Group Message Authentication*

Bartosz Przydatek
Google Switzerland
przydatek@google.com

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

Abstract. Group signatures is a powerful primitive with many practical applications, allowing a group of parties to share a signature functionality, while protecting the anonymity of the signer. However, despite intensive research in the past years, there is still no fully satisfactory implementation of group signatures in the plain model. The schemes proposed so far are either too inefficient to be used in practice, or their security is based on rather strong, non-standard assumptions.
We observe that for some applications the full power of group signatures is not necessary. For example, a group signature can be verified by any third party, while in many applications such a universal verifiability is not needed or even not desired. Motivated by this observation, we propose a notion of *group message authentication*, which can be viewed as a relaxation of group signatures. Group message authentication enjoys the group-oriented features of group signatures, while dropping some of the features which are not needed in many real-life scenarios. An example application of group message authentication is an implementation of an *anonymous* credit card.
We present a generic implementation of group message authentication, and also propose an efficient concrete implementation based on standard assumptions, namely strong RSA and DDH.

1 Introduction

A typical sequence of events in an offline credit card purchase is as follows: A card holder authenticates himself using his card and leaves a receipt of purchase to the merchant. The merchant then gives the receipt to the bank and the bank transfers money from the card holder's account to the merchant's account.

A natural way to improve the security of this scheme is to use smartcards and let a smartcard digitally authenticate each purchase transaction on behalf of the card holder. Message authentication can be achieved using a digital signature scheme. A drawback of this approach, and also of the original scheme, is that it reveals the identity of the card holder to the merchant.

Chaum and van Heyst [9] introduced group signatures to resolve this, and other similar privacy problems. When using a group signature scheme the signatures computed by different signers are indistinguishable, i.e., they provide unlinkability and anonymity within the group of signers. On the other hand a special party, called the group manager, has the ability to open any valid signature and identify the signer. In the application described above, the bank would

* Work done in part at ETH Zurich. The full version of this paper is available on Cryptology ePrint Archive [29]

play the role of the group manager and each credit card would use a unique signing key. This solution still reveals the correspondence between purchases and card holders to the bank, but in practice this is not a serious problem. Not only would customers leave a bank if it did not treat customer information carefully, but in most countries banks are required to do so by law and they are typically under supervision of some authority.

In principle, group signatures can be constructed under general assumptions [3], but these constructions are prohibitively inefficient for practical purposes. There are efficient schemes, e.g., Ateniese *et al.* [1] or Boyen and Waters [5], but the security of these schemes rests on non-standard pairing-based assumptions, which are still controversial in the cryptographic community. There are also efficient and provably secure realizations of group signatures in the random oracle model, e.g., the scheme given by Camenisch and Groth [6], but the random oracle model is not sound [7]. Thus, a proof of security in this model does not necessarily imply that the scheme is secure when the random oracle is instantiated by an efficiently computable function.

To summarize, despite intensive research there is still no *efficient* group signature scheme provably secure in the *plain model* under *standard* assumptions. This motivates the study of relaxed notions for special applications of group signatures that allows for a simpler solution.

A closer look at our motivating anonymous credit cards problem reveals that group signatures provide several features that are not essential in this setting:

- Signatures are publicly verifiable, while in our setting only the merchant and the bank must be able to verify the authenticity of a transaction, as no other party even receives a signature.
- Group signatures are non-interactive. This is crucial for the bank, since it may receive a large number of transactions from the numerous senders. However, in many applications it is not essential that the merchant is able to verify the authenticity of a transaction without interacting with the sender.
- Although there are exceptions, it is typically required from a group signature scheme that the group manager is unable to frame signers, i.e., he cannot compute signatures on their behalf. This property is not essential in a credit card system, since the bank is trusted to manage all the transactions properly anyway, and would quickly lose all customers if it framed a few of them.

Contributions. Our main contribution is threefold:

- Motivated by our observation that in some classical applications of group signatures a fully blown group signature scheme is not really needed, we formalize a relaxed notion that we call *group message authentication*.
- We give a generic construction of a group message authentication scheme that satisfies our relaxed security definitions.
- We instantiate the generic construction efficiently and in a provably secure way in the plain model under the decision Diffie-Hellman assumption and the strong RSA assumption.

Thus, for an important special case of the original motivating application of group signatures we give the first fully satisfactory solution. We also give the first reduction of the security of the Cramer-Shoup cryptosystem with labels over a cyclic group of composite order to its security over each subgroup. A direct proof can be achieved by adapting the original proof due to Cramer and Shoup (cf. [26]), but we think our analysis is of independent interest.

In group signatures in the random oracle model (cf. [6]), the signer encrypts a token and gives a non-interactive Fiat-Shamir [15] proof, with the message as a prefix to the hashfunction, that the ciphertext was formed in this way. It is tempting to conclude that if we skip the Fiat-Shamir transform, we get a secure group message authentication scheme, but this does not work, since the random oracle is used for three purposes: (a) to embed the message and provide unforgeability, (b) to prove that a ciphertext contains a token, and (c) to prove knowledge of an encrypted valid token and thereby provide the CCA2-like security needed in group signature schemes. One can use a CCA2-secure cryptosystem to avoid (c), but without the Fiat-Shamir proof there is nowhere to embed the message. We could (along the lines of [3]) encrypt a *standard* signature of the message along with the signer's public key and a certificate thereof, but no *efficient* proof that a plaintext has this form is known. We instead use a CCA2-secure cryptosystem with labels and *embed the message in the label*.

