The proceedings version of this paper appears in *ESORICS 2012*, Lecture Notes in Computer Science vol. ?, Springer, pp. ?–?, 2012. This is the full version of the paper.

Unique Group Signatures

Matthew Franklin * Haibin Zhang †

July 13, 2012

Abstract

We initiate the study of *unique group signature* such that signatures of the same message by the same user will always have a large common component (i.e., unique identifier). It enables an efficient detection algorithm, revealing the identities of illegal users, which is fundamentally different from previous primitives. We present a number of unique group signature schemes (without random oracles) under a variety of security models that extend the standard security models of ordinary group signatures. Our work is a beneficial step towards mitigating the well-known group signature paradox, and it also has many other interesting applications and efficiency implications.

Keywords: anonymity, anonymous authentication, detection algorithm, group signature, unique signature, verifiable random function.

*Department of Computer Science, University of California at Davis, Davis, California, 95616, USA. E-mail: franklin@cs.ucdavis.edu WWW: http://www.cs.ucdavis.edu/∼franklin/

†Department of Computer Science, University of California at Davis, Davis, California, 95616, USA. E-mail: hbzhang@cs.ucdavis.edu WWW: http://csiflabs.cs.ucdavis.edu/∼hbzhang/

1 Introduction

Group signatures, introduced by Chaum and van Heyst [25], are very useful tools in applications where the signer's privacy should be protected and in case of abuse some authorities can identify the misbehaving user. However, a well-known group signature "paradox" is that it is difficult for the group manager to identify a "misbehaving" user since all of signatures are anonymous. The group manager obviously cannot afford to open *all* of group signatures signed, for this is inefficient, and more importantly, it would compromise the privacy of every signer. Typically, the group manager identifies possible misbehaving users by observing whether some surprising documents are signed, or a huge amount of documents are signed within a short period, or some other "rules" are broken. These empirical test methods only provide the group manager with rough estimation about what signatures are suspicious. Trying to open and reveal the identities of suspicious signatures has a risk of jeopardizing legal users, while the illegal users may still be well-hidden.

Let us consider the motivating example of group signature due to Chaum and van Heyst [25]:

> "A company has several computers, each connected to the local network. Each department of that company has its own printer (also connected to the network) and only persons of that department are allowed to use their department's printer. Before printing, therefore, the printer must be convinced that the user is working in that department. At the same time, the company wants privacy: the user's name may not be revealed. If, however, someone discovers at the end of the day that a printer has been used too often, the director must be able to discover who misused that printer, to send him a bill."

The above opening policy, in practice, is problematic: it is not fair to reveal all identities of the persons who use the printer that is "used too much", since the identities of legal users might as well be revealed. It does not even make sense to say what is "used too much", as a dedicated adversary might use the same printer every day such that the times of uses are always slightly below the daily threshold, while the others would not dare to use the printer.

In this case, the rule that this company would like to enforce is to limit the number of times within some period that group members can use the service. If anyone who accessed the service beyond the allowed quota then its identity should be revealed by the group authority. At the same time, it is equally desirable for this company to detect *other* malicious printing any time—for instance, one printing process that uses up all the paper—which is prohibitive. In other words, once a user signs a message more than a predetermined value then it shall be almost always (efficiently) detected, but the group manager can always open signatures any time in case of other misbehavior.

We define *unique group signature* as a first step towards mitigating this paradox. We may say that a group signature scheme is "unique" if it is computationally infeasible for a signer to produce two different group signatures of the same message, such that both will pass the verification procedure (by analogy with the well-studied notion of uniqueness for ordinary signature schemes). We adopt a less stringent but more general definition such that if a signer produces two different group signatures of the same message, then both signatures will always have a large common component (hereinafter *unique identifier*) which is otherwise highly unlikely to occur. Ideally, if one user indeed signs two different signatures on one message then there should be an (efficient) *detection algorithm* that can reveal the identity of this user. With carefully defined other security notions, this primitive (still called unique group signature) serves as a perfect solution of dealing with the above problem.

In fact, a closely related question was first asked by Damgård, Dupont, and Pedersen [26] in their paper on *unclonable group identification scheme*. An unclonable group identification scheme

enables a user to authenticate to a server with complete anonymity provided that no other users try to use the first user's secret key to authenticate to the server within the same time period ("cloning attack"), while allowing the user's identity to be traced if they do misbehave in this way. In the introduction of their paper, Damgård et al. point out the inadequacy of existing group signature schemes for this purpose: "One may also consider using group signatures and have users identify themselves by signing a message chosen by the verifier (using his current system time, for instance). This achieves anonymity but does not protect against cloning. To do this, one would need the property that if the same user signs the same message twice, this would result in signatures that could be detected as coming from the same user. This...is actually false for known schemes, since these are probabilistic and produce randomly varying signature even if the message is fixed. A similar comment applies to identity escrow schemes...." Our work on unique group signature can be deemed as important progress on this interesting open question, and it also has many applications beyond unclonable group identification.

Informally speaking, unique group signatures (suitably defined) are adequate for unclonable group identification. For example, the user might send identification requests that include a signed message of the form "service_name || date" where || denotes concatenation. The server accepts if the signature is valid, and if it doesn't have the same large common component as another identification request received earlier in the day. For this application (and many others), we further need a *non-colliding* property for a unique group signature. A unique group signature is non-colliding if two different signers almost never produce the same unique identifier of the same message.

In another application, the user might send authentication requests that include a signed message of the form "service_ name || date || j", where j is any integer between 1 and the (daily) authentication bound k. The server accepts if the signature is valid, and if it doesn't have the same large common component as another authentication request from earlier in the day. This yields a variant of *periodic k-times anonymous authentication scheme* [18, 51, 53, 54]. Of course, many variations are possible by varying the space of messages to be signed.

Notice that for both of these applications, the server can choose whether or not to ask the group manager to reveal the identities of misbehaving users. For minor misbehavior (such as an attempt to authenticate to a service a few more times than the allowed bound, which might be due to innocent human or software or network error) the extra attempts could be detected and ignored. This lets the service provider reserve the relatively harsh penalty of anonymity revocation for more significant (sustained and persistent) misbehavior.

Also note that the deterministic and uniqueness property of our unique signature can lead to very fast processing of data. For example, a service provider carrying out a "first come, first kept" policy on a stream of ℓ requests would need only $\mathcal{O}(\ell \log \ell)$ operations (via appropriate tree structures), or $\mathcal{O}(\ell)$ expected operations (via hash tables). This is particularly useful when there are many users to be processed.

Though it can also deal with some applications that k-times anonymous authentication and more generalized e-token system [18] can, our primitive (even in this respect) is in essence a different one with distinct features and benefits (further discussion and comparison coming shortly).

TWO MODELS. This paper studies both the static group signature setting due to Bellare, Micciancio, and Warinschi (BMW) [8] and the dynamic group signature setting due to Bellare, Shi, and Zhang (BSZ) [9]. Intuitively, the static setting has a single authority (called the group manager), which the dynamic setting splits into two: an issuer for enrolling members, and an opener for tracing identities.

One might feel that studying static setting is not quite necessary as one could focus on the more involved and generalized dynamic group signature setting. First, this does not make sense

syntactically, since a dynamic group model is not simply an extension of a static group model. Static group signature models realistic scenarios that the group manager takes full control of the group user generation, and the secret signing key is distributed to each member, preferably, *without* interaction. (Otherwise, the members have to be supported by a trusted PKI, which usually is not the case in such a setting.) Instead, in the dynamic group setting, PKI support and interactive Join/Issue between the group issuer and group users are both *inevitable*. Second, this does not make sense *technically*, as we shall see, asking for non-interaction raises a few subtle issues in the static setting, making constructing an efficient scheme equally difficult. Third, we believe that static group signature is still *conceptually* more simple and starting from such a non-trivial point will make our presentation much clearer. Last, *constructionally*, our results for static unique group signature are both general and more efficient, while for the dynamic group setting our results are only semi-modular and a little less efficient.

How to model unique group signature? As an extension of group signature with an efficient detection algorithm, we provide the syntax and security definitions by extending ones for group signatures. We offer the "strongest" achievable definitions of security for both settings, but here we only highlight the case of dynamic model. On the one hand, the security requirements of dynamic unique group signatures are all simple and clear. Three of them (i.e., CCA-anonymity, traceability, and non-frameability) are based on previous security definitions of ordinary group signatures, while the uniqueness requirement is a quite natural and intuitive one. This is good, whether for understanding the definitions, or for designing the constructions. The uniqueness security notion formalizes the intuition that one signer can only sign one message once. Jumping ahead, we argue that defining uniqueness in the group signature setting raises subtle issues that must be carefully treated. The partly deterministic property of our unique group signature implies that we cannot target for the CCA-anonymity as formalized in [9]. We can still model the strongest achievable anonymity definition (which we still call CCA-anonymity): as long as the secret keys of challenge identities are not revealed and they did not sign the same message twice, no privacy is leaked, even if adversary fully controls the issuer and corrupts all the other users.

On the other hand, they are in fact very carefully defined *on the whole*. Recall that our goal is to present a group signature system where each group member can only sign any message once, equipped with an (efficient) detection algorithm such that the identities of ones who disobey such a rule can be revealed and should otherwise be never leaked. All of definitions of security are designed to this end. A few seemingly reasonable variants of definitions turn out to be inadequate.

The detection algorithm of our dynamic CCA-anonymous unique group signature is as simple as one could imagine: if the detection authority (i.e., the opener) ever found two different valid group signatures on the same message with the same unique identifier, then it runs the opening algorithm Open to extract their identities i and j (possibly i equals j), and adds them (it) to the misbehaving user set. However, all of these on detection algorithm have to be formally defined, otherwise it leaves one without any notion for what it means to have a *good* detection algorithm. Also note that our defined security properties do *not* even involve any properties of detection algorithms. Instead, we show that once the group system satisfies the four basic security requirements, it gives rise to a good (*complete* and *sound*) detection algorithm. We regard this particularly different strategy of dealing with security definitions as an important contribution of the paper.

Relaxations and Separations. We build the strongest achievable notions that we can imagine for our unique group signatures, and show that they meet our need. But there might be circumstances where one does not need such strong definitions. Also, one might be able to come up with more efficient constructions with relaxed security notions. Therefore, we consider various meaningful relaxations of secure unique group signatures. We give constructions for them and also show

separation results on definitions. We stress that these results actually serve to motivate our final instantiations.

CONSTRUCTIONS. In this paper, we present both the general constructions and efficient instantiations for both static and dynamic group models without relying on random oracles.

In the static setting, our general scheme follows the BMW two-level signature construction but uses a *verifiable random function* (VRF) [50] as the second-level signature. We also give a simpler construction for a unique group signature that is secure in a relaxed yet reasonable model. They together lead to our final efficient instantiation using Groth-Sahai proof system [40]. All of our constructions (either general or specific) are constant-size, and the instantiation is as efficient as the-state-of-the-art. We still find room for improvement to the first-level signature for *all* of the above, showing that a signature that is secure in the sense of unforgeability under random-message attacks suffices.

Our construction for the unique group signatures in the dynamic setting is semi-modular, and can be instantiated efficiently. The construction can even admit efficient *concurrent-join* which allows many entities concurrently engage in the Join/Issue protocol with the issuer.

In building the schemes, we identify new and useful techniques that we believe can be used in other privacy-preserving primitives. We highlight two of them. The first one is a PRF with NIZK proof that can degenerate into a unique signature. In many signature-related primitives, one not only need prove a deterministic function in a zero-knowledge sense but also prove knowledge of input to the function. There are many existing techniques, but ours gives the constructions that can be more efficient and rely on weaker assumptions. The other technique is what we call "*double-chaining certification*," which is used to achieve our unique group signature in the dynamic setting. In essence, this allows us to separate the unique identifier generation process from tracing process, thereby resulting in efficient and intuitive constructions.

APPLICATIONS AND COMPARISON BETWEEN OTHER PRIMITIVES. Our primitive is designed to mitigate the group signature paradox and also motivated by other privacy-preserving constructions, such as k-times anonymous authentication, unclonable group identification protocol, and more generalized e-token systems (periodic k-times anonymous authentication) [18]. The latter primitives are closely related to group signatures, but do *not* have an opening authority that can *always* de-anonymize signed messages.

On the other hand, our primitive can be as well used in applications where (periodic) k-times anonymous authentication is needed as illustrated earlier. Indeed, one can simply use a range proof to extend unique group signature to handle cases for $k > 1$, or one can easily achieve constant-size scheme by registering k public keys for one user at a time. (Note one of our instantiations supports efficient concurrent-join.) However, our primitive, in this respect, has distinct features.

First, the detection algorithms for other primitives are made public, meaning that if the a user signs more than the authentication bound k then its identity can be publicly known. This can be both *good* and *bad*: if an honest user accidentally signs slightly more than what is required because of hardware breakdown or clock desynchronization, then the public identity disclosure might not be the most reasonable choice. In fact, we are not aware of any implementations with such stringent mechanisms in *real* applications. Our unique group signature in the dynamic group setting supports in essence a different identity disclosure strategy where the detection authority (other than the group provider) is responsible to detect and reveal disobeyers by the detection algorithm Det. Anyone including the group provider and group members can find publicly misbehaving signatures and report to the detection authority. In our setting, this algorithm is even coupled with a detection proving algorithm DetProve that ensures the detection authority to behave correctly with a proof that the revealed identities are ones of the disobeyers. The opener reserves the right to open

persistent misbehaving users to the public, or contact and warn them privately, or send the identity and the corresponding proof into court as it sees fit. As far as we are concerned, two flavors of revelation are both interesting and should be used depending on specific applications.

Second, it was argued in [53], for their applications only, of course, that it is preferable that the users (who honestly follow the protocol specification) should enjoy anonymity even from the group provider. For the traditional group signature schemes, this requirement is not satisfied. But in the dynamic group model, the group provider might be a distinct entity from the opener who acts as the detection authority. Indeed, the reliance on some other party is inevitable if we do not want to enforce public identity discovery.

Third, in the context of k-times anonymous authentication, to the best of our knowledge, all previous constructions (e.g., [18, 51, 53, 54]) uses an idea originally from e-cash system. The detection algorithm of our primitive is fundamentally different from those. It turns out, perhaps somewhat counter-intuitive, that modeling and achieving "right" detection without using public discovery is actually more challenging (which has been highlighted earlier and see Section 3.3 for our treatment).

Last, as mentioned earlier, our primitives can be used in a more efficient way such that no detection algorithm is involved. Namely, the deterministic and unique property of our unique signature lead to very fast processing of data—on a stream of ℓ requests it would need only $\mathcal{O}(\ell \log \ell)$ operations via appropriate tree structures, or $\mathcal{O}(\ell)$ expected operations via hash tables. We are not aware of other primitives admitting such efficient detection.

FURTHER RELATED WORK. The topic of *unique signatures* and *verifiable random functions* have been an active research branch, which are of great value from both theoretical and applied perspectives [1, 17, 24, 27, 28, 41, 49, 50].

For a long time, most of the proposed group signatures rely on random oracles. Bellare, Micciancio and Warinschi [8] gave security definitions for group signature in the static setting where all members were given their signing keys and provided a trapdoor-permutation based construction. Boneh and Waters [15, 16] proposed group signatures under a weaker version of the BMW model where for the anonymity notion the adversary is not given any openings of group signatures [13]. The setting of dynamic groups was formalized by Bellare, Shi and Zhang [9], and they also offered a trapdoor-permutation based construction. Groth [38] suggested the first constant size group signature scheme without random oracles in the BSZ model but the constant is huge and the scheme is not yet practical. Groth [39] proposed a practical and fully anonymous group signature without random oracles. Recently, Abe et al. [2] proposed group signatures with efficient concurrent join. Traceable signatures, introduced by Kiayias, Tsiounis, and Yung [44], extend group signatures by enabling an efficient tracing of all signatures by a misbehaving party without revealing identities of any other users in the system. Chow [52] studied real traceable signatures that are essentially non-interactive k-times anonymous authentication schemes. Jarecki and Shmatikov [42] studies the linkable but anonymous authentication in the context of transaction escrow scheme. Similar notions also appear in the ring signature setting (i.e., linkable ring signatures and traceable ring signatures [5, 32, 33, 48, 55]).

