variations. The goal of the adversary is now to distinguish among two possible receivers given the encryption of a same witness under two different public keys. Of course, the adversary does not control the opening authority (and so is given the public opening key pk_{OA}).

The formal definition of selective-label anonymity against weak chosen-ciphertext attacks (in short, ANO-sl-wCCA) follows. The notion of ANO-sl-wCCA is met if the success probability of any polynomial-time adversary $\mathcal{A}$ is at most negligibly better than $1/2$. The formal definition of ANO-CCA (anonymity against chosen-ciphertext attacks) is obtained by modifying the experiment as for the IND-CCA notion (see above). Similarly to [23, 12], we introduce the following notations:

- $\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot)$: is a stateless opening oracle, for the key pk_{OA}, which is restricted not to open ciphertexts w.r.t. the label L.
- $\mathsf{CH}^{b}_{\mathsf{anon}}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_0, pk_1, w, L)$: is a challenge oracle that is only queried once. It returns $\psi, coins_{\psi}$ such that $\psi \leftarrow \mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_b, cert_b, w, L)$ and $coins_{\psi}$ denote the random coins used to produce ψ.
- $\mathsf{User}(pk_{\mathsf{GM}})$: is a stateful oracle that simulates two executions of J_{User} to introduce two honest users in the group. It uses a string keys where the outputs of the two executions are written.

Experiment $\mathbf{Exp}^{\mathsf{ANO\text{-}sl\text{-}wCCA}}_{\mathcal{A}}(\kappa)$

1. $param \leftarrow \mathtt{setup}(1^{\kappa})$; $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}}) \leftarrow \mathtt{keygen}_{\mathsf{OA}}(1^{\kappa}, param)$;
2. $(aux, pk_{\mathsf{GM}}, L) \leftarrow \mathcal{A}(param)$; $aux \leftarrow \mathcal{A}^{\mathsf{User}(pk_{\mathsf{GM}}),\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot)}(aux, pk_{\mathsf{OA}})$;
   If $\mathsf{keys} \neq (pk_0, sk_0, cert_{pk_0}, pk_1, sk_1, cert_{pk_1})$ return 0;
3. $(aux, x, w, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux)$;
4. If $(x, w) \notin \mathcal{R}$ return 0;
5. $b \stackrel{R}{\leftarrow} \{0,1\}$; $(\psi, coins_{\psi}) \leftarrow \mathsf{CH}^{b}_{\mathsf{anon}}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_0, pk_1, w, L)$;
6. $b^{\star} \leftarrow \mathcal{A}^{\mathtt{prove}_{\mathcal{P}(w, coins_{\psi}, pk_b, cert_{pk_b})}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi),\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux, \psi)$;
7. If $(b = b^{\star})$ return 1 else return 0.

Soundness In a soundness attack, the adversary creates adaptively the intended group of receivers communicating with the genuine group manager. The adversary is successful if it can produce a ciphertext ψ and a corresponding proof of validity w.r.t. a relation $\mathcal{R}$ with a chosen $pk_{\mathcal{R}}$ such that (1) ψ is invalid, or (2) opening ψ results in an invalid public key or a value which is not equal to the public key of any group member. We adhere to the same formal definition of [23, 12] which we recall in Appendix B.

3 Building Group Encryption Schemes

In this section we present our new strategy to build efficient group encryption schemes. We start by providing a construction which achieves "weak" security properties from "weakly secure" components. Next, we use a technique evocative of the Canetti-Halevi-Katz transformation to upgrade the security of the resulting construction into full-fledged CCA security.

In the rest of this document, and in order to avoid confusion, we use of a dot notation to refer the different components; for instance, $\Gamma.\mathtt{encrypt}()$ refers to the encryption algorithm of public-key scheme Γ, $\Sigma.pk$ to the public key of signature scheme Σ, etc.

3.1 A generic construction

Our construction for group encryption departs from the specific constructions in [23, 12] in encrypting only an *alias* to the public key (computed using a function H) instead of encrypting the entire public key. As will become apparent, such a change drastically reduces the cost and size of the resulting encryption. Moreover, and similarly to [12], it does not include the commitment on the public key (and potentially on its certificate) in the ciphertext.

Let $\Gamma^{\mathsf{User}} = (\mathtt{keygen}, \mathtt{encrypt}, \mathtt{decrypt})$ and $\Gamma^{\mathsf{OA}} = (\mathtt{keygen}, \mathtt{encrypt}, \mathtt{decrypt})$ be two public-key encryption schemes with labels. Let further $\Sigma = (\mathtt{keygen}, \mathtt{sign}, \mathtt{verify})$ be a signature scheme. We assume that the message space of Σ includes the public-key space of Γ^{User}. Finally, let H denote a collision-resistant function from the public key space of Γ^{User} to the message space of Γ^{OA}.

The properties required for H to guarantee an efficient `prove` algorithm/protocol are described in the next section. The definitions of the different building blocks involved are summarized in Appendix A.

$\mathtt{setup}(1^\kappa)$ This algorithm invokes the setup algorithms for the building blocks (namely, Γ^{User}, Γ^{OA}, and Σ), and outputs *param*. The public parameters *param* are input to all the subsequent algorithms/protocols and further include the description of a relation $\mathcal{R}$ along with a key pair $(pk_{\mathcal{R}}, sk_R)$ necessary for sampling pairs (x, w) where $(x, w) \in \mathcal{R}$ and w belongs to the message space of Γ^{User}. Finally, `setup` outputs also a description of a collision-resistant function H which maps elements from the public key space of Γ^{User} into elements in the message space of Γ^{OA}.

$\mathtt{keygen}_{\mathsf{GM}}(1^\kappa)$ This algorithm invokes $\Sigma.\mathtt{keygen}(1^\kappa)$ and outputs $(pk_{\mathsf{GM}}, sk_{\mathsf{GM}}) = (\Sigma.pk, \Sigma.sk)$.

$\mathtt{keygen}_{\mathsf{OA}}(1^\kappa)$ This algorithm invokes $\Gamma^{\mathsf{OA}}.\mathtt{keygen}(1^\kappa)$ and outputs $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}}) = (\Gamma^{\mathsf{OA}}.pk, \Gamma^{\mathsf{OA}}.sk)$.

$\mathtt{keygen}_{\mathsf{User}}(1^\kappa)$ This algorithm invokes $\Gamma^{\mathsf{User}}.\mathtt{keygen}(1^\kappa)$ and outputs $(pk_{\mathsf{User}}, sk_{\mathsf{User}}) = (\Gamma^{\mathsf{User}}.pk, \Gamma^{\mathsf{User}}.sk)$ as a key pair for the group member User.

$\mathtt{join} = \langle J_{\mathsf{User}}, \mathsf{GM}(sk_{\mathsf{GM}})\rangle(pk_{\mathsf{GM}})$ The potential joining group member J_{User} wishing to join the group sends her public key pk_{User} (obtained after calling $\mathtt{keygen}_{\mathsf{User}}$) to GM, and the latter replies with the certificate $cert_{pk_{\mathsf{User}}} = \Sigma.\mathtt{sign}_{\Sigma.sk}(pk_{\mathsf{User}})$. GM then stores $(pk_{\mathsf{User}}, H(pk_{\mathsf{User}}), cert_{pk_{\mathsf{User}}})$ in a public directory *database*. (We stress that for any two different pk_{User} and pk'_{User}, the values of $H(pk_{\mathsf{User}})$ and $H(pk'_{\mathsf{User}})$ will be different.)

$\mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathsf{User}}, w, L)$ This algorithm first produces an encryption ψ_1 using Γ^{User} on w with the public key pk_{User} under label L, and then encrypts $H(pk_{\mathsf{User}})$ in ψ_2 using Γ^{OA} under label L with public key pk_{OA}. The ciphertext consists of the pair (ψ_1, ψ_2).

$\mathtt{prove} = \langle \mathcal{P}(w, coins_\psi), \mathcal{V}(param)\rangle(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, \psi, L)$ $\mathcal{P}$ who created the ciphertext $\psi = (\psi_1, \psi_2)$ uses the coins used to produce ψ in order to prove to $\mathcal{V}$:

- knowledge of the message underlying ψ_1 and that it forms a witness for the instance x w.r.t. the relation $\mathcal{R}$;
- knowledge of the decryption of ψ_2 and that it corresponds to the value of the function H on the public key under which ψ_1 is created;
- knowledge of a certificate on the public key used to create ψ_1.

These proofs are detailed in Section 4.

$\mathtt{decrypt}(sk_{\mathsf{User}}, \psi, L)$ This algorithm parses ψ as (ψ_1, ψ_2), invokes $\Gamma^{\mathsf{User}}.\mathtt{decrypt}$ on (ψ_1, L), and outputs the result of this decryption, say w, if $(x, w) \in \mathcal{R}$, and $\perp$ otherwise.

$\mathtt{open}(sk_{\mathsf{OA}}, \psi, L)$ This algorithm parses ψ as (ψ_1, ψ_2), invokes $\Gamma^{\mathsf{OA}}.\mathtt{decrypt}$ on (ψ_2, L), then looks up *database* for the preimage w.r.t. the function H of such a decryption, and finally outputs the result of this search.

Theorem 1. *The construction of § 3.1 yields a group encryption scheme with*

1. IND-sI-wCCA *message security if Γ^{User} is a tag-based encryption scheme having indistinguishability of encryptions under selective-tag weak chosen-ciphertext attacks, and* `prove` *is zero knowledge.*
2. ANO-sI-wCCA *anonymity if Γ^{User} is a tag-based encryption scheme having indistinguishability of keys under selective-tag weak chosen-ciphertext attacks, Γ^{OA} is a tag-based encryption scheme having indistinguishability of encryptions under selective-tag weak chosen-ciphertext attacks, and* `prove` *is zero knowledge.*
3. *soundness if the proof underlying* `prove` *is sound and the used certification scheme is EUF-CMA secure.*

A formal proof is provided in Appendix C.

3.2 A Canetti-Halevi-Katz like paradigm for group encryption

Canetti, Halevi, and Katz [10] provide a method that transforms any selective-identity chosen-plaintext secure identity-based scheme into one with full-fledged chosen-ciphertext security. The transformation, referred to as the *CHK transform*, consists in signing the ciphertext, result of encryption with the weakly secure identity-based encryption scheme, using a one-time signature scheme, wherein the "identity" is given by the verification key. Later in [7], Boneh and Katz improve the CHK transform using a MAC instead of a one-time signature; see also [6]. Concurrently, MacKenzie, Reiter, and Yang [26] present a method for converting a weakly chosen-ciphertext secure tag-based encryption scheme to a fully secure public-key encryption scheme. Finally, Kiltz [24] combines the ideas of [10, 26, 7] in order to derive chosen-ciphertext secure public-key encryption schemes from selective-tag weakly chosen-ciphertext secure tag-based encryption schemes using one-time signatures.

Interestingly and analogously to [24], we can now turn a weakly secure group encryption scheme as per Theorem 1 into a group encryption scheme with full message security (*i.e.*, IND-CCA) and full anonymity (*i.e.*, ANO-CCA). Let $\mathcal{GE}^\star$ be a group encryption satisfying the notions of IND-sI-wCCA and ANO-sI-wCCA. Given $\mathcal{GE}^\star$, we construct a group encryption scheme $\mathcal{GE}$ meeting the strong notions of IND-CCA and ANO-CCA as depicted in Fig. 1. The conversion uses a one-time signature scheme $\mathfrak{S} = (\texttt{keygen}, \texttt{sign}, \texttt{verify})$.

Theorem 2. *The group encryption scheme $\mathcal{GE}$ obtained from the conversion in Fig. 1 has* IND-CCA *message-security and* ANO-CCA *anonymity if $\mathcal{GE}^\star$ is* IND-sI-wCCA *and* ANO-sI-wCCA, *and $\mathfrak{S}$ is a strongly secure one-time signature scheme.*

The proof can be found in Appendix D.

$\boxed{\mathcal{GE}.\texttt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathsf{User}}, w, L)}$

1. $(\mathfrak{S}.pk, \mathfrak{S}.sk) \leftarrow \mathfrak{S}.\texttt{keygen}(1^\kappa)$;
2. $\psi^\star \leftarrow \mathcal{GE}^\star.\texttt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathsf{User}}, w, \mathfrak{S}.pk)$;
3. $\sigma \leftarrow \mathfrak{S}.\texttt{sign}(\mathfrak{S}.sk, \psi^\star \| L)$;
4. Return $\psi \leftarrow (\psi^\star, \mathfrak{S}.pk, \sigma)$.

$\boxed{\mathcal{GE}.\texttt{decrypt}(sk_{\mathsf{User}}, \psi, L)}$

1. Parse ψ as $(\psi^\star, \mathfrak{S}.pk, \sigma)$;
2. If $\mathfrak{S}.\texttt{verify}(\mathfrak{S}.pk, \psi^\star \| L) = \perp$ then return $\perp$;
3. Else return $\mathcal{GE}^\star.\texttt{decrypt}(sk_{\mathsf{User}}, \psi^\star, \mathfrak{S}.pk)$.

Fig. 1. Conversion

Remark 2. This transformation can be also used to upgrade the security in TBE (from st-wCCA to full CCA indistinguishability and anonymity). In [24], Kiltz suggests to achieve this task via a CCA secure public key encryption (PKE): first derive a CCA secure PKE from a st-wCCA secure TBE, then identify the pair "(message,tag)" in the TBE by the message "message||tag" in the PKE.

Our transform has the merit of preserving the algebraic structure of the message to be encrypted. This impacts positively the verifiability of the encryption, i.e. proving knowledge of the message underlying the resulting CCA encryption is as efficient as proving knowledge of the message underlying its st-wCCA encryption.

4 Efficient Instantiations

The sender of the message, in the construction provided in the previous section, is compelled to provide the following proof in the prove procedure:

$$
\begin{aligned}
\mathsf{PoK} = \{(pk, cert_{pk}, w) : \; & cert_{pk} = \Sigma.\mathtt{sign}_{\Sigma.sk}(pk) \\
& \psi_1 = \Gamma^{\mathsf{User}}.\mathtt{encrypt}_{pk}(w) \;\wedge \\
& \psi_2 = \Gamma^{\mathsf{OA}}.\mathtt{encrypt}_{pk_{\mathsf{OA}}}(H(pk)) \\
& (x, w) \in \mathcal{R} \\
\} \; & (\psi_1, \psi_2, \Sigma.pk, pk_{\mathsf{OA}}, x)
\end{aligned}
$$

where the private input of the prover are the coins used to form ψ_1, ψ_2 in addition to pk, $cert_{pk}$, and m.

According to whether we want to provide an interactive or a non-interactive prove procedure, the components underlying the construction have to satisfy different conditions:

Non-interactive setting We can provide a non-interactive zero knowledge (NIZK) prove if the language defined by this procedure is compatible with the Groth-Sahai proof system [21] (Appendix A.5). In fact, [21] provides efficient NIZK or NIWI proofs for a number of languages that cover pairing-product equations, multi-scalar multiplication, and quadratic equations.
In this case, the common reference string (CRS) needs to be part of the setup algorithm. Besides, the private input of the prover has to consist of only group elements. Moreover, the building blocks, namely Σ, Γ^{User}, Γ^{OA}, H, and $\mathcal{R}$ need to perform only group or pairing (if bilinear groups are involved) operations on this private input. For example, we can consider structure-preserving signatures [19, 17, 1] for the certification scheme Σ, Kiltz'[24] or Cash *et al.*'s [11](described in Figure 8) encryption schemes for both Γ^{User} or Γ^{OA}, the discrete logarithm or the Diffie-Hellman relation for $\mathcal{R}$, and any function H performing group or pairing operations on the input. Note however that the statments underlying prove that consist of pairing-product equations need to be of special form (see Appendix A.5) in order to accept zero knowledge proofs (otherwise prove will be only witness-indistinguishable).

Interactive setting While having a number of useful properties, non-interactive proofs built from Groth-Sahai's proof system suffer the high verification cost due to the pairing evaluations in the verification. Therefore, it would judicious to support the construction in the previous section with an interactive variant of prove. This will decree different conditions on the building blocks. The essence of these conditions consists in manipulating the private input, which has to comprise only group elements, through homomorphic maps.