Even taken as a group signature scheme (using Fiat-Shamir to eliminate interaction), our construction is novel and, interestingly, its security holds in the *plain* model where the group manager plays the role of the verifier, i.e., he can detect signatures forged due to the failure of Fiat-Shamir heuristic.

Related Work. In addition to the intensive work on group signatures mentioned above, there has been substantial interest in other aspects of group-oriented cryptography.[1] In particular, many researchers have explored numerous variations of group signatures with additional properties such as: traceable signatures [21], multi-group and subgroup signatures [2, 24], or hierarchical group signatures [33]. In contrast to these various extensions of group signatures, group message authentication is actually a relaxation of group signatures.

Another related primitive is identity escrow [23], which applies key-escrow ideas to the problem of user identification. In contrast to group signatures or group message authentication, identity escrow does not allow any form of signing a message. More precisely, identity escrow employs an identity token that allows the holder to anonymously authenticate itself as a member of a group, but it does not allow the holder to sign any messages. Furthermore, identity escrow introduces an additional party, *escrow agent*, and requires separability: the agent remains dormant during normal operation of the identification system, and is "woken up" only when there is a request to revoke anonymity. This feature and the requirement of resistance to impersonation imply that not every group signature scheme can be used as an identity escrow scheme.

[1] In independent work, Laur and Pasini [25] use the term "group message authentication" in a different context.

Other notions related to group message authentication include designated verifier signatures [20] and designated confirmer signatures [8]. Recently, Kiayias *et al.* [22] and Qin *et al.* [30] proposed a group-oriented notion of cryptosystems.

Notation. We denote the set $\{j, j+1, j+2, \ldots, k\}$ of integers by $[j,k]$. We write: $\mathbb{Z}_N$ for the integers modulo N, $\mathbb{Z}_N^*$ for its multiplicative group, and SQ_N for the subgroup of squares in $\mathbb{Z}_N^*$. We say that a prime q is safe if $(q-1)/2$ is prime. We use n as our main security parameter and say that a function $\epsilon(n)$ is negligible if for every constant c, $\epsilon(n) < n^{-c}$ for every sufficiently large n. We say that the probability of an event is overwhelming if it is at least $1-\epsilon(n)$, where $\epsilon(n)$ is negligible. Given a public key cpk of a cryptosystem we denote the plaintext space by $\mathcal{M}_{cpk}$, the ciphertext space by $\mathcal{C}_{cpk}$, and the randomizer space by $\mathcal{R}_{cpk}$. We use PT to denote the set of deterministic polynomial time algorithms, PPT the set of probabilistic polynomial time algorithms, and IPPT the set of interactive, probabilistic polynomial time algorithms (sometimes with oracles). Given $P, V \in$ IPPT, we denote by $\langle P(w), V(z)\rangle(x)$ the output of V when executed on input (z,x) and interacting with P on input (w,x). We write $V[\![(P_i(w_i))_{i\in[1,k]}]\!]$ when V interacts over separate communication tapes with k copies of $P_i(w_i)$ running on inputs $w_1, \ldots, w_k$ respectively, i.e., it has "oracle access" to these machines.

2 Group Message Authentication Schemes

To avoid confusion with group signatures and related notions we refer to the parties in a group message authentication (GMA) scheme as the *receiver*, *proxies*, and *senders*, and we say that a sender computes an *authentication tag*. Compared to a group signature scheme the role of the receiver is similar to the group manager. It can verify and open an authentication tag, but in a GMA scheme it may need its secret key also to verify a tag. Thus, we combine these two operations into a single *checking algorithm* that outputs an identity if the authentication tag is valid, and outputs $\perp$ otherwise. The role of a sender is similar to a signer, except that when it hands an authentication tag to a proxy, it also executes an interactive *authentication protocol* that convinces the proxy that the receiver will accept it. The role of a proxy corresponds to the holder of a signature, except that it can not hand the signature to anybody but the receiver.

Definition 1 (Group Message Authentication Scheme). *A* group message authentication scheme *consists of four algorithms* $(\mathsf{RKg}, \mathsf{AKg}, \mathsf{Aut}, \mathsf{Check})$ *and a protocol* π_a*, associated with a polynomial* $\ell(\cdot)$*:*

1. *A receiver key generation algorithm* $\mathsf{RKg} \in$ PPT*, that on input* 1^n *outputs a public key* pk *and a secret key* sk*.*
2. *An authentication key generation algorithm* $\mathsf{AKg} \in$ PPT*, that on input* 1^n*, a receiver key* sk*, and an integer* $i \in [1, \ell(n)]$ *outputs an authentication key* ak_i*.*
3. *An authentication algorithm* $\mathsf{Aut} \in$ PPT*, that on input a public receiver key* pk*, an authentication key* ak_i*, and a message* $m \in \{0,1\}^*$ *outputs an authentication tag* σ*.*

4. *A checking algorithm* $\mathsf{Check} \in \mathrm{PT}$*, that on input the secret receiver key* sk*, a message* $m \in \{0,1\}^*$*, and a candidate authentication tag* σ*, outputs an integer* $i \in [1, \ell(n)]$ *or* $\perp$*.*
5. *An interactive 2-party authentication protocol* $\pi_a = (P_a, V_a) \in \mathrm{IPPT}^2$*, such that the output of the verifier* V_a *is a bit.*

For every $n \in \mathbb{N}$*, every* $(pk, sk) \in \mathsf{RKg}(1^n)$*, every integer* $i \in [1, \ell(n)]$*, every authentication key* $ak_i \in \mathsf{AKg}_{sk}(1^n, i)$*, every message* $m \in \{0,1\}^*$*, and every* $r \in \{0,1\}^*$ *the following holds: if* $\sigma = \mathsf{Aut}_{ak_i,r}(pk, m)$*, then* $\mathsf{Check}_{sk}(m, \sigma) = i$ *and the probability* $\Pr[\langle P_a(ak_i, r), V_a\rangle(pk, m, \sigma) = 1]$ *is overwhelming.*

2.1 Definition of Security

Conceptually, the security requirements of a GMA scheme must guarantee: that authentication tags are indistinguishable, that it is infeasible to forge an authentication tag, that the receiver can always trace the sender, and that the proxy is never convinced that an invalid authentication tag is valid. We formalize these properties with two experiments similarly as is done in [3] for group signatures.