2 Preliminaries

NOTATIONS. If x is a string then $|x|$ denotes its length. The empty string is denoted ε. If S is a set then $|S|$ denotes its size and $s \xleftarrow{\$} S$ denotes the operation of selecting an element s of S uniformly at random. $\emptyset$ denotes the empty set, while $\boldsymbol{\emptyset}$ denotes a vector of empty sets. If n

is an integer $[n]$ denotes the set $\{1, 2, \cdots, n\}$. If $\mathcal{A}$ is a randomized algorithm then we write $z \xleftarrow{\$} \mathcal{A}(x, y, \cdots)$ to indicate the operation that runs $\mathcal{A}$ on inputs $x, y, \cdots$ and a uniformly selected r from an appropriately required domain and outputs z. We write $z \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$ to indicate the operation that runs $\mathcal{A}$ having access to oracles $\mathcal{O}_1, \mathcal{O}_2, \cdots$ on inputs $x, y, \cdots$ and outputs z. A function $\epsilon(\lambda)$: $\mathbb{N} \to \mathbb{R}$ is *negligible* if, for any positive number d, there exists some constant $\lambda_0 \in \mathbb{N}$ such that $\epsilon(\lambda) < (1/\lambda)^d$ for any $\lambda > \lambda_0$.

2.1 Primitives

Pseudo-random function. *Pseudo-random function (PRF)* was first introduced by Goldreich, Goldwasser, and Micali [35]. We define a PRF family F: $\mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ where $\mathcal{S}$ is the *key space*, $\mathcal{X}$ is the *message space*, and $\mathcal{Y}$ is the *range*. We write $F_s(\cdot)$ to denote a PRF for every $s \in \mathcal{S}$. Let Γ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. Define the PRF advantage of $\mathcal{A}$ against F as

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{A}) = \Pr[\, s \xleftarrow{\$} \mathcal{S} :\ \mathcal{A}^{F_s} = 1\,] - \Pr[\, f \xleftarrow{\$} \Gamma :\ \mathcal{A}^{f} = 1\,].$$

Digital signatures. A *digital signature* $\mathcal{DS}$ consists of three algorithms $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vrf})$. A *key generation* algorithm Gen takes the security parameter λ and generates a *verification key* vk and a *signing key* sk. A *signing* algorithm Sig computes a signature σ for input message m using the signing key sk. A *verification* algorithm Vrf takes as input vk and a message-signature pair (m, σ) and outputs a single bit b. It is required that for all the messages m it holds that $\Pr[\mathsf{Vrf}(vk, m, \mathsf{Sig}(sk, m)) = 1] = 1$. The standard security notion of a digital signature is *existential unforgeability against adaptive chosen message attacks* [36]. Formally, given a signature scheme $\mathcal{DS}$, we associate to an adversary $\mathcal{A}$ the following experiment:

Experiment $\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{DS}}(\mathcal{A})$
$(vk, sk) \xleftarrow{\$} \mathcal{DS}.\mathsf{Gen}(1^\lambda)$
$(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sig}(sk, \cdot)}(vk)$
if $\mathsf{Vrf}(vk, m, \sigma) = 0$ **then return** 0
return 1

where m was not a query of $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathcal{DS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{DS}}(\mathcal{A}) = 1].$$

We also consider two less stringent notions of security which we call *existential unforgeability under a weak chosen message attack* (a.k.a. *a generic chosen message attack*), and *existential unforgeability under a random message attack* [29]. In a weak chosen message attack, we require that the adversary submit all signature queries before seeing the verification key vk. In a random message attack, the adversary given vk has access to an oracle that on query i returns a message-signature pair (r_i, σ_i) where each of the r_i's is uniformly and independently from the message space. We will also use a *strong one-time signature* scheme. It is secure against weak chosen message attack if no probabilistic polynomial-time adversary that has access to a single weak chosen message attack oracle can create a new message-signature pair (m, σ).

We also need a notion of *unique signature* (or *verifiable unpredictable function*), which asks that for any message there exists only one signature that can pass the verification algorithm.

Verifiable random function. *Verifiable random function* (VRF), introduced by Micali et al. [50], combines the properties of PRF and digital signature. Namely, a VRF is a PRF with

a non-interactive *proof of the correctness* of the input. A VRF $\mathcal{VRF}$ consists of four algorithms $(\mathsf{Gen}, \mathsf{Eva}, \mathsf{Prove}, \mathsf{Ver})$ with *input domain* $\mathcal{X}$ and *output range* $\mathcal{Y}$. A *key generation* algorithm Gen takes the security parameter λ and outputs a pair of keys (vk, sk) (where we use the same notation as digital signature, and there should be no ambiguity from context). An *evaluation* algorithm Eva takes as input sk and some x and outputs a value y. A *proving* algorithm Prove takes as input sk and some x and outputs ν which is the *proof* of correctness. A *verification* algorithm Ver takes as input vk and (x, y, ν) and outputs a single bit b. Formally, we require:

- $\mathsf{Provability/Correctness}$. If $y \leftarrow \mathsf{Eva}(sk, x)$ and $\nu \xleftarrow{\$} \mathsf{Prove}(sk, x)$ then $\mathsf{Ver}(vk, x, y, \nu) = 1$.
- $\mathsf{Unconditional\ Uniqueness}$. There do not exist $(vk, x, y_1, y_2, \nu_1, \nu_2)$ such that $y_1 \neq y_2$, but $\mathsf{Ver}(vk, x, y_1, \nu_1) = \mathsf{Ver}(vk, x, y_2, \nu_2) = 1$. Note that uniqueness in the definition above can be relaxed so as to hold *computationally* as opposed to *unconditionally*.
- $\mathsf{Pseudorandomness}$. We associate to an adversary $\mathcal{A}$ the following experiment:

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\mathrm{pr}}_{\mathcal{VRF}}(\mathcal{A}) \\
\quad (vk, sk) \xleftarrow{\$} \mathcal{VRF}.\mathsf{Gen}(1^\lambda) \\
\quad (x, \mathsf{s}) \xleftarrow{\$} \mathcal{A}^{\mathsf{Eva}(sk,\cdot),\mathsf{Prove}(sk,\cdot)}(pk) \\
\quad y_0 \leftarrow \mathsf{Eva}(sk, x);\ y_1 \xleftarrow{\$} \mathcal{Y} \\
\quad b \xleftarrow{\$} \{0,1\}; b' \xleftarrow{\$} \mathcal{A}(y_b, \mathsf{s}) \\
\quad \textbf{if } b' \neq b \textbf{ then return } 0 \\
\quad \textbf{return } 1
\end{array}
$$

  where the adversary did *not* query its oracles with x. We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{pr}}_{\mathcal{VRF}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{pr}}_{\mathcal{VRF}}(\mathcal{A}) = 1] - 1/2.$$

  A VRF scheme $\mathcal{VRF}$ is said to have the pseudorandomness property if for any polynomial-time adversary $\mathcal{A}$ the function $\mathbf{Adv}^{\mathrm{pr}}_{\mathcal{VRF}}(\mathcal{A})$ is negligible in the security parameter.

2.2 Complexity Assumptions

Bilinear groups. We recall the definition of a *bilinear group* $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order q, g and h generate $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable bilinear map. We call a bilinear group *symmetric* if $\mathbb{G}_1 = \mathbb{G}_2$, otherwise we call it *asymmetric*. All of assumptions that we use should hold in certain bilinear groups. We refer the reader to [34] for details.

Symmetric eXternal Diffie-Hellman assumption (SXDH) [56]. Given a bilinear group of prime order as described above, the symmetric external Diffie-Hellman assumption is that the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. This setting implies that there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Decisional Linear assumption (DLIN) [13]. This assumption is first proposed in the setting of symmetric bilinear groups of prime order: Given $(g, g^\alpha, g^\beta, g^{r\alpha}, g^{s\beta}, g^t)$, it is computationally hard to distinguish whether $t = r + s$ or t is random. This assumption can be generalized to hold in other settings.

Diffie-Hellman Inversion assumption (DHI) [28]. Given $g, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^l} \in \mathbb{G}$, where $\alpha \xleftarrow{\$} \mathbb{Z}_q$, it is computationally hard to compute $g^{1/\alpha}$. For our constructions, we require that this assumption holds in $\mathbb{G}_1$ or $\mathbb{G}_2$.

Decisional Diffie-Hellman Inversion assumption (DDHI)[7, 19]. Given input $g, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^l} \in \mathbb{G}$, where $\alpha \xleftarrow{\$} \mathbb{Z}_q$, it is computationally hard to distinguish $g^{1/\alpha}$ from a random element of $\mathbb{G}$. This assumption is generically false in the symmetric setting. We require that this assumption holds either in $\mathbb{G}_1$ or $\mathbb{G}_2$. Also note that this assumption is slightly stronger than the DBDHI assumption in [12, 28].

Strong Decisional Diffie-Hellman Inversion assumption (SDDHI) [18]. This assumption is proposed in standard groups. Given a random generator $g, g^{\alpha} \in \mathbb{G}$ where $\alpha \xleftarrow{\$} \mathbb{Z}_q$, and an oracle $\mathcal{O}_{\alpha}$ that on input m outputs $g^{1/(\alpha+m)}$, it is computationally hard to distinguish $g^{1/(\alpha+m')}$ from random on a new m'. Apparently, this assumption can be generalized to hold in a group with a bilinear map (which must not be symmetric).

Strong Diffie-Hellman Inversion assumption (SDHI). The above SDDHI assumption simply asks the function $g^{1/(\alpha+\cdot)}$ to behave like a "random" function, but for our constructions, we need a weak version of the assumption that we call strong Diffie-Hellman inversion assumption (SDHI) where the above function is only needed to be "unpredictable": Given a random generator $g, g^{\alpha} \in \mathbb{G}$ where $\alpha \xleftarrow{\$} \mathbb{Z}_q$, and an oracle $\mathcal{O}_{\alpha}$ that on input m outputs $g^{1/(\alpha+m)}$, it is computationally hard to compute $g^{1/(\alpha+m')}$ on a new m'.

2.3 Groth-Sahai Proof System and Related Tools

Groth-Sahai proof system [40] provides efficient (composable) NIWI proofs and NIZK proofs[1] in the *common reference string model* for a large set of statements involving bilinear groups, including *pairing product equations*, *multi-scalar multiplication equations*, and *quadratic equations*. This system can be instantiated under three assumptions: SXDH assumption (in asymmetric bilinear groups), DLIN assumption (in symmetric bilinear groups), and subgroup decision assumption (in composite order bilinear groups). There are two types of common reference strings (which are computationally indistinguishable) yielding perfect soundness and perfect witness-indistinguishability (or zero-knowledge) respectively. A Groth-Sahai proof system consists of four algorithms $(\mathsf{Gen}, P, V, \mathsf{Extr})$. The *key generation* algorithm Gen takes a security parameter and outputs a common reference string crs together with an *extraction key* xk. The *prover* P takes as input crs and witnesses of equations and outputs a proof π. The *verifier* V takes as input crs and π and outputs a bit b with respect to a set of equations. The Extr algorithm taking as input the extraction key xk can extract the *group elements witnesses*. Therefore, for the equations whose witnesses are group elements the above proof as well provides *proofs of knowledge(PoK)*. Such NIZKPoK proof systems are powerful tools to construct signature-related protocols. However, it cannot extract the witnesses which are scalars but some function f of the scalar witnesses. This is formally called *f-extractable non-interactive proofs of knowledge* [6]. It therefore entails stronger F-unforgeability notion (to construct their P-signature scheme). The other solution is to use structure-preserving signatures [2] which are compatible with Groth-Sahai proof system. A signature scheme is *structure-preserving* if its verification keys, messages, and signatures are group elements and verification algorithm is a set of pairing product equations.

We shall use the blind signature scheme by Fuchsbauer [31]. We briefly recall this primitive as follows. The protocol is run between a signer who has the secret key x and a user who wants to get

[1] See Appendix A for standard notions of NIWI and NIZK proofs.

a blind signature on (M, N) which is a DH pair w.r.t. two fixed random generators g and h. The user sends a randomized message and obtains a pre-signature from the signer, and eventually the user can produce a blind signature on (M, N). The security of this scheme (as a blind signature and as a "signing on committed value" protocol [20–22]) can be justified under ADH-SDH and SXDH (or ASH-SDH, WF-CDH and DLIN in symmetric groups) assumptions [31].

We also use two other tools related to Groth-Sahai proof system. One is a PRF with a NIZK proof by Belenkiy et al. [7], which builds on a VRF by Dodis and Yampolskiy [28]. It can be instantiated in both the SXDH and DLIN setup. We will use a variant of this function.

The other tool is Kiltz's selective-tag weakly CCA-secure encryption [46]. We now recall the scheme in the context of symmetric bilinear groups $(q, \mathbb{G}, \mathbb{G}_T, e, g)$. The public key pk is $(X_1, X_2, Y_1, Y_2) \in \mathbb{G}^4$ and the secret key is $(x_1, x_2) \in \mathbb{Z}_q^2$ such that $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$. To encrypt a message m with a tag t, one computes ciphertext $C = (X_1^{r_1}, X_2^{r_2}, (g^t Y_1)^{r_1}, (g^t Y_2)^{r_2}, g^{r_1+r_2} m)$ where $(r_1, r_2) \in \mathbb{Z}_q^2$ are randomness used. Given a ciphertext $(c_1, c_2, d_1, d_2, c_0)$, the validity is publicly verified by checking if $e(X_1, d_1) = e(c_1, g^t y_1)$ and $e(X_2, d_2) = e(c_2, g^t y_2)$. If this is the case then the receiver (owning the secret key) computes $m = c_0 c_1^{-1/x_1} c_2^{-1/x_2}$. The scheme is selective-tag weakly CCA-secure under DLIN assumption. Groth [39] used it to construct a CCA-anonymous group signature scheme.

3 Unique Group Signature Models

In this section we present models of unique group signatures in the static setting (following BMW [8]) and in the dynamic setting (following BSZ [9]).

3.1 Static Setting Model

Following [8], a *static group signature scheme* $\mathcal{SGS}$ consists of four algorithms $(\mathsf{GK}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open})$. There is only one group authority which we call the *group manager*. The *group key generation* algorithm GK takes as input the security parameter λ to form a fixed-size group with n members where n may be related to λ, returning a tuple $(gpk, gmsk, \boldsymbol{gsk})$, where gpk is the *group public key*, $gmsk$ is the *group manager secret key*, and $\boldsymbol{gsk}$ is an n-vector of *secret signing keys* with $gsk[i]$ for each user i. The secret keys are usually distributed to members without interaction. The *group signing* algorithm GS takes as input $gsk[i]$ and a message m to return a signature σ under $gsk[i]$. The *group verification* algorithm GV takes as input the group public key gpk, a message m, and a signature σ for m to return a single bit b. We say that σ is a *valid* signature of m if $\mathsf{GV}(gpk, m, \sigma) = 1$. The *opening* algorithm Open takes the group public key gpk, group manager secret key $gmsk$, a message m, and a signature σ to return an identity i or $\perp$ (indicating failure). Basic *correctness* property is required: for all security parameter λ and integer n, all $(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathsf{GK}(1^\lambda)$, all $i \in [n]$, and all message $m \in \{0,1\}^*$, it holds that $\mathsf{GV}(gpk, m, \mathsf{GS}(gsk[i], m)) = 1$ and $\mathsf{Open}(gpk, gmsk, m, \mathsf{GS}(gsk[i], m)) = i$.

For our purposes, we consider *static unique group signatures* where the signatures should have the form of $(m, \sigma) = (m, \tau, \psi)$ where τ is the *unique identifier* for the message m and some group member i, and ψ is the rest of the signature. (One can view the unique identifier as a special *tag*.) We define for static unique group signature three security requirements: uniqueness, anonymity, and traceability. The uniqueness requirement formalizes the intuition that one user can only sign one message once, while the last two requirements are adapted from ones for the regular static group signatures with the restraints of being unique.[2]

[2]For this reason, we choose to first present the uniqueness property.