The rest of this section is organized as follows. Subsection 1 will introduce formally the classes of the different components Σ, Γ^{User}, Γ^{OA}, H, and $\mathcal{R}$ that will lead to an efficient *interactive* prove. Subsection 2 describes explicitly the *interactive* prove protocol in case the building blocks are instantiated from the previously presented classes. Finally, we provide a concrete realization of group encryption in Subsection 3 and compare the resulting performances with the those of the prior proposals [23, 12].

4.1 Building blocks

Definition 1 (The class $\mathbb{S}$ of certification/signature schemes). *$\mathbb{S}$ is the set of all digital signatures Σ for which there exists a pair of efficient algorithms,* convert *and* retrieve*, where* convert *inputs a verification key vk, a key pk (to be certified), and a valid signature $cert_{pk}$ (w.r.t. vk) on pk, and outputs a tuple (S, R) such that:*

1. *R is information theoretically independent from $cert_{pk}$ and pk. I.e. There exists an algorithm* simulate *that inputs a verification key vk from the verification key space and outputs a string statistically indistinguishable from R.*
2. *There exists an algorithm* compute *that on the input vk and R, computes a description of a map $F : (\mathbb{G}_S, *_S) \times (\mathbb{G}_{pk}, *_{pk}) \to (\mathbb{G}_F, \circ_F)$:*
   - *where $(\mathbb{G}_S, *_S)$ and $(\mathbb{G}_{pk}, *_{pk})$ are groups and $\mathbb{G}_F$ is a set equipped with the binary operation $\circ_F$,*

- $\forall (S, pk), (S', pk') \in (\mathbb{G}_S, *_S) \times (\mathbb{G}_{pk}, *_{pk})\colon F(S *_S S', pk *_{pk} pk') = F(S, pk) \circ_F F(S', pk')$.

and an I such that $F(S, pk) = I$.

3. *The* `retrieve` *algorithm inputs a candidate tuple* (S, R, pk) *(satisfying the above conditions) and vk, and outputs a key* $\widetilde{pk}$ *and a valid certificate* $\widetilde{cert}_{\widetilde{pk}}$ *on it w.r.t. vk.*

Informally, this class includes signature schemes where the signature on a given message can be converted into a "simulatable" part (denoted by R in the definition) that does not reveal any information about the signature or the message to be signed (denoted by pk), and a "vital" part (denoted by S) such that S and pk form a preimage, by a homomorphic map F, of some quantity I computed only from R and the public parameters. The last condition dictated by the `retrieve` algorithm guarantees the non-triviality of the map F; given (S, R, pk) satisfying $F(S, pk) = I$ (I computed as prescribed by the definition), one can come up with a pair of a message and a valid signature on it w.r.t. the same verification key.

Signatures from this class accept efficient proofs of knowledge of a signature/certificate $cert_{pk}$ on a *hidden* message (pk). In fact, the prover will first convert $cert_{pk}$ into (S, R), then reveal R to the verifier, and finally prove knowledge of the preimage of I (computed as in the definition) via the map F. The last proof can be efficiently carried out thanks to the homomorphic property of F. Moreover, existence of the `retrieve` algorithm guarantees the special soundness (SpS) property of the protocol. Finally, this class accepts many instantiations for instance the signature in [1]. See Appendix E.1 for the details.

Definition 2 (The class $\mathbb{R}$ of relations). *$\mathbb{R}$ is the set of relations $\mathcal{R}$ such that there exists an algorithm which inputs an instance x from the set of instances $\mathbb{G}_x$ (in addition to the public parameters) and outputs a description of a map $F_{\mathcal{R}}\colon (\mathbb{G}_w, *_w) \to (\mathbb{G}_{\mathcal{R}}, \circ_{\mathcal{R}})$:*

- *where $\mathbb{G}_w$ is the set of witnesses which is a group for $*_w$, and $\mathbb{G}_{\mathcal{R}}$ is a set equipped with the binary operation $\circ_{\mathcal{R}}$,*
- $\forall w, w' \in \mathbb{G}_w, : F_{\mathcal{R}}(w *_w w') = F_{\mathcal{R}}(w) \circ_{\mathcal{R}} F_{\mathcal{R}}(w')$.

and an $I_{\mathcal{R}}$ such that $F_R(w) = I_{\mathcal{R}} \Leftrightarrow (x, w) \in \mathcal{R}$.

Examples of such functions include the discrete logarithm or the Diffie-Hellman function. Likewise, one can prove knowledge of a witness corresponding to a given instance thanks to the homomorphic property of $F_{\mathcal{R}}$.

Definition 3 (The class $\mathbb{E}_1$ of encryption schemes). *$\mathbb{E}_1$ is the set of tag-based encryption (TBE) schemes Γ that have the following properties:*

1. *The message space $\mathbb{G}_w$ and the public key space $\mathbb{G}_{pk}$ are groups with respect to $*_w$ and $*_{pk}$ respectively.*
2. *Let $w \in \mathbb{G}_w$ be a message and e its encryption with respect to a tag t under a public key pk. On the common input pk, w, e, and t, there exists an efficient zero knowledge proof of w being the decryption of e with respect to the key pk and the tag t. The private input of the prover is the randomness used to produce the encryption e.*
3. *Given an encryption e of some message under some public key w.r.t. a given tag t, there exists an efficient algorithm* `compute` *which inputs e and outputs a public key $pk' \in \mathbb{G}_{pk}$, a message w', and its encryption $e' = \Gamma.\mathtt{encrypt}_{pk'}(w', t)$, under the key pk' w.r.t. the same tag t, such that:*
   - *The probability distributions of the random variables $pk' \in \mathbb{G}_{pk}$ and $w' \in \mathbb{G}_w$ are indistinguishable from uniform, where the probability is taken over the ciphertext e, the tag t, and the random coins of* `compute`.
   - *One can define a group operation $\circ_e$ on the set*

$$\mathcal{E} = \{e'\colon (pk', w', e') \leftarrow \Gamma.\mathtt{compute}(e, t)\}$$

such that $\Gamma.\mathtt{encrypt}_{pk' *_{pk} pk}(w' *_w w, t) = e' \circ_e e$, where w and pk are the message and public key underlying the encryption e respectively. Moreover, given the randomnesses used to produce e and e', one can deduce (using only the public parameters) the randomness used to produce $e' \circ_e e$ on $w' *_w w$ under the key $pk' *_{pk} pk$.

The class $\mathbb{E}_1$ informally comprises encryption schemes that possess efficient proofs of correctness of decryption (i.e. proofs that a given ciphertext correctly decrypts to a given message) in addition to being homomorphic w.r.t. both the message and the public key. This might seem restrictive at a first glance, however, it turns out that the ElGamal-based family of encryption schemes satisfy nicely the properties required in the above definition. We note as an illustration the tag-based variant of the modified Cramer-Shoup scheme [11] described in Figure 3 (we provide the analysis of this class in Appendix E.2).

Definition 4 (The class $\mathbb{H}$ of functions). *$\mathbb{H}$ is the set of functions $H\colon \mathbb{G}_{pk} \to \mathbb{G}_H$ such that:*

- *$\mathbb{G}_{pk}$ is a group w.r.t. some binary operation $*_{pk}$, and $\mathbb{G}_H$ is a set equipped with a binary operation $*_H$.*
- *$\forall pk, pk' \in \mathbb{G}_H\colon H(pk *_{pk} pk') = H(pk) *_H H(pk')$.*

It is natural to require that $H(pk)$ for some public key pk in *database* identifies pk uniquely; i.e. there are no different pk and pk' in *database* that are mapped to the same value by the function H. In this sense, requiring H to be collision-resistant seems natural, however we remark that in our application of group encryption, GM has some control over the public keys she certifies, and therefore may proceed to simple measures (see the instantiation in Appendix E.3) in case a collision occurs.

Definition 5 (The class $\mathbb{E}_2$ of encryption schemes). *$\mathbb{E}_2$ is the set of tag-based encryption schemes Γ that have the following properties:*

1. *The message space is a group $\mathbb{G}_H$ w.r.t. some binary operation $*_H$ and the ciphertext space C is a set equipped with some binary operation $\circ_c$.*
2. *Let $h \in \mathbb{G}_H$ be a message and e its encryption with respect to a given key pk and a given tag t. On the common input pk, t, h, and e, there exists an efficient zero knowledge proof of h being the decryption of e with respect to t under the key pk. The private input of the prover is the randomness used to produce the encryption e.*
3. *$\forall\ h, h' \in \mathbb{G}_H$, $\forall\ pk$, $\forall t\colon \Gamma.\mathtt{encrypt}_{pk}(h *_H h', t) = \Gamma.\mathtt{encrypt}_{pk}(h, t) \circ_c \Gamma.\mathtt{encrypt}_{pk}(h', t)$. Moreover, given the randomness used to encrypt h in $\Gamma.\mathtt{encrypt}_{pk}(h, t)$ and h' in $\Gamma.\mathtt{encrypt}_{pk}(h', t)$, one can deduce (using only the public parameters) the randomness used to produce $\Gamma.\mathtt{encrypt}_{pk}(h, t) \circ_c \Gamma.\mathtt{encrypt}_{pk}(h', t)$ on $h *_H h'$.*

Examples of encryption schemes in the above class include Kiltz' [24] and Cash *et al.*'s [11] (described in Figure 8) tag-based encryption schemes. In fact, the encryption algorithm in both schemes is homomorphic w.r.t. the message, and both schemes accept efficient proofs of the correctness of a decryption, namely the proof of knowledge of discrete logarithms which satisfy known predicates (linear relations). The analysis of this class is provided in Appendix E.4.

4.2 The `prove` protocol

In this paragraph, we instantiate the construction in Section 3 with the following constituents:

1. A signature scheme Σ from Class $\mathbb{S}$ with key pair $(sk_{\mathsf{GM}}, pk_{\mathsf{GM}})$, and corresponding function $F : (\mathbb{G}_S, *_S) \times (\mathbb{G}_{pk}, *_{pk}) \to (F(\mathbb{G}_S \times \mathbb{G}_{pk}), \circ_F)$.
2. An encryption scheme Γ_1 from Class $\mathbb{E}_1$ with public key space $(\mathbb{G}_{pk}, *_{pk})$, message space $(\mathbb{G}_w, *_w)$, and with ciphertext subset $(\mathcal{E}, \circ_e)$ (as defined in Definition 3).
3. A relation $\mathcal{R}$ from Class $\mathbb{R}$ with instance space $\mathbb{G}_x$ and witness space $(\mathbb{G}_w, *_w)$.
4. A function H from Class $\mathbb{H}$ with domain $(\mathbb{G}_{pk}, *_{pk})$ and codomain $(\mathbb{G}_H, *_H)$.
5. An encryption scheme Γ_2 from Class $\mathbb{E}_2$ with key pair $(sk_{\mathsf{OA}}, pk_{\mathsf{OA}})$, message space $(\mathbb{G}_H, *_H)$, and ciphertext space $(C, \circ_c)$.

Theorem 3. *The* `prove` *protocol depicted in Figure 2 is an efficient zero knowledge proof of knowledge with the SpS property.*

Theorem 4 (Soundness of the construction in Section 3). *The construction in Section 3 is sound if the* `prove` *protocol satisfies the SpS property, and the used certification scheme is EUF-CMA secure.*

The proofs are provided in Appendix E.5 and Appendix E.6 respectively.

Prover P		Verifier V
		Compute $I_{\mathcal{R}}$ as in Definition 2
Run $\texttt{convert}(pk_{\mathsf{GM}}, pk, cert_{pk})$ to get (S, R) Compute $(pk', w', e'_1) \leftarrow \Gamma_1.\texttt{compute}(e_1)$ Choose $S' \xleftarrow{R} \mathbb{G}_S$ Compute $f' = F(S', pk')$ Compute $f'_{\mathcal{R}} = F_{\mathcal{R}}(w')$ Compute $h' = H(pk')$ Compute $e'_2 = \Gamma_2.\texttt{encrypt}_{pk_{\mathsf{OA}}}(h')$		
	$\xrightarrow{\quad R, f', e'_1, f'_{\mathcal{R}}, e'_2 \quad}$	
		Compute I as in Definitions 1
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{z_S = S' *_S S^b, z_{pk} = pk' *_k pk^b}$	
	$\xrightarrow{z_w = w' *_w w^b, z_h = h' * h^b}$	
	$\xleftrightarrow{\mathsf{PoK}_1\{e'_1 \circ_e e_1^b = \Gamma.\texttt{encrypt}_{z_{pk}}(z_w, t)\}}$	
	$\xleftrightarrow{\mathsf{PoK}_2\{e'_2 \circ_c e_2^b = \Gamma.\texttt{encrypt}_{pk_{\mathsf{OA}}}(z_h, t)\}}$	
		Accept if: $F(z_S, z_{pk}) = f' \circ_F I^b$, $F_{\mathcal{R}}(z_w) = f'_{\mathcal{R}} \circ_{\mathcal{R}} I^b_{\mathcal{R}}$, $H(z_{pk}) = z_h$, PoK_1 and PoK_2 are valid.

Fig. 2. Proof system for membership to the language $\{(w, pk, cert_{pk})\colon e_1 = \Gamma_1.\texttt{encrypt}_{pk}(w, t) \wedge e_2 = \Gamma_2.\texttt{encrypt}_{pk_{\mathsf{OA}}}(H(pk), t) \wedge cert_{pk} = \Sigma.\texttt{sign}_{sk_{\mathsf{GM}}}(pk) \wedge (x, w) \in \mathcal{R}\}$ Common input: $(e_1, e_2, t, x, pk_{\mathsf{OA}}, pk_{\mathsf{GM}})$ and Private input: $(w, pk, cert_{pk})$ and randomness used to produce e_1 and e_2.

4.3 A concrete realization

In this subsection, we consider bilinear groups with $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Moreover, we instantiate this system with the Λ_{sxdh} setting which refers to the case of asymmetric pairings for which the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

assume that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

We instantiate the construction in Section 3 with the following bricks:

1. The signature scheme from [1]. The scheme signs in particular messages in $\mathbb{G}_2^4$.
2. The encryption scheme described in Figure 3 to instantiate both Γ^{User} and Γ^{OA}. The message space of both Γ^{User} and Γ^{OA} is the group $\mathbb{G}_2$ (we provide in Appendix F the security analysis of this scheme).
3. The relation $\mathcal{R}$: $(x = [X, Y], w) \in \mathcal{R} \leftarrow e(X, Y) = e(g, w)$, where g is a known generator of $\mathbb{G}_1$.
4. The function H mapping an element $(X_1, \ldots, X_4) \in \mathbb{G}_2^4$ to $\prod_{i=1}^4 X_i^{a_i}$, where $a_1, \ldots, a_4$ are public elements in $\mathbb{Z}_d$ (d is order of $\mathbb{G}_2$). We provide in Appendix E.3 the analysis of the collision-resistance of H.
5. The one-time signature from [20].

The ciphertext will then comprise 13 group elements, which amounts to 0.4 kB if we consider an implementation using a 256-bit group size. This is definitely more compact than the realization in [12] (1.25 kB using a 256-bit group order) or [23] (2.5 kB using 1024-bit moduli).

Moreover, the `prove` procedure can be carried out interactively or non-interactively.

We summarize in this chart the performances of our realization compared to those of [23] and [12] (see Appendix G and Appendix H for the details).

[Setup]	Choose a group $(\mathbb{G}, \cdot)$ generated by g with prime order d.
[Key generation]	Choose $x_1, \widetilde{x}_1, x_2, \widetilde{x}_2 \xleftarrow{R} \mathbb{Z}_d$ then compute $X_i \leftarrow g^{x_i}$ and $\widetilde{X}_i \leftarrow g^{\widetilde{x}_i}$ for $i = 1, 2$ set $pk \leftarrow \{X_i, \widetilde{X}_i\}_{i=1,2}$ and $sk \leftarrow \{x_i, \widetilde{x}_i\}_{i=1,2}$.
[Encryption]	For a message $m \in \mathbb{G}$ and a tag $t \in \mathbb{Z}_d$: choose $r \xleftarrow{R} \mathbb{Z}_d$, compute $c_1 \leftarrow g^r$, $c_2 \leftarrow (X_1^t \widetilde{X}_1)^r$, $c_3 \leftarrow (X_2^t \widetilde{X}_2)^r$, and $c_4 = mX_1^r$, set the ciphertext to (c_1, c_2, c_3, c_4).
[Decryption]	Given a ciphertext $c = (c_1, c_2, c_3, c_4)$ and a tag t: check that $c_2 = c_1^{tx_1 + \widetilde{x}_1}$ and that $c_3 = c_1^{tx_2 + \widetilde{x}_2}$ if it is not the case, return $\perp$, otherwise: compute the plaintext as $m \leftarrow c_4 c_1^{-x_1}$.

