Published in M. Jacobson *et al.*, Eds, *Applied Cryptography and Network Security (ACNS 2013)*, vol. 7954 of *Lecture Notes in Computer Science*, pp. 237-252, Springer, 2013.

Toward Practical Group Encryption*

Laila El Aimani[1,**] and Marc Joye[2]

[1] Gemalto, 6 rue de la verrerie, 92197 Meudon Cedex, France
[2] Technicolor, 975 avenue des Champs Blancs, 35576 Cesson-Sévigné 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.

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

1 Introduction

Basically, group signature schemes [7] 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 [12] 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

* The full version [2] is available at the Cryptology ePrint Archive.
** This work was done while the first author was working at Technicolor.

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 [12, 6].

Related work The concept of group encryption was first formalized by Kiayias, Tsiounis, and Yung [12]. 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.* [15] 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 [6] provided the first non-interactive realization of group encryption in the standard model. Their ciphertext and proof are also significantly shorter than those of [12] (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 pairings) due to the recourse to Groth-Sahai [11]'s 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.

It 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 constructions.

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 [12, 6] 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, [12, 6] use a chosen ciphertext 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 resorting to the opening function is only done in case of disputes and occurs thus 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 [6]) for the verification.

Finally, we note that due to space constraints, all technical details, proofs, and analyzes of our results, are deferred to the long version [2].

2 Group Encryption: Syntax and Security Model

In this section, we review the formal definition of group encryption, as introduced in [12]. 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 $\mathcal{A}^{\mathtt{oracle}(\cdot)}$ to denote that adversary $\mathcal{A}$ has access to oracle $\mathtt{oracle}(\cdot)$. When a query is not allowed, we use the symbol $\neg$: $\mathcal{A}^{\mathtt{oracle}^{\neg(\text{some query})}(\cdot)}$.

2.1 Syntax

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

$\mathtt{setup}(1^\kappa)$ 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.

$(\mathcal{G}_r, \mathcal{R}, \mathtt{sample}_{\mathcal{R}})$ This tuple of algorithms is part of the `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 [12, 6], $sk_{\mathcal{R}}$ 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 $\mathtt{sample}_{\mathcal{R}}$ 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 $pk_{\mathcal{R}}$.

$\mathtt{keygen}_{\mathsf{E}}(param)$ 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.

$\mathtt{join} = \langle \mathsf{J}_{\mathsf{User}}(param), \mathsf{GM}(sk_{\mathsf{GM}})\rangle(pk_{\mathsf{GM}})$ This is an interactive protocol between GM and the potential joining group member $\mathsf{J}_{\mathsf{User}}$. The latter sends her public key pk to GM and prospectively proves the correctness of her key, whereas GM issues (at the end) 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*.

$\mathtt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, w, L)$ 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).

$\mathtt{prove} = \langle \mathcal{P}(w, coins_{\psi}), \mathcal{V}(param)\rangle(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk_{\mathcal{R}}, x, \psi, L)$ 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.

$\mathtt{decrypt}(sk, \psi, L)$ 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 failure symbol $\perp$).

$\mathtt{open}(sk_{\mathsf{OA}}, \psi, L)$ 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. The verifiability of encryption is optional; if it is not desired, the relation $\mathcal{R}$ can be set to the trivial relation that includes any string of fixed size as a witness.

2.2 Security model

In addition to correctness, we require the following properties in a group encryption scheme.

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 [12, 6]. This definition involves an oracle $\texttt{reg}(sk_{\mathsf{GM}}, \cdot)$ that simulates the group manager GM and maintains a repository *database* that comprises 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

$$\left\{\texttt{encrypt}(pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, pk, w, L)\colon (x, w) \in \mathcal{R} \text{ and } pk \in \mathit{database}\right\}$$

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. $param \leftarrow \texttt{setup}(1^{\kappa})$;
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. $(aux, pk_{\mathcal{R}}, x, \psi, L) \leftarrow \mathcal{A}^{\texttt{reg}(sk_{\mathsf{GM}},\cdot)}(param, pk_{\mathsf{GM}}, pk_{\mathsf{OA}}, sk_{\mathsf{OA}})$;
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. $pk \leftarrow \texttt{open}(sk_{\mathsf{OA}}, \psi, L)$;
7. If $(pk \notin \mathit{database})$ 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.