Uniqueness. Unlike defining uniqueness for a stand-alone signature (i.e., unique signature), it is "tricky" to do so in the context of group signature that involves multiple users. We achieve this goal step by step. In general, uniqueness is to protect the system from adversarial group members. In this setting, any single group member should not generate more than one valid signatures for any message m. In this sense, we may say that a group signature $\mathcal{SGS}$ satisfies the uniqueness requirement, if for any polynomial-time adversary $\mathcal{A}$ in possession of a secret signing key for member j, it holds that

$$\Pr[(m, \sigma_1 = (\tau_1, \psi_1), \sigma_2 = (\tau_2, \psi_2)) \xleftarrow{\$} \mathcal{A}(gpk, gsk[j]) \\ : \tau_1 = \tau_2 \textbf{ and } \mathsf{GV}(gpk, m, \sigma_1) = \mathsf{GV}(gpk, m, \sigma_2) = 1] \leq \epsilon(\lambda).$$

The above formalization captures the spirit of the uniqueness property of group signatures by analogy with the well-studied notion of uniqueness for ordinary unique signature schemes. However, it is not quite adequate, for, an adversary may (adaptively) corrupt multiple group members to gain an additional advantage. We thus give adversary access to a *user secret oracle*, $\mathsf{USK}(\cdot)$, which, when queried with an identity $i \in [n]$, answers with the secret signing key $gsk[i]$ for user i. In the static group signature setting, once the secret key of a user is revealed then it is said to be *corrupted*.[3] We let $\mathtt{CU}$ denote a set of corrupted users. Since the group has a fixed-size n, a set of uncorrupted (i.e., honest) users is $[n]/\mathtt{CU}$. The adversary is also given access to a *user signing oracle*, $\mathsf{GS}(\cdot, \cdot)$, which when queried with an identity i of a user and a message m, returns $\mathsf{GS}(gsk[i], m)$. Note that we do not require that adversary only ask uncorrupted users for this oracle. Let $\mathtt{GS}$ denote a set of message-signature pairs queried via the $\mathsf{GS}(\cdot, \cdot)$ oracle. We write $\mathtt{GS}_m$ to denote a set of users with which adversary calls $\mathsf{GS}(\cdot, m)$. We write $\mathtt{GS}_{\mathbf{M}}$ where $\mathbf{M}$ is a set of the messages queried to denote a vector of sets with $\mathtt{GS}_m$ for each $m \in \mathbf{M}$. For maximal security, we also provide adversary with the secret of the group manager $gmsk$. Formally, given a static signature scheme $\mathcal{SGS}$ of a fixed-size n, we associate to an adversary $\mathcal{A}$ the following experiment:

Experiment $\mathbf{Exp}^{\text{unique}}_{\mathcal{SGS},n}(\mathcal{A})$

$(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathcal{SGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{GS} \leftarrow \emptyset$
$(m, \sigma_1, \cdots, \sigma_{|\mathtt{CU}|+1}) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot), \mathsf{GS}(\cdot,\cdot)}(gpk, gmsk)$
for $i \leftarrow 1$ **to** $|\mathtt{CU}| + 1$ **do**
  if $\mathsf{GV}(gpk, m, \sigma_i) = 0$ **or** $(m, \sigma_i) \in \mathtt{GS}$ **then return** 0
for $i, j \leftarrow 1$ **to** $|\mathtt{CU}| + 1$ **do**
  if $i \neq j$ **and** $\tau_i = \tau_j$ **then return** 0
return 1

where, above, each σ_i is of the form (τ_i, ψ_i). We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\text{unique}}_{\mathcal{SGS},n}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{unique}}_{\mathcal{SGS},n}(\mathcal{A}) = 1].$$

In the above experiment, adversary is expected to output *exactly* $|\mathtt{CU}| + 1$ *new* and *valid* signatures which have *distinct* unique identifiers with respect to the *same* message.

A CAVEAT. We first emphasize that the above notion is the one that we shall use in this paper. However, we do point out some "inadequacies" by considering the following scenario: it is possible to design a scheme such that a single user can only sign any message once, but when they collude

[3] Jumping ahead, we stress that this is not the case for the dynamic group signature scheme, and the circumstances there are more complex.

they are able to gain an additional advantage. (In Appendix B, we give a separation result, showing that there exist schemes satisfying a weakened uniqueness definition where the adversary can only corrupt one user but not the standard uniqueness that we defined.) Our security experiment above does not well capture this situation. Informally speaking, it is entirely possible that some of keys correspond to one same unique identifier (i.e., they "collide"), while some other keys might generate more unique identifiers than *required*. To put it differently, it might be the case that a set of users of size k who do not collude ought to create $k-1$ unique identifiers as two of them collide, but when they collude they can create k unique identifiers. This does not contract our uniqueness security, but such a collusion clearly makes them sign messages beyond their own.

Non-colliding property. In light of this (and as required by some applications mentioned in the introduction), we impose a restriction on our static unique group signature. We say that a group signature is *non-colliding* if any of two different (honest) signers (who follow the scheme specification) almost never produce the same *unique identifier* of the same message. We stress that one should think of this as a correctness property rather than a security notion. More formally, for all security parameter λ and integer n, all $(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathsf{GK}(1^\lambda)$, all $i, j \in [n]$ and $i \neq j$, and all message $m \in \{0,1\}^*$, it holds that

$$\Pr[(\tau_i, \psi_i) \xleftarrow{\$} \mathsf{GS}(gsk[i], m); (\tau_j, \psi_j) \xleftarrow{\$} \mathsf{GS}(gsk[j], m) : \ \tau_i = \tau_j] \leq \epsilon(\lambda).$$

Above, the probability is taken over the coins of the group key generation algorithm and group signing algorithm.

The above requirement can resolve the "issue" above. Indeed, if the above-mentioned circumstance happens then an adversary who corrupted a set of group members can always "honestly" generate signatures *again* and pick "enough" signatures with different unique identifiers to attack the uniqueness property. It also makes our primitive justifiable in a few applications—only via this property one can safely achieve the functionality of restricted anonymous authentication (as mentioned in the introduction). Jumping ahead, we claim that the non-colliding property is needed as well in justifying the security of the detection algorithm of unique group signature.

One may also consider a natural way of solving this problem by formalizing the intuition that even a collusion of group members cannot produce more unique identifiers than they *are able to* (if they follow the specification of the protocols). This is not that easy to formalize, and more importantly, this is not really about uniqueness.

Another natural concern is what if the adversary both asks the $\mathtt{HSK}(\cdot)$ and $\mathtt{GS}(\cdot,\cdot)$ for some user i. Recall that the adversary is expected to output *new* message-signature pairs. Following our uniqueness definition, the adversary should not output the signatures returned by the $\mathtt{GS}(\cdot,\cdot)$ for some corrupted users. (We stress that this requirement is crucial for our formalization.) This, however, does not restrict the adversary's capability of attacking the uniqueness experiment, since it can easily honestly generate other signatures using the secret key for the corrupted user as long as the group signature algorithm is not fully deterministic.

Discussion. Since the main goal of setting up the uniqueness requirement is to protect the system from the dishonest users, it is debatable whether or not to provide adversary with the group manager secret key $gmsk$. It does make sense and suffices for many applications if we did not give adversary this secret, but this provides a less strong definition. A natural separation result regarding this will be provided in Section 4.

Our uniqueness definition is described in a *computational* sense where it should hold for polynomial time adversaries. This can, however, be easily defined in the *unconditional* setting to protect against all-powerful adversaries—as usually required in the conventional uniqueness definition of security for a unique signature scheme.

Note that this definition remains non-trivial even when the adversary has made the maximum number of corruption queries (i.e., when $|\mathtt{CU}| = n$).

In defining the above experiment, we could let the adversary output N signatures (rather than exactly $|\mathtt{CU}| + 1$ signatures). We impose no further restrictions on N as long as it contains valid signatures which has $|\mathtt{CU}| + 1$ different unique identifiers. (Also note we do not even need to ask $N \geq |\mathtt{CU}| + 1$—which has to be the case if an adversary is intended to win.) To allow flexible verification queries turns out to be a slightly stronger definition. But this definitional choice is nonsignificant since the advantage of an adversary in this alternative definition can increase by at most N.

In fact, there are other seemingly "correct" definition of uniqueness. The reason why we do not choose them is due to the overall consideration of a "good" detection algorithm as we shall describe shortly. Namely, some of these definitions (even with carefully defined other definitions) do not seem to lead to a provably secure detection algorithm.

Anonymity. Due to the uniqueness property, we cannot achieve the strongest anonymity definition of security as defined in BMW [8]. (The group signature signed by each member i is a partly deterministic function of the gpk, $gsk[i]$, and the message m. If the adversary is given all of the secret keys $\boldsymbol{gsk}$ then it can attack the full-anonymity game simply by re-computing.) Thus a slightly weaker yet still very strong anonymity security notion is used: the adversary can adaptively corrupt the users of the group; for uncorrupted users, adversary is given a signing oracle; in the $\mathsf{challenge}$ stage, adversary is not allowed to submit challenge queries with identities of corrupted users, and not allowed to submit challenge queries with at least one of the identities and the message being the same as ones queried before. We write $\mathsf{Open}(\cdot,\cdot)$ to denote the *opening oracle*, which when queried with a message m and a candidate signature σ, answers with $\mathsf{Open}(gpk, gmsk, m, \sigma)$. Specifically, given a static group signature scheme $\mathcal{SGS}$ of a fixed-size n, we associate to an adversary $\mathcal{A}$ the following experiment:

Experiment $\mathbf{Exp}^{\mathrm{anon}}_{\mathcal{SGS},n}(\mathcal{A})$

$(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathcal{SGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{GS}_{\mathbf{M}} \leftarrow \boldsymbol{\emptyset}$

$(i_0, i_1, m, \mathsf{s}) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot),\mathsf{Open}(\cdot,\cdot)}(\mathsf{find}, gpk)$

$b \xleftarrow{\$} \{0,1\}$; $\sigma \xleftarrow{\$} \mathsf{GS}(gsk[i_b], m)$

$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot),\mathsf{Open}(\cdot,\cdot)}(\mathsf{guess}, \sigma, \mathsf{s})$

if $b' \neq b$ **then return** 0

return 1

where it is mandated that for each $d \in \{0,1\}$ we have $i_d \notin \mathtt{CU}$ and $i_d \notin \mathtt{GS}_m$, and in the guess phase the adversary $\mathcal{A}$ did not query $\mathsf{Open}(\cdot,\cdot)$ with m and σ. We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{SGS},n}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{anon}}_{\mathcal{SGS},n}(\mathcal{A}) = 1] - 1/2.$$

The above formalization considers the strongest possible anonymity definition in the context of static unique group signature. The adversary is given access to the opening oracle in both phases of the experiment.

We use the term "CPA-anonymity" to denote the following weakening of the security definition for anonymity in the static setting [13]: The adversary is never given access to the opening oracle (but otherwise the definition is identical). For consistency, we interchangeably use "CCA-anonymity" and "anonymity" to describe the stronger anonymity definition (with access to the

opening oracle).

Traceability. The traceability security definition is the same as one in BMW [8]. We recall it by considering the following experiment that is associated to an adversary $\mathcal{A}$:

Experiment $\mathbf{Exp}^{\text{trace}}_{\mathcal{SGS},n}(\mathcal{A})$

$(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathcal{SGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{GS}_\mathbf{M} \leftarrow \boldsymbol{\emptyset}$
$(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot)}(gpk, gmsk)$
if $\mathsf{GV}(gpk, m, \sigma) = 0$ **then return** 0
if $\mathsf{Open}(m, \sigma) = \perp$ **then return** 1
if $\mathsf{Open}(m, \sigma) = i$ **and** $i \notin \mathtt{CU}$ **and** $i \notin \mathtt{GS}_m$ **then return** 1
return 0

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{SGS},n}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{trace}}_{\mathcal{SGS},n}(\mathcal{A}) = 1].$$

It is in fact possible to combine the uniqueness and traceability definitions of security to form an integrated one, but we choose not to, for we think that this may give readers less intuitive definitions. (A similar argument applies to the dynamic case.)

Remark and Discussion. BMW [8] shows that anonymity and traceability imply *all* existing (informal) security requirements. Specifically, unforgeability, exculpability, functional traceability, coalition resistance, framing are implied by traceability, while anonymity (without opening oracle) and unlinkability are implied by anonymity. In our setting, the former clearly holds as well, since our traceability definition is the same as the one in BMW. On the other hand, we intentionally want to link two signatures signed by the same user on the same message. The uniqueness definition is a step to achieve this goal. Our anonymity definition is the strongest achievable one (that we can imagine) with the restrains of being unique. In BMW, the authors comment that "$\cdots$ anonymity and unlinkability are technically the same property, and only the easier to define anonymity property needs to be included in the security definition." While we consent this viewpoint if one pursues complete anonymity, our results (in the following sections) in fact show that one can achieve meaningful linkability with still strong and maximal anonymity.

3.2 Dynamic Setting Model

In the dynamic group setting, there are two more features: it allows one to add members to the group; the authority is separated into the *opener* and the *issuer*. An issuer is responsible to enroll members, while an opener traces the identities of signatures signed by the users enrolled. The signatures of members should be otherwise anonymous. The issuer maintains a registry $\boldsymbol{reg}$ which contains information that both the issuer and user agree on. The registry $\boldsymbol{reg}$ is a vector with $reg[i]$ containing the identification information of member i. We assume, without loss of generality, that the content in $reg[i]$ is signed by the user i (using, for instance, its external PKI secret key). Enrollment involves an explicit $\mathsf{Join}/\mathsf{Issue}$ protocol, at the end of which the user will get its secret signing key $gsk[i]$.

A *dynamic group signature scheme* $\mathcal{DGS} = (\mathsf{GK}, \mathsf{Join}/\mathsf{Issue}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open}, \mathsf{Judge})$ is specified as follows:

- GK: The *group key generation* algorithm takes as input the security parameter λ outputs a *group public key* gpk, the *issuer key* ik that is provided to the issuer, and the *opening key* ok that is provided to the opener.
- UK: A user i who wants to join the group runs the *user key generation* algorithm taking as input the security parameter λ, and outputs the *user public and private key pair* $(upk[i], usk[i])$.
- Join/Issue: This is an *interactive* algorithm between a user and the issuer. If being successful, they register user's public key in $reg[i]$, and the user obtains its *secret signing key* $gsk[i]$.
- GS: The *group signing* algorithm GS takes as input $gsk[i]$, and a message m to return a signature σ under $gsk[i]$.
- GV: The *group verification* algorithm GV takes as input the group public key gpk, a message m, a signature σ for m to return a single bit b. We say that σ is a *valid* signature of m if $\mathsf{GV}(gpk, m, \sigma) = 1$.
- Open: The opener, who has access to the registration table ***reg*** and takes as input the group public key gpk, its opening key ok, a message m, and a valid signature σ on m to output (i, ω). If $i > 0$ then it means the opener identifies that user i produced the signature; ω is a *proof* to support its claim that user i indeed signed the message. If $i = 0$ then it is claiming that the issuer forged (m, σ).
- Judge: The *judge* algorithm, taking as input gpk, the opening (i, ω), a message m, and a valid signature σ of m, is to verify that openings of signatures are indeed correct. We say that the opening is *correct* if the judge algorithm returns 1.

As in this static setting, we also consider *dynamic unique group signatures* where the signatures should have the form of $(m, \sigma) = (m, \tau, \psi)$ where τ is the *unique identifier* for the message m and some group member i, and ψ is the rest of the signature. A secure unique group signature in the dynamic setting should satisfy correctness and four security notions: uniqueness, anonymity, traceability and non-frameability.

We write HU and CU to denote a set of honest users and corrupted users respectively. Below, we briefly recall the oracles provided to adversaries. Please refer [9] for further details.

- $\mathsf{AddU}(\cdot)$–*add user oracle*: adversary can add user i to the honest user list. We do not consider honest users who do not faithfully complete the Join/Issue protocol.
- $\mathsf{Crpt}(\cdot, \cdot)$–*corrupt user oracle*: adversary corrupts user i who wants to join the group and sets its user public key as $upk[i]$.
- $\mathsf{SndToI}(\cdot, \cdot)$–*send to issuer oracle*: adversary runs a corrupted user i to run Join/Issue algorithm with the honest issuer.
- $\mathsf{SndToU}(\cdot, \cdot)$–*send to user oracle*: adversary uses a corrupted group issuer to run Join/Issue algorithm with an honest user i.
- $\mathsf{USK}(\cdot)$–*user secret keys oracle*: adversary can call this oracle to get the user signing key $gsk[i]$ and user private key $usk[i]$ of i. (In this setting, obtaining the secret keys of the user does not mean that it corrupted the user which can only be achieved via $\mathsf{Crpt}(\cdot, \cdot)$ oracle.)
- $\mathsf{RReg}(\cdot)$–*read registration oracle*: adversary can call it with argument i to read the content of $reg[i]$.
- $\mathsf{WReg}(\cdot, \cdot)$–*write registration oracle*: adversary is given a write access to the content of $reg[i]$ by calling the oracle with argument i.
- $\mathsf{GS}(\cdot, \cdot)$–*signing oracle*: adversary is given the group signature for an honest user i and message m.
- $\mathsf{Open}(\cdot, \cdot)$–*opening oracle*: given a message-signature pair, this oracle outputs the identity of signer and the corresponding proof.

Note that we do not include *challenge oracle*, for we would like to defer it to the anonymity game. We write $\texttt{GS}_m$ to denote a set of *honest* users which adversary calls $\mathsf{GS}(\cdot,\cdot)$ on m. We write $\texttt{GS}_{\mathbf{M}}$ where $\mathbf{M}$ is a set of *all* messages queried to denote a vector of sets with $\texttt{GS}_m$ for each $m \in \mathbf{M}$. We also use the notation $\texttt{USK-HU}$ to denote a set of *honest* users whose secret keys are given to the adversary.