Fig. 3. TBE variant of the Modified Cramer-Shoup [11]

	[23]	[12]	Our scheme
Ciphertext size (kB)	2.5	1.25	0.4
Interactive Proof size (kB)	70	-	1
Non-interactive Proof size (kB)	-	16.125	2
Interactive proof verification cost (# of pairings computations)	0	-	14
Non-interactive proof verification cost (# of pairings computations)	-	3895	325

Fig. 4. Comparison

Remark 3. There is a technique, called batch verification [2], which reduces the cost of verification of GS proofs at the expense of soundness. Applying this technique to our non-interactive proof reduces the required number of pairings to approximately 81 (vs 974 pairings for the proof in [12]). It is also worth-noting that the group encryption scheme in [12] would only need 1182 (295 using the batch technique) pairings if instantiated with the certification scheme in [1], which remains still higher than the 325 (81) pairings required in our proof. Besides, the realization in [23] which has the merit of not requiring any pairings in its (interactive) proof does comprise however an expensive computation which consists in repeating 50 times a sub-protocol in order to reduce the knowledge error.

5 Summary and Perspectives

We presented a new security model for group encryption which captures all the required security features (anonymity, message security and soundness) in a weaker attack model, namely selective-label weak chosen ciphertext attacks (sl-wCCA). We also proposed a construction for sl-wCCA secure group encryption from weakly secure components (st-wCCA secure encryption). Our construction departs from the previous constructions in encrypting only an *alias* or *hash* of the recipient's public key instead of the entire key, and thus has the merit of avoiding the linear dependency between the key and its encryption. We then applied a method reminiscent of the Canetti-Halevi-Katz paradigm in order to upgrade the security of the resulting construction to full adaptive CCA security. This detour made to get fully secure group encryption allows to use homomorphic (tag-based) encryption as building blocks which impacts positively the `prove` procedure. In fact, we define classes of building blocks that lead to group encryption schemes where `prove` is an efficient Σ protocol. The advantages of our technique are manifest in the realization we provide as we achieve efficient and shorter ciphertexts. Moreover, our proofs are also shorter and can be carried out with or without interaction with the verifier.

A future direction of research consists in developing more efficient proofs (interactive or non-interactive) of knowledge of a certified public key, since this is the main bottleneck in group encryption design. So far, we resorted to structure-preserving signatures to achieve this goal. One could also envisage further solutions.

Proofs of set membership The public keys along with their certificates are published in a public database. Therefore, one could simply prove the membership of the public key in question (committed to in a public value) to this database. Camenisch *et al.* [8] presented an efficient solution using as sub-protocol the proof of knowledge of a signature on a hidden message; the instantiation they provide uses Boneh-Boyen [4]'s signature as the considered messages are 160-bit exponents (further instantiations using [9, 14] signatures or cryptographic accumulators [9] are also possible). Obviously, if we use this technique in our case, then we will have again recourse to structure-preserving signatures as the considered messages lie in a group. Therefore, finding new efficient techniques for set membership proofs, where the set elements lie in a group, would have a direct impact on group encryption.

Signatures on randomized ciphertexts Blazy *et al.* [3] introduced a new primitive which allows to, given a signature on a ciphertext, produce a new ciphertext on the same message along with a fresh signature on it (the new ciphertext). They give a concrete realization of this primitive using Waters' [30] signature and ElGamal's [16] or the linear encryption [5]. The scheme satisfies a reasonable security notion (difficulty to come up with a valid signature on an encryption of a *new* message) under standard assumptions (CDH or slightly stronger variants), however it suffers the high verification cost (due to the recourse to Groth-Sahai's proof system) in addition to the consequent size of the ciphertexts and corresponding signatures.

We could use this primitive in our application of group encryption; GM will publish the public keys together with their encryptions and signatures on these encryptions using this mechanism. GM will further publish the coins used to encrypt the public key in order to convince of the well-formedness of the ciphertext. In this case, the sub-protocol of `prove` which proves knowledge of the certified public key will consist of a randomization of the pair (ciphertext,signature). This solution presents the advantage of having a security under standard assumptions (compared to the assumptions underlying the proposed structure-preserving signatures), however, it is afflicted with the expensive aforementioned cost and bandwidth. Therefore, any significant improvement in this primitive will translate immediately in an improvement in group encryption design.

References

1. Masayuki Abe, Georg Fuchsbauer, Jen Groth, Kristian Haralambiev, and Miyako Ohkubo. Structure-Preserving Signatures and Commitments to Group Elements. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223, pages 209–236. Springer, 2010.
2. Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In J. Zhou and M. Yung, editors, *Applied Cryptography and Network Security (ACNS 2010)*, volume 6123, pages 218–235. Springer, 2010.
3. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In D. Catalano et al., editors, *Public Key Cryptography – PKC 2011*, volume 6571, pages 403–422. Springer, 2011.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027, pages 56–73. Springer, 2004.
5. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In M. K. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152, pages 41–55. Springer, 2004.
6. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.
7. Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3027, pages 87–103. Springer, 2005.
8. Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In J. Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350, pages 234–252. Springer, 2008.

9. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Advances in Cryptology − CRYPTO 2002*, volume 2442, pages 61–76. Springer, 2002.
10. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology − EUROCRYPT 2004*, volume 3027, pages 207–222. Springer, 2004.
11. David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009. Earlier version in EUROCRYPT 2008.
12. Julien Cathalo, Benoît Libert, and Moti Yung. Group encryption: Non-interactive realization in the standard model. In M. Matsui, editor, *Advances in Cryptology − ASIACRYPT 2009*, volume 5912, pages 179–196. Springer, 2009.
13. David Chaum and Eugène van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology − EUROCRYPT '91*, volume 547, pages 257–265. Springer, 1991.
14. Benoît Chevallier-Mames and Marc Joye. A practical and tightly secure signature scheme without hash function. In M. Abe, editor, *Topics in Cryptology − CT-RSA 2007*, volume 4377, pages 347–356. Springer, 2007.
15. Laila El Aimani. Anonymity from public key encryption to undeniable signatures. In B. Preneel, editor, *Progress in Cryptology − AFRICACRYPT 2009*, volume 5580, pages 217–234. Springer, 2009.
16. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.
17. G. Fuchsbauer. Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. Cryptology ePrint Archive, Report 2009/320, 2009. http://eprint.iacr.org/.
18. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
19. J. Groth. Homomorphic Trapdoor Commitments to Group Elements. *IACR Cryptology ePrint Archive*, 2009:7, 2009.
20. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *Advances in Cryptology − ASIACRYPT 2006*, volume 4284, pages 444–459. Springer, 2006.
21. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *Advances in Cryptology − EUROCRYPT 2008*, volume 4965, pages 415–432. Springer, 2008.
22. Shai Halevi. A sufficient condition for key privacy. IACR Cryptology ePrint Archive, Report 2005/005, 2005. http://eprint.iacr.org/.
23. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group encryption. In K. Kurosawa, editor, *Advances in Cryptology − ASIACRYPT 2007*, volume 4833, pages 181–199. Springer, 2007.
24. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *Theory of Cryptography (TCC 2006)*, volume 3876, pages 581–600. Springer, 2006.
25. Joseph K. Liu, Patrick P. Tsang, and Duncan S. Wong. Efficient verifiable ring encryption for ad hoc groups. In R. Molva et al., editors, *Security and Privacy in Ad-hoc and Sensor Networks (ESAS 2005)*, volume 3813, pages 1–13. Springer, 2005.
26. Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications. In M. Naor, editor, *Theory of Cryptography (TCC 2004)*, volume 2951, pages 171–190. Springer, 2004.
27. Bo Qin, Qianhong Wu, Willy Susilo, and Yi Mu. Publicly verifiable privacy-preserving group decryption. In M. Yung et al., editors, *Information Security and Cryptology (Inscrypt 2008)*, volume 5487, pages 72–83. Springer, 2008.
28. Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. IACR Cryptology ePrint Archive, Report 2007/074, 2007. http://eprint.iacr.org/.
29. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002. Earlier version in EUROCRYPT 1998.
30. Brent Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Advances in Cryptology − EUROCRYPT 2005*, volume 3494, pages 114–127. Springer, 2005.

A Building Blocks

A.1 Signature schemes

A signature scheme Σ comprises three algorithms, namely the key generation algorithm `keygen`, the signing algorithm `sign`, and the verification algorithm `verify`. The standard security notion for a signature scheme is existential unforgeability under chosen message attacks (EUF-CMA), which was introduced in [18]. Informally, this notion refers to the hardness of, given a signing oracle, producing a valid pair of message and corresponding signature such that message has not been queried to the signing oracle. There exists also the stronger notion, SEUF-CMA (strong existential unforgeability under chosen message attack), which allows the adversary to produce a forgery on a previously queried message, however the corresponding signature must not be obtained from the signing oracle.

A.2 Tag-based encryption (TBE)

Tag-based encryption, also referred to as encryption with labels, was first introduced in [29]. In these schemes, the encryption algorithm takes as input, in addition to the public key pk and the message m intended to be encrypted, a tag t which specifies information related to the message m and its encryption context. Similarly, the decryption algorithm takes additionally to the ciphertext and the private key the tag under which the ciphertext was created. Security notions are then defined as usual except that the adversary specifies to his challenger the tag to be used in the challenge ciphertext, and in case he (the adversary) is allowed to query oracles, then he cannot query them on the pair formed by the challenge ciphertext and the tag used to form it. There are also weakened security models for this type of encryption where the adversary specifies the challenge tag before getting the parameters of the scheme, and during the game, he (the adversary) is not allowed to query the allowed oracles w.r.t. the challenge tag; we talk in this case about selective tag security. We specify in the following the formal definitions of IND-st-wCCA and ANO-st-wCCA security for tag-based encryption:

Definition 6 (IND-st-wCCA indistinguishability). *Let Γ be a tag-based encryption scheme scheme. Let further $\mathcal{A}$ denote a PPTM. We consider the following random experiment:*

$\boxed{\textit{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND-st-wCCA}}(1^\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $t \leftarrow \mathcal{A}(param)$;
3. $(sk, pk) \leftarrow \Gamma.\mathtt{keygen}(param, 1^\kappa)$;
4. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(pk)$;
5. $b \xleftarrow{R} \{0,1\}$; $e_b \leftarrow \Gamma.\mathtt{encrypt}_{pk}(m_b, t)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(e_b)$;
7. *If* $b = b^\star$ *return* 1 *else* 0.

We define the advantage *of $\mathcal{A}$* via*:*

$$\mathbf{Adv}_{\Gamma,\mathcal{A}}^{\textit{ind-st-wCCA}}(1^\kappa) = \left|\Pr\left[\mathbf{Exp}_{\Gamma,\mathcal{A}}^{\textit{ind-st-wCCA}-b}(1^\kappa) = b\right] - \frac{1}{2}\right|.$$

Given $(t, q_d) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a (t, ε, q_d)-ind-st-wCCA adversary against Γ if, running in time t and issuing q_d decryption queries, $\mathcal{A}$ has $\mathbf{Adv}_{\Gamma,\mathcal{A}}^{\mathsf{ind-st-wCCA}}(1^\kappa) \geq \varepsilon$. The scheme Γ is said to be (t, ε, q_d)-ind-st-wCCA secure if no (t, ε, q_d)-ind-st-wCCA adversary against it exists.

Definition 7 (ANO-st-wCCA anonymity). *Let Γ be a tag-based encryption scheme scheme. Let further $\mathcal{A}$ denote a PPTM. We consider the following random experiment:*

$\boxed{\textit{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{ANO-st-wCCA}}(1^\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $t \leftarrow \mathcal{A}(param)$;
3. $(sk_0, pk_0) \leftarrow \Gamma.\mathtt{keygen}(param, 1^\kappa)$; $(sk_1, pk_1) \leftarrow \Gamma.\mathtt{keygen}(param, 1^\kappa)$
4. $m \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk_0,.),\mathtt{decrypt}^{\neg(-,t)}(sk_1,.)}(pk_0, pk_1)$;
5. $b \xleftarrow{R} \{0,1\}$; $e_b \leftarrow \Gamma.\mathtt{encrypt}_{pk_b}(m, t)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk_0,.),\mathtt{decrypt}^{\neg(-,t)}(sk_1,.)}(e_b)$;

7. *If $b = b^\star$ return* 1 *else* 0.

We define the advantage *of $\mathcal{A}$* via*:*

$$\mathbf{Adv}^{\textsf{ind-st-wCCA}}_{\Gamma,\mathcal{A}}(1^\kappa) = \left|\Pr\left[\mathbf{Exp}^{\textsf{ind-st-wCCA}-b}_{\Gamma,\mathcal{A}}(1^\kappa) = b\right] - \frac{1}{2}\right|.$$

Given $(t, q_d) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a (t, ε, q_d)-ind-st-wCCA adversary against Γ if, running in time t and issuing q_d decryption queries, $\mathcal{A}$ has $\mathbf{Adv}^{\textsf{ind-st-wCCA}}_{\Gamma,\mathcal{A}}(1^\kappa) \geq \varepsilon$. The scheme Γ is said to be (t, ε, q_d)-ind-st-wCCA secure if no (t, ε, q_d)-ind-st-wCCA adversary against it exists.

A.3 One-time signatures

One-time digital signature schemes can be used to sign, at most, one message; otherwise, signatures can be forged. A new public key is required for each message that is signed. One-time signature schemes have the advantage that signature generation and verification are very efficient. This is due to the fact that they rely on one-way functions without trapdoors. They are, similarly to normal digital signatures, defined by the key generation algorithm `keygen`, the signing algorithm `sign`, and the verification algorithm `verify`. The strong security for one-time signatures is defined as follows.

Definition 8. *Let $\Sigma = (\texttt{keygen}, \texttt{sign}, \texttt{verify})$ be a one-time signature scheme, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, where κ is a security parameter:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{strongsecurity}}_{\Sigma,\mathcal{A}}(\kappa)}$

1. $(pk, sk) \leftarrow \Sigma.\texttt{keygen}(\kappa)$
2. $(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathfrak{S}}(pk)$
   $\mathfrak{S} : m \longmapsto \sigma = \Sigma.\texttt{sign}_{sk}(m)$
3. *return 1 if and only if the following properties are satisfied:*
   - $\Sigma.\texttt{verify}_{pk}[\sigma^\star, m^\star] = \{1\}$
   - $(m, \sigma) \neq (m^\star, \sigma^\star)$ *($\mathfrak{S}$ is invoked on at most one message m)*

We define the success *of $\mathcal{A}$* via*:*

$$\mathbf{Succ}^{\textit{strong security}}_{\Sigma,\mathcal{A}}(\kappa) = \Pr\left[\mathbf{Exp}^{\textit{strong security}}_{\Sigma,\mathcal{A}}(\kappa) = 1\right].$$

Given $t \in \mathbb{N}$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is said to be a (t, ε) strong adversary against Σ if, running in time t, $\mathcal{A}$ has $\mathbf{Succ}^{\textit{strong security}}_{\Sigma,\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme Σ is called (t, ε) strong one-time secure secure if no (t, ε) strong adversary against it exists.

A.4 Proofs of Knowledge (PoK)

A proof of knowledge is modeled by an interactive proof system, first introduced by Goldwasser, Micali and Rackoff GoldwasserMicaliRackoff1989. Such a system informally consists of a prover P trying to convince a verifier V that an instance x belongs to a language L. x refers to the common input whereas $(P, V)(x)$ denotes the proof instance carried between P and V at the end of which V is (not) convinced with the membership of the alleged instance x to L:

$$(P, V)(x) \in \{\textsf{Accept}, \textsf{Reject}\}$$

P is modeled by a probabilistic Turing machine whereas V is modeled by a *polynomial* probabilistic Turing machine. During $(P, V)(x)$, the parties exchange a sequence of messages called the proof transcript.

A proof of knowledge should satisfy **completeness** which denotes the property of successfully running the protocol is both parties are honest. A further property required in proofs of knowledge is **soundness** notion which captures the inability of a cheating prover P to convince the verifier V with an invalid statement.