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 [12, 6].

– $\texttt{decrypt}^{\neg(\cdot,L)}(sk, \cdot)$: 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 uniformly 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 [12, 6], designing an efficient simulator $\mathcal{P}'$ is part of proving the security.

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. $\langle pk, sk, cert_{pk} \mid aux, pk, cert_{pk} \rangle \leftarrow \langle \mathsf{J}_{\mathsf{User}}(param), \mathcal{A}(aux) \rangle (pk_{\mathsf{GM}})$;
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.

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 [12, 6], we introduce the following notations:

– $\mathsf{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot)$: 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 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 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. $(aux,pk_{\mathsf{GM}},L) \leftarrow \mathcal{A}(param)$;$aux \leftarrow \mathcal{A}^{\mathsf{User}(pk_{\mathsf{GM}}),\mathsf{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot)}(aux,pk_{\mathsf{OA}})$; If keys $\neq (pk_0,sk_0,cert_{pk_0},pk_1,sk_1,cert_{pk_1})$ return 0;
3. $(aux,x,w,pk_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathsf{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux)$;
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. $b^\star \leftarrow \mathcal{A}^{\mathtt{prove}_{\mathcal{P}},\mathsf{open}^{\neg(\cdot,L)}(sk_{\mathsf{OA}},\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_0,\cdot),\ \mathtt{decrypt}^{\neg(\cdot,L)}(sk_1,\cdot)}(aux,\psi)$;
7. If $(b = b^\star)$ return 1 else return 0.

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 paper, and in order to avoid confusion, we use a dot notation to refer the different components; for instance, Γ.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 [12, 6] 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 [6], 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. Actually, even the collision-resistance property can be weakened as we will see later since GM has some control over the public keys she certifies, and therefore may proceed to simple measures in case a collision occurs.

`setup`(1^κ) 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.

`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.)

`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).

`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.

`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.

`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. H of such a decryption, and outputs the result of this search.

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

1. IND-sl-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-sl-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.* □

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

Canetti, Halevi, and Katz [4] 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. Concurrently, MacKenzie, Reiter, and Yang [14] present a method for converting a weakly chosen-ciphertext secure tag-based encryption scheme to a fully secure public-key encryption scheme. Finally, Kiltz [13] combines the ideas of [4, 14, 3] 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 [13], 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-sl-wCCA and ANO-sl-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 $\mathfrak{S}$ = (`keygen`, `sign`, `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-sl-wCCA *and* ANO-sl-wCCA*, and* $\mathfrak{S}$ *is a strongly secure one-time signature scheme.* □

Remark 2. This transformation can be also used to upgrade the security in TBE (from sl-wCCA to full CCA indistinguishability and anonymity). In [13], Kiltz suggests to achieve this task via a CCA secure public key encryption (PKE): first derive a CCA secure PKE from an sl-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 CCA encryption is as efficient as proving knowledge of the message underlying its sl-wCCA encryption.

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

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

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:

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

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 [11]. In fact, [11] 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 [9, 8, 1] for the certification scheme Σ, Kiltz'[13] or Cash *et al.*'s [5] (described in Fig. 3) 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 statements underlying `prove` that consist of pairing-product equations need to be of special form 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 those of the prior proposals [12, 6].

4.1 Building blocks

Definition 1 (The class $\mathbb{S}$ of certification 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 $F : (\mathbb{G}_S, *_S) \times (\mathbb{G}_{pk}, *_{pk}) \rightarrow (\mathbb{G}_F, \circ_F)$:*
   - *where $(\mathbb{G}_S, *_S)$ and $(\mathbb{G}_{pk}, *_{pk})$ are groups and $\mathbb{G}_F$ is a set equipped with the binary operation $\circ_F$,*
   - *$\forall (S, pk), (S', pk') \in (\mathbb{G}_S, *_S) \times (\mathbb{G}_{pk}, *_{pk})$: $F(S *_S S', pk *_{pk} pk') = F(S, pk) \circ_F F(S', pk')$.*

   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.