Correctness. The correctness requirement asks that the signature should be valid on even adversarially chosen message, the opening algorithm should identify the real signer, and the judge algorithm should accept the opening. For a dynamic group signature scheme $\mathcal{DGS}$, we formalize correctness using the following experiment involving a computationally unbounded adversary $\mathcal{A}$:

Experiment $\mathbf{Exp}^{\text{correct}}_{\mathcal{DGS}}(\mathcal{A})$
$(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\texttt{CU} \leftarrow \emptyset$; $\texttt{HU} \leftarrow \emptyset$
$(i, m) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}(\cdot),\mathsf{RReg}(\cdot)}(gpk)$
$\sigma \xleftarrow{\$} \mathsf{GS}(gsk[i], m)$
if $\mathsf{GV}(gpk, m, \sigma) = 0$ **then return** 1
$\mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma) = (j, \omega)$
if $i \neq j$ **then return** 1
if $\mathsf{Judge}(gpk, (i, \omega), m, \sigma, \boldsymbol{reg}) = 0$ **then return** 1
return 0

We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\text{correct}}_{\mathcal{DGS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{correct}}_{\mathcal{DGS}}(\mathcal{A}) = 1].$$

We say that $\mathcal{DGS}$ is *correct* if $\mathbf{Adv}^{\text{correct}}_{\mathcal{DGS}}(\mathcal{A}) = 0$ for any computationally unbounded adversary $\mathcal{A}$.

Uniqueness. Compared to the uniqueness definition in the static group signature, the case here is more involved and the adversary is also provided a few additional attack capabilities. For clarity, consider the following typical scenario: A service provider has a list of registered public keys corresponding to all users who have purchased a single access to some confidential service for that day (requiring anonymous authentication). It is important that no set of users could sign messages beyond their own, since otherwise that set of users could purchase the daily service legitimately, and then maliciously gain more daily accesses than they paid for.

The adversary is not given issuer key ik, otherwise it can create arbitrarily many dummy honest users and easily attack the uniqueness property. Instead, the adversary is allowed to create honest group members using $\mathsf{AddU}(\cdot)$ oracle, and obtains both of personal private key and user signing key of a user using $\mathsf{USK}(\cdot)$ oracle and obtains signatures of an honest user using $\mathsf{GS}(\cdot,\cdot)$. The adversary is also allowed to run $\mathsf{RReg}(\cdot)$ oracle. Besides, it can corrupt users and interact with the issuer using $\mathsf{Crpt}(\cdot,\cdot)$ and $\mathsf{SndToI}(\cdot,\cdot)$ oracles. We consider the following experiment $\mathbf{Exp}^{\text{unique}}_{\mathcal{DGS}}(\mathcal{A})$:

Experiment $\mathbf{Exp}^{\text{unique}}_{\mathcal{DGS}}(\mathcal{A})$
$(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\texttt{CU} \leftarrow \emptyset$; $\texttt{HU} \leftarrow \emptyset$; $\texttt{GS} \leftarrow \emptyset$
$(m, \sigma_1, \cdots, \sigma_{|\texttt{CU}|+|\texttt{USK-HU}|+1}) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot),\mathsf{SndToI}(\cdot,\cdot),\mathsf{AddU}(\cdot),\mathsf{RReg}(\cdot),\mathsf{Crpt}(\cdot,\cdot)}(gpk, ok)$
for $i \leftarrow 1$ **to** $|\texttt{CU}| + |\texttt{USK-HU}| + 1$ **do**
  if $\mathsf{GV}(gpk, m, \sigma_i) = 0$ **or** $(m, \sigma_i) \in \texttt{GS}$ **then return** 0

for $i, j \leftarrow 1$ **to** $|\mathtt{CU}| + |\mathtt{USK\text{-}HU}| + 1$ **do**
  if $i \neq j$ **and** $\tau_i = \tau_j$ **then return** 0
return 1

where, above, each σ_i is of the form (τ_i, ψ_i). We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\text{unique}}_{\mathcal{DGS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{unique}}_{\mathcal{DGS}}(\mathcal{A}) = 1].$$

Non-colliding property. The same caveat applies to dynamic unique group signatures. One may think that it would not be adequate to define only the non-colliding property for an honest user (who follows the protocol) as we did in the static group setting since the secret signing key or a group signing algorithm of a corrupted user in this case is not even well-defined. In fact, we only have to define the non-colliding property. It would be the adversary's own failure if, via interacting with an honest issuer, the adversary gets a key (if this key really exists) and signs a signature that collides with some other group signature (whether that other signature was honestly generated or not).

We formally define non-colliding property for unique signature in the dynamic setting involving an adversary:

Experiment $\mathbf{Exp}^{\text{nc}}_{\mathcal{DGS}}(\mathcal{A})$
  $(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{HU} \leftarrow \emptyset$
  $(i, j, m) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}(\cdot),\mathsf{RReg}(\cdot)}(gpk)$ where $i \neq j$
  $(\tau_i, \psi_i) \xleftarrow{\$} \mathsf{GS}(gsk[i], m)$; $(\tau_j, \psi_j) \xleftarrow{\$} \mathsf{GS}(gsk[j], m)$
  if $\tau_i = \tau_j$ **then return** 1
  return 0

We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\text{nc}}_{\mathcal{DGS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{nc}}_{\mathcal{DGS}}(\mathcal{A}) = 1].$$

We say that $\mathcal{DGS}$ is *non-colliding* if $\mathbf{Adv}^{\text{nc}}_{\mathcal{DGS}}(\mathcal{A}) \leq \epsilon(\lambda)$ for any polynomial-time adversary $\mathcal{A}$.

Anonymity. The basic idea is the same as the one for the static group signature setting. We cannot target the strongest anonymity definition as defined in BMW [8] due to the uniqueness property. Nevertheless, we still provide the strongest achievable anonymity definition that we can imagine. Given a dynamic group signature scheme $\mathcal{DGS}$, we associate to an adversary $\mathcal{A}$ the following experiment:

Experiment $\mathbf{Exp}^{\text{anon}}_{\mathcal{DGS}}(\mathcal{A})$
  $(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{HU} \leftarrow \emptyset$; $\mathtt{GS}_{\mathbf{M}} \leftarrow \mathbf{\emptyset}$
  $(i_0, i_1, m, \mathsf{s}) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot),\mathsf{Open}(\cdot,\cdot),\mathsf{Crpt}(\cdot,\cdot),\mathsf{SndToU}(\cdot,\cdot),\mathsf{WReg}(\cdot,\cdot)}(\mathsf{find}, gpk, ik)$
  $b \xleftarrow{\$} \{0,1\}$; $\sigma \xleftarrow{\$} \mathsf{GS}(gsk[i_b], m)$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot),\mathsf{Open}(\cdot,\cdot),\mathsf{Crpt}(\cdot,\cdot),\mathsf{SndToU}(\cdot,\cdot),\mathsf{WReg}(\cdot,\cdot)}(\mathsf{guess}, \sigma, \mathsf{s})$
  if $b' \neq b$ **then return** 0
  return 1

where it is mandated that for each $d \in \{0,1\}$ we have $i_d \in \mathtt{HU}$ and $i_d \notin \mathtt{GS}_m$, and in the guess phase the adversary $\mathcal{A}$ did not query $\mathsf{Open}(\cdot,\cdot)$ with m and σ. We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{DGS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{anon}}_{\mathcal{DGS}}(\mathcal{A}) = 1] - 1/2.$$

Discussion. Note that the adversary is given the issuer key ik, as considered for a conventional dynamic group signature in BSZ [9]. It turns out that giving adversary such an attack capability would complicate our construction by much. One can consider a weakening of definition that we call CPA-anonymity where the adversary is never given access to the opening oracle.

Traceability and Non-frameability. The traceability and non-frameability definitions are the same as ones in BSZ [9], which we recall here. First consider an experiment involving some traceability adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\mathrm{trace}}_{\mathcal{DGS}}(\mathcal{A})$

$(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{HU} \leftarrow \emptyset$
$(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}(\cdot),\mathsf{USK}(\cdot),\mathsf{Crpt}(\cdot,\cdot),\mathsf{SndToI}(\cdot,\cdot),\mathsf{RReg}(\cdot)}(gpk, ok)$
if $\mathsf{GV}(gpk, m, \sigma) = 0$ **then return** 0
$(i, \omega) \leftarrow \mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$
if $i = 0$ **or** $\mathsf{Judge}(gpk, i, \boldsymbol{reg}, m, \sigma, \omega) = 0$ **then return** 1
return 0

We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathcal{DGS},}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{trace}}_{\mathcal{DGS}}(\mathcal{A}) = 1].$$

We now recall the non-frameability definition by considering the following experiment involving an adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\mathrm{nf}}_{\mathcal{DGS}}(\mathcal{A})$

$(gpk, ik, ok) \xleftarrow{\$} \mathcal{DGS}.\mathsf{Gen}(1^\lambda)$; $\mathtt{CU} \leftarrow \emptyset$; $\mathtt{HU} \leftarrow \emptyset$
$(m, \sigma, i, \omega) \xleftarrow{\$} \mathcal{A}^{\mathsf{USK}(\cdot),\mathsf{GS}(\cdot,\cdot)\mathsf{Crpt}(\cdot,\cdot),\mathsf{SndToU}(\cdot,\cdot),\mathsf{WReg}(\cdot,\cdot)}(gpk, ik, ok)$
if $\mathsf{GV}(gpk, m, \sigma) = 0$ **then return** 0
$(i, \omega) \leftarrow \mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$
if $i \in \mathtt{HU}$ **and** $i \notin \mathtt{USK\text{-}HU}$ **and** $i \notin \mathtt{GS}_m$ **and**
  $\mathsf{Judge}(gpk, i, \boldsymbol{reg}, m, \sigma, \omega) = 1$ **then return** 1
return 0

We define the advantage of $\mathcal{A}$ in the above experiment as

$$\mathbf{Adv}^{\mathrm{nf}}_{\mathcal{DGS}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{nf}}_{\mathcal{DGS}}(\mathcal{A}) = 1].$$

3.3 Detection Algorithms

We show how our security definitions in *both* settings imply efficient *detection algorithms* that can find who do not follow the algorithm specification *and* disobey the rule that one group member can only sign any message once. Here we only focus on the more involved dynamic setting, and one can easily get similar (but weak) results for the static group setting.

The detection algorithm Det takes as input two different group signatures σ_1 and σ_2 for the same message m and outputs $\perp$ or $\mathcal{I}$ or (b, i, j, θ) for $b \in \{0, 1\}$. The algorithm returns $\perp$ if at least for one of σ_1 and σ_2 it holds that $\mathsf{GV}(gpk, m, \sigma_t) = 0$ ($t \in \{0, 1\}$). If $b = \mathcal{I}$ then the detection algorithm is claiming that at least one of the two signatures was not generated by the group members registered in the ***reg***. (Note that group *issuer* can always generate group signatures on his own by adding dummy users.) In this case, it might have an additional output μ that is a proof that at least one of the signatures was generated by the group issuer. If $b = 0$ then it is claiming that two signatures were generated by two different signers—a rule that the system would like to enforce. In this case, it does not need a proof of the claim. (But one could ask a proof if desired.) In case $b = 1$, it is claiming that two signatures were generated by rule disobeyers i and j, where $i, j \geq 1$, i could be equal to j, and θ is a proof of this claim that is verified by the $\mathsf{DetProve}$ algorithm.[4]

The *detection proving* algorithm $\mathsf{DetProve}$ takes as input the group public key gpk, two valid signatures σ_1 and σ_2 of m, and a vector (b, i, j, θ) output from $\mathsf{Det}(m, \sigma_1, \sigma_2)$ where $b = 1$, $i, j \geq 1$, and θ is a non-empty string to output a single bit d indicating whether θ is a correct proof that both of i and j disobey the rule.

The detection algorithm should satisfy *completeness* and *soundness* properties described below.

Completeness. The set $\mathtt{LU}$ of legal users (who follow the rule that one signer can only sign one message once)[5] will almost never be wrongly detected by the detection algorithm.

Soundness. If $\mathsf{Det}(m, \sigma_1, \sigma_2) = (1, i, j, \theta)$ and $\mathsf{DetProve}(gpk, m, \sigma_1, \sigma_2, \mathsf{Det}(m, \sigma_1, \sigma_2)) = 1$ then both i and j are illegal users (who did not follow the specification of the protocol or the rule).[6]

Discussion. The soundness definition is *strong* in the sense that once the detection algorithm outputs two different identities with a correct proof of this claim then *neither* of them followed the specification of the protocol or the rule. The detection algorithm would not work well if only one of the two users is dishonest but the other still follows the rule, since in this case the latter user is actually framed by some adversary.

Unique group signature implies an efficient detection algorithm. Our dynamic CCA-anonymous unique group signature immediately has an *efficient* complete and sound detection algorithm Det coupled with a detection proving algorithm $\mathsf{DetProve}$. In a nutshell, the detection algorithm takes as input the group opening key and proceeds as follows: If ever found two different valid group signatures on the same message with the same unique identifier, then it runs the opening algorithm Open to extract their identities i and j (where possibly i equals j) and their corresponding proofs θ_i and θ_j, and adds them (it) to the misbehaving user set. Anyone can run the Judge algorithm as the $\mathsf{DetProve}$ algorithm. More specifically, given a dynamic unique group signature, we have the following detection algorithm Det and detection proving algorithm $\mathsf{DetProve}$ as illustrated in Figure 1.

We stress that the completeness and soundness properties of the above detection algorithm are not guaranteed by merely using uniqueness property. For instance, we cannot rule out that there is an adversary who obtains one signature that points to some honest user i who has not been queried with $\mathsf{USK}(\cdot)$ oracle and can forge a different signature still with the same unique identifier that points to the same user i. Clearly, in this case, it does not contradict the uniqueness property, but such an adversary successfully frames user i. In what follows, we formally justify the detection algorithm by providing the following theorem (with proof in Appendix C.1).

[4] Note that in the static group setting we do not have such an algorithm.

[5] This set does not mean only the honest user set $\mathtt{HU}$ but a set of corrupted users who still follow the algorithm.

[6] Recall that i could be equal to j.

Algorithm $\mathsf{Det}(m, \sigma_1 = (\tau_1, \psi_1), \sigma_2 = (\tau_2, \psi_2))$
  if $\mathsf{GV}(m, \sigma_1) = 0$ **or** $\mathsf{GV}(m, \sigma_1) = 0$ **then**
    return $\perp$
  $(i, \omega_i) \leftarrow \mathsf{Open}(m, \sigma_1)$; $(j, \omega_j) \leftarrow \mathsf{Open}(m, \sigma_2)$
  if $i = 0$ **or** $j = 0$ **then**
    return $(\mathcal{I}, \mu)$
  if $\tau_1 = \tau_2$ **then**
    return $(1, i, j, (\omega_i, \omega_j))$
  return $(0, \varepsilon)$

Algorithm $\mathsf{DetProve}(m, \sigma_1, \sigma_2)$
  if $\mathsf{Det}(m, \sigma_1, \sigma_2) \neq (1, i, j, (\omega_i, \omega_j))$ **then**
    return 0
  if $\mathsf{Judge}(gpk, (i, \omega_i), m, \sigma_1, \boldsymbol{reg}) = 1$ **and**
    $\mathsf{Judge}(gpk, (j, \omega_j), m, \sigma_2, \boldsymbol{reg}) = 1$ **then**
    return 1
  return 0

Figure 1: Det and DetProve algorithms. If Det identifies that one of signatures was produced by the group issuer then it can also output μ which is either ε or a proof of the claim, depending on what the system requires.

Theorem 1 *Given a dynamic unique group signature $\mathcal{DGS}$, if it is correct and non-colliding, and satisfies CCA-anonymity, uniqueness, traceability, and non-frameability requirements, then the Det algorithm given in Figure 1 is complete and sound.* ∎

Further Comments. It is quite easy to generalize the results to handle more complex privacy-preserving circumstances as we have done in the introduction. It is also possible to provide some privileges to a small set of members who pay more by providing them with more uniqueness keys.

Note that identity detection algorithm is *optional*. It makes sense that the system only prevents repeated unique identifers (and denies the request for service), and no further action is taken. As we mentioned in the introduction, if revealing the identities of misbehaving users is not a must then this variant leads to an extremely efficient construction.

4 Unique Group Signature Construction – Static Setting

In this section, we first present general constructions for CCA-anonymous unique group signature and for its meaningful relaxations in the static setting. They together motivate efficient instantiations by using Groth-Sahai proof system.

4.1 A General CCA-Anonymous Unique Group Signature

Our construction basically follows the general two-level signature constructions of [8]. The difference is that we replace the second-level signature with a verifiable random function, where its public key is signed by the certification key of group manager. Normal CPA-anonymous group signatures are studied in [13, 15, 16]. It is pointed out by [15, 16] that it is not necessary to use a full-fledged adaptive chosen message attack secure signature for the first level. The construction in [16] uses a first-level signature that provides security against a special form of random message attack.[7] This random message attack is slightly non-standard, namely, it only provides the adversary with random signatures but not the message. This observation gives greater flexibility and more candidates in designing static group signature, yielding efficient constructions. It is interesting to know whether using signature schemes secure under weaker attacks for the first level is possible for unique group signature.