ZK proofs of knowledge. Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **zero knowledge** if for every $x \in L$, the proof transcript $(P, V)(x)$ can be produced by a probabilistic polynomial-time algorithm (in the size of the input) S with indistinguishable probability distributions.

Σ protocols. A *public-coin protocol* is a protocol in which the verifier chooses all its messages randomly from publicly known sets. A *three-move protocol* can be written in a canonical form in which the messages sent in the three moves are often called commitment, challenge, and response. The protocol is said to have the *honest-verifier zero-knowledge property (HVZK)* if there exists an algorithm that is able, provided the verifier behaves as prescribed by the protocol, to produce, without the knowledge of the secret, transcripts that are indistinguishable from those of the real protocol. The protocol is said to have the *special soundness property (SpS) property* if there exists an algorithm that is able to extract the secret from two accepting transcripts of the protocol with the same commitment and different challenges. Finally, a three-move public-coin protocol with both the HVZK and SpS properties is called a *Σ protocol*.

A.5 Groth-Sahai's proof system

The Groth-Sahai (GS) proof system gives efficient non-interactive zero knowledge (NIZK) and non-interactive witness-indistinguishable (NIWI) proofs for algebraic statements involving elements in groups equipped with a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \leftarrow \mathbb{G}_T$ (we assume that), e.g.

A pairing-product equation over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$

$$\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, Y_i)^{\gamma_{i,j}} = t_T,$$

for constants $A_1, \ldots, A_n \in \mathbb{G}_1$, $B_1, \ldots, B_m) \in \mathbb{G}_1$, and $\Gamma = (\gamma_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le n}} \in \mathbb{Z}_d^{m \times n}$, and $t_T \in \mathbb{G}_T$.

A multi-scalar multiplication over variables $y_1, \ldots, y_m \in \mathbb{Z}_d$ and $X_1, \ldots, X_m \in \mathbb{G}_1$

$$\prod_{i=1}^{n} A_i^{y_i} \cdot \prod_{i=1}^{m} X_i^{b_i} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} X_j^{y_i \gamma_{i,j}} = T,$$

for constants $A_1, \ldots, A_n \in \mathbb{G}_1$, $b_1, \ldots, b_m \in \mathbb{Z}_d$, $\Gamma \in \mathbb{Z}_d^{m \times n}$, and $T \in \mathbb{G}_1$. A multi-scalar multiplication equation over variables in $\mathbb{G}_2$ is defined analogously.

A quadratic equation over variables $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ in $\mathbb{Z}_d$

$$\sum_{i=1}^{n} a_i y_i + \sum_{i=1}^{m} x_i b_i + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{i,j} x_i y_j = t,$$

For constants $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ in $\mathbb{Z}_d$, constant matrix $\Gamma \in \mathbb{Z}_d^{m \times n}$, and $t \in \mathbb{Z}_d$.

The principle of the GS proof system consists in using the setup parameters and the common reference string (CRS) to commit to the witness components, then generate proof elements that these values committed to satisfy the equations underlying the statement to be proved. Note that proofs for pairing-product equations are only (WI) when $t_T \in \mathbb{G}_T$ has no particular structure. They can be however transformed into ZK proofs (at the expense of an additional cost) if t_T is equal to $1_{\mathbb{G}_T}$ or its representation as a pairing product equation is known.

The GS proof system could be instantiated under different (mild) assumptions, in particular the Λ_{sxdh} setting which refers to the case of asymmetric pairings for which the DDH assumptions holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{Z}_d$
Variables $x \in \mathbb{Z}_d, X \in \mathcal{G}_1$	2	0	0
Variables $y \in \mathbb{Z}_d, Y \in \mathcal{G}_1$	0	2	0
Pairing product equations	4	4	0
- Linear equations: $\prod_{i=1}^{n} e(A_i, Y_i) = t_T$	2	0	0
- Linear equations: $\prod_{i=1}^{n} e(X_i, B_i) = t_T$	0	2	0
Multi-scalar multiplication equations in $\mathbb{G}_1$	2	4	0
- Linear equations: $\prod_{i=1}^{n} A_i^{y_i} = T_1$	1	0	0
- Linear equations: $\prod_{i=1}^{n} X_i^{b_i} = T_1$	0	0	2
Multi-scalar multiplication equations in $\mathbb{G}_2$	4	2	0
- Linear equations: $\prod_{i=1}^{n} Y_i^{a_i} = T_2$	0	0	2
- Linear equations: $\prod_{i=1}^{n} B_i^{x_i} = T_2$	0	1	0
Quadratic equations in $\mathbb{Z}_d$	2	2	0
- Linear equations: $\sum_{i=1}^{n} a_i y_i = t$	0	0	1
- Linear equations: $\sum_{i=1}^{m} x_i b_i = t$	0	0	1

Fig. 5. Cost of each variable and equation in terms of group elements from $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{Z}_d$

Pairing product equations	$5m + 3n + 16$
Multi-scalar multiplication equations in $\mathbb{G}_1$	$8m + 2n + 14$
Multi-scalar multiplication equations in $\mathbb{G}_2$	$8n + 2m + 14$
Quadratic equations in $\mathbb{Z}_d$	$8m + 8n + 12$

Fig. 6. Number of pairings per proof verification (m and n refer to the number of different types of involved variables - see [21])

B Soundness of Group Encryption Schemes

The formal definition of soundness involves an oracle $\mathtt{reg}(sk_{\mathsf{GM}}, \cdot)$ that simulates the group manager GM and maintains a repository *database* that maintains the registered public keys along with their certificates. The space of valid ciphertexts is denoted by $\mathcal{L}_{ciphertext}^{x,L,pk_{\mathcal{R}},pk_{\mathsf{GM}},pk_{\mathsf{OA}},pk}$ and is given by

$$\mathcal{L}_{ciphertext}^{x,L,pk_{\mathcal{R}},pk_{\mathsf{GM}},pk_{\mathsf{OA}},pk} = \left\{\mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, w, L) \colon (x, w) \in \mathcal{R} \text{ and } pk \in database\right\}$$

The space of valid public keys is denoted by $\mathcal{L}_{PK}^{param}$. A group encryption scheme satisfies soundness if for any polynomial-time adversary $\mathcal{A}$, the experiment below returns 1 with negligible probability.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{soundness}}(\kappa)$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(pk_{\mathsf{GM}}, sk_{\mathsf{GM}}) \leftarrow \mathtt{keygen}_{\mathsf{GM}}(1^\kappa, param)$; $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}}) \leftarrow \mathtt{keygen}_{\mathsf{OA}}(1^\kappa, param)$;
3. $(aux, pk_{\mathcal{R}}, x, \psi, L) \leftarrow \mathcal{A}^{\mathtt{reg}(sk_{\mathsf{GM}},\cdot)}(param, pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, sk_{\mathsf{OA}})$;
4. $\langle done \mid out\rangle \leftarrow \langle \mathcal{A}(aux), \mathcal{V}(param)\rangle(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, \psi, L)$;
5. If $(out = 0)$ return 0;
6. $pk \leftarrow \mathtt{open}(sk_{\mathsf{OA}}, \psi, L)$;
7. If $(pk \notin database)$ or $(pk \notin \mathcal{L}_{PK}^{param})$ or $(\psi \notin \mathcal{L}_{ciphertext}^{x,L,pk_{\mathcal{R}},pk_{\mathsf{GM}},pk_{\mathsf{OA}},pk})$ return 1 else return 0.

C Proof of Theorem 1

C.1 Message security

Proof. Let $\mathcal{A}$ be a (t, ϵ, q_d)-IND-sl-wCCA against the above construction. We will build a (t, ϵ, q_d)-IND-st-wCCA adversary $\mathcal{B}$ against Γ^{User} by using a sequence of games $\mathsf{Game_i}$ where the first one corresponds to the standard security game for the IND-sI-wCCA property for group encryption, and the last one corresponds to the standard game for the IND-st-wCCA security notion for TBE.

In the rest of the proof, S_i denotes the event that the experiment in $\mathsf{Game_i}$ returns 1.

Game_0. $\mathcal{B}$ plays the real IND-sI-wCCA security game with $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND\text{-}sI\text{-}wCCA}}(\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(aux, pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, L) \leftarrow \mathcal{A}(param)$;
3. $\langle pk, sk, cert_{pk} \mid aux, pk, cert_{pk}\rangle \leftarrow \langle \mathsf{J}_{\mathsf{User}}(param), \mathcal{A}(aux)\rangle(pk_{\mathsf{GM}})$;
4. $(aux, x, w, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(\cdot,L)}(sk,\cdot)}(aux)$; ▷ Find stage
5. If $(x, w) \notin \mathcal{R}$ then abort;
6. $b \xleftarrow{R} \{0,1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}^b_{\mathsf{ror}}(1^\kappa, pk, w, L)$;
7. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi), \mathtt{decrypt}^{\neg(\cdot,L)}(sk,\cdot)}(aux, \psi)$; ▷ Guess stage
8. If $(b = b^\star)$ return 1 else return 0.

In this game, we use the zero knowledge (ZK) simulator $\mathcal{S}$ (of `prove` since it is ZK by assumption) as the prover $\mathcal{P}'$. In fact, $\mathcal{S}$ is more powerful than $\mathcal{P}'$ as it ($\mathcal{S}$) can simulate the prover in the standard `prove` procedure without knowledge of $pk, cert_{pk}$.

By definition, $\Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_1. In this game, $\mathcal{B}$ forwards the label L received from $\mathcal{A}$ (in line 2) to her own challenger as a challenge tag, and receives in return the challenge public key pk. Then, she presents this very public key to $\mathcal{A}$ in line 3 in order to get a certificate.

Clearly $\Pr[S_1] = \Pr[S_0]$.

Game_2. In this game, $\mathcal{B}$ substitutes the decryption oracle in line 4 by her own decryption oracle for Γ^{User}. $\mathcal{B}$ is able to answer perfectly the decryption queries made by $\mathcal{A}$ as they are by definition w.r.t. labels/tags different from the challenge tag L. Therefore $\Pr[S_2] = \Pr[S_1]$.

Game_3. In this game, $\mathcal{B}$ changes the construction of the challenge in line 6 as follows. Upon receipt of the pair (x, w) (with $(x, w) \in \mathcal{R}$), $\mathcal{B}$ chooses a random witness w' from the message space (as done by the oracle $\mathsf{CH}^b_{\mathsf{ror}}$) and forwards $(w_0, w_1) \leftarrow (w', w)$ to her own challenger. She receives as a challenge ψ^b_1, which corresponds to the encryption of w_b for $b \xleftarrow{R} \{0,1\}$ under the label/tag L using the public key pk. $\mathcal{B}$ further encrypts $H(pk)$ in ψ_2 using Γ^{OA} under the challenge label/tag L and hands $\psi_b = (\psi^b_1, \psi_2)$ to $\mathcal{A}$ as a challenge. Note that ψ_b is perfectly indistinguishable from the output of the oracle $\mathsf{CH}^b_{\mathsf{ror}}(1^\kappa, pk, w, L)$. Therefore $\Pr[S_3] = \Pr[S_2]$.

Game_4. In this game, $\mathcal{B}$ substitutes the decryption oracle in line 7 by her own decryption oracle for Γ^{User}. Again, $\mathcal{B}$ is able to answer perfectly the decryption queries made by $\mathcal{A}$ as they are by definition w.r.t. labels different from L. Moreover, $\mathcal{B}$ runs the ZK simulator $\mathcal{S}$ of `prove` instead of the oracle $\mathtt{prove}^b_{\mathcal{P},\mathcal{S}}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi)$. Clearly, this modification does not impact the probability space of S_4 due to the zero-knowledge property of `prove` (the execution of `prove` using the real prover is indistinguishable from that using the simulator).

Therefore $\Pr[S_4] = \Pr[S_3]$.

Finally, when $\mathcal{A}$ answers b', $\mathcal{R}$ will return the same answer to her challenger. $\mathcal{R}$ will succeed in solving her challenge with advantage at least $\Pr[S_4] - \frac{1}{2} = \epsilon$ after making q_d queries to her own decryption oracle. ☐

C.2 Anonymity

Proof. Let $\mathcal{A}$ be a (t, ϵ, q_d)-ANO-sI-wCCA attacker against the construction which is assumed to use a $(t, \epsilon_{\mathsf{sec}}, q_d)$-IND-st-wCCA secure tag-based encryption Γ^{OA}. We will build a $(t, \frac{\epsilon}{2} - \epsilon_{\mathsf{sec}}, q_d)$-ANO-st-wCCA adversary against the underlying encryption Γ^{User} as follows.

Likewise, we will proceed using a sequence of games Game_i and will denote by S_i the event that the experiment in $\mathsf{Game_i}$ returns 1.

Game_0. $\mathcal{B}$ plays the real ANO-sl-wCCA security game of with $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{ANO\text{-}sl\text{-}wCCA}}(\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$; $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}}) \leftarrow \mathtt{keygen}_{\mathsf{OA}}(1^\kappa, param)$;
2. $(aux, pk_{\mathsf{GM}}, L) \leftarrow \mathcal{A}(param)$; $aux \leftarrow \mathcal{A}^{\mathsf{User}(pk_{\mathsf{GM}}),\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot)}(aux, pk_{\mathsf{OA}})$;
   If keys $\neq (pk_0, sk_0, cert_{pk_0}, pk_1, sk_1, cert_{pk_1})$ return 0;
3. $(aux, x, w, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux)$;
4. If $(x, w) \notin \mathcal{R}$ return 0;
5. $b \xleftarrow{R} \{0,1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}^b_{\mathsf{anon}}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_0, pk_1, w, L)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}_{\mathcal{P}(w,coins_\psi,pk_b,cert_{pk_b})}(pk_{\mathsf{GM}},pk_{\mathsf{OA}},pk_{\mathcal{R}},x,L,\psi),\mathtt{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux, \psi)$;
7. If $(b = b^\star)$ return 1 else return 0.

By definition, $\Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_1. In this game, $\mathcal{B}$ substitutes the public keys pk_0, pk_1 in line 2 by those obtained from her own challenger as challenge keys after having communicated the label L as a challenge tag. $\mathcal{B}$ further obtains the certificates on both pk_0, pk_1 from $\mathcal{A}$. Clearly $\Pr[S_1] = \Pr[S_0]$.

Game_2. In this game, $\mathcal{B}$ replaces the decryption oracles in lines 3 and 6 (w.r.t. both keys pk_0, pk_1) by her own decryption oracles which accept all decryption queries with labels different from L. Moreover, $\mathcal{B}$ changes the prover $\mathcal{P}$ in line 6 by the ZK simulator $\mathcal{S}$ of $\mathtt{prove}$. Due to the ZK property of $\mathtt{prove}$, we have $\Pr[S_2] = \Pr[S_1]$.

Game_3. In this game, $\mathcal{B}$ changes the construction of the challenge in line 5 as follows. Upon receipt of (x, w) from $\mathcal{A}$, she will forward it to her own challenger, she obtains in return an encryption ψ_1^b of w under pk_b for some $b \xleftarrow{R} \{0,1\}$. $\mathcal{B}$ further produces an encryption ψ_2^0 of $H(pk_0)$. Finally she hands the pair (ψ_1^b, ψ_2^0) to $\mathcal{A}$ as a challenge ANO-sl-wCCA group encryption.

Let Z denote the event where $b = 0$. Clearly $\Pr[S_3|Z] = \Pr[S_2|Z]$. We claim that any significant distance between Game_3 and Game_2 conditioned on the event $\neg Z$ will give rise to a $(t, \epsilon_{\mathsf{sec}}, q_d)$ IND-st-wCCA adversary $\mathcal{B}'$ against Γ^{OA}. In fact, $\mathcal{B}'$ will execute the standard ANO-sl-wCCA game with $\mathcal{A}$ with the exception of substituting the opening oracle in lines 3, 6 by her own decryption oracle for pk_{OA} (the decryption oracles for pk_0, pk_1 will be simulated internally as $\mathcal{B}'$ knows the corresponding privates keys). $\mathcal{B}'$ gives $H(pk_0), H(pk_1)$ as challenge messages to her Γ^{OA} challenger, and receives an encryption $\psi_2^{b'}$, $b' \xleftarrow{R} \{0,1\}$, of $H(pk_{b'})$. Let b'' be the output of $\mathcal{B}'$. b'' corresponds to the output of Game_2 if $\psi_2^{b'}$ encrypts $H(pk_1)$, and to the output of Game_3 otherwise. By definition:

$$
\begin{aligned}
\mathsf{adv}_{\Gamma^{\mathsf{OA}}}^{\mathsf{IND\text{-}st\text{-}wCCA}}(\mathcal{B}') &\geq |\Pr[b'' = b'] - \frac{1}{2}| \\
&= |\Pr[b'' = 0, b' = 0] + \Pr[b'' = 1, b' = 1] - \frac{1}{2}| \\
&= \frac{1}{2}|\Pr[b'' = 0|b' = 0] + \Pr[b'' = 1|b' = 1] - 1| \\
&= \frac{1}{2}|\Pr[b'' = 1|b' = 0] - \Pr[b'' = 1|b' = 1]| \\
&= \frac{1}{2}|\Pr[S_3|\neg Z] - \Pr[S_2|\neg Z]|
\end{aligned}
$$

It follows that $|\Pr[S_3] - \Pr[S_2]| = |\Pr[S_3|\neg Z] - \Pr[S_2|\neg Z]| = 2\epsilon_{\mathsf{sec}}$ (as $\Pr[S_3|Z] = \Pr[S_2|Z]$).