Experiment 1 (Anonymity, $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n)$**).**

$$
\begin{aligned}
(pk, sk) &\leftarrow \mathsf{RKg}(1^n) && \text{// receiver key}\\
ak_i &\leftarrow \mathsf{AKg}_{sk}(1^n, i) \ \textit{ for } \ i \in [1, \ell(n)] && \text{// auth. keys}\\
(m, i_0, i_1, \mathrm{state}) &\leftarrow A^{\mathsf{Check}_{sk}(\cdot,\cdot)}(pk, ak_1, \ldots, ak_{\ell(n)}) && \text{// choose ids}\\
\sigma &\leftarrow \mathsf{Aut}_{ak_{i_b},r}(pk, m), \ \textit{with random } r \in \{0,1\}^* && \text{// challenge}\\
d &\leftarrow \langle P_a(ak_{i_b}, r), A^{\mathsf{Check}_{sk}(\cdot,\cdot)}(\mathrm{state})\rangle(pk, \sigma, m) && \text{// guess}
\end{aligned}
$$

If the $\mathsf{Check}_{sk}(\cdot,\cdot)$*-oracle was never queried with* (m, σ)*, then output* d*, otherwise output* 0*.*

Anonymity. We define one experiment for each value of $b \in \{0,1\}$ and then require that the distributions of the two experiments are close. The adversary is given the receiver's public key and all authentication keys. Then it chooses two identities of senders and a message, and hands these to the experiment. The bth experiment chooses the bth of the identities and computes an authentication tag of the given message using the authentication key of this identity. Then it hands the authentication tag to the adversary and executes the authentication protocol on behalf of the chosen identity. Finally, the adversary must guess which of the two authentication keys was used to authenticate the message and execute the authentication protocol. During the experiment the adversary also has access to a checking oracle to which it may send any query, except the challenge message-and-authentication tag pair.[2]

[2] No oracle for authentication or running the authentication protocol is needed, since the adversary can simulate these using the authentication keys $ak_1, \ldots, ak_{\ell(n)}$. A

Definition 2 (Anonymity). *A group message authentication scheme* GMA *is anonymous if* $\forall A \in \text{IPPT}$ *the following expression is negligible*

$$|\Pr[\mathsf{Exp}^{\mathsf{anon}-0}_{\mathsf{GMA},A}(n)=1] - \Pr[\mathsf{Exp}^{\mathsf{anon}-1}_{\mathsf{GMA},A}(n)=1]|\,.$$

Experiment 2 (Traceability, $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n)$).

$$\begin{aligned}
(pk, sk) &\leftarrow \mathsf{RKg}(1^n) && \textit{// receiver key}\\
ak_i &\leftarrow \mathsf{AKg}_{sk}(1^n, i) \ \textit{for}\ i \in [1,\ell(n)] && \textit{// auth. keys}\\
\textit{Define}\ ak(i) &= \begin{cases} ak_i & \textit{if } i \in [1,\ell(n)] \\ \perp & \textit{otherwise} \end{cases} && \textit{// auth. key oracle}\\
(m,\sigma,\text{state}) &\leftarrow A^{ak(\cdot),\mathsf{Check}_{sk}(\cdot,\cdot)}[\![(P_a^+(pk,ak_i))_{i\in[1,\ell(n)]}]\!](pk) && \textit{// forge auth. tag...}\\
d &\leftarrow \langle A^{ak(\cdot),\mathsf{Check}_{sk}(\cdot,\cdot)}(state), V_a\rangle(pk,m,\sigma) && \textit{// ..or authenticate}\\
& && \textit{// an invalid tag}
\end{aligned}$$

Let $\mathcal{C}$ *be the set of queries asked by* A *to the* $ak(\cdot)$*-oracle. If* $\mathsf{Check}_{sk}(m,\sigma) \in [1,\ell(n)] \setminus \mathcal{C}$ *and* P_a^+ *has never output* σ*, or if* $\mathsf{Check}_{sk}(m,\sigma) = \perp$ *and* $d = 1$*, then output* 1*, otherwise output* 0*.*

Traceability. The adversary is given the receiver's public key, and during the experiment it has access to a checking oracle, it may interact with honest senders, and it may corrupt any sender to acquire its authentication key. To succeed, the adversary must either forge an authentication tag that checks to the identity of an uncorrupted sender, or it must output an authentication tag that checks to $\perp$ and convince the honest verifier of the authentication protocol that the tag checks to an identity.[3]

Denote by $P_a^+ \in \text{IPPT}$ the machine that accepts (pk, ak_i) as input and repeatedly waits for messages on its communication tape. Given an input (Aut, m) on its communication tape P_a^+ computes and outputs $\sigma = \mathsf{Aut}_{ak_i,r}(pk, m)$, and then executes P_a on common input (pk, m, σ) and private input (ak_i, r). In the traceability experiment the adversary has "oracle access" to several copies of P_a^+. We stress that although the "oracle access" to each copy P_a^+ running on some input (pk, ak_i) is sequential by the definition of P_a^+, the adversary may interact concurrently with different copies.[4] This is essential for the definition to be realistic, as we can not assume that different senders are aware of each other.