We give our general construction using a first-level signature scheme that provides security against standard random message attacks. To make it easier to understand, we present our re-

[7] Precisely, the signature is based on a q-type assumption and the hidden scalars are not necessarily uniformly distributed. However, asking them to be chosen uniformly at random is a typical choice.

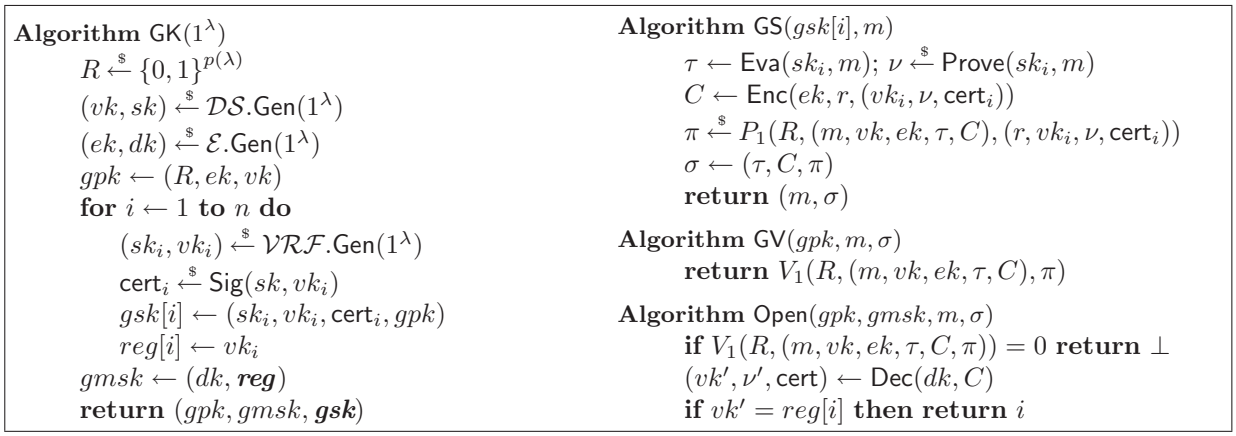

Figure 2: *General construction for static unique group signature* $\mathcal{SGS}_1 = (\mathsf{GK}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open})$. We write $\boldsymbol{reg}$ to denote $reg[1] \cdots reg[n]$. R is the common reference string for the underlying NIZK proof system (P_1, V_1). $\mathcal{SGS}_1$ is a secure CCA-anonymous unique group signature, if $\mathcal{DS}$ is unforgeable under random message attacks, $\mathcal{E}$ is CCA-secure, and $\mathcal{VRF}$ is a verifiable random function, and (P_1, V_1) is a one-time simulation-sound NIZK proof system. $\mathcal{SGS}_1$ is CPA-anonymous, if $\mathcal{E}$ is semantically secure and (P_1, V_1) is a regular NIZK proof system.

sult using a combination of an encryption scheme and a NIZK proof system, instead of using an equivalent notion of NIZKPoK. Define a verifiable random function $\mathcal{VRF} = (\mathsf{Gen}, \mathsf{Eva}, \mathsf{Prove}, \mathsf{Ver})$ with input domain $\mathcal{X}$ and output range $\mathcal{Y}$ (which is also the domain and range of unique identifier respectively). Let $\mathcal{DS} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vrf})$ be a signature scheme. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. Let (P_1, V_1) be a NIZK proof system for a language $\mathcal{L}_1 := \{(m, vk, ek, \tau, C) | \exists (r, vk', \nu', \mathsf{cert})[\mathsf{Vrf}(vk, vk', \mathsf{cert}) = 1, \mathsf{Ver}(vk', m, \tau, \nu') = 1, \text{and } C = \mathsf{Enc}(ek, r, (vk', \nu', \mathsf{cert}))]\}$ where we write $\mathsf{Enc}(ek, r, M)$ for the encryption of a message M under the public key ek using the randomness r. We define a group signature scheme $\mathcal{SGS}_1 = (\mathsf{GK}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open})$ in Figure 2. The following theorem establishes its security:

Theorem 2 *If $\mathcal{VRF}$ is a verifiable random function, $\mathcal{DS}$ is a secure signature against random message attack,*[8] *$\mathcal{E}$ is a CCA-secure encryption scheme, and the underlying NIZK proof system (P_1, V_1) is adaptively sound and adaptively zero-knowledge and one-time simulation-sound then the construction $\mathcal{SGS}_1$ in Figure 2 is a secure CCA-anonymous unique group signature in the static setting.* ∎

4.2 Relaxations and Separations

The above construction is general but does not seem to immediately give rise to efficient instantiations. This is due, first, to the fact current *simulation-sound* NIZK proof systems are not efficient—even for the Groth-Sahai instantiations for some specific languages [14, 38]. This is further due to the fact that the VRF proof ν may be incompatible with the efficient proof systems.

In light of this, we consider two meaningful relaxations of CCA-anonymous unique group signature. The first natural relaxation is to consider CPA-anonymous unique group signature where the anonymity adversary is never given the opening oracle. This immediately helps avoid using simulation-sound property of NIZK proof system and chosen ciphertext security for the underlying

[8] We require an additional but natural property for the underlying second-level VRF scheme, where each pair of VRF public/secret keys (vk, sk) are both uniformly and independently distributed. Similar requirements are needed for Theorems 3-5.

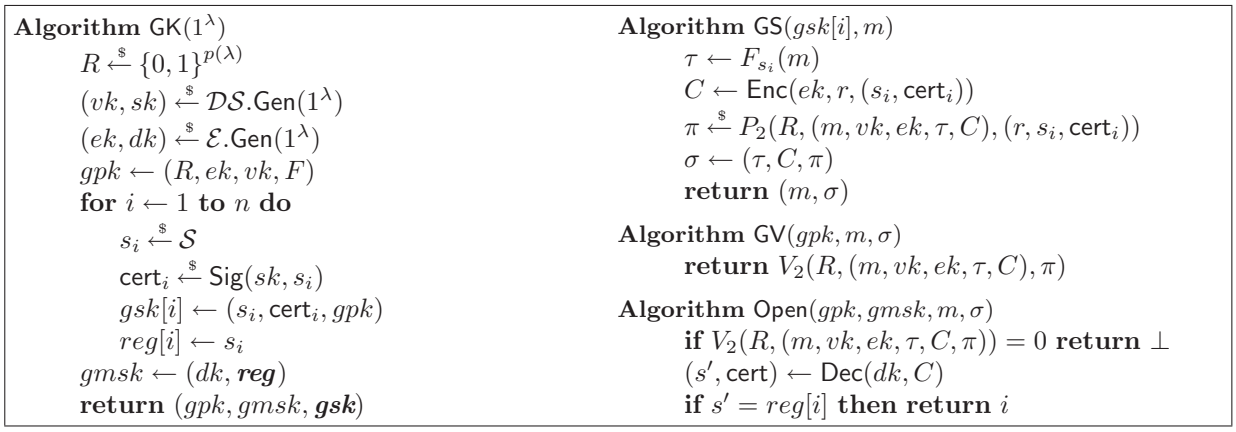

Figure 3: *Static unique group signature* $\mathcal{SGS}_2$ with *relaxed* uniqueness and traceability notions, where the adversaries are not allowed to give the group manager secret key $gmsk$.

encryption scheme. Namely, we have a group signature the same as illustrated in Figure 2 except that we only use a regular NIZK proof system and a semantic-secure encryption.

Theorem 3 *If $\mathcal{VRF}$ is a verifiable random function, $\mathcal{DS}$ is a secure signature against random message attack, $\mathcal{E}$ is a CPA-secure encryption scheme, and the underlying NIZK proof system (P_1, V_1) is adaptively sound and adaptively zero-knowledge, then the construction $\mathcal{SGS}_1$ in Figure 2 is a secure CPA-anonymous unique group signature in the static setting.* ∎

The other meaningful relaxation that we consider is that we no longer give the uniqueness and traceability adversaries the group manager secret key $gmsk$. This relaxation makes sense as an external adversary usually does not obtain the opening key of group manager unless it corrupts the group manager which looks less likely. We find that if we restrict the adversary in such a way then we can simply use PRF instead of VRF such that the second problem can be solved.

Define a PRF family $F\colon \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ where $\mathcal{S}$ is the key space, $\mathcal{X}$ is the message space, and $\mathcal{Y}$ is the range. We write $F_s(\cdot)$ to denote a PRF for every $s \in \mathcal{S}$. Let $\mathcal{DS}$ and $\mathcal{E}$ be a digital signature and a public key encryption scheme respectively. Let (P_2, V_2) be a NIZK proof system for a language $\mathcal{L}_2 := \{(m, vk, ek, \tau, C) | \exists (r, s, \mathsf{cert})[\tau = F_s(m), \mathsf{Vrf}(vk, s, \mathsf{cert}) = 1, \text{ and } C = \mathsf{Enc}(ek, r, (s, \mathsf{cert}))]\}$. We define a group signature scheme $\mathcal{SGS}_2 = (\mathsf{GK}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open})$ as illustrated in Figure 3. The following theorem establishes the security of this construction.

Theorem 4 *If F is a PRF, $\mathcal{DS}$ is a secure signature against random message attack, $\mathcal{E}$ is a CCA2 secure encryption scheme, and the underlying NIZK proof system (P_2, V_2) is adaptively sound and adaptively zero-knowledge and one-time simulation-sound then the construction $\mathcal{SGS}_2$ given in Figure 3 is a CCA-anonymous unique group signature with relaxed uniqueness and traceability where the adversaries are not given gmsk.* ∎

Separations. One can verify that $\mathcal{SGS}_1$ (i.e., the CPA-anonymous construction) may be not CCA-anonymous, and $\mathcal{SGS}_2$ may be not secure in the sense of standard uniqueness and traceability. Thus, they give natural separations results for these definitions of security. See Appendix B for proofs and discussion. (Recall that we also give a separation result on uniqueness definitions in Appendix B.)

4.3 Efficient Instantiations

The above concerns do not rule out *ad hoc* constructions in the strongest model defined. It turns out that we can provide efficient constructions using the Groth-Sahai proof system. The encryption

Algorithm $\mathsf{GK}(1^\lambda)$
- $(\mathsf{crs}, xk) \xleftarrow{\$} \mathsf{Groth\text{-}Sahai.Gen}(1^\lambda)$
- $(vk, sk) \xleftarrow{\$} \mathcal{DS}.\mathsf{Gen}(1^\lambda)$
- $gpk \leftarrow (\mathsf{crs}, vk)$
- **for** $i \leftarrow 1$ **to** n **do**
  - $s_i \xleftarrow{\$} \mathbb{Z}_q$
  - $\mathsf{cert}_i \xleftarrow{\$} \mathsf{Sig}(sk, h^{s_i})$
  - $gsk[i] \leftarrow (s_i, \mathsf{cert}_i, gpk)$
  - $reg[i] \leftarrow h^{s_i}$
- $gmsk \leftarrow (xk, \boldsymbol{reg})$
- **return** $(gpk, gmsk, \boldsymbol{gsk})$

Algorithm $\mathsf{GS}(gsk[i], m)$
- $\tau \leftarrow g^{1/(s_i+m)}$
- $C_s \xleftarrow{\$} \mathsf{Com}(h^{s_i})$
- $\theta \xleftarrow{\$} \mathsf{Sig}(sk, h^{s_i})$
- **return** $(m, \tau, C_s, C_\theta, \pi_1, \pi_2)$

Algorithm $\mathsf{GV}(gpk, m, \sigma)$
- **return** $V_3((m, \tau, C_s), \pi_1) \wedge V_4(C_s, C_\theta, vk), \pi_2)$

Algorithm $\mathsf{Open}(gpk, gmsk, m, \sigma)$
- **if** $\mathsf{GV}(gpk, m, \sigma) = 0$ **return** $\perp$
- $S' \leftarrow \mathsf{Extr}(xk, C_s)$
- **if** $S' = reg[i]$ **then return** i

Figure 4: *Efficient CPA-anonymous unique group signatures.* We let V_3 and V_4 be the corresponding verification algorithms for the languages $\mathcal{L}_3$ and $\mathcal{L}_4$. The common reference string crs contains the bilinear map parameter $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ besides the Groth-Sahai proof parameter.

scheme can be replaced with a Groth-Sahai extractable commitment scheme. Given a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, a commitment to $x \in \mathbb{G}$ (either $\mathbb{G}_1$ or $\mathbb{G}_2$) with randomness r_x is denoted $\mathsf{Com}(x, r_x)$, and an extraction algorithm Extr takes as input the extraction key xk and a commitment C to return a group element.

The key component is a PRF that supports efficient NIZK proof that can *degenerate* into a unique signature scheme where they share the *same* tag. In general, the former helps achieve the anonymity security, where the tag has to be random, while the latter is used to prove the uniqueness and traceability security, where the tag only needs to be unique and unpredictable.

Specifically, we make use of a *variant* of the PRF with NIZK proof proposed by Belenkiy et al. [7]. We define a language $\mathcal{L}_3 := \{(m, \tau, C_s) | \exists (s, r_s)[\tau = F_s(m) \text{ and } C_s = \mathsf{Com}(h^s, r_s)]\}$, where $F_s(\cdot) := g^{1/(s+\cdot)}$. The corresponding NIZK proof π_1 is of the form $(C_\tau, \pi_\tau, C'_s, \pi_s, \pi')$, where C_τ is a commitment to τ and π_τ is a NIZK proof for that C_τ is a commitment to τ, C'_s is a commitment to h^s, π_s is a NIZK proof that C_s and C'_s are commitments to the same value, and π' is a witness-indistinguishable proof that C_τ is a commitment to $\bar{\tau}$, C'_s is a commitment to S such that $e(\bar{\tau}, Sh^m) = e(g, h)$. The above proof system is a NIZK proof system for $\mathcal{L}_3$ if DDHI assumption holds and Groth-Sahai proof system is secure. As shown in Section 4.2, if we directly let group manager sign each secret key $s \in \mathbb{Z}_q$ (and add each s to $\boldsymbol{reg}$ which is part of $gmsk$) and run a corresponding NIZKPoK then we can get a CPA-anonymous unique group signature yet with relaxed uniqueness and traceability security. Still, this appears hard to find an efficient instantiation in the framework of Groth-Sahai proof system, since the secret s is a scalar rather a group element.

Note that we cannot as well expose the value h^s in the above PRF with NIZK proof system, because neither the above system would be zero-knowledge nor we are able to prove its security based on DDHI assumption. We can, however, degenerate the above PRF with NIZK proof to get a unique signature scheme, where one can view h^s as the public key and $g^{1/(s+m)}$ as the signature of m.[9] Then, the manager can sign each h^s instead of s, and add h^s to $\boldsymbol{reg}$. Fortunately, we can show that uniqueness property and standard unforgeability security (rather than pseudorandomness) suffice to give the security of uniqueness and traceability. This prevents us from using rather strong assumptions such as SDDHI assumption [18] in bilinear groups. In fact, one can prove security of the unforgeability under DHI assumption [28] (with less tight reduction) or SDHI assumption that we formalize where the adversary is only asked to output a new message-signature pair to win (see

[9] This is the Boneh-Boyen weakly secure signature [11].

Section 2.2).

It remains to be shown how to choose the first-level signature. Recall that Groth-Sahai commitment, given the extraction trapdoor, can only extract group elements rather than the scalars. There are currently two ways of settling the problem. The first solution is to use the F-unforgeable signature by Belenkiy et al. [6]. They proposed two F-unforgeable signature schemes, one of which has a simple structure, yet using an interactive assumption (i.e., interactive Hidden SDH assumption). We can build our scheme on this signature, while the security can be proven using a weaker and more natural non-interactive q-type assumption. The other method is to employ a structure-preserving signature that is only needed to be secure in the weak random message attack (e.g., one from [37]) to sign h^s directly. To be as general as possible, we let $\mathcal{DS} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vrf})$ be the first-level signature that can sign at least one group element and π_2 be a corresponding Groth-Sahai NIZK proof for the language $\mathcal{L}_4 := \{(C_s, C_\theta, vk) | \exists (S, r_s, \theta, r_\theta)[C_s = \mathsf{Com}(S, r_s)$, and $C_\theta = \mathsf{Com}(\theta, r_\theta)$, and $\mathsf{Vrf}(vk, S, \theta) = 1\}$. [10] The construction is illustrated in Figure 4 and we have the following theorem.