Now, back to the reduction $\mathcal{B}$. This algorithm will output, to her Γ^{User} ANO-sl-wCCA challenger, the output $b^\star$ returned by $\mathcal{A}$. Note, that $\mathcal{B}$ can simulate the opening oracle in lines 3, 6 internally as she knows sk_{OA}.

Clearly $\mathsf{adv}_{\Gamma^{\mathsf{User}}}^{\mathsf{ANO\text{-}sl\text{-}wCCA}}(\mathcal{B}) \geq \Pr[b = 0]|\Pr[S_3] - \frac{1}{2}| \geq \frac{\epsilon}{2} - \epsilon_{\mathsf{sec}}$. □

C.3 Soundness

Similarly to the proofs in [23, 12], the soundness of the construction follows as a direct consequence of the soundness of the prove procedure and the EUF-CMA security of the underlying signature/certification scheme. We provide further (in Appendix E.6) a simple soundness proof in case the construction is instantiated from the building blocks described in Section 4.

D A Canetti-Halevi-Katz like Paradigm for Group Encryption

D.1 Message security

Proof. Let $\mathcal{A}$ be an IND-CCA adversary against $\mathcal{GE}$. We will build from $\mathcal{A}$ an IND-sl-wCCA adversary $\mathcal{R}$ against the underlying $\mathcal{GE}^\star$. We proceed by using a sequence of games $\mathsf{Game_i}$ where the first one mirrors the standard game for IND-CCA security for group encryption (GE) whereas the last one corresponds to that of IND-sl-wCCA security for GE. We will denote by S_i the event that the experiment in $\mathsf{Game_i}$ returns 1.

Game_0. $\mathcal{R}$ plays the real IND-CCA security game of $\mathcal{GE}$ with $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND\text{-}sl\text{-}wCCA}}(\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $(aux, pk_{\mathsf{GM}}, pk_{\mathsf{OA}}) \leftarrow \mathcal{A}(param)$;
3. $\langle pk, sk, cert_{pk} \mid aux, pk, cert_{pk}\rangle \leftarrow \langle \mathsf{J}_{\mathsf{User}}(param), \mathcal{A}(aux)\rangle(pk_{\mathsf{GM}})$;
4. $(aux, x, w, L, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{decrypt}(sk,\cdot)}(aux)$; ▷ `Find stage`
5. If $(x, w) \notin \mathcal{R}$ then abort;
6. $b \xleftarrow{R} \{0,1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}^b_{\mathsf{ror}}(1^\kappa, pk, w, L)$;
7. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi), \mathtt{decrypt}^{\neg(\psi,L)}(sk,\cdot)}(aux, \psi)$; ▷ `Guess stage`
8. If $(b = b^\star)$ return 1 else return 0.

By definition, $\Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_1. $\mathcal{R}$ replaces *param* (in line 1) by those obtained from her own challenger, say C. She further forwards pk_{GM} and pk_{OA} (obtained from $\mathcal{A}$ in line 2) to C. $\mathcal{R}$ further chooses a suitable (t, ϵ') strongly secure one-time signature scheme $\mathfrak{S}$ and generates a key pair $(\mathfrak{S}.pk, \mathfrak{S}.sk)$. Finally, $\mathcal{R}$ presents $\mathfrak{S}.pk$ to C as a challenge label.
Clearly, $\Pr[S_1] = \Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_2. In this game, $\mathcal{R}$ simulates the decryption oracle decrypt on queries (ψ_i, L_i) (line 4). First parse ψ_i as $(\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i)$. If $\mathfrak{S}.\mathtt{verify}_{vk_i}(\sigma_i, \psi_i^{\mathsf{sl-wCCA}} \| L_i) = 0$, then respond with $\perp$. Otherwise, two cases manifest: either $vk_i \neq \mathfrak{S}.pk$ or not. In the first case, $\mathcal{R}$ requests the decryption of $q_i = (\psi_i^{\mathsf{sl-wCCA}}, vk_i)$ from her challenger whose answer will be forwarded to $\mathcal{A}$. In the second case $\mathcal{R}$ aborts the simulation.
Let F_1 be the event corresponding to $\mathcal{A}$ submitting a valid ciphertext $\psi_i = (\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i)$, w.r.t. label L_i, where $vk_i = \mathfrak{S}.pk$. We have $\Pr[F_1] \leq \epsilon'$, since $\mathcal{R}$ can output $(\psi_i^{\mathsf{sl-wCCA}} \| L_i, \sigma_i)$ as a forgery on $\mathfrak{S}$.
$\Pr[S_2] \geq \Pr[S_1]\Pr[\neg F_1] \geq (1-\epsilon')^{q_d}\Pr[S_1]$.

Game_3. In this game, $\mathcal{R}$ changes the construction of the challenge in line 6. $\mathcal{A}$ outputs $(x, w, pk_{\mathcal{R}})$, such that $(x, w) \in \mathcal{R}$, along with the challenge label L. At that point, $\mathcal{R}$ transfers the former items to C as a challenge, and receives $\psi^{\mathsf{sl-wCCA}}$ as a challenge encryption: $\psi^{\mathsf{sl-wCCA}} = \mathcal{GE}^\star.\mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, w_b, \mathfrak{S}.pk)$ for some $b \xleftarrow{R} \{0,1\}$ such that $w_1 = w$ and w_0 is a random plaintext uniformly chosen from the plaintext space. Upon receipt of the challenge, $\mathcal{R}$ will produce a one-time signature σ on $\psi^{\mathsf{sl-wCCA}} \| L$ using $\mathfrak{S}.sk$. Finally $\mathcal{R}$ will hand the triple $\psi = (\psi^{\mathsf{sl-wCCA}}, \mathfrak{S}.pk, \sigma)$ as a challenge encryption.
Note that ψ is a valid IND-CCA challenge for $\mathcal{GE}$. Moreover, $\mathcal{R}$ can perfectly handle queries to $\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi)$ using her own (external) oracle $\mathtt{prove}^b_{\mathcal{P},\mathcal{P}'}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi^{\mathsf{sl-wCCA}})$.
Therefore $\Pr[S_3] = \Pr[S_2]$.

Game_4**.** In this game, $\mathcal{R}$ simulates the decryption oracle $\mathtt{decrypt}$ in line 7. Consider the decryption query $q_i = [\psi_i, L_i] = [(\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i), L_i]$. Naturally $q_i \neq [\psi, L]$. We distinguish two cases of queries. If $vk_i \neq \mathfrak{S}.pk$, then $\mathcal{R}$ proceeds as usual using her own decryption oracle. Otherwise, $\mathcal{R}$ responds with $\perp$ if the output of $\mathfrak{S}.\mathtt{verify}_{vk_i}(\sigma_i, \psi_i^{\mathsf{sl-wCCA}} \| L_i)$ is 0, and aborts otherwise.

Let F_2 be the event corresponding to $\mathcal{A}$ submitting a valid ciphertext $\psi_i = (\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i)$, w.r.t. label L_i, where $vk_i = \mathfrak{S}.pk$. We distinguish two cases:

1. If $L_i = L$, and thus $\psi_i \neq \psi$ (since $q_i \neq (\psi, L)$), then we have $(\psi_i^{\mathsf{sl-wCCA}}, \sigma_i) \neq (\psi^{\mathsf{sl-wCCA}}, \sigma)$. It follows that $(\psi_i^{\mathsf{sl-wCCA}} \| L_i, \sigma_i) \neq (\psi^{\mathsf{sl-wCCA}} \| L, \sigma)$, which means that $(\psi_i^{\mathsf{sl-wCCA}} \| L_i, \sigma_i)$ is a strong forgery on $\mathfrak{S}$.
2. If $L_i \neq L$, then $\psi_i^{\mathsf{sl-wCCA}} \| L_i \neq \psi^{\mathsf{sl-wCCA}} \| L$. It follows again that $(\psi_i^{\mathsf{sl-wCCA}} \| L_i, \sigma_i) \neq (\psi^{\mathsf{sl-wCCA}} \| L, \sigma)$, which means that $(\psi_i^{\mathsf{sl-wCCA}} \| L_i, \sigma_i)$ is a strong forgery on $\mathfrak{S}$.

Therefore, $\Pr[S_4] \geq \Pr[S_3] \Pr[\neg F_2] \geq (1 - \epsilon')^{q_d} \Pr[S_3]$. Finally, when $\mathcal{A}$ answers b', $\mathcal{R}$ will return the same answer to her challenger. $\mathcal{R}$ will succeed in solving her challenge with advantage at least $\epsilon(1 - \epsilon')^{q_d}$.

□

D.2 Anonymity

Proof. Let $\mathcal{A}$ be an ANO-CCA adversary against $\mathcal{GE}$. We will build from $\mathcal{A}$ an ANO-sl-wCCA adversary $\mathcal{R}$ against the underlying $\mathcal{GE}^\star$. We proceed similarly to the previous proof by using a sequence of games $\mathsf{Game_i}$ where the first one mirrors the standard game for ANO-CCA security for GE and the last one corresponds to that of ANO-sl-wCCA security for GE. We will denote likewise by S_i the event that the experiment in Game_i returns 1.

Game_0**.** $\mathcal{R}$ plays the real ANO-CCA security game of $\mathcal{GE}$ with $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{ANO\text{-}sl\text{-}wCCA}}(\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$; $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}}) \leftarrow \mathtt{keygen}_{\mathsf{OA}}(1^\kappa, param)$;
2. $(aux, pk_{\mathsf{GM}}) \leftarrow \mathcal{A}(param)$; $aux \leftarrow \mathcal{A}^{\mathsf{User}(pk_{\mathsf{GM}}), \mathsf{open}(sk_{\mathsf{OA}}, \cdot)}(aux, pk_{\mathsf{OA}})$;
   If $\mathsf{keys} \neq (pk_0, sk_0, cert_{pk_0}, pk_1, sk_1, cert_{pk_1})$ return 0;
3. $(aux, x, w, L, pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathtt{open}(sk_{\mathsf{OA}}, \cdot),\ \mathtt{decrypt}(sk_0, \cdot),\ \mathtt{decrypt}(sk_1, \cdot)}(aux)$;
4. If $(x, w) \notin \mathcal{R}$ return 0;
5. $b \xleftarrow{R} \{0, 1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}_{\mathsf{anon}}^b(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_0, pk_1, w, L)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}_{\mathcal{P}(w, coins_\psi, pk_b, cert_{pk_b})}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi), \mathtt{open}^{\neg(\psi, L)}(sk_{\mathsf{OA}}, \cdot),\ \mathtt{decrypt}^{\neg(\psi, L)}(sk_0, \cdot),\ \mathtt{decrypt}^{\neg(\psi, L)}(sk_1, \cdot)}(aux, \psi)$;
7. If $(b = b^\star)$ return 1 else return 0.

By definition, $\Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_1**.** $\mathcal{R}$ replaces $param$ and $(pk_{\mathsf{OA}}, sk_{\mathsf{OA}})$ (in lines 1) by those obtained from her own challenger C. She also forwards pk_{GM} (obtained from $\mathcal{A}$ in line 2) to C. $\mathcal{R}$ further generates a key pair $(\mathfrak{S}.pk, \mathfrak{S}.sk)$ for a suitable (t, ϵ') strongly secure one-time signature scheme $\mathfrak{S}$, then presents $\mathfrak{S}.pk$ to C as a challenge tag. Finally $\mathcal{R}$ issues the certificates on the challenge keys pk_0, pk_1 using $\mathcal{A}$.

Clearly, $\Pr[S_1] = \Pr[S_0] = \epsilon + \frac{1}{2}$.

Game_2**.** $\mathcal{R}$ simulates in this game the decryption and opening oracles in line 3. Decryption queries (ψ_i, L_i) with respect to pk_0 or pk_1 are answered as in the message security proof. I.e., first parse ψ_i as $(\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i)$; if $\mathfrak{S}.\mathtt{verify}_{vk_i}(\sigma_i, \psi_i^{\mathsf{sl-wCCA}} \| L_i) = 0$, then respond with $\perp$, otherwise reply with the answer of C to the decryption query $(\psi_i^{\mathsf{sl-wCCA}}, vk_i)$ (w.r.t. the proper decryption oracle) if $vk_i \neq \mathfrak{S}.pk$ and abort otherwise. Opening queries (ψ_i, L_i) are answered similarly, namely $\mathcal{R}$ first checks the consistence of ψ_i (validity of σ_i on $\psi_i^{\mathsf{sl-wCCA}} \| L_i$, w.r.t. vk_i, where $\psi_i = (\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i)$). In case it is not consistent, then respond with $\perp$, otherwise $\mathcal{R}$ will use her own opening oracle in the case where $vk_i \neq \mathfrak{S}.pk$ and aborts the simulation in the opposite one.

Similarly, a (decryption or opening) query causing $\mathcal{R}$ to abort will naturally lead to a forgery on $\mathfrak{S}$. Thus $\Pr[S_2] \geq (1 - \epsilon')^{q_d + q_o} \Pr[S_1]$.

Game_3. In this game, $\mathcal{R}$ changes the construction of the challenge in line 5 as follows. $\mathcal{A}$ will output $(x, w, L, pk_{\mathcal{R}})$, such that $(x, w) \in \mathcal{R}$, as a challenge to $\mathcal{R}$. This latter will forward the same quantities except L to her challenger C. She receives as a challenge encryption $\psi^{\mathsf{sl-wCCA}} = \mathcal{GE}^{\star}.\mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_b, w, \mathfrak{S}.pk)$ where $b \xleftarrow{R} \{0, 1\}$. Construction of $\mathcal{A}$'s challenge is done in a straightforward manner, namely produce a one-time signature σ on $\psi^{\mathsf{sl-wCCA}} \| L$ using $\mathfrak{S}.sk$ and set $\psi = (\psi^{\mathsf{sl-wCCA}}, \mathfrak{S}.pk, \sigma)$ as a challenge encryption. It is easy to see that this challenge is a valid $\mathcal{GE}$ encryption of w under label L using either pk_0 or pk_1. Moreover, $\mathcal{R}$ uses her external $\mathtt{prove}_{\mathcal{P}(w, coins_{\psi^{\mathsf{sl-wCCA}}}, pk_b, cert_{pk_b})}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi^{\mathsf{sl-wCCA}})$ oracle to handle queries to the $\mathtt{prove}_{\mathcal{P}(w, coins_{\psi}, pk_b, cert_{pk_b})}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, L, \psi)$ oracle. Thus $\Pr[S_3] = \Pr[S_2]$.

Game_4. In this game, $\mathcal{R}$ simulates the decryption and opening oracles in line 6. Let $q_i = [\psi_i, L_i] = [(\psi_i^{\mathsf{sl-wCCA}}, vk_i, \sigma_i), L_i]$ be a decryption/opening query. Naturally $q_i \neq (\psi, L)$. Likewise we distinguish two cases. Either $vk_i \neq \mathfrak{S}.pk$ in which case $\mathcal{R}$ proceeds as usual using her own decryption/opening oracles. Or $vk_i = \mathfrak{S}.pk$ in which case $\mathcal{A}$ aborts (we assume that ψ is consistent since otherwise $\mathcal{R}$ can perfectly simulate the real decryption/opening algorithms by rejecting the ciphertext). Similarly, we have $\Pr[S_4] \geq (1 - \epsilon')^{q_d + q_o} \Pr[S_3]$.