---

standard hybrid argument then shows that it suffices to consider a single invokation of the authentication protocol as in the definition.

[3] Traceability could alternatively be formalized by two separate experiments, where each experiment captures one type of attack, but we think this is less natural.

[4] This is a benign form of concurrency, since each copy of P_a^+ executes using its own independently generated private input (only the public inputs are dependent). Thus, our setting is the sequential setting in disguise.

Definition 3 (Traceability). *A group message authentication scheme* GMA *is traceable if for* $\forall A \in \mathrm{IPPT}$ *the probability* $\Pr[\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n) = 1]$ *is negligible.*

Definition 4 (Security). *A group message authentication scheme* GMA *is secure if it is anonymous and traceable.*

We stress, that while the new notion is related to variants of group signatures, it is essentially different from previous work. In particular, in contrast to various enhancements of group signatures, like traceable signatures [21] or identity escrow [23] (cf. Sect. 1), group message authentication it is a *relaxation* of group signatures, aiming at typical applications, and improved efficiency and security.

3 Tools

To construct the algorithms of the group message authentication scheme we use two basic primitives: a bounded signature scheme secure against chosen message attacks and a CCA2-secure cryptosystem with labels. The authentication protocol is loosely speaking a "zero-knowledge proof", but we use relaxed notions that allows efficient instantiation.

Bounded Signature Schemes. Each sender in a GMA scheme is given a unique authentication key that it later uses to authenticate messages. In our construction an authentication key is a signature of the identity of the holder, but a fully blown signature scheme is not needed. Note that standard signature schemes can be used to sign any message from an exponentially large space of strings, but in our setting we only need to sign a polynomial number of different integers. We call a signature scheme with this restriction *bounded*.

Definition 5 (Bounded Signature Scheme). *A* bounded signature scheme *consists of three algorithms* $(\mathsf{SKg}, \mathsf{Sig}, \mathsf{Vf})$ *associated with a polynomial* $\ell(n)$*:*

1. *A key generation algorithm* $\mathsf{SKg} \in \mathrm{PPT}$*, that on input* 1^n *outputs a public key spk and a secret key ssk.*
2. *A signature algorithm* $\mathsf{Sig} \in \mathrm{PPT}$*, that on input a secret key ssk and a message* $m \in [1, \ell(n)]$ *outputs a signature* s*.*
3. *A verification algorithm* $\mathsf{Vf} \in \mathrm{PT}$*, that on input a public key spk, a message* $m \in [1, \ell(n)]$*, and a candidate signature* s*, outputs a bit.*

For every $n \in \mathbb{N}$*, every* $(spk, ssk) \in \mathsf{SKg}(1^n)$*, every message* $m \in [1, \ell(n)]$*, and every* $s \in \mathsf{Sig}_{ssk}(m)$*, it must hold that* $\mathsf{Vf}_{spk}(m, s) = 1$*.*

The standard definition of security against chosen message attacks (CMA) [19] is then directly applicable. The existence of an ordinary signature scheme clearly implies the existence of a bounded one.

Cryptosystems With Labels. Cryptosystems with labels were introduced by Shoup and Gennaro [31]. The idea of this notion is to associate a label with a ciphertext without providing any secrecy for the label. One simple way of constructing such cryptosystems is to append to the plaintext before encryption a collision-free hash digest of the label, but there are simpler constructions in practice. As a result, the input to the cryptosystem is not only a message, but also a label, and similarly for the decryption algorithm. Below we recall the definition, with a small modification — in the standard definition the decryption algorithm outputs only the message, without the label. Clearly, this is a minor modification, but we need it to allow certain joint encodings of the label and message in the analysis.

Definition 6 (Public Key Cryptosystem With Labels [31]). *A* public key cryptosystem with labels *consists of three algorithms* $(\mathsf{CKg}, \mathsf{Enc}, \mathsf{Dec})$*:*

1. *A key generation algorithm* $\mathsf{CKg} \in \mathrm{PPT}$*, that on input* 1^n *outputs a public key* cpk *and a secret key* csk*.*
2. *An encryption algorithm* $\mathsf{Enc} \in \mathrm{PPT}$*, that on input a public key* cpk*, a label* l*, and a message* $m \in \mathcal{M}_{cpk}$ *outputs a ciphertext* c*.*
3. *A decryption algorithm* $\mathsf{Dec} \in \mathrm{PT}$*, that on input a secret key* csk*, a label* l*, and a ciphertext* c *outputs the label and a message,* $(l, m) \in \{0,1\}^* \times \mathcal{M}_{cpk}$*, or* $\perp$*.*

For every $n \in \mathbb{N}$*, every* $(csk, cpk) = \mathsf{CKg}(1^n)$*, every label* $l \in \{0,1\}^*$*, and every* $m \in \mathcal{M}_{cpk}$ *it must hold that* $\mathsf{Dec}_{csk}(l, \mathsf{Enc}_{cpk}(l, m)) = (l, m)$*.*

Security against chosen ciphertext attacks is then defined as for standard cryptosystems except for some minor changes. In addition to the challenge messages, the adversary outputs a label to be used in the construction of the challenge ciphertext c. The adversary may also ask any query except the pair (l, c).