Theorem 5 *The construction in Figure 4 is a CPA-anonymous unique group signature if DDHI and DHI (or SDHI) assumptions hold and Groth-Sahai proof system is secure, and the $\mathcal{DS}$ is structure-preserving and unforgeable under random message attack (or F-unforgeable under random message attack).* ∎

The above unique group signature scheme is only CPA-anonymous. While it is possible to use the technique from [14, 38] to achieve CCA-anonymity, more efficient would be using an explicit chosen-ciphertext attack secure encryption as we shall describe in the next section.

5 Unique Group Signature Construction – Dynamic Setting

Overview. Similar to the construction of Section 4, the starting point for a CPA-anonymous unique group signature scheme in the dynamic setting is a two-level certification protocol: the issuer signs the verification key of users, and the users can then sign their own messages. This process should be achieved in a zero-knowledge sense.[11]

To facilitate understanding our final CCA-anonymous construction, we start by sketching the above idea in some more details (in the framework of Groth-Sahai proof system). Let $\mathcal{DS}_1$ and $\mathcal{DS}_2$ be two digital signature schemes. A dynamic group signature (without uniqueness) can be given as follows: key generation algorithm runs $\mathsf{Groth\text{-}Sahai.Gen}(1^\lambda)$ to set up an appropriate common reference string together with a trapdoor xk for the Groth-Sahai NIWI proof system $(P_{\text{NIWI}}, V_{\text{NIWI}})$. It also generates a pair of signature keys (vk, sk) by running the key generation algorithm of $\mathcal{DS}_1$. The certification key sk is sent to the issuer, while opening key xk is given to the opener. Then, a user and the issuer run an interactive $\mathsf{Join/Issue}$ protocol. The user i generates its own signature key pair (vk_i, sk_i) by running the key generation algorithm of $\mathcal{DS}_2$ and sends vk_i to the issuer. The issuer sends $\mathsf{cert}_i \xleftarrow{\$} \mathcal{DS}_1.\mathsf{Sig}(sk, vk_i)$ as the membership certificate of i. To sign a message m, member i computes $\phi \xleftarrow{\$} \mathcal{DS}_2.\mathsf{Sig}(sk_i, m)$ and a NIWI proof π such that $\mathcal{DS}_1.\mathsf{Vrf}(vk, vk_i, \mathsf{cert}_i) = 1$ and $\mathcal{DS}_2.\mathsf{Vrf}(vk_i, m, \phi) = 1$, where vk_i, ϕ, and cert_i are Groth-Sahai proof system variables, and vk and m are constants. The group signature is (m, π). The group verification algorithm GV

[10] Precisely, θ might be multiple group elements and in this case $\mathsf{Com}(\theta, r_\theta)$ denotes a vector of commitments. Meanwhile, there are possibly additional group elements as signature messages besides h^s that should be also included in the proof system.

[11] This idea originated from [23, 45], and very efficient instantiations [2, 39] without relying on random oracles are just recently given using the Groth-Sahai proof system.

simply verifies the correctness of NIWI proof of π. Open algorithm uses the trapdoor xk to extract $(\phi, vk_i, \mathsf{cert}_i)$. It looks up the registration table to identify the signer. The Judge algorithm simply checks the correctness of the verification chain.

To make the above-mentioned group signature *unique*, one can consider using a PRF F instead of a secure signature scheme at the second level. Moreover, an interactive protocol is used to get a signature of the secret PRF key s_i of user i under vk, without letting the issuer know this secret value. To sign a message m, group signing algorithm GS computes $\tau := F_{s_i}(m)$, which we would like to use as the unique identifier. The GS algorithm then computes a NIZK proof of knowledge π that there exists a certification chain (s_i, cert_i) such that $\tau = F_{s_i}(m)$ and $\mathcal{DS}_1.\mathsf{Vrf}(vk, s_i, \mathsf{cert}_i) = 1$. The group signature is now (m, τ, π).

It is important that the issuer should not learn the PRF keys that it signs, or the issuer may now attack the CPA-anonymity by simply checking which of the PRF keys could have produced a given unique identifier. (Similar argument applies to the case where one tries to use VRF instead of PRF. The issuer should not learn the verification key of the VRF scheme.)

In general, we can resort to two-party secure computation. More efficiently, in order for the user i to get cert_i without letting the issuer know s_i (or g^{s_i}, for our construction), the user i and issuer can first run a "signing on a committed value" protocol to get a certification of signature, and user later makes a proof of knowledge of that signature. (These protocols are known as "CL-signatures" [20–22], and such signatures with non-interactive proofs of knowledge are termed as P-signatures [6]).

However, this above process does not make the tracing and judging algorithm available. To solve this, we introduce a second chaining of two-level certification; namely, two new signature schemes $\mathcal{DS}'_1$ and $\mathcal{DS}'_2$ are selected. This time, we use Groth-Sahai commitments such that the witnesses can be extracted using the trapdoor given to the opener. Moreover, we can also use this chain to combine the technique of Groth [39] to achieve CCA anonymity. We call this technique "double-chaining certification".

There are a few further issues that we need to consider and address:

1) The registration table should not contain any information regarding the first-chaining certification. This part is not intended to trace users. More severely, it might cause additional problems. For example, an adversary might be able to repeat the certification process of some existing user, using the first-chaining information, though it does not even know the secrets of the existing user. (This is the case for our efficient instantiation that we describe shortly.) This adversary can successfully frame the existing user by signing the same message of the exiting user. It is crucial to point out what property it violates—the non-colliding property.

2) If we use a Groth-Sahai instantiation for our signing on committed value protocol then we have to use an independent common reference string.

3) Since τ has been made publicly known, we have to resort to NIZK proof rather than NIWI proof to complete the proof.

4) If user i (resp., user j) has a signing key $(s_i, \mathsf{cert}_i, sk'_i, vk'_i, \mathsf{cert}'_i)$ (resp., $(s_j, \mathsf{cert}_j, sk'_j, vk'_j, \mathsf{cert}'_j)$), then $(s_i, \mathsf{cert}_i, sk'_j, vk'_j, \mathsf{cert}'_j)$ and $(s_j, \mathsf{cert}_j, sk'_i, vk'_i, \mathsf{cert}'_i)$ are both valid signing keys. However, this does not contradict any security notions of unique group signature.

Our algorithm. The CCA-anonymous unique group signature is illustrated in Figure 5. We define a PRF family $F\colon \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ with key space $\mathcal{S}$. Let $\mathcal{DS}_1$, $\mathcal{DS}'_1$, and $\mathcal{DS}'_2$ be three signature schemes, all of which are secure under adaptive chosen message attacks. Issuer runs $(vk, sk) \xleftarrow{\$} \mathcal{DS}_1.\mathsf{Gen}(1^\lambda)$ and $(vk', sk') \xleftarrow{\$} \mathcal{DS}'_1.\mathsf{Gen}(1^\lambda)$, where (vk, sk) is used to certify PRF keys, and (vk', sk') is used for double-chaining certification. Correspondingly, we use two Groth-Sahai

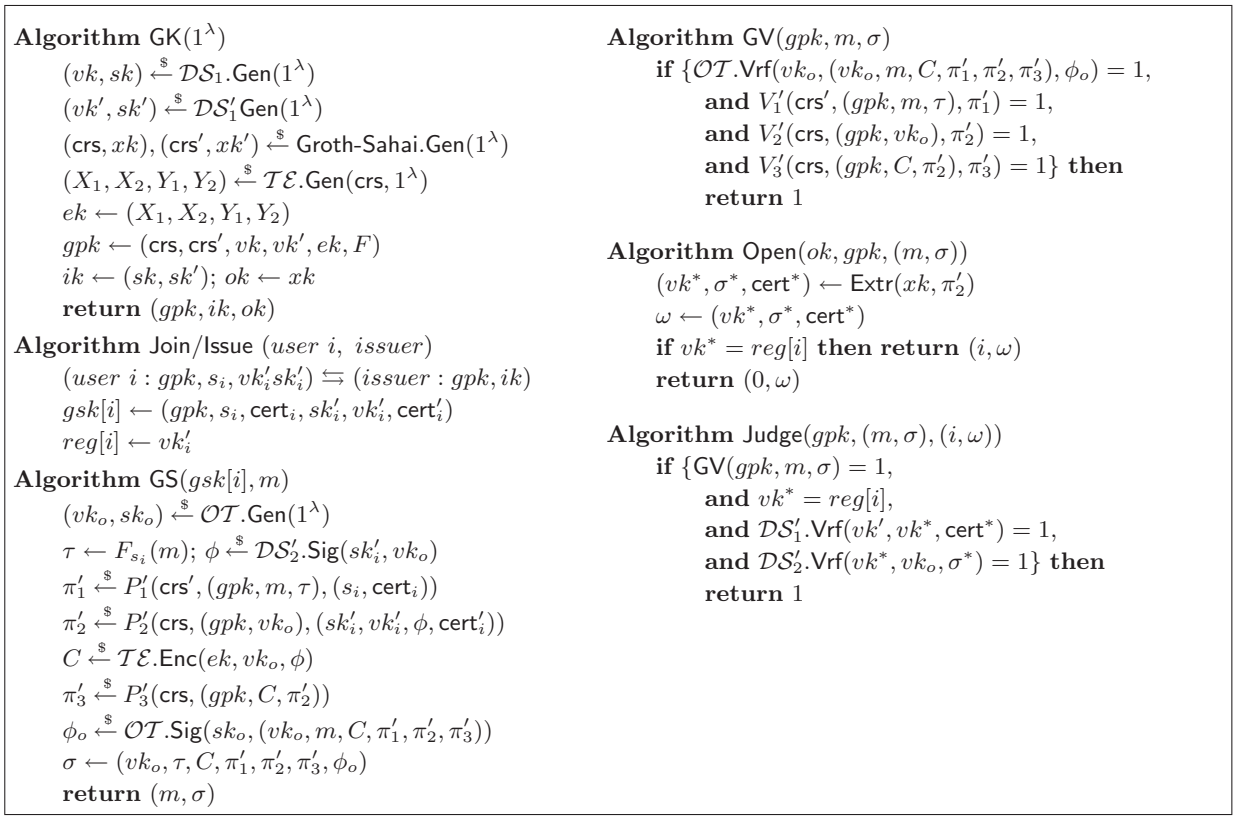

Figure 5: *CCA-anonymous unique group signature—Dynamic Setting.* In the group key generation algorithm GK, two independent Groth-Sahai proof common reference strings are generated—(crs, xk) for the "double-chaining" (i.e., for $\mathcal{L}'_2$ and $\mathcal{L}'_3$), and the other for certifying the PRF secret key and the language $\mathcal{L}'_1$. The secret keys of Kiltz's encryption and xk' can be safely discarded.

proof systems with the same security parameter but with independently generated common reference strings (crs, xk) and (crs', xk')—the former for the double-chaining certification and the latter for certifying the PRF protocol and proving the knowledge of the corresponding signature. Let $\mathcal{OT}$ be a strong one-time signature scheme secure against weak chosen message attacks. Let $\mathcal{TE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be Kiltz's selective-tag weakly CCA-secure encryption scheme [46], with the public key compatible with Groth-Sahai proof system setup. We write $\mathsf{Enc}(ek, t, M)$ for the encryption of a message M under the public key ek and a tag t. User i and the issuer run an interactive $\mathsf{Join/Issue}$ protocol. This includes two steps. First, user i randomly picks its PRF key s_i; the user and issuer run a protocol on signing on committed value s_i, and finally the user gets a signature cert_i on s_i such that $\mathcal{DS}_1.\mathsf{Vrf}(vk, s_i, \mathsf{cert}_i) = 1$. Second, user i runs $(vk'_i, sk'_i) \xleftarrow{\$} \mathcal{DS}'_2.\mathsf{Gen}(1^\lambda)$, sends vk'_i to the issuers, and obtains a cert'_i such that $\mathcal{DS}'_1.\mathsf{Vrf}(vk', vk'_i, \mathsf{cert}'_i) = 1$. After the $\mathsf{Join/Issue}$ procedure, user will get its secret key $(s_i, \mathsf{cert}_i, sk'_i, vk'_i, \mathsf{cert}'_i)$, while the issuer puts vk'_i to $reg[i]$.

We now specify the three NIZK proof systems in a general NIZK framework. (P'_1, V'_1) is a NIZK proof system for a language $\mathcal{L}'_1 := \{(gpk, m, \tau) | \exists (s, \mathsf{cert})[\tau = F_s(m)$ and $\mathcal{DS}_1.\mathsf{Vrf}(vk, s, \mathsf{cert}) = 1]$. (P'_2, V'_2) is a NIZK proof system for a language $\mathcal{L}'_2 := \{(gpk, vk_o) | \exists (\overline{vk'}, \phi', \mathsf{cert}')[\mathcal{DS}'_1.\mathsf{Vrf}(vk', \overline{vk'}, \mathsf{cert}') = 1$ and $\mathcal{DS}'_2.\mathsf{Vrf}(\overline{vk'}, vk_o, \phi') = 1]$. (P'_3, V'_3) is a NIZK proof system that the plaintext of C and second-level signature in π'_3 are the same (see [39] for details).

All of the primitives used in the above construction can be efficiently instantiated using Groth-Sahai proofs. In particular, the first chaining (including the signing on committed value protocol and $\mathcal{L}'_1$) can be achieved by combining the PRF with NIZK proof [7] and the P-signatures [6] (that relies on F-unforgeability). We can use the technique in Section 4 (PRF with NIZK proof that can degenerate into a unique signature) to improve the security as well as achieve extractability. $\mathcal{L}'_2$ can be instantiated using any structure-preserving signature combining any signature whose public keys are group elements. We have the following theorem:

Theorem 6 *The construction illustrated in Figure 5 is a secure unique group signature (CCA-anonymous, dynamic setting).* ∎

Efficient instantiation with concurrent join. Yet more efficiently, we use the Fuchsbauer's blind signature (also an efficient signing on committed value protocol [31, Remark 6]) to achieve the $\mathsf{Join/Issue}$ protocol. We note that this protocol is perfectly compatible with the PRF with NIZK proof [7]. Specifically, the user can simply run the Obtain procedure of blind signature on the message $(M, N) = (g^s, h^s)$ (where s is the PRF key). Once receiving the pre-signature by the issuer, it can generate a valid automorphic signature on (M, N). The additional commitment to g^s does not influence the security the PRF scheme, which can be easily justified via a standard hybrid argument. The above process clearly can be realized in two moves. Note that the double-chaining $\mathsf{Join/Issue}$ protocol can also be achieved in two moves if we use an independent structure-preserving signature for the first level. In fact, these two $\mathsf{Join/Issue}$ protocols for both chaining can be easily run together using external PKI to allow very efficient concurrent join.

Acknowledgments

The authors would like to thank Sherman Chow and anonymous reviewers for their helpful and insightful comments. This work was supported by NSF grant CNS-0831547.

References

[1] M. Abdalla, D. Catalano, and D. Fiore. Verifiable random functions from identity-based key encapsulation. *EUROCRYPT 2009*, LNCS vol. 5479, Springer, pp. 554–571, 2009.

[2] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-perserving signatures and commitments to group elements. *CRYPTO 2010*, LNCS vol. 6223, Springer, pp. 209–236, 2010.

[3] M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. *CRYPTO 2011*, LNCS vol. 6841, Springer, pp. 649–666, 2011.

[4] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. http://eprint.iacr.org.

[5] M.H. Au, S.S.M. Chow, W. Susilo, and P.P. Tsang. Short linkable ring signatures revisited. *EUROPKI 2006*, LNCS vol. 4043, Springer, pp. 101–115, 2006.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and non-interactive anonymous credentials. *TCC 2008*, LNCS vol. 4948, Springer, pp. 356–374, 2008.

[7] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. *Pairing 2009*, LNCS vol. 5671, Springer, pp. 114–131, 2009.

[8] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003*, LNCS vol. 2656, Springer, pp. 614–629, 2003.

[9] M. Bellare, H. Shi and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. *CT-RSA 2005*, LNCS vol. 3376, Springer, pp. 136–153, 2005.

[10] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge proof systems and applications. *STOC '88* ACM, pp. 103-112, 1988.

[11] D. Boneh and X. Boyen. Short signatures without random oracles. *EUROCRYPT 2004*, LNCS vol. 3027, Springer, pp. 56–73, 2004.

[12] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. *EUROCRYPT 2004*, LNCS vol. 3027, Springer, pp. 223–238, 2004.

[13] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. *CRYPTO 2004*, LNCS vol. 3152, Springer, pp. 41–55, 2004.

[14] X. Boyen, C. Chevalier, G. Fuchsbauer, and D. Pointcheval. Strong cryptography from weak secrets—Building efficient PKE and IBE from distributed passwords. *AFRICACRYPT 2010*, LNCS vol. 6055, Springer, pp. 297–315, 2010.