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 that 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 (in bilinear groups) functions. 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 $e' = \Gamma.\texttt{encrypt}_{pk'}(w', t)$, 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*

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

   such that $\Gamma.\texttt{encrypt}_{pk' *_{pk} pk}(w' *_w w, t) = e' \circ_e e$, 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 [5] described in Fig. 3.

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 map 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 [2, 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. $\forall h, h' \in \mathbb{G}_H, \forall pk, \forall t\colon \Gamma.\texttt{encrypt}_{pk}(h *_H h', t) = \Gamma.\texttt{encrypt}_{pk}(h, t) \circ_c \Gamma.\texttt{encrypt}_{pk}(h', t)$*. Moreover, given the randomness used to encrypt h in* $\Gamma.\texttt{encrypt}_{pk}(h, t)$ *and h′ in* $\Gamma.\texttt{encrypt}_{pk}(h', t)$*, one can deduce (using only the public parameters) the randomness used to produce* $\Gamma.\texttt{encrypt}_{pk}(h, t) \circ_c \Gamma.\texttt{encrypt}_{pk}(h', t)$ *on* $h *_H h'$*.*

Examples of encryption schemes in the above class include Kiltz' [13] and Cash *et al.*'s [5] (described in Fig. 3) tag-based encryption schemes.

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 Fig. 2 is an efficient zero knowledge proof of knowledge with the special soundness property.* □

Theorem 4. *The construction in Section 3 is sound if the* prove *protocol satisfies the special soundness property, and the used certification scheme is* EUF-CMA *secure.* □