Experiment 3 (CCA2-Security With Labels [31], $\mathsf{Exp}^{\mathsf{cca2}-b}_{\mathsf{CSL},A}(n)$).

$$
\begin{aligned}
(cpk, csk) &\leftarrow \mathsf{CKg}(1^n) \\
(l, m_0, m_1, \mathrm{state}) &\leftarrow A^{\mathsf{Dec}_{csk}(\cdot,\cdot)}(\mathsf{choose}, cpk) \\
c &\leftarrow \mathsf{Enc}_{cpk}(l, m_b) \\
d &\leftarrow A^{\mathsf{Dec}_{csk}(\cdot,\cdot)}(\mathsf{guess}, \mathrm{state}, c)
\end{aligned}
$$

If $\mathsf{Dec}_{csk}(\cdot,\cdot)$ *was queried on* (l, c)*, then output 0, otherwise output* d*.*

Definition 7 (CCA2-Security). *Let* CSL *denote a public key cryptosystem with labels. We say that the cryptosystem* CSL *is CCA2-secure if for every adversary* $A \in \mathrm{PPT}$ *the quantity* $\left|\Pr[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathsf{CSL},A}(n) = 1] - \Pr[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathsf{CSL},A}(n) = 1]\right|$ *is negligible.*

A Relaxed Notion of Computational Soundness. Another tool used in our constructions, allowing for more efficient protocols, is a relaxed notion of soundness. In contrast to the standard (computational) soundness [18, 28], we do not require that the protocol is sound for all inputs, but only when a part of the input is chosen according to a specific distribution, and the rest of the input is chosen by the adversary. Such a relaxation allows to capture scenarios where it is safe to assume that some parameters are chosen according to some prescribed distribution. For example, a bank will usually pick faithfully the keys guarding its transactions. A similar notion (without the oracle) has been used implicitly in several papers and is sometimes called a "computationally convincing proof", see e.g., [13]. In fact, non-interactive computationally sound proofs may be viewed as an instance of this notion.

Definition 8 ((T,O)-Soundness). *Let $T \in$ PPT and let $O \in$ PT be an oracle. Define a random variable $(t_1, t_2, t_3) = T(1^n)$. A 2-party protocol $(P, V) \in$ IPPT$\times$ IPPT is (T, O)-sound for language L if for every instance chooser $I \in$ PT and prover $P^* \in$ PT the following holds: If $(y, z) = I^{O(t_3,\cdot)}(t_1, t_2)$ and $x = (t_1, y)$, then $\Pr[x \notin L \wedge \langle P^{O(t_3,\cdot)*}(z), V\rangle(x) = 1]$ is negligible.*

In the above definition we use generic names T, I, O and t_1, t_2, t_3, y, z to denote abstractly the involved algorithms and the information exchanged between them. The actual meaning and function of these parameters depends on a concrete scenario. For example, in the context of group message authentication algorithm T generates keys for both a signature scheme and a public-key encryption scheme, algorithm O computes signatures, and the parameters t_1, t_2, t_3 correspond to public and secret keys generated faithfully by the receiver (bank) using algorithm T: t_1 denotes (signature and encryption) public keys, t_2 denotes encryption secret key, and t_3 denotes signature secret key (cf. Construction 1). Obviously, if a protocol is sound in the standard sense, it is (T, O)-sound for any T, O. In a slight abuse of notation, we write (X, O)-sound also when X is a polynomially samplable random variable.

A Relaxed Notion of Computational Zero-Knowledge. Recall that a protocol is zero-knowledge if it can be simulated for *every* instance. Goldreich [17] introduced the notion of uniform zero-knowledge to capture the fact that for uniform adversaries it is sufficient to require that no instance for which the protocol leaks knowledge can be found. Wikström [36] generalized this idea to capture settings where the choice of instance is somehow restricted by an instance compiler F and randomized by some sampling algorithm T out of control of the adversary. We generalize Wikström's definition. As in the case of relaxed computational soundness, we use generic names T, I, O, F and t_1, t_2, y, z to denote the involved algorithms and the information exchanged between them. In the concrete context of group message authentication these names gain concrete meaning, e.g., T is a key generation algorithm of a public-key cryptosystem, O is a decryption oracle, and t_1, t_2 denote the public and secret keys, respectively, generated faithfully by the receiver using algorithm T (cf. Construction 1).

A sampling algorithm T outputs a sample $t = (t_1, t_2)$. The first part, t_1, is given to an instance chooser I which outputs a tuple (y, z), where y influences the choice of instance x, and z is an auxiliary input. An instance compiler F takes (t_1, y) as input and forms an instance (x, w) according to some predefined rule. A protocol is said to be (T, F, O)-zero-knowledge, for some oracle O, if for every malicious verifier V^* and every constant $c > 0$ there is a simulator M such that for every instance chooser I as above no distinguisher D can distinguish a real view of V^* from the view simulated by M with advantage better than n^{-c}, when all of these machines have access to the oracle $O(t_2, \cdot)$. Thus, the algorithms T, F, and O represent a class of environments in which the protocol remains zero-knowledge in the ϵ-zero-knowledge sense of Dwork, Naor, and Sahai [14]. We use the following experiment to define our notion formally.