[15] X. Boyen and B. Waters. Compact group signatures without random oracles. *EUROCRYPT 2006*, LNCS vol. 4004, Springer, pp. 427–444, 2006.

[16] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. *PKC 2007*, LNCS vol. 4450, Springer, pp. 1–15, 2007.

[17] Z. Brakerski, S. Goldwasser, G. Rothblum, and V. Vaikuntanathan. Weak verifiable random functions. *TCC 2009*, LNCS vol. 5444, Springer, pp. 558–576, 2009.

[18] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clone wars: efficient periodic n-times anonymous authentication. *ACM CCS 2006*, ACM, pp. 201–210, 2006.

[19] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-cash. *EUROCRYPT 2005*, LNCS vol. 3494, Springer, pp. 302–321, 2005.

[20] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *EUROCRYPT 2001*, LNCS vol. 2045, Springer, pp. 93–118, 2001.

[21] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. *SCN '02*, LNCS vol. 2576, Springer, pp. 268–289, 2002.

[22] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. *CRYPTO 2004*, LNCS vol. 3152, Springer, pp. 56–72, 2004.

[23] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. *CRYPTO '97*, LNCS vol. 1294, Springer, pp. 410–424, 1997.

[24] M. Chase and A. Lysyanskaya. Simulatable VRFs with applications to multi-theorem NIZK. *CRYPTO '07*, LNCS vol. 4622, Springer, pp. 303–322, 2007.

[25] D. Chaum and E. Heyst. Group signatures. *EUROCRYPT '91*, LNCS vol. 547, Springer, pp. 257–265, 1991.

[26] I. Damgård, K. Dupont, and M. Pedersen. Unclonable group identification. *EUROCRYPT 2006*, LNCS vol. 4004, Springer, pp. 555–572, 2006.

[27] Y. Dodis. Efficient construction of (distributed) verifiable random functions. *PKC 2003*, LNCS vol. 2567, Springer, pp. 1–17, 2003.

[28] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. *PKC 2005*, LNCS vol. 3386, Springer, pp. 416–431, 2005.

[29] S. Even, O. Goldreich, and S. Micali. On-line/Off-line digital signatures. *CRYPTO '89*, LNCS vol. 435, Spinger, pp. 263–275, 1990.

[30] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Computing*, 29(1):1–28, 1999.

[31] G. Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive: Report 2009/320. http://eprint.iacr.org

[32] E. Fujisaki and K. Suzuki. Traceable ring signature. *IEICE Transactions 91-A(1)*: 83–93 (2008). Preliminary version in *PKC 2007*.

[33] E. Fujisaki. Sub-linear size traceable ring signatures without random oracles. *CT-RSA '11*, LNCS vol. 6558, Springer, pp. 393–415, 2011.

[34] S. Galbraith, K. Paterson and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics (2007)*, vol. 156, Issue 16, pp. 3113-3121, September, 2008.

[35] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[36] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[37] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. *ASIACRYPT 2008*, LNCS vol. 5350, Springer, pp. 179–197, 2008.

[38] J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. *ASIACRYPT 2006*, LNCS vol. 4284, Springer, pp. 444–459, 2006.

[39] J. Groth. Fully anonymous group signatures without random oracles. *ASIACRYPT 2007*, LNCS vol. 4833, Springer, pp. 164–180, 2007.

[40] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT 2008*, LNCS vol. 4965, Springer, pp. 415–432, 2008.

[41] S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. *EUROCRYPT 2010*, LNCS vol. 6110, Springer, pp. 656–672, 2010.

[42] S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. *EUROCRYPT 2004*, LNCS vol. 3027, Springer, pp. 590-608, 2004.

[43] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. *EUROCRYPT 2007*, LNCS vol. 4515, Springer, pp. 97–114, 2007.

[44] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. *EUROCRYPT 2004*, LNCS vol. 3027, Springer, pp. 571–589, 2004.

[45] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. *EUROCRYPT 2005*, LNCS vol. 3494, Springer, pp. 198–214, 2005.
[46] E. Kiltz. Chosen-ciphertext security from tag-based encryption. *TCC '06*, LNCS vol. 3876, Springer, pp. 581–600, 2006.
[47] B. Libert and M. Yung. Efficient traceable signatures in the standard model. *Pairing 2009*, LNCS vol. 5671, Springer, pp. 187–205, 2009.
[48] J.K. Liu, V.K. Wei, and D.S. Wong. Linkable spontaneous anonymous group signatures for ad hoc groups. *ACISP 2004*, LNCS vol. 3108, Springer, pp. 325–335, 2004.
[49] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. *CRYPTO '02*, LNCS vol. 2442, Springer, pp. 597–612, 2002.
[50] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. *FOCS 1999*, IEEE Computer Society, pp. 120–130, 1999.
[51] L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. *ACNS* 2005, LNCS vol. 3531, Springer, pp. 318–333, 2005.
[52] S. Chow. Real traceable signatures. *SAC 2009*. LNCS vol. 5867, Springer, pp. 92–107, 2009.
[53] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication. *Asiacrypt 2004*, LNCS vol. 3329, Springer, pp. 308–322, 2004.
[54] I. Teranishi and K. Sako. k-times anonymous authentication with a constant proving cost. *PKC 2006*, LNCS vol. 3958, Springer, pp. 525–542, 2006.
[55] P.P. Tsang, V.K. Wei, T.K. Chan, M.H. Au, J.K. Liu, and D.S. Wong. Separable linkable threshold ring signatures. *INDOCRYPT 2004*, LNCS vol. 3348, Springer, pp. 389–398, 2004.
[56] M. Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. http://eprint.iacr.org

A Definitions of NIZK and NIWI

Non-interactive zero-knowledge proof systems. We shall use a notion of NIZK proofs of membership in NP languages, which was introduced by Blum, Feldman, and Micali [10]. Let $\rho(\cdot,\cdot)$ be a polynomially bounded binary relation. If $(x,w)\in\rho$ we will call that x is a *theorem* and w is a proof of x. Let $\mathcal{L}_\rho$ denote the langauge associated with relation ρ: $\mathcal{L}_\rho=\{x|\exists w[(x,w)\in\rho]\}$.

Consider two polynomial-time algorithms (P,V), both of which have access to a *common reference string* η. (If the string is randomly chosen then we will call it *common random string*.) Call (P,V) is a *non-interactive proof system* for $\mathcal{L}_\rho$ if there exists some polynomial $p(\cdot)$ such that it satisfies the following two conditions:

- **Completeness**: For every $\lambda\in\mathbb{N}$, every $(x,w)\in\rho$,
$$\Pr[\eta\xleftarrow{\$}\{0,1\}^{p(\lambda)};\pi\xleftarrow{\$}P(\lambda,x,w,\eta):\ V(\lambda,x,\pi,\eta)=1]=1.$$
- **(Adaptive) soundness**: For every $\lambda\in\mathbb{N}$, any prover $\hat{P}$, and every $x\notin\mathcal{L}_\rho$,
$$\Pr[\eta\xleftarrow{\$}\{0,1\}^{p(\lambda)};(x,\pi)\xleftarrow{\$}\hat{P}(\lambda,\eta):\ V(\lambda,x,\pi,\eta)=1]\leq\epsilon(\lambda).$$

We let $\mathbf{Adv}^{\text{sound}}_{(P,V)}(\hat{P})$ denote the above soundness advantage of $\hat{P}$ against a non-interactive proof system (P,V).

Given a polynomial time simulator $S=(S_1,S_2)$, define the zero-knowledge advantage of $\mathcal{A}$ against a non-interactive proof system (P,V) as $\mathbf{Adv}^{\text{zk}}_{(P,V)}(\mathcal{A})=\Pr[\eta\xleftarrow{\$}\{0,1\}^{p(\lambda)};(x,w)\xleftarrow{\$}\mathcal{A}(1^\lambda,\eta);\pi\xleftarrow{\$}P(\lambda,x,w,\eta):\ \mathcal{A}(\lambda,x,\pi,\eta)=1]-\Pr[(\eta',\mathsf{s})\xleftarrow{\$}S_1(1^\lambda);(x,w)\xleftarrow{\$}\mathcal{A}(1^\lambda,\eta');\pi'\xleftarrow{\$}S_2(x,\eta',\mathsf{s}):\ \mathcal{A}(\lambda,x,\pi',\eta')=1]$, where s is the state information. We say that a non-interactive proof system (P,V)

for $\mathcal{L}_\rho$ is (adaptive) zero-knowledge if there exists a probabilistic polynomial time simulator (S_1, S_2) such that for any probabilistic polynomial time adversary $\mathcal{A}$, it holds that $\mathbf{Adv}^{\mathrm{zk}}_{(P,V)}(\mathcal{A}) \leq \epsilon(\lambda)$.

One-time simulation-sound NIZK. Let (P, V) be an adaptive NIZK proof system for $\mathcal{L}_\rho$ and $S = (S_1, S_2)$ be a simulator for (P, V). We define the one-time simulation-soundness advantage of $\mathcal{A}$ against (P, V, S) as $\mathbf{Adv}^{\mathrm{otss}}_{(P,V,S)}(\mathcal{A}) = \Pr[(\eta, \mathsf{s}) \xleftarrow{\$} S_1(1^\lambda); (x, \mathsf{a}) \xleftarrow{\$} \mathcal{A}(1^\lambda, \eta); \pi \xleftarrow{\$} S_2(x, \eta, \mathsf{s}); (x', \pi') \xleftarrow{\$} \mathcal{A}(\lambda, x, \eta, \pi, \mathsf{a}) : x' \notin \mathcal{L}_\rho \wedge (x', \pi') \neq (x, \pi) \wedge V(\lambda, x', \pi', \eta) = 1]$, where a is the state information for $\mathcal{A}$. We say that (P, V, S) is one-time simulation-sound, if for any probabilistic polynomial-time adversary $\mathcal{A}$, it holds that $\mathbf{Adv}^{\mathrm{otss}}_{(P,V,S)}(\mathcal{A}) \leq \epsilon(\lambda)$.

Non-interactive witness-indistinguishable proofs. We also use *non-interactive witness-indistinguishable (NIWI) proof system*. We define the WI-*advantage* of $\mathcal{A}$ against a non-interactive proof system (P, V) for a language $\mathcal{L}_\rho$ as $\mathbf{Adv}^{\mathrm{wi}}_{(P,V)}(\mathcal{A}) = \Pr[\eta \xleftarrow{\$} \{0,1\}^{p(\lambda)}; (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(1^\lambda, \eta); \pi \xleftarrow{\$} P(\lambda, x, w_0, \eta) : \mathcal{A}(\lambda, x, \pi, \eta) = 1] - \Pr[\eta \xleftarrow{\$} \{0,1\}^{p(\lambda)}; (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(1^\lambda, \eta); \pi \xleftarrow{\$} P(\lambda, x, w_1, \eta) : \mathcal{A}(\lambda, x, \pi, \eta) = 1]$, where we require that $(x, w_0), (x, w_1) \in \rho$. We say that a non-interactive proof system (P, V) is *witness indistinguishable*, if for any probabilistic polynomial time adversaries $\mathcal{A}$ it holds that $\mathbf{Adv}^{\mathrm{wi}}_{(P,V)}(\mathcal{A}) \leq \epsilon(\lambda)$.

B Separations

B.1 A Separation Result Regarding Colluding Adversary Attacking Uniqueness Security

In defining uniqueness security of unique group signature in both static and dynamic settings, we argue that it is important to consider colluding attack where the adversary might corrupt a multiple of group users to gain an additional advantage. Instead, we call the weaker definition of security where the adversary can only corrupt one group user *weak uniqueness*.

Here, we show that these two definitions are different in the sense that there exist schemes satisfying the weak uniqueness definition but not the standard uniqueness that we defined.

For the static model case, we consider the CCA-anonymous unique group signature scheme $\mathcal{SGS}_1 = (\mathsf{GK}, \mathsf{GS}, \mathsf{GV}, \mathsf{Open})$ in Figure 2. Given a symmetric encryption $(\mathsf{Enc}, \mathsf{Dec})$, we define a double-encryption scheme $(\mathsf{D\text{-}Enc}, \mathsf{D\text{-}Dec})$ such that $\mathsf{D\text{-}Enc}(k_1, k_2, M) := \mathsf{Enc}(k_1, E(k_2, M))$ and $\mathsf{D\text{-}Dec}(C, k_1, k_2) := \mathsf{Dec}_{k_2}(\mathsf{Dec}_{k_1}(C))$. We construct the following scheme: the group key generation is the same as GK, except that every two consecutive users is given a double-encryption of a VRF public/secret key pair and a certificate of the public key, where each of the double-encryption keys is given to one of the users. Meanwhile, add the additional verification key to both the corresponding positions of the two users in the registration table. The new signature scheme is secure in the weak uniqueness sense (and secure in the sense of the standard anonymity and traceability notions), but is not secure in the sense of standard uniqueness, where the adversary can simply corrupt any two of them to win the game.

Similarly, for the dynamic model case, given the construction Figure 5, we construct a new signature scheme where the group issuer sends two consecutive users one more uniqueness key and a corresponding signature using double-encryption and they can collaboratively decrypt the ciphertext. (We do not need to modify the registration table, since we separate the unique identifier generation process from the tracing functionality for our construction.) One can verify that the new signatures serves as an example that is not secure in the standard uniqueness security (but is still secure under other security definitions).

B.2 Separations Results for Section 4.2

It is easy to see that the CPA-anonymous $\mathcal{SGS}_1$ may not be CCA-anonymous, since both the encryption and NIZK proof system may be malleable.

We now show that $\mathcal{SGS}_2$ may not be secure in the sense of standard uniqueness and traceability. We give simple attacks. Recall that group manager secret key $gmsk$ for $\mathcal{SGS}_2$ in Figure 3 is of the form $(dk, \boldsymbol{reg})$ where $\boldsymbol{reg} = \{s_i\}_1^n$. An adversary in possession of $gmsk$ especially dk can make one group signing oracle $\mathsf{GS}(\cdot,\cdot)$ with some arbitrary user $i \in \mathtt{HU}$ and some arbitrary message m. The oracle returns (m, τ, C, π). It can then use the decryption key dk to obtain the underlying plaintext that contains the secret signing key $gsk[i]$. In this case, the user i remains uncorrupted *technically*, but the adversary can use the secret signing key to attack the traceability and uniqueness properties.

Discussion. It seems that the above adversary can successfully attack the schemes because it obtains the decryption key dk. It would be interesting to know if the adversary that only knows $\{s_i\}_1^n$ can attack them as well. It turns out that this is true for both cases. For traceability, we can construct the following adversary. An adversary in possession of $\{s_i\}_1^n$ can make one group signing oracle $\mathsf{GS}(\cdot,\cdot)$ with some arbitrary user $i \in \mathtt{HU}$ and some arbitrary message m. The oracle returns (m, τ, C, π). The adversary then outputs (m', τ', C, π) where $\tau' = F_{s_i}(m')$ as a valid signature on some new message m' such that $i \notin \mathtt{GS}_{m'}$. Note that the adversary can compute $F_{s_i}(m')$ since it obtains s_i. Also note that π is a correct proof for instance (m', vk, ek, τ', C), since the original witnesses (r, s, cert) for the instance (m, vk, ek, τ, C) are also the witnesses for this new instance! (Check the definition for $\mathcal{L}_2$ for details.) In other words, the adversary can produce any valid signatures under some secret signing key $gsk[i]$ given a valid signature under this key on some message. Similar attack applies to the uniqueness case.

C Proofs of Theorems

C.1 Proof of Theorem 1

Proof: It is easily seen that if every user obeys the rule then the probability that there are two different group signatures with one unique identifier is negligible. The crux of the proof is to show the soundness of the algorithm. This requires some careful case analysis:

Case 1. Consider the case where adversary $\mathcal{A}$ can come up with two new valid group signatures (m, σ_1) and (m, σ_2) (where $\sigma_1 \neq \sigma_2$) such that they point to user i and j, respectively (where possibly $i = j$), and adversary does not obtain the secret keys of both of them. In this case, one can construct another adversary that violates non-frameability.

Case 2. Consider the case where $\mathcal{A}$ has seen a group signature (m, σ_1) that points to i (which the adversary may not know) and $\mathcal{A}$ does not obtain the secret signing key of user i. If adversary $\mathcal{A}$ can come up with another signature (m, σ_2) such that $\sigma_2 \neq \sigma_1$ and (m, σ_2) points to i, then we can use it to construct an adversary who attacks CCA-anonymity of the group signature. If it points to 0 then we can construct an adversary who attacks traceability. If it points to user j who has not been corrupted then we can construct an adversary violating non-frameability.

Case 3. Consider the case where adversary $\mathcal{A}$ can come up with two valid group signatures (m, τ, ψ_1) and (m, τ, ψ_2) (where $\psi_1 \neq \psi_2$) and they point to different users, say, i and j, such that $i, j \neq 0$, and adversary corrupted one of them, say, i, and did not query (j, m). We can construct adversary violating the non-frameability property.