Finally, when $\mathcal{A}$ responds with b', $\mathcal{R}$ will forward the same guess to her own challenger. $\mathcal{R}$ will succeed in solving her challenge with advantage at least $\epsilon(1 - \epsilon')^{q_d + q_o}$.

□

E Efficient Instantiations

E.1 The class $\mathbb{S}$ of signatures

Fact 1 *The signature [1] belongs to the class $\mathbb{S}$.*

Proof. Let $cert_{pk} = (z, r, s, t, u, v, w)$ be a signature produced on a message $pk = (X_1, \ldots, X_n)$ w.r.t. a public key $cpk = \{g_z, h_z, g_r, h_r, \{g_i, h_i\}_{i=1}^n\}$. We apply the $\mathsf{SigRand}$ partial randomization algorithm of [1, Subsection 4.4, Partial randomizability] to get the new signature on pk, say $cert'_{pk} = (z, r', s', t', u', v', w')$. $cert'_{pk}$ has the following properties:

1. (s', t', v', w') is information-theoretically independent of $cert_{pk}$ and pk. See [1] for further details. Moreover, we define:
$$I = [A \cdot e(s', t')^{-1}, B \cdot e(v', w')^{-1}]$$
and
$$F\,[z, r, u, (X_1, \ldots, X_n)] = \left[e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^{n} e(g_i, X_i), e(h_z, z) \cdot e(h_u, u) \cdot \prod_{i=1}^{n} e(h_i, X_i) \right]$$
It is easy to check that:
$$F\left[(z, r, u) *_S (z', r', u'), (X_1, \ldots, X_k) *_{pk} (X'_1, \ldots, X'_k)\right] = F\,[z, r, u, (X_1, \ldots, X_n)] \circ_F F\left[z', r', u', (X'_1, \ldots, X'_n)\right]$$
where $\circ_F$ is the component-wise product in $\mathbb{G}_T^2$, $*_S$ is the component-wise product in $\mathbb{G}_2^3$, and $*_{pk}$ is the component-wise product in $\mathbb{G}_2^n$.
We further define $\mathtt{convert}(cpk, pk, cert_{pk}) = \mathsf{SigRand}(cpk, cert_{pk}) = (z, r', s', t', u', v', w')$, $R = (s', t', v', w')$, and $S = (z, r', u')$.
2. Since $(z, r', s', t', u', v', w')$ is itself a signature on pk, then the $\mathtt{retrieve}$ algorithm is nothing but the identity function. □

Theorem 5. *The protocol depicted in Figure 7 is a zero knowledge proof of knowledge with the SpS property, for proving knowledge of a signature, produced using a scheme from Class $\mathbb{S}$ on some message.*

Prover P		Verifier V
Run $\texttt{convert}(cpk, pk, cert_{pk})$ to get (S, R) as in Definition 1		
Choose $(S', pk') \xleftarrow{R} \mathbb{G}_S \times \mathbb{G}_{pk}$		
Compute $f' = F(S', pk')$	$\xrightarrow{\quad f', R \quad}$	
		Compute I as in Definition 1
	$\xleftarrow{\quad b \quad}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
	$\xrightarrow{(z_S, z_{pk}) = (S' *_S S^b, pk' \star_{pk} pk^b)}$	
		Accept if $F(z_S, z_{pk}) = f' \circ_F I^b$

Fig. 7. Proof system for membership to the language $\{(cert_{pk}, pk) \colon cert_{pk} = \Sigma.\texttt{sign}_{csk}(pk)\}$ Common input: cpk and Private input : $(cert_{pk}, pk)$

We first remark that the function F used in the definition of the class $\mathbb{S}$ induces a group law in $F(\mathbb{G}_S \times \mathbb{G}_{pk})$ for the operation $\circ_F$ (actually, the order of $F(\mathbb{G}_S \times \mathbb{G}_{pk})$ is the lcm of the orders of both $\mathbb{G}_S$ and $\mathbb{G}_{pk}$). Moreover, we have $1_{F(\mathbb{G}_S \times \mathbb{G}_{pk})} = F(1_{\mathbb{G}_S}, 1_{\mathbb{G}_{pk}})$ and $\forall (s, k) \in \mathbb{G}_S \times \mathbb{G}_{pk}$: $F(s, k)^{-1} = F(s^{-1}, k^{-1})$.

Proof. For completeness, it is clear that if both parties follow the protocol, the prover will always be able to provide a proof that the verifier will accept.

The protocol satisfies further the special soundness property. In fact, suppose we have for the same (f', R) two accepting answers (z_S, z_{pk}) and $(\overline{z}_S, \overline{z}_{pk})$ for two different challenges b and $\overline{b}$ respectively. Then $F(z_S, z_{pk}) = f' \circ_F I^b$ and $F(\overline{z}_S, \overline{z}_{pk})) = f' \circ_F I^{\overline{b}}$, which leads to $F[z_S^{-1} *_S \overline{z}_S, z_{pk}^{-1} *_{pk} \overline{z}_{pk}] = I^{\overline{b}-b}$ (we assume w.l.g. that $\overline{b} > b$). This means that $F[z_S^{-1} *_S \overline{z}_S, z_{pk}^{-1} *_{pk} \overline{z}_{pk}]^{1/(\overline{b}-b)} = I$ (we assume that both orders of $\mathbb{G}_S$ and $\mathbb{G}_{pk}$ are prime (powers), and thus $1/(\overline{b} - b)$ (modulo the lcm of both orders) exists). Let $(\widetilde{S}, \widetilde{pk}) = [(z_S^{-1} *_S \overline{z}_S)^{1/(\overline{b}-b)}, (z_M^{-1} *_{pk} \overline{z}_{pk})^{1/(\overline{b}-b)}]$. Applying $\texttt{retrieve}$ to $(\widetilde{S}, R, \widetilde{pk})$ gives a valid certificate (signature) on the key $\widetilde{pk}$.

To prove that the protocol is ZK, we provide the following simulator:

1. Run simulate to generate R, and compute I accordingly.
2. Choose $(z_S, z_{pk}) \xleftarrow{R} \mathbb{G}_S \times \mathbb{G}_{pk}$, then compute $f' = F(z_S, z_{pk}) \circ_F I^{-\hat{b}}$, for some $\hat{b} \xleftarrow{R} \{0, 1\}$, and send it to the verifier along with R.
3. Get b from the verifier.
4. If $b = \hat{b}$, the simulator sends back (z_S, z_{pk}). Otherwise, it goes to Step 2 (*rewinds* the verifier).

The prover's first message in the protocol is the value of F on a random point $(S', pk') \in \mathbb{G}_S \times \mathbb{G}_{pk}$ and so is the simulator's. Moreover, the distributions of the responses of the prover and of the simulator are again identical. Finally, we observe that the simulator runs in expected time 2^ℓ since the probability of not rewinding the verifier is:

$$\begin{aligned}
\Pr[b = \hat{b}] &= \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i, \hat{b} = b_i] \\
&= \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i] \Pr[\hat{b} = b_i] \\
&= 2^{-\ell} \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i] \\
&= 2^{-\ell}
\end{aligned}$$

Adjusting ℓ to a factor logarithmic in the security parameter ensures that the simulator will run in expected polynomial time. □

E.2 The Class $\mathbb{E}_1$ of TBE

Fact 2 *The scheme described in Figure 3 belongs to the class* $\mathbb{E}_1$.

Proof. 1. The message space is $\mathbb{G}$ and the key space is $\mathbb{G}^4$.

2. Given a message m and its encryption $e = (e_1, e_2, e_3, e_4)$ w.r.t. a key pk and a tag t, one can efficiently prove knowledge of this statement using the randomness used to produce c; the proof of equality of discrete logarithms.
3. Given a ciphertext $e = (e_1, e_2, e_3, e_4)$ produced on some message w under some key pk w.r.t. a given tag t:
   (a) Generate some random $sk' = \{x'_i, \widetilde{x}'_i\}_{i=1,2} \xleftarrow{R} \mathbb{Z}_d^4$ and some random message $m' \xleftarrow{R} \mathbb{G}$
   (b) Compute $pk = \{X'_i, \widetilde{X}'_i\}_{i=1,2} \leftarrow \{g^{x'_i}, g^{\widetilde{x}'_i}\}_{i=1,2}$.
   (c) Compute the ciphertext e' as $(g^r, (X'^t_1 \widetilde{X}'_1)^r, (X'^t_2 \widetilde{X}'_2)^r, w' X'^r_1), \left(\{X'^r_i, (\widetilde{X}'_i)^r\}_{i=1,2} \leftarrow \{c_1^{x'_i}, c_1^{\widetilde{x}'_i}\}_{i=1,2}\right)$.

   It is easy to see that both pk' and w' are random.
4. Moreover the set $\mathcal{E} = \{e' : (pk', w', e') \leftarrow \Gamma.\mathtt{compute}(e)\}$ is obviously equal to $\{c_1\} \times \mathbb{G}^3$, on which we define the following operation $\circ_e$:

$$(c_1, a, b, c) \circ_e (c_1, a', b', c') = (c_1, a \cdot a', b \cdot b', c \cdot c')$$

   $\mathcal{E}$ is naturally a group operation w.r.t. $\circ_e$ with identity $(c_1, 1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})$ and inverse element for (c_1, a, b, c) the element $(c_1, a^{-1}, b^{-1}, c^{-1})$. Moreover the ciphertext $e'' = e' \circ_e e$ is an encryption of $w \cdot w'$ under the key $pk' \cdot pk = (X_1 X'_1, \widetilde{X}_1 \widetilde{X}'_1, X_2 X'_2, \widetilde{X}_2 \widetilde{X}'_2)$ w.r.t. the same tag t. Finally, the randomness used to produce e is the same used to produce e' and e''. □

Prover P		Verifier V
		Compute $I_{\mathcal{R}}$ as in Definition 2
Compute $(pk', w', e'_1) \leftarrow \Gamma_1.\mathtt{compute}(e_1, t)$ Compute $f' = F(pk')$ and Compute $f'_{\mathcal{R}} = F_{\mathcal{R}}(w')$		
	$\xrightarrow{e'_1, f', f'_{\mathcal{R}}}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
	$\xrightarrow{z_{pk} = pk' *_{pk} pk^b, z_w = w' *_w w^b}$	
	$\xleftrightarrow{\mathsf{PoK}\{e'_1 \circ_e e_1^b = \Gamma_1.\mathtt{encrypt}_{z_{pk}}(z_w, t)\}}$	
		Accept if PoK is valid, $f' \circ_F f^b = F(z_{pk})$, and $F_{\mathcal{R}}(z_w) = f'_{\mathcal{R}} \circ_{\mathcal{R}} I_{\mathcal{R}}^b$

Fig. 8. Proof system for membership to the language $\{(w, pk) : e_1 = \Gamma_1.\mathtt{encrypt}_{pk}(w, t) \wedge f = F(pk) \wedge (x, w) \in \mathcal{R}\}$ Common input: (e_1, t, f, x) and Private input: (w, pk) and randomness used to produce e_1.

Remark 4. The presence of the map F in Figure 8 is needed to bind the public key underlying the encryption e_1 to a specific value. Presence of the relation $\mathcal{R}$ adds a further constraint on the decryption of e_1.

Theorem 6. *Let Γ_1 be an encryption scheme from the above class $\mathbb{E}_1$. Let further F be a homomorphic function with domain the key space of Γ_1. Let finally $\mathcal{R}$ be a relation from Class $\mathbb{R}$. The protocol depicted in Figure 8 is a zero knowledge proof of knowledge with the SpS property.*

Proof. We first note that $1_{\mathcal{E}} = \mathit{\Gamma}_1.\mathtt{encrypt}_{1_{\mathbb{G}_{pk}}}(1_{\mathbb{G}_w}, t)$ ($\mathcal{E}$ is the set underlying $\mathit{\Gamma}_1$, see Definition 3), and $[\mathit{\Gamma}_1.\mathtt{encrypt}_{pk}(w, t)]^{-1} = \mathit{\Gamma}_1.\mathtt{encrypt}_{pk^{-1}}(w^{-1}, t)$.

Completeness is straightforward. For the SpS property, we assume that we have for the same commitment $(e_1', f', f_{\mathcal{R}}')$ two accepting answers (z_{pk}, z_w) and $(\overline{z}_{pk}, \overline{z}_w)$ (along with the corresponding proofs PoK and $\overline{\mathsf{PoK}}$) for two different challenges b and $\overline{b}$ respectively. We have thereby:

- $f' \circ_F f^b = F(z_{pk})$ and $f' \circ_F f^{\overline{b}} = F(\overline{z}_{pk})$.
- $F_{\mathcal{R}}(z_w) = f_{\mathcal{R}}' \circ_{\mathcal{R}} I_{\mathcal{R}}^b$ and $F_{\mathcal{R}}(\overline{z}_w) = f_{\mathcal{R}}' \circ_{\mathcal{R}} I_{\mathcal{R}}^{\overline{b}}$.
- $e_1' \circ_e e_1^b = \mathit{\Gamma}_1.\mathtt{encrypt}_{z_{pk}}(z_w, t)$ and $e_1' \circ_e e_1^{\overline{b}} = \mathit{\Gamma}_1.\mathtt{encrypt}_{\overline{z}_{pk}}(\overline{z}_w, t)$, provided both PoK and $\overline{\mathsf{PoK}}$ are sound.

This leads to $f = F[(z_{pk}^{-1} *_{pk} \overline{z}_{pk})^{1/\overline{b}-b}]$, and $e_1 = \mathit{\Gamma}_1.\mathtt{encrypt}_{[(z_{pk}^{-1} *_{pk} \overline{z}_{pk})^{1/\overline{b}-b}]}[(z_w^{-1} *_w \overline{z}_w)^{1/\overline{b}-b}, t]$, and $(x, z_w^{-1} *_w \overline{z}_w^{1/\overline{b}-b}) \in \mathcal{R}$.

For the zero-knowledgeness, we describe the following simulator:

1. Generate uniformly a random challenge $\hat{b} \xleftarrow{R} \{0,1\}^{\ell}$. Run $\mathit{\Gamma}_1.\mathtt{compute}$ on (e_1, t) to get $(z_{pk}, z_w, e_1'') \in \mathbb{G}_{pk} \times \mathbb{G}_w \times \mathcal{E}$. Compute $e_1' = e_1'' \circ_e e_1^{-\hat{b}}$, $f'' = F(z_{pk})$, and $f_{\mathcal{R}}'' = F_{\mathcal{R}}(z_w)$. Finally, send $(e_1', f' = f'' \circ_F f^{-\hat{b}}, f_{\mathcal{R}}' = f_{\mathcal{R}}'' \circ_{\mathcal{R}} I_{\mathcal{R}}^{-\hat{b}})$ to the verifier.
2. Get b from the verifier.
3. If $b = \hat{b}$, the simulator sends back (z_{pk}, z_w), and simulates the proof PoK for e_1'' being the encryption of z_w under z_{pk} w.r.t. the tag t (the proof is simulatable since it is zero knowledge by assumption). Otherwise, it goes to Step 2 (*rewinds* the verifier).

The prover's first message is always an encryption of a random message w' w.r.t. t under a random public key pk' in addition to the value of F on pk', and the value of $F_{\mathcal{R}}$ on w'. The simulator's first message is also an encryption of a random message $z_w *_w w^{-\hat{b}}$ w.r.t. t under a random public key $z_{pk} *_{pk} pk^{-\hat{b}}$, the value of the function F on the same key, and the value of $F_{\mathcal{R}}$ on the same random message. Since $\hat{b}$ is chosen uniformly at random from $\{0,1\}^{\ell}$, then the probability that the simulator does not rewind the verifier is $2^{-\ell}$, and thus the simulator runs in expected polynomial time if ℓ is logarithmic in the security parameter. Finally, the distributions of the answers of the prover and of the simulator are the same (both answers comprise random values (z_{pk}, z_w) in $\mathbb{G}_{pk} \times \mathbb{G}_w$ and a proof that z_w is the decryption of $e_1' \circ_e e_1^{\hat{b}}$ w.r.t. t under the key z_{pk}). We conclude that above proof is zero knowledge. $\square$

E.3 Instantiation of the class $\mathbb{H}$

Let $(\mathbb{G}, \cdot)$ be a group, denoted multiplicatively, with prime order d. Let further $a_1, \ldots, a_n$ be elements in $\mathbb{Z}_d$, where n is some integer. We consider the following map:

$$\begin{aligned} H\colon \quad \mathbb{G}^n &\longrightarrow \mathbb{G} \\ (X_1, \ldots, X_n) &\longmapsto \prod_{i=1}^{n} X_i^{a_i} \end{aligned}$$

It is easy to see that $\mathbb{G}^n$ is a group where the group operation, say $\odot$, is the component-wise product. Moreover for every $(X_1, \ldots, X_n)$ and $(X_1', \ldots, X_n')$ in $\mathbb{G}^n$, we have:

$$H[(X_1, \ldots, X_n) \odot (X_1', \ldots, X_n')] = H(X_1, \ldots, X_n) \cdot H(X_1', \ldots, X_n')$$

Let $pk = (X_1, \ldots, X_n) \in \mathbb{G}^n$ be a public key from *database*. We assume that $X_i = g_i^{x_i}$, $i = 1 \ldots n$, where the g_i's are known generators of $\mathbb{G}$ and $(x_1, \ldots, x_n)$ denotes the private key sk corresponding to pk. Let further g be a known generator of $\mathbb{G}$. We denote by $\log_g g_i$ the discrete logarithm of g_i in base g.