Experiment 4 (Zero-Knowledge, $\mathsf{Exp}^{(T,F,O)-\mathsf{zk}-b}_{\pi,R,I,V^*,M,D}(n)$).

$$
\begin{aligned}
(t_1, t_2) &\leftarrow T(1^n) \\
(y, z) &\leftarrow I^{O(t_2,\cdot)}(1^n, t_1) \\
(x, w) &\leftarrow F(t_1, y) \\
d &\leftarrow \begin{cases} D^{O(t_2,\cdot)}(x, z, \langle P(w), V^{*O(t_2,\cdot)}(z)\rangle(x)) & \text{if } b{=}0 \\ D^{O(t_2,\cdot)}(x, z, M^{O(t_2,\cdot)}(z, x)) & \text{if } b{=}1 \end{cases}
\end{aligned}
$$

If $R(x, w) = 0$ or if the output of V^*_a or M respectively does not contain the list of oracle queries as a postfix, then output 0, otherwise output d.

The requirement that the list of queries made is output is quite natural. It captures that the simulator should not be able to ask more queries, or more powerful queries than the real verifier.

Definition 9 ((T, F, O)-Zero-Knowledge). *Let $\pi = (P, V)$ be an interactive protocol, let $T \in$ PPT be a sampling algorithm, let $F \in$ PT be an instance compiler, let $O \in$ PT be an oracle, and let R be a relation. We say that π is (T, F, O)-zero-knowledge for R if for every verifier $V^* \in$ PPT and every constant $c > 0$ there exists a simulator $M \in$ PPT such that for every instance chooser $I \in$ PPT and every distinguisher $D \in$ PPT:*

$$
\left| \Pr[\mathsf{Exp}^{(T,F,O)-\mathsf{zk}-0}_{\pi,R,I,V^*,M,D}(n) = 1] - \Pr[\mathsf{Exp}^{(T,F,O)-\mathsf{zk}-1}_{\pi,R,I,V^*,M,D}(n) = 1] \right| < n^{-c} \quad .
$$

In our security proof we exploit that a protocol that satisfies the definition can be simulated polynomially many times sequentially, where the instance chooser chooses a new common and private input for each execution (see [29] for details).

4 A Generic Construction

The idea of our GMA scheme is simple. The group manager generates a key pair (spk, ssk) of a bounded signature scheme $\mathsf{BSS} = (\mathsf{SKg}, \mathsf{Sig}, \mathsf{Vf})$, and a key pair

(cpk, csk) of a CCA2-secure cryptosystem with labels $\mathsf{CSL} = (\mathsf{CKg}, \mathsf{Enc}, \mathsf{Dec})$. The ith user is given the authentication key $ak_i = \mathsf{Sig}_{ssk}(i)$. To compute an authentication tag σ of a message m, the user simply encrypts its secret key using the message m as a label, i.e., he computes $\sigma = \mathsf{Enc}_{cpk}(m, ak_i)$.

We assume that SKg is implemented in two steps SKg_1 and SKg_2: on input 1^n the algorithm SKg_1 outputs a string spk_1, that is given as input to SKg_2, which in turn outputs a key pair (spk, ssk), where $spk = (spk_1, spk_2)$. We also assume that CKg can be divided into CKg_1, and CKg_2 in a similar way and that $\mathsf{CKg}_1(1^n)$ is identically distributed to $\mathsf{SKg}_1(1^n)$. Note that any pair of a signature scheme and a cryptosystem can be viewed in this way by letting $\mathsf{SKg}_1(1^n) = \mathsf{CKg}_1(1^n) = 1^n$. This allows the signature scheme and the cryptosystem to generate dependent keys which share algebraic structure.

Construction 1 (Group Message Authentication Scheme GMA). Given a polynomial $\ell(n)$, a bounded signature scheme $\mathsf{BSS} = ((\mathsf{SKg}_1, \mathsf{SKg}_2), \mathsf{Sig}, \mathsf{Vf})$ associated with $\ell(n)$, and a cryptosystem with labels $\mathsf{CSL} = ((\mathsf{CKg}_1, \mathsf{CKg}_2), \mathsf{Enc}, \mathsf{Dec})$, the group message authentication scheme $\mathsf{GMA} = (\mathsf{RKg}, \mathsf{AKg}, \mathsf{Aut}, \mathsf{Check}, \pi_a)$ is constructed as follows:

Receiver Key Generation. On input 1^n the algorithm RKg computes $spk_1 = \mathsf{SKg}_1(1^n)$, $(spk, ssk) = \mathsf{SKg}_2(spk_1)$, and $(cpk, csk) = \mathsf{CKg}_2(spk_1)$, where spk_1 is a prefix of both spk and cpk, and outputs $(pk, sk) = ((cpk, spk), (csk, ssk))$.

Authentication Key Generation. On input $(1^n, sk, i)$ the algorithm AKg outputs an authentication key $ak_i = \mathsf{Sig}_{ssk}(i)$.

Authentication Algorithm. On input (pk, ak_i, m) the algorithm Aut outputs the authentication tag $\sigma = \mathsf{Enc}_{cpk}(m, ak_i)$.

Checking Algorithm. On input (sk, m, σ) the algorithm Check returns the smallest[5] $i \in [1, \ell(n)]$ such that $\mathsf{Vf}_{spk}(i, \mathsf{Dec}_{csk}(m, \sigma)) = 1$ or $\perp$ if no such i exists.