Case 4. Consider the case where adversary $\mathcal{A}$ can come up with two valid group signatures (m, τ, ψ_1)

and (m, τ, ψ_2) (where $\psi_1 \neq \psi_2$), and they point to different users i and j such that $i, j \neq 0$, and adversary corrupted i, and it did query (j, m) and the signing oracle returns some $(m, \sigma') = (m, \tau', \psi')$.[12] If $\tau = \tau'$ and (m, τ, ψ_2) is different from every signature returned by the signing oracle $\mathsf{GS}(j, m)$, then we actually obtain an adversary violating the CCA-anonymity. If $(m, \tau, \psi_2) = (m, \tau', \psi')$ then by the non-colliding property and uniqueness property the above probability is negligible. More concretely, an adversary can honestly run the group signature signing algorithm using user i's secret signing key. The signature produced in this way with overwhelming probability has a different unique identifier from τ by the non-colliding property. Thus, an adversary can output this new signature and (m, τ, ψ_1) to attack the uniqueness property.

Note that the above two cases consider the possibilities that the detection algorithm might wrongly output the identity of an honest user j who never violated the rule.

Case 5. Adversary $\mathcal{A}$ has corrupted two users i and j, and produces two signatures (m, τ, ψ_1) and (m, τ, ψ_2) where $\psi_1 \neq \psi_2$ and they point to i and j, respectively. That is what the algorithm manages to detect. (To make it easier to understand how this type of attacks works, we take our CCA-anonymous unique group signature in Section 5 for example. If user i (resp., user j) has a signing key $(s_i, \mathsf{cert}_i, sk'_i, vk'_i, \mathsf{cert}'_i)$ (resp., $(s_j, \mathsf{cert}_j, sk'_j, vk'_j, \mathsf{cert}'_j)$), then $(s_i, \mathsf{cert}_i, sk'_j, vk'_j, \mathsf{cert}'_j)$ and $(s_j, \mathsf{cert}_j, sk'_i, vk'_i, \mathsf{cert}'_i)$ are both valid signing keys. Adversary who has corrupted both i and j (or equivalently, i and j collude) can now produce two different signatures with the same unique identifier, and in this case both the identities of i and j can be output by our detection algorithm.)

Case 6. Adversary $\mathcal{A}$ has corrupted user i and produces two signatures (m, τ, ψ_1) and (m, τ, ψ_2) where $\psi_1 \neq \psi_2$ and they both point to i. This case is handled successfully by the algorithm as well.

Case 7. Adversary $\mathcal{A}$ has corrupted two users i and j, and produces two signatures (m, τ_1, ψ_1) and (m, τ_2, ψ_2) where $\tau_1 \neq \tau_2$ and they point to i and j, respectively. In this case, adversary obeys the rules and behaves like legal users. Moreover, it is by uniqueness that it is computationally difficult for adversary $\mathcal{A}$ to generate valid signatures with more than two different unique identifiers if it only obtains two secret signing keys.

This completes the proof of the claim. ▌

C.2 Proof Sketch of Theorem 2, 3, and 4

Proof sketch of Theorem 2 and 3. The proof for CCA-anonymity of Theorem 2 largely follows from one in [8]. The pseudorandomness of the VRF implies that the unique identifier does not leak any "useful" information under the attack of the restricted anonymity adversary. If the underlying encryption is only IND-CPA secure and NIZK proof system just satisfies the regular adaptive zero-knowledge property then the $\mathcal{SGS}_1$ is CPA-anonymous.

The proof for traceability is almost the same as one in [8]. The only difference is that the second-level signature is replaced with a verifiable random function, which does not essentially complicate the proof. To show that the security unforgeability under random message attacks suffices for the first-level signature suffices, we require an additional (and weak) assumption that the verification key for the second-level signature should randomly and independently distributed if the corresponding signing key is chosen uniformly at random.

We now sketch the proof for uniqueness. We consider two cases: the first case is that among all the signatures that adversary outputs there at least exists one group signature (m, τ, C, π) such

[12] Note even if the adversary repeats the query it will only return signatures with the same unique identifier τ' due to the partly deterministic property of unique group signature. Precisely, once the user j outputs two signatures on the same query m, it technically violates the rule but remains honest.

that $(m, vk, ek, \tau, C) \notin \mathcal{L}_1$ but $V_1((m, \tau, C), \pi) = 1$. The probability is bounded by the soundness advantage $\mathbf{Adv}^{\text{sound}}_{(P_1,V_1)}(\mathcal{A}_1)$ for some adversary $\mathcal{A}_1$; Otherwise, each plaintext under any ciphertext C is of the form $(vk', \nu', \mathsf{cert})$ such that $\mathsf{Vrf}(vk, vk', \mathsf{cert}) = 1$ and $\mathsf{Ver}(vk', m, \tau, \nu') = 1$. This is further divided into two cases: vk' was not certified by vk and vk' was certified by vk. The former case implies that there is an adversary that attacks the unforgeability under *random message attack* of the first-level signature scheme. The latter case implies that either at least one unique identifier corresponds to some uncorrupted users or there are *at least* two unique identifiers corresponding to one same corrupted vk'. This either contracts the traceability property or contradicts uniqueness property of the underlying VRF.

Last, the non-colliding property follows from the pseudorandomness property of the underlying VRF scheme.

For Theorem 4. The proof for Theorem 4 can be likewise obtained following from the above. The difference is that PRF with a NIZK proof naturally has the uniqueness property.

C.3 Proof of Theorem 5

Now we sketch the proof for Theorem 5 that does not immediately follow from the ones for Theorems 2-4. Before we proceed, it is important to identify the second-level primitive. For anonymity, we use the PRF with corresponding proof as the first primitive. While proving the traceability and uniqueness properties, we deem h^s as the signature verification key and $\tau = g^{1/(s+m)}$ as the signature on a message m. The security that we require for the latter two properties is only unforgeability. (Of course, we still need the uniqueness property, but this deterministic signature itself has this property.) Note that here its security (even with the additional proof) cannot be proven under the pseudorandom property of PRF with NIZK proof, since at least the "verification key" h^s *has to* be exposed to the uniqueness adversary. Thus, its security does not follow from DDHI assumption and Groth-Sahai proof system. This signature can still be shown unforgeable under DHI assumption in [28] with a less tight proof. Its security can be also justified under SDHI assumption that we formalize and is weaker than SDDHI assumption that is often used to solve similar privacy-preserving tasks. One can view this signature is a degenerate variant of the PRF with NIZK proof [7] where they share the same tag. This tag has to be pseudorandom when it comes to anonymity, and it only needs to be unforgeable when it comes to uniqueness and traceability. This gives the basic idea, however, one has to deal with other issues like *simulatability* when proving the latter two security requirements.

Non-colliding property. The non-colliding property is impled by the pseudorandomness of the PRF scheme.

CPA-anonymity. This basically follows from the fact that the unique identifier is pseudorandom, the commitment schemes are perfectly hiding (in the witness-indistinguishability setting), and proofs are zero-knowledge (in the witness-indistinguishability setting). Since here we are only concerned with chosen plaintext-attack adversary, the case is much easier. As we mentioned, the PRF with NIZK proof that we use is a variant of Belenkiy et al. [7], which is secure if DDHI assumption holds and Groth-Sahai proof system is secure.

Uniqueness. To justify the uniqueness security, we consider four different kinds of adversaries.

Case 1. Adversary can come up with a group signature $(m, \tau, C_s, C_\theta, \pi_1, \pi_2)$ such that $(m, \tau, C_s) \notin \mathcal{L}_3$ or $(C_s, C_\theta, vk) \notin \mathcal{L}_4$ but passes the group signature verification procedure. We can bound the probability that an adversary can give such a kind of forgery by the soundness error of Groth-Sahai proof system that is zero in the soundness setting.

Case 2. Otherwise, in the soundness setting, for any valid signature, there exist exactly one S and one (vector) of θ such that $\tau = F_s(m)$, $C_s = \mathsf{Com}(S, r_s)$, $C_\theta = \mathsf{Com}(\theta, r_\theta)$, and $\mathsf{Vrf}(vk, S, \theta) = 1$ where $S = h^s$. (For simplicity, we assume the message for signatures is one group S, and θ is the corresponding signature.) We first consider the case where there is at least one group signature such that the verification key S of the second-level signature was not certified by the group manager. We show in this case that one can construct an adversary that can attack the unforgeability under random message attack.

Namely, given an adversary $\mathcal{A}$ that can produce such a group signature, we now construct an adversary $\mathcal{B}$ such that it attacks the underlying first-level signature $\mathcal{DS}$ in the sense of unforgeability under random message attack. More concretely, we consider a signature scheme $\mathcal{DS}$ with a random message oracle (see Section 2.1 for definition). For each query $i \in [n]$ to the random message oracle, adversary $\mathcal{B}$ first selects randomly and independently $s_i \in \mathbb{Z}_q$ and computes $vk_i = h^{s_i}$. It is clear that each vk_i is uniformly and independently from the message space of the Groth-Sahai commitment scheme. The secret signing key $(s_i, h^{s_i}, \mathsf{cert}_i)$ (where cert_i is the answer returned by the first-level signature oracle) is given to each group member. Adversary $\mathcal{B}$ also prepares the Groth-Sahai proof system parameter including the bilinear group parameter, Groth-Sahai commitment parameter, and a common reference string in the soundness setting. Besides, adversary $\mathcal{B}$ keeps the extraction key. $\mathcal{B}$ can answer any queries $\mathcal{A}$ makes, including group user signing queries and user secret queries, since it possesses the secret keys of all group members. When $\mathcal{A}$ finally produces a group signature $(m, \tau, C_s, C_\theta, \pi_1, \pi_2)$ of the assumed type, $\mathcal{B}$ simply uses its extraction key to extract S from C_s and θ from C_θ—which would not cause any problems since we have required the first-level signature to be structure-preserving. $\mathcal{B}$ outputs (S, θ) as its forgery. Since the group signature belongs to the given type, it is a successful forgery for the first-level signature $\mathcal{DS}$. A similar argument applies to the case where the first-level signature is F-unforgeable against random message attack.

Case 3. The third case is that adversary can come up with group signatures such that at least one of the signatures corresponds to some uncorrupted group member and the adversary did not ask group signing oracle on this message. (This is in fact a type of attacks for traceability adversary.) Since we do not know which user corresponds to this forgery on the same message, the adversary $\mathcal{A}$ has to guess the identity i. Given the second-level signature with the verification key h^s with a signing oracle $\mathsf{Sig}(s, \cdot)$ which returns $g^{1/(s+m)}$ on message m, adversary $\mathcal{C}$ now simulates the environment that $\mathcal{A}$ runs in. It first randomly generates all but one second-level signature signing-verification keys $\{s_j, h^{s_j}\}_{j \neq i}$. It also generates a pair of signing/verification keys for first-level signature scheme and signs the verification key for every user including user i. Then $\mathcal{C}$ runs $\mathcal{A}$ and answers the queries (i.e., group user secret oracle queries and group user signing oracle queries) that $\mathcal{A}$ makes. Adversary $\mathcal{C}$ can answer requests of type (j, m) such that $j \neq i$, since it knows the secret signing key of each user. But it is not obvious for $\mathcal{C}$ to answer requests of the form (i, m) for some message m, as $\mathcal{C}$ does not obtain s for user i that is part of the witnesses for the NIZK proof for the PRF. Although the adversary $\mathcal{C}$ does not have s, it can still get the signature $g^{1/(s+m)}$ on m via the signature oracle of user i (and of course the verification key h^s). Adversary $\mathcal{C}$ can in fact use these instead of s to build the first part of NIZK proof π_1. Specifically, one can check that $\pi_1 = (C_\tau, \pi_\tau, C'_s, \pi_s, \pi')$ can all be given without directly using s! The adversary $\mathcal{A}$ does not request the secret signing key for user i, since otherwise it would have corrupted i such that the forgery is not successful. Therefore, adversary $\mathcal{C}$ can answer all requests that $\mathcal{A}$ makes. If adversary $\mathcal{A}$ finally produces signatures that contains at least one assumed forgery then adversary $\mathcal{C}$ can give a successful forgery for the second-level signature with verification key h^s.

Case 4. The last case is that adversary can come up with group signatures such that all S was

certified by the group manager and these signatures do not correspond to any of the uncorrupted group members. If in this case there is an adversary $\mathcal{A}$ can attack the uniqueness property then we can construct another adversary $\mathcal{D}$ that attacks the uniqueness property of the first-level unique signature scheme. Given a second-level unique signature scheme, the adversary $\mathcal{D}$ simulates the environment that the adversary $\mathcal{A}$ runs in. Specifically, the adversary $\mathcal{D}$ starts by preparing the common reference string and keeps the extraction key. It then generates all signing/verification keys $\{s_i, h^{s_i}\}$ for each user i. The adversary $\mathcal{D}$ can easily answer all of the requests that adversary $\mathcal{A}$ makes, for it possesses all of the secret signing keys of users. If finally adversary $\mathcal{A}$ comes up with $|\mathtt{CU}| + 1$ signatures on some message m with distinct unique identifiers, then this implies that there are *at least* two unique identifiers corresponding to one same registered but uncorrupted user with some second-level signature verification key S'. Adversary $\mathcal{D}$ can extract the necessary witnesses and outputs (m, S', τ', τ''), where τ' and τ'' are both valid signatures on m relative to S'.

Traceability. Like the proofs for Theorem 2-4, we classify the types of forgeries into three cases that are similar to *Case 1-3* for the uniqueness property proof. This is not surprising since we use signatures and NIZK proofs as main tools to achieve our construction.

C.4 Proof of Theorem 6

We outline the proof of Theorem 6 as follows.

Non-colliding property. This follows from the fact that pseudorandomness property of the PRF function.

CCA-anonymity. The proof is analogous to Groth's construction [39] but is more complicated. (The basic idea is to use explicitly an encryption scheme secure under chosen-ciphertext attacks to extend the underlying CPA-anonymous group signature.) We outline the proof as follows. Let Game 0 be the original CCA-anonymity game involving an anonymity adversary $\mathcal{A}$. We assume that this is in the soundness setting. In Game 1, we simulate the opening oracle queries by decrypting the corresponding Kiltz's encryption C. This is guaranteed by the soundness property of the NIZK proof system (P_3', V_3'). In Game 2, we turn to the witness-indistinguishability setting. In Game 3, we simulate all the proofs. In Game 4, the unique identifier is replaced with a random string. This does not cause any problem due to PRF security and property on signing on committed value protocol. More precisely, what we require here is the user privacy property (see definition from [6]). They are indistinguishable even for adversary in possession of the issuing key. In Game 5, an adversary that attacks the security of CCA-anonymity of the group signature can be transferred to an adversary that attacks the underlying tag-based encryption. This can be informally justified as follows: the adversary cannot produce a group signature with the same tag as one for the challenge signature, otherwise one can attack the unforgeability of the one-time signature. The rest of the opening oracle queries (with some other tags) can be answered by the decryption oracle of tag-based encryption. The additional unique identifier is random and thus it makes no difference for attacking the tag-based encryption.

Uniqueness. For uniqueness property, we can focus our attention on the first-chaining. This is mainly due to the uniqueness property of the unique signature and the unforgeability property of the signing on committed value protocol. If an adversary outputs signatures than required then it implies that there exists an adversary can attack the unforgeability of the first-level signature for the first-chaining with a proof analogous to the one for Theorem 5. Note that one must make sure that the forgery can be extracted (which is the case for the construction using Fuchsbauer's blind signature), otherwise we have to resort to strong f-extractable proof of knowledge (which is for the

construction using Belenkiy et al.'s P-signature).

Traceability and Non-frameability. Due to the (perfect) knowledge soundness of Groth-Sahai proof system, the opener can extract $(vk^*, \sigma^*, \mathsf{cert}^*)$ such that $\mathcal{DS}'_1.\mathsf{Vrf}(vk', vk^*, \mathsf{cert}^*) = 1$ and $\mathcal{DS}'_2.\mathsf{Vrf}(vk^*, vk_o, \sigma^*) = 1$. If vk^* does not correspond to any group member registered then we can construct an adversary that attacks the unforgeability of $\mathcal{DS}'_1$. Therefore, the traceability follows.

For non-frameability, we first notice that any forgery to frame a user requires a signature σ^* on a new value vk_o, which is due to the strong unforgeability of the one-time signature. Thus, any successful framing in fact corresponds to a successful forgery on the $\mathcal{DS}'_2$. Of course, the adversary has to guess in advance which user to frame such that the reduction loses a factor of n that denotes the number of users added by $\mathcal{A}$ via the $\mathsf{AddU}(\cdot)$ oracle.