Let E denote the event that corresponds to having two different pk and pk' that map to the same value by H. Occurrence of the event E translates in having the multivariate polynomial (defined over $\mathbb{Z}_d$):

$$P(y_1, \ldots, y_n) = \sum_{i=1}^{n} a_i \log_g(g_i) y_i$$

evaluate to zero at the point $(x_1 - x'_1, \ldots, x_n - x'_n)$, where $sk = (x_1, \ldots, x_n)$ and $sk' = (x'_1, \ldots, x'_n)$ are the private keys corresponding to pk and pk' respectively. If one of the private keys sk or sk' is random, then the event E has probability $\frac{1}{d}$ of occurrence according to Schwartz-Zippel lemma:

Lemma 1 (Schwartz-Zippel). *Let P be a non-zero affine multivariate polynomial in $\mathbb{Z}_d[X_1, \ldots, X_n]$, where d is prime. We have:*

$$\Pr_{x_1,\ldots,x_n \xleftarrow{R} \mathbb{Z}_d} [P(x_1, \ldots, x_n) = 0] \leq 1/d.$$

□

To force the private keys sk to be random, GM can proceed to a systematic randomization of the corresponding pk as follows: upon receipt of $pk = (X_1, \ldots, X_n)$ from the group member, GM computes the new key as $X_i \leftarrow X_i^{r_i}$, $i = 1 \ldots n$, for a randomly selected vector $r = (r_1, \ldots, r_n) \xleftarrow{R} \mathbb{Z}_d^n$. The corresponding private key can be computed once GM publishes the vector r.

E.4 The Class $\mathbb{E}_2$ of TBE

Prover P		Verifier V
Choose $pk' \xleftarrow{R} \mathbb{G}_{pk}$		
Compute $f' = F(pk')$		
Compute $h' = H(pk')$		
Compute $e'_2 = \Gamma_2.\mathtt{encrypt}_{\Gamma_2.pk}(h', t)$	$\xrightarrow{(f', e'_2)}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell$ $(b \in \mathbb{N})$
	$\xrightarrow{z_h = h' *_H h^b,\ z_{pk} = pk' *_k pk^b}$	
	$\xleftrightarrow{\mathsf{PoK}\{e'_2 \circ_c e_2^b = \Gamma_2.\mathtt{encrypt}_{\Gamma_2.pk}(z_h, t)\}}$	
		Accept if: $F(z_{pk}) = f' \circ_F f^b$, $H(z_{pk}) = z_h$, and PoK is valid.

Fig. 9. Proof system for membership to the language $\{(h, pk) \colon e_2 = \Gamma_2.\mathtt{encrypt}(h, t) \wedge h = H(pk) \wedge f = F(pk)\}$ Common input: $(f, e_2, t, \Gamma_2.pk)$ and Private input: (h, pk) and randomness used to produce e_2.

Theorem 7. *Let Γ_2 be a tag-based encryption scheme from Class $\mathbb{E}_2$. Let further F be a homomorphic map and H denote a map from Class $\mathbb{H}$. We assume that both maps have for domain the message space of Γ_2. The protocol depicted in Figure 9 is a zero knowledge proof of knowledge, with the SpS property.*

Proof. To prove this theorem, we first remark that the `encrypt` algorithm, with respect to a given public key pk, induces a group law on the ciphertext space C.

Completeness is straightforward. For the SpS property, we assume that we have for the same commitment e'_2 two accepting answers (z_h, z_{pk}) and $(\overline{z}_h, \overline{z}_{pk})$ (along with the corresponding proofs PoK and $\overline{\mathsf{PoK}}$) for two different challenges b and $\overline{b}$ respectively. We have:

– $F(z_{pk}) = f' \circ_F f^b$ and $F(\overline{z}_{pk}) = f' \circ_F f^{\overline{b}}$.
– $H(z_{pk}) = z_h$ and $H(\overline{z}_{pk}) = \overline{z}_h$.
– $e'_2 \circ_c e_2^b = \Gamma_2.\mathtt{encrypt}_{pk_{\mathsf{OA}}}(z_h, t)$ and $e'_2 \circ_c e_2^{\overline{b}} = \Gamma_2.\mathtt{encrypt}_{pk_{\mathsf{OA}}}(\overline{z}_h, t)$, provided both PoK and $\overline{\mathsf{PoK}}$ are sound.

This leads to $e_2 = \Gamma_2.\mathtt{encrypt}_{pk_{\mathsf{OA}}}[(z_h^{-1} *_H \overline{z}_h)^{1/\overline{b}-b}, t]$, $f = F[(z_{pk}^{-1} *_{pk} \overline{z}_{pk})^{1/\overline{b}-b}]$, and $H[(z_{pk}^{-1} *_{pk} \overline{z}_{pk})^{1/\overline{b}-b}] = z_h^{-1} *_H \overline{z}_h)^{1/\overline{b}-b}$.

For the zero-knowledgeness, we describe the following simulator:

1. Generate uniformly a random challenge $\hat{b} \xleftarrow{R} \{0,1\}^\ell$ and a random $z_{pk} \xleftarrow{R} \mathbb{G}_{pk}$. Then compute $z_h = H(pk)$, $f' = F(z_{pk}) \circ_F f^{-\hat{b}}$, and $e'_2 = \Gamma_2.\mathtt{encrypt}_{\Gamma_2.pk}(z_h, t) \circ_c e_2^{-\hat{b}}$. And finally send (f', e') to the verifier.
2. Get b from the verifier.
3. If $b = \hat{b}$, the simulator sends back (z_h, z_{pk}) and runs the proof PoK for $e'_2 \circ_c e_2^b$ being the encryption of z_h w.r.t. t (with the randomness used to produce $\Gamma_2.\mathtt{encrypt}_{\Gamma_2.pk}(z_h, t)$ as private input). Otherwise, it goes to Step 2 (*rewinds* the verifier).

The distribution of the messages of the prover and of the simulator is the same. Moreover, the probability that the simulator does not rewind the verifier is $2^{-\ell}$, since $\hat{b}$ is chosen uniformly at random from $\{0,1\}^\ell$. We conclude then that above proof is perfectly zero knowledge as the simulator runs in expected polynomial time if ℓ is logarithmic in the security parameter. □

E.5 Proof of Theorem 3

Proof. The protocol depicted in Figure 2 is a parallel composition of the protocols given in Figures 7 & 8 & 9. Therefore, completeness and the SpS property follow in straightforward way. Likewise, the ZK knowledge simulator of the protocol is a parallel composition of the ZK simulators of the mentioned protocols, which concludes the proof. □

E.6 Proof of Theorem 4

We first note the following remark.

Remark 5. If the function F is not second-preimage resistant, i.e. given (S, pk), then one can find a different pair $(\widetilde{S}, \widetilde{pk})$ such that $F(\widetilde{S}, \widetilde{pk}) = F(S, pk)$, then one can existentially forge certificates given any valid certificate and corresponding key. In fact, the adversary can compute from the valid certificate, say $cert_{pk}$ and key pk the tuple (S, R) such that $F(S, pk) = g(cpk, R)$. Then call the `retrieve` algorithm on $(\widetilde{S}, \widetilde{pk}, R)$ in order to compute a certificate $\widetilde{cert}_{\widetilde{pk}}$ on $\widetilde{pk}$.

Proof. The proof is similar to that of [23, Theorem 3.1] except that we use in our case the SpS property of `prove` instead of the extractability and binding property of the commitment. Let $\mathcal{A}$ be a soundness adversary against the construction. We build an adversary $\mathcal{R}$ against the signature scheme underlying the construction as follows.

$\mathcal{R}$ gets the signature scheme public parameters, generates further the remaining parameters for the construction, and finally submits these to $\mathcal{A}$. $\mathcal{R}$ further uses her signing oracle in order to issue valid certificates on the public keys presented by $\mathcal{A}$.

Due to the SpS property (which implies soundness) of `prove`, the only possibility for $\mathcal{A}$ is to successfully carry out `prove` for $pk \notin database$. $\mathcal{R}$ can then use the knowledge extractor (described in the proof of Theorem 5) in order to extract a pair $(\widetilde{S}, \widetilde{pk})$ satisfying $F(\widetilde{S}, \widetilde{pk}) = F(S, pk) = I$. If $\widetilde{pk} \neq pk$, then according to Remark 5, we can build an EUF-CMA forger against the construction, otherwise we call `retrieve` on (S, R, pk) in order to extract the certificate, say $cert_{pk}$, on pk. Since $pk \notin database$, then $\mathcal{R}$ never requested his challenger for a signature on pk and thus $(cert_{pk}, pk)$ forms a valid existential forgery on the signature scheme.

F Security Analysis of the Encryption Scheme in Figure 3

F.1 The Twin Diffie-Hellman problem

The Twin Diffie-Hellman problem was introduced in [11] as a slight modification to the original Diffie-Hellman (DH) problem which is as hard as the ordinary DH problem, *even given access to a corresponding decision oracle*. It is defined as follows.

Let $\mathbb{G}$ be a prime-order cyclic group generated by some g. Let further dh further denotes the Diffie-Hellman function (it inputs $(X, Y) \in \mathbb{G}^2$ and returns Z such that (g, X, Y, Z) is Diffie-Hellman quadruple). The *Twin DH function is defined as follows*

$$\mathrm{2dh}\colon \begin{array}{c} \mathbb{G}^3 \to \mathbb{G}^2 \\ (X_1, X_2, Y) \to (\mathrm{dh}(X_1, Y), \mathrm{dh}(X_1, Y)) \end{array}$$

Cash *et al.* define further a corresponding *twin DH predicate*:

$$\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) := \mathrm{2dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2).$$

Finally the *Strong Twin Diffie-Hellman* assumption states that distinguishing the two distributions $(X_1, X_2, Y, \mathrm{dh}(X_1, Y))$ and (X_1, X_2, Y, Z) for random $X_1, X_2, Y, Z \in \mathbb{G}$ is hard even in the presence of a decision oracle for the predicate $\mathrm{2dhp}(X_1, X_2, -, -, -)$ which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2))$ returns $\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

Theorem 8. *The DDH assumption holds if and only if the Strong Twin DDH assumptions holds.*

The proof is given in [11].

F.2 Indistinguishability

Theorem 9. *The encryption scheme in Figure 3 is IND-st-wCCA secure under the Strong Twin Diffie-Hellman assumption.*

Proof. The proof is similar to that of Theorem 7 in [11]. We consider a sequence of games $\mathsf{Game_i}$ and we denote by $\mathsf{S_i}$ the event corresponding to $\mathsf{Game_i}$ returning 1.

Game_0 We consider the standard game played with a (t, ϵ, q_d)-IND-st-wCCA attacker $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND-st-wCCA}}(1^\kappa)}$

1. $(\mathbb{G}, d, g) \leftarrow \mathtt{setup}(1^\kappa)$;
2. $t \leftarrow \mathcal{A}(param)$;
3. $sk = (x_1, \tilde{x}_1, x_2, \tilde{x}_2) \stackrel{R}{\leftarrow} \mathbb{Z}_d^4$;
4. $pk = (X_1, \tilde{X}_1, X_2, \tilde{X}_2)) \leftarrow (g^{x_1}, g^{\tilde{x}_1}, g^{x_2}, g^{\tilde{x}_2})$;
5. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(pk)$;
6. $r \leftarrow \mathbb{Z}_d\ b \stackrel{R}{\leftarrow} \{0, 1\}; e_b \leftarrow (g^r, (X_1^t \tilde{X}_1)^r, (X_1^t \tilde{X}_1)^r)$;
7. $b^\star \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(e_b)$;
8. If $b = b^\star$ return 1 else 0.

By definition, $\Pr[\mathsf{S}_0] = \epsilon + \frac{1}{2}$.

Game_1 We change in this game the generation of the key in Lines 3. and 4. as follows:

3. $x_1, x_2, a_1, a_2 \stackrel{R}{\leftarrow} \mathbb{Z}_d$;
4. $pk \leftarrow (X_1, \tilde{X}_1, X_2, \tilde{X}_2) = (g^{x_1}, X_1^{-t} g^{a_1}, g^{x_2}, X_2^{-t} g^{a_2}))$;

We change further the decryption oracle $\mathtt{decrypt}^{\neg(-,t)}(sk, .)$ in Lines 4. and 6. as follows. For a ciphertext $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{C})$ w.r.t. a tag $t^\star \neq t$, compute $\bar{Z}_i = (\hat{Z}_i^{/} \hat{Y}_i^{a_i})^{1/(\hat{t}-t)}$ for $i = 1, 2$. Then, check the consistency of the ciphertext by checking if

$$\hat{Y}^{x_1} = \bar{Z}_1 \text{ and } \hat{Y}^{x_2} = \bar{Z}_2$$

If the ciphertext is consistent, then output $m = C \cdot \bar{Z}_1^{-1}$ as a decryption.
Clearly $\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0]$.

Game_2 Consider a Strong Twin Diffie-Hellman adversary $\mathcal{R}$ with instance (Y, X_1, X_2, Z). $\mathcal{R}$ will execute Game_1 with the following changes:

3. $a_1, a_2 \xleftarrow{R} \mathbb{Z}_d$;
4. $pk \leftarrow (X_1, X_1^{-t} g^{a_1}, X_2, X_2^{-t} g^{a_2})$;

Furthermore, the simulation of the decryption oracle in Lines 4 and 6 is done using the 2dhp oracle (provided for $\mathcal{R}$). Actually, the oracle is invoked on the input $(X_1, X_2, \hat{Y}, \bar{Z}_1, \bar{Z}_1)$ to check the consistency of the queried ciphertext $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{C})$ (with $\bar{Z}_i = (\hat{Z}_i^{/} \hat{Y}_i^{a_i})^{1/(\hat{t}-t)}$ for $i = 1, 2$) since $\mathcal{R}$ does not know x_1 and x_2.

Again we have $\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1]$.

Game_3 In this game, $\mathcal{B}$ plugs further her instance (Y, X_1, X_2, Z) in the challenge ciphertext as follows:

5. $b \xleftarrow{R} \{0, 1\}$ $e_b \leftarrow (Y, Y^{a_1}, Y^{a_2}, m_b Z)$;

If we denote by T the event corresponding to $Z = dh(X_1, Y)$, then we clearly have $\Pr[\mathsf{S}_3|\mathsf{T}] = \Pr[\mathsf{S}_2|\mathsf{T}]$. $\mathcal{R}$ will simply forward the result of the outcome of Game_3 to his challenger. We have:

$$\begin{aligned}
\mathsf{Adv}(\mathcal{R}) &= \left|\Pr[\mathsf{Game}_3(1^\kappa) = 1 \wedge \mathsf{T}] + \Pr[\mathsf{Game}_3(1^\kappa) = 0 \wedge \neg\mathsf{T}] - \frac{1}{2}\right| \\
&= \frac{1}{2} \left|\Pr[\mathsf{Game}_3(1^\kappa) = 1|\mathsf{T}] + \Pr[\mathsf{Game}_3(1^\kappa) = 0|\neg\mathsf{T}] - 1\right| \\
&= \frac{1}{2} \left|\Pr[\mathsf{S}_3|\mathsf{T}] + \Pr[\mathsf{Game}_3(1^\kappa) = 0|\neg\mathsf{T}] - 1\right| \\
&= \frac{1}{2} \left|\Pr[\mathsf{S}_0|\mathsf{T}] - \frac{1}{2}\right| \\
&= \frac{\mathsf{Adv}(\mathcal{A})}{2}
\end{aligned}$$

The last but one equation is due to $\Pr[\mathsf{S}_3|\mathsf{T}] = \Pr[\mathsf{S}_2|\mathsf{T}]$ and $\Pr[\mathsf{S}_2] = \Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0] = \Pr[\mathsf{S}_0|\mathsf{T}]$. Moreover, if $\neg\mathsf{T}$ occurs, then Z is random in $\mathbb{G}$, and so is $m_b Z$ which is consequently independent of m_b. Then the probability that the output of Game_3 equals zero is exactly $\frac{1}{2}$. □

F.3 Anonymity

Here, we extend the result of [22, 15] which derives anonymity from indistinguishability in public key encryption schemes, to tag-based encryption schemes.