Authentication Protocol. Let R_a denote the relation consisting of pairs $((pk, m, \sigma), (ak_i, r))$, s.t. $\mathsf{Vf}_{spk}(i, ak_i) = 1$ and $\sigma = \mathsf{Enc}_{cpk}(m, ak_i, r)$, and let L_a denote the language corresponding to R_a, i.e., $L_a = \{x : \exists y \text{ s.t. } (x, y) \in R_a\}$ are the honestly encrypted valid tags. Let $F_{\mathsf{Enc}} \in \mathrm{PT}$ take as input a tuple $(cpk, (s, m, i, s', r))$. First F_{Enc} computes $(spk, ssk) = \mathsf{SKg}_2(cpk, s)$ and $ak_i = \mathsf{Sig}_{ssk}(i, s')$, where s resp. s' specify the randomness[6] to be used by SKg_2 resp. Sig_{ssk}. If $i \in [1, \ell(n)]$ holds, then the oracle F_{Enc} outputs $(((cpk, spk), m, \mathsf{Enc}_{cpk}(m, ak_i, r)), (ak_i, r))$, and otherwise it outputs $\perp$.
The authentication protocol π_a must be overwhelmingly complete, $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge for R_a, and $(((cpk, spk), csk, ssk), \mathsf{Sig})$-sound for the language L_a.

Proposition 1. *The construction* GMA *associated with a polynomial* $\ell(n)$ *is a group message authentication scheme. If* CSL *is CCA2-secure, and if* BSS *associated with* $\ell(n)$ *is CMA-secure, then* GMA *is secure.*

[5] In our concrete instantiation at most one index i has this property.

[6] Note that requiring explicit randomness as input ensures that the signature public key spk and the signature ak_i are correctly formed.

A proof is given in [29], but we outline the main ideas of the proof here. The functional property of the scheme is clear by inspection. For anonymity and traceability we argue as follows.

Anonymity. The idea of the proof is to turn a successful adversary A in the anonymity experiment into a successful adversary A' against the CCA2-security of the cryptosystem CSL. We do this in two steps:

1. We replace the invokation of the authentication protocol π_a in the anonymity experiment by a simulation. Due to its $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge property this changes the success probability of the adversary by an arbitrarily small amount. This allows simulation of the experiment without using the secret key of the cryptosystem.
2. We construct a new adversary A' that simulates the modified experiment and breaks the CCA2-security of the cryptosystem. When A outputs (m, i_0, i_1), A' hands (m, ak_{i_0}, ak_{i_1}) to its CCA2-experiment and forwards the challenge ciphertext σ it receives as a challenge authentication tag to A. All checking queries are computed by A' using its decryption oracle, and A' outputs the result of the simulated modified experiment. Thus, when A guesses correctly in the anonymity experiment, A' guesses correctly in the CCA2-experiment.

Traceability. The idea of the proof is to turn a successful attacker A against the traceability of GMA into a successful attacker A' against the CMA-security of the bounded signature scheme BSS. We do this in four steps:

1. We replace each invokation of P_a by a simulation. Due to the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge of π_a and the sequential composition lemma [29] this changes the advantage of the adversary by an arbitrarily small amount.
2. We replace the authentication tags computed by P_a^+ by encryptions of 0. This only reduces the advantage of A negligibly, since CSL is CCA2-secure. The CCA2-security is essential for this argument, since we must be able to simulate the $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle to A without the secret decryption key csk.
3. We use the $(((cpk, spk), csk, ssk), \mathsf{Sig})$-soundness of π_a to argue that the probability that A convinces the honest verifier V_a that an invalid authentication tag is valid is negligible.
4. We show that the adversary in the modified traceability experiment can be used to break the CMA security of the bounded signature scheme. To see this, note that in the modified experiment we may postpone the computation of any authentication key ak_i until the adversary requests it, and to be successful in the modified experiment, A must produce an authentication tag that checks to an uncorrupted sender, i.e., A must produce an encrypted forged bounded signature of a sender's identity, and the simulator holds the secret decryption key.

5 An Efficient Instantiation

We give an efficient instantiation of the above generic scheme and prove its security under the strong RSA assumption and the decision Diffie-Hellman as-

sumption. The strong RSA assumption says that given a modulus $N = pq$, where p and q are random safe primes of the same bit-size, and a random $g \in SQ_N$, it is infeasible to compute (g', e) such that $(g')^e = g \bmod N$ and $e \neq \pm 1$. Let G_q be a prime order q subgroup of $\mathbb{Z}_N^*$ generated by g such that $\log q = O(\log N)$, i.e., the subgroup has "large order". The decision Diffie-Hellman (DDH) assumption for G_q says that if $a, b, c \in \mathbb{Z}_q$ are randomly chosen, then it is infeasible to distinguish the distributions of (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), where we compute modulo N (cf. full version [29] for formal definitions).

5.1 A Bounded Signature Scheme

We construct a bounded signature scheme $\mathsf{BSS}^{\mathrm{rsa}} = (\mathsf{SKg}^{\mathrm{rsa}}, \mathsf{Sig}^{\mathrm{rsa}}, \mathsf{Vf}^{\mathrm{rsa}}, \ell(n))$ for every polynomial $\ell(n)$, that some readers may recognize as a component of several cryptographic constructions based on the strong RSA-assumption. Denote by n_p an additional security parameter, whose value is determined by the authentication protocol.

Construction 2 (Bounded Signature Scheme $\mathsf{BSS}^{\mathrm{rsa}}$).