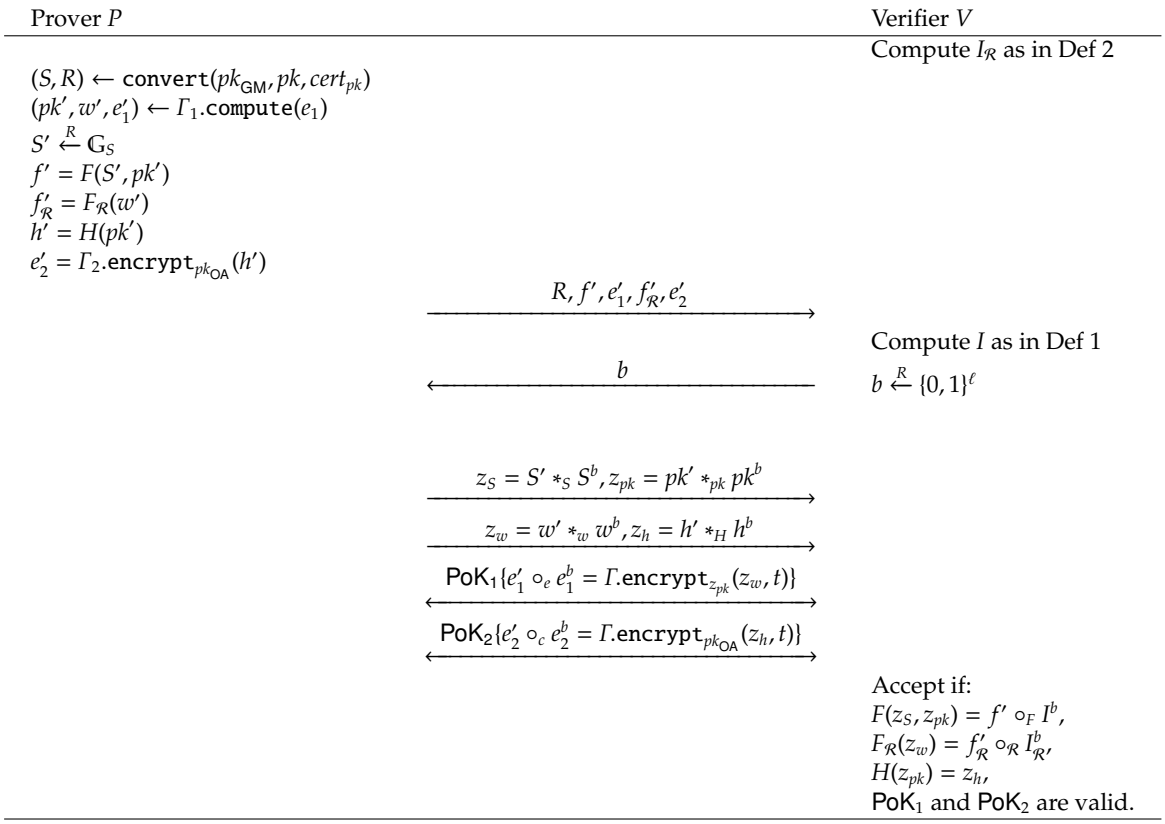

Fig. 2. Proof system for membership to the language $\{(w, pk, cert_{pk})$: $e_1 = \Gamma_1.\texttt{encrypt}_{pk}(w, t) \wedge e_2 = \Gamma_2.\texttt{encrypt}_{pk_{\mathsf{OA}}}(H(pk), t) \wedge cert_{pk} = \Sigma.\texttt{sign}_{sk_{\mathsf{GM}}}(pk) \wedge (x, w) \in \mathcal{R}\}$ Common input: $(e_1, e_2, t, x, pk_{\mathsf{OA}}, pk_{\mathsf{GM}})$ 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 $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. 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 $\mathbb{G}_2$.

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

1. The signature scheme from [1]. The scheme signs messages in $\mathbb{G}_2^4$.
2. The encryption scheme described in Fig. 3 to instantiate both Γ^{User} and Γ^{OA}. The message space of both Γ^{User} and Γ^{OA} is the group $\mathbb{G}_2$.
3. The relation $\mathcal{R}$: $(x = [X, Y], w) \in \mathcal{R} \Leftrightarrow e(X, Y) = e(g, w)$, 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 $\mathbb{G}_2$). The collision-resistance of H is analyzed in the full version of the paper.

5. The one-time signature from [10].

[Setup]	Choose a group $(\mathbb{G}, \cdot)$ generated by g with prime order d.
[Keygen]	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 $sk \leftarrow \{x_i, \widetilde{x}_i\}_{i=1,2}$.
[Encrypt]	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).
[Decrypt]	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 [5]

We summarize in this chart the performances of our realization compared to those of [12] and [6]. (IP stands for interactive proof, whereas NIP stands for non-interactive proof).

	[12]	[6]	Our scheme
Ciphertext (kB)	2.5	1.25	0.4
IP size (kB)	70	–	1
# of pairings in IP	0	–	14
NIP size (kB)	–	16.125	2
# of pairings in NIP	–	3895	325

Fig. 4. Comparison

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 of *Lecture Notes in Computer Science*, pages 209–236. Springer, 2010.
2. Laila El Aimani and Marc Joye. Toward practical group encryption. IACR Cryptology ePrint Archive, Report 2012/155, 2012. http://eprint.iacr.org/.
3. 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 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.

4. 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 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.
5. 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.
6. 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 of *Lecture Notes in Computer Science*, pages 179–196. Springer, 2009.
7. David Chaum and Eugène van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
8. 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/.
9. J. Groth. Homomorphic Trapdoor Commitments to Group Elements. *IACR Cryptology ePrint Archive*, 2009:7, 2009.
10. 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 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.
11. 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 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.
12. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group encryption. In K. Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 181–199. Springer, 2007.
13. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *Theory of Cryptography (TCC 2006)*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.
14. 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 of *Lecture Notes in Computer Science*, pages 171–190. Springer, 2004.
15. 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 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2008.