Let Γ be a tag-based encryption scheme given by the three algorithms $\Gamma.\mathtt{keygen}$, $\Gamma.\mathtt{encrypt}$, and $\Gamma.\mathtt{decrypt}$. Let further M and C denote the message and ciphertext space of Γ respectively. We stress that C depends solely on the considered security parameter and not on a particular key. We consider the following property:

Property A Let κ be a security parameter. Let further t be a fixed tag, and (pk, sk) be an output of $\Gamma.\mathtt{keygen}$. Consider the uniform distribution on M. Then, the distribution on C corresponding to the random variable $\Gamma.\mathtt{encrypt}_{pk}(m, t)$ where $m \xleftarrow{R} \mathsf{M}$, is indistinguishable from uniform.

It is clear that the scheme in Figure 3 satisfies easily **Property A** as a for a fixed key $(X_1, \widetilde{X}_1, X_2, \widetilde{X}_2)$ and a fixed tag the encryption of a random message m is a random ciphertext $(g^r, (X_1^t \widetilde{X}_1)^r, (X_2^t \widetilde{X}_2)^r, m X_1^r)$ in the ciphertext space (due to the randomness of r).

Theorem 10. *Let Γ be tag-based encryption scheme which has Property **A**. Let further $(t, q_d) \in \mathbb{N}^2$ and $\epsilon \in [0, 1]$. Γ is (t, ϵ, q_d)-ANO-st-wCCA secure if it is $(t, \frac{\epsilon}{2}, q_d)$-IND-st-wCCA secure.*

Proof. We proceed in this proof using the game-hopping technique which considers a sequence of games; the first one mirrors that of an anonymity adversary whereas the last one corresponds to the standard indistinguishability game. We denote by S_i the event that Game_i returns 1.

Game_0**.** We consider the standard game played with a (t, ϵ, q_d)-ANO-st-wCCA attacker $\mathcal{A}$.

$\boxed{\text{Experiment } \mathbf{Exp}_{\mathcal{A}}^{\mathsf{ANO\text{-}st\text{-}wCCA}}(1^\kappa)}$

1. $param \leftarrow \mathtt{setup}(1^\kappa)$;
2. $t \leftarrow \mathcal{A}(param)$;
3. $(sk_0, pk_0) \leftarrow \Gamma.\mathtt{keygen}(param, 1^\kappa)$; $(sk_1, pk_1) \leftarrow \Gamma.\mathtt{keygen}(param, 1^\kappa)$
4. $m_0 \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk_0,.),\mathtt{decrypt}^{\neg(-,t)}(sk_1,.)}(pk_0, pk_1)$;
5. $b \xleftarrow{R} \{0,1\}$; $e_b \leftarrow \Gamma.\mathtt{encrypt}_{pk_b}(m_0, t)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk_0,.),\mathtt{decrypt}^{\neg(-,t)}(sk_1,.)}(e_b)$;
7. If $b = b^\star$ return 1 else 0.

By definition, $\Pr[\mathsf{S}_0] = \epsilon + \frac{1}{2}$.

Game_1. In this game, we change Line 5 by 5. $m_1 \xleftarrow{R} \mathsf{M}$; $b \xleftarrow{R} \{0,1\}$; $e_b \leftarrow \Gamma.\mathtt{encrypt}_{pk_0}(m_b, t)$; and Line 7. by

7. If $b^\star = 0$ return 1 else 0.

If we denote by Z the event corresponding to $b = 0$, then we clearly have $\Pr[\mathsf{S}_1|\mathsf{Z}] = \Pr[\mathsf{S}_0|\mathsf{Z}]$.

Now, consider the following IND-st-wCCA adversary $\mathcal{R}$ against Γ who executes Game_1 as follows; $\mathcal{R}$ replaces *param* by those got from her challenger, and forwards to this latter the tag t. She further sets pk_0 to her challenge public key, and simulates the oracle $\mathtt{decrypt}^{\neg(-,t)}(sk_0,.)$ in Lines 4 and 6 by her own decryption oracle (the simulation is perfect as she is allowed to query any ciphertext w.r.t. any tag different from t). Finally, she generates m_1 internally and pass both m_0 and m_1 to her challenger to get in response the challenge e_b that she gives to $\mathcal{A}$. If the output of Game_1 is 1, then $\mathcal{R}$ will output $b' = 0$ to her challenger, otherwise she will respond with $b' = 1$.

By definition $\mathsf{adv}(\mathcal{R}) = \left|\Pr[b' = b] - \frac{1}{2}\right|$:

$$
\begin{aligned}
\mathsf{Adv}(\mathcal{R}) &= \left|\Pr[b' = b] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Game}_1(1^\kappa) = 1 \wedge b = 0] + \Pr[\mathsf{Game}_1(1^\kappa) = 0 \wedge b = 1] - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|\Pr[\mathsf{Game}_1(1^\kappa) = 1|\mathsf{Z}] + \Pr[\mathsf{Game}_1(1^\kappa) = 0|\neg\mathsf{Z}] - 1\right| \\
&= \frac{1}{2}\left|\Pr[\mathsf{S}_1|\mathsf{Z}] + \Pr[\mathsf{Game}_1(1^\kappa) = 0|\neg\mathsf{Z}] - 1\right| \\
&= \frac{1}{2}\left|\Pr[\mathsf{S}_0|\mathsf{Z}] - \frac{1}{2}\right| \\
&= \frac{\mathsf{Adv}(\mathcal{A})}{2}
\end{aligned}
$$

The last but one equation is justified by the fact that conditioned on the event Z, the two events S_0 and S_1 are identical. Moreover, we have $\Pr[\mathsf{Game}_1(1^\kappa) = 0|\neg\mathsf{Z}] = \frac{1}{2}$ by virtue of **Property A** if $b = 1$, then e_b is a random element in C, and with overwhelming probability it is not an encryption of m_0 under either keys. Finally, we have $\Pr[\mathsf{S}_0|\mathsf{Z}] = \Pr[\mathsf{S}_0|\neg\mathsf{Z}] = \Pr[\mathsf{S}_0] = \mathsf{adv}(\mathcal{A}) + \frac{1}{2}$. □

G Analysis of the prove Procedure in the Realization in 4.3

Interactive proof The interactive version of prove is that obtained by instantiating the protocol depicted in Figure 2 with the previously mentioned bricks. This results in a commitment comprising two elements from $(\mathbb{G}_1 \times \mathbb{G}_2)^2$ (for R), two elements from $\mathbb{G}_T$ (for f'), one element from $\mathbb{G}_T$ (for x'), and eight elements from $\mathbb{G}_2$ (for the encryptions e'_1 and e'_2), which totals 15 group elements. The response comprises three elements from $\mathbb{G}_2$ (for z_S), four elements from $\mathbb{G}_2$ (for z_{pk}), one element from $\mathbb{G}_2$ (for z_h), and one element in $\mathbb{G}_2$ (for z_m). It further comprises two proofs of equality of discrete logarithms where each needs two elements in $\mathbb{G}_2$, a challenge, and one element from the exponent group, say $\mathbb{Z}_d$ (d is $\mathbb{G}_2$'s order). Therefore the total communication cost amounts to 30 group elements in addition to 3ℓ, where $2^{-\ell}$ is the soundness error, say 1 kB if we use a 256-bit group size and we consider a soundness error of 2^{-50}. This is again significantly smaller than 16.125 kB required for [12] (using the same group order bit-size) or 70 kB required for [23] (achieving the same soundness error).

Non-interactive proof We recall again the equations underlying prove (the variables are emphasized by a bold font):

$$A = e(g_z, \mathbf{z})e(g_r, \mathbf{r'})e(s', t') \prod_{i=1}^{4} e(g_i, \mathbf{X_i}) \tag{1}$$

$$B = e(h_z, \mathbf{z})e(g_u, \mathbf{u'})e(v', w') \prod_{i=1}^{4} e(h_i, \mathbf{X_i}) \tag{2}$$

$$(C_1, C_2, C_3, C_4) = (g^{\mathbf{k}}, (Y_1^t Y_3)^{\mathbf{k}}, (Y_2^t Y_4)^{\mathbf{k}}, \mathbf{h} Y_1^{\mathbf{k}}) \tag{3}$$

$$\mathbf{h} = \prod_{i=1}^{4} \mathbf{X_i}^{a_i} \tag{4}$$

$$(D_1, D_2, D_3, D_4) = (g^{\ell}, (\mathbf{X_1}^{\mathbf{t}}\mathbf{X_3})^{\ell}, (\mathbf{X_2}^{\mathbf{t}}\mathbf{X_4})^{\ell}, \mathbf{m}\mathbf{X_1}^{\ell}) \tag{5}$$

$$e(P, \mathbf{m}) = x \tag{6}$$

The equations involve 11 variables $(z, r', u', X_1, X_2, X_3, X_4, k, h, \ell, m)$and thus will increase the size of the proof by 22 group elements. Equations (1) and (2) costs two group elements each (linear pairing equation). Equation (3) comprises four sub-equations; the first three cost one group element each (linear equation) and the last one costs 6 group elements (multi-exponentiation equation). Equation (4) costs 6 group elements (multi-exponentiation equation). Equation (5) comprises four sub-equations, the first one costs 1 group element, and the last three ones cost 6 elements each. Finally Equation (6) costs 2 group elements. The overall size of prove amounts then to $22 + 2 + 2 + 3 + 6 + 6 + 1 + 3 \cdot 6 + 2 = 62$ group elements, say 2 kB versus 16.125 kB [12] (using the same group order bit-size, namely 256-bit size).

The verification cost in terms of pairing computations is computed as follows. Equation (1) costs $5 \cdot 0 + 3 \cdot 6 + 16 = 34$ pairings. Equation (2) costs similarly 34 pairings. Equation (3) comprises four sub-equations: the first three ones need $8 \cdot 0 + 2 \cdot 1 + 14 = 16$ pairings each; the last sub-equation demands $8 \cdot 1 + 2 \cdot 1 + 14 = 24$; so to say Equation (3) needs $3 \cdot 16 + 24 = 72$ pairings. Equation (4) needs $8 \cdot 5 + 2 \cdot 0 + 14 = 54$ pairings. Equation (5) comprises four sub-equations; the first one needs $2 + 14 = 16$ pairings and the last three costs $8 \cdot 2 + 2 \cdot 1 + 14 = 32$ pairings each. Finally Equation (6) needs $5 \cdot 0 + 3 \cdot 1 + 16 = 19$ pairings each. The total cost adds up to $34 + 34 + 72 + 54 + 16 + 3 \cdot 32 + 19 = 325$ pairings.

H Analysis of the prove Procedure in [12]

The authors in [12] use Shacham [28]'s scheme to encrypt the witness (which is a Diffie-Hellman key), Kiltz [24]'encryption scheme to encrypt the receiver's key, and finally they use a certification scheme they develop in the same paper in order to certify the receivers keys.

More precisely, the equations underlying the `prove` procedures are (the variables are emphasized by a bold font)

$$e(\mathbf{C_{i,1}}, f) = e(f_1, \mathbf{D_{i,1}})e(f_{3,1}, \mathbf{D_{i,1}}) \quad i = 1 \ldots 6 \tag{1}$$

$$e(\mathbf{C_{i,2}}, f) = e(f_2, \mathbf{D_{i,2}})e(f_{3,2}, \mathbf{D_{i,3}}) \quad i = 1 \ldots 6 \tag{2}$$

$$e(\mathbf{C_{i,3}}, f) = e(\mathbf{X_i} \cdot \mathbf{D_{i,1}} \cdot \mathbf{D_{i,2}}, f)e(f_{3,3}, \mathbf{D_{i,3}}) \quad i = 1 \ldots 6 \tag{3}$$

$$e(\mathbf{S_1}, \Omega \cdot \mathbf{S_1}) = A \tag{4}$$

$$e(\mathbf{S_2}, u) = e(g, \mathbf{S_3}) \tag{5}$$

$$e(\mathbf{S_4}, g) = e(u_0, \mathbf{S_2}) \cdot \prod_{i=1}^{n} (e(u_{i,1}, \mathbf{S_{5,i,1}}) \cdot e(u_{i,2}, \mathbf{S_{5,i,2}}) \cdot e(u_{i,3}, \mathbf{S_{5,i,3}})) \tag{6}$$

$$e(\mathbf{S_{5,i,j}}, g) = e(\mathbf{C_{i,j}}, \mathbf{S_2}) \quad i = 1 \ldots 6, j = 1 \ldots 3 \tag{7}$$

$$(V_{i,1}, V_{i,2}, V_{i,3}, V_{i,4}, V_{i,5}) = \left(Y_1^{\mathbf{w_{i,1}}}, Y_2^{\mathbf{w_{i,2}}}, (g^t Y_3)^{\mathbf{w_{i,1}}}, (g^t Y_4)^{\mathbf{w_{i,2}}}, \mathbf{X_i} \cdot g^{\mathbf{w_{i,1}+w_{i,2}}}\right) \quad i = 1 \ldots 6 \tag{8}$$

$$(U_1, U_2, U_3, U_4, U_5) = \left(g_1^{\mathbf{r}}, g_2^{\mathbf{s}}, g^{\mathbf{r+s}}, \mathbf{W} \cdot \mathbf{X_5}^{\mathbf{r}}\mathbf{X_6}^{\mathbf{s}}, (\mathbf{X_1}\mathbf{X_3}^{\alpha})^{\mathbf{r}} \cdot (\mathbf{X_2}\mathbf{X_4}^{\alpha})^{\mathbf{s}}\right) \tag{9}$$

$$e(X, Y) = e(g, \mathbf{W}) \tag{10}$$

Equation (1) comprises 6 sub-equations, each involves 3 variables, thus it requires $6(12 \cdot 3 + 27) = 378$ pairingd. Similarly equation (2) requires 378 pairings. Equation (3) contains 6 sub-equations each involves 5 variables thus it requires $6(12 \cdot 5 + 27) = 522$ pairings. Equations (4) and (5) both involve two variables and thus require $12 \cdot 2 + 27 = 51$ pairings each. Equation (6) involves 20 variables and thus requires $12 \cdot 20 + 27 = 267$ pairings. Equation (7) comprise 18 sub-equations, each involves 3 variables, thus the equation needs $18(12 \cdot 3 + 27) = 1134$. Equation (8) comprises 6 sub-equations, each sub-equation comprises itself three sub-equations, where the first two involve one variable (the exponent) thus require $9 \cdot 1 + 12 \cdot 0 + 27 = 36$ pairings each, whereas the third one comprise two variables (one exponent and one group element) thus needs $9 \cdot 2 + 12 \cdot 1 + 27 = 57$. Thus Equation (8) requires $6(36 + 36 + 57) = 774$. Equation (9) comprises five sub-equations: the first two involves one variable each and thus require $9 \cdot 1 + 12 \cdot 0 + 27 = 36$ pairings each; the third one involves two variables and thus requires $9 \cdot 2 + 12 \cdot 0 + 27 = 45$; the fourth one involves two exponent variables and three group element variables thus needs $9 \cdot 2 + 12 \cdot 3 + 27 = 81$; finally the fifth sub-equation involves two exponent variables and four group elements variables thus needs $9 \cdot 2 + 12 \cdot 4 + 27 = 93$. Thus equation (9) needs 291 pairings. Finally equation (10) involves one variable and thus requires $12 \cdot 1 + 27 = 49$ pairings.

The overall `prove` requires then 3895 pairing computations.