Key Generation. On input 1^n the algorithm $\mathsf{SKg}_1^{\mathrm{rsa}}$, i.e., the first step of the key generator, picks two random $n/2$-bit safe primes p and q, then defines and outputs $spk_1 = N = pq$. The key generator $\mathsf{SKg}_2^{\mathrm{rsa}}$ on input spk_1 picks a random $g' \in SQ_N$, and defines $g = (g')^2 \prod_{i=1}^{\ell(n)} \rho_i$, where ρ_i is the ith positive prime integer larger than 2^{n_p} with $\rho_i = 3 \bmod 8$. This allows for computing roots of g, as required for the computation of signatures (see next step). Finally $\mathsf{SKg}_2^{\mathrm{rsa}}$ outputs the key pair $(spk, ssk) = ((N, g), g')$.

Signature Algorithm. On input a secret key ssk and a message $m \in [1, \ell(n)]$, the signature algorithm $\mathsf{Sig}^{\mathrm{rsa}}$ computes $\omega = g^{1/(2\rho_m)} \bmod N$ and outputs ω. More precisely, ω is computed as $(g')^{\prod_{i=1, i\neq m}^{\ell(n)} \rho_i} \bmod N$.

Verification Algorithm. On input a public key spk, a message $m \in [1, \ell(n)]$, and a candidate signature ω, the verification algorithm $\mathsf{Vf}^{\mathrm{rsa}}$ verifies that $|\rho_m| > 2$ and $\omega^{2\rho_m} = g \bmod N$ or $\omega^{-2\rho_m} = g \bmod N$.

Equivalently, we could define the keys of $\mathsf{BSS}^{\mathrm{rsa}}$ as $(spk, ssk) = ((N, g), (p, q))$, where $g \in SQ_N$ is picked at random. Then to compute a signature on a message m we would just compute the $2\rho_m$-th root of g modulo N directly, using the factorization of N. This would give exactly the same functionality and the same distribution of the signatures, but would not fit our framework with the two-step key generation, which is why we present the above variant. A simple proof of the proposition below, following older work [12, 16], is given in [29].

Proposition 2. *For every polynomial $\ell(n)$, the scheme $\mathsf{BSS}^{\mathrm{rsa}}$ is a CMA-secure bounded signature scheme under the strong RSA assumption.*

5.2 A Cramer-Shoup Cryptosystem

The original cryptosystem of Cramer and Shoup [11] was given over a group of prime order, but this is not essential. We may view the key generation as consisting of first choosing a group with a generator and then running the key generation of the cryptosystem as described in [11] using these parameters. Denote by n_r an additional security parameter such that 2^{-n_r} is negligible.

Construction 3 (Cramer-Shoup Cryptosystem CSL^{cs} in SQ_N).

Key Generation. On input 1^n, the first key generator CKg_1^{cs} runs exactly as $\mathsf{SKg}^{\mathrm{rsa}}(1^n)$, and outputs N, i.e., a product of two random safe primes. The group is defined as the group SQ_N of squares modulo N. The second key generator CKg_2^{cs} is the original key generator of Cramer and Shoup with some minor modifications. It chooses $g_1, g_2 \in SQ_N$ and $z, x_1, x_2, y_1, y_2 \in [0, N2^{n_r}]$ randomly, and computes $h = g_1^z$, $c = g_1^{x_1} g_2^{x_2}$, and $d = g_1^{y_1} g_2^{y_2}$. Then a collision-free hash function $H : \{0,1\}^* \to [0, \sqrt{N}/2]$ is generated and $(cpk, csk) = ((N, H, g_1, g_2, h, c, d), (z, x_1, x_2, y_1, y_2))$ is output.

Encryption. On input a public key cpk, a label $l \in \{0,1\}^*$, and a message $m \in SQ_N$, the encryption algorithm Enc^{cs} chooses a random $r \in [0, N2^{n_r}]$ and outputs $(u_1, u_2, e, v) = (g_1^r, g_2^r, h^{2r}m, c^r d^{rH(l,u_1,u_2,e)})$.

Decryption. On input a secret key csk, a label $l \in \{0,1\}^*$, and a ciphertext (u_1, u_2, e, v), the decryption algorithm Dec^{cs} checks if

$$u_1^{2x_1} u_2^{2x_2} (u_1^{y_1} u_2^{y_2})^{2H(l,u_1,u_2,e)} = v^2 \,.$$

If so, it outputs eu_1^{-2z} and otherwise it outputs $\perp$.

We view (u_1, u_2, e, v) and (u_1', u_2', e', v') as encodings of the *same* ciphertext if $(u_1, u_2, e) = (u_1', u_2', e')$ and $v^2 = (v')^2$. A maliciously constructed, but valid, ciphertext may have $u_1, u_2, e \notin SQ_N$, but this is not a problem as explained in the proof of the proposition [29].

Proposition 3. *The cryptosystem CSL^{cs} is CCA2-secure under the decision Diffie-Hellman assumption.*

5.3 An Efficient Authentication Protocol

Given our implementations of a bounded signature scheme and of a cryptosystem with labels, the authentication protocol boils down to convincing the proxy that the plaintext of a Cramer-Shoup ciphertext is a non-trivial root. The basic idea is to first show that the ciphertext is valid, i.e., that the ciphertext (u_1, u_2, e, v) satisfies $u_1^{2x_1} u_2^{2x_2} (u_1^{y_1} u_2^{y_2})^{2H(m,u_1,u_2,e)} = v^2$, and then show that (u_1, e) is on the form $(g_1^r, h^{2r}\omega)$ for some 2ρth root ω. The latter is equivalent to showing that $(u_1^{2\rho}, e^{2\rho}/g)$ is on the form (g_1^s, h^{2s}), for some $|\rho| \geq 3$. Standard methods for proofs of logarithms over groups of unknown order, e.g. [4], could be used to construct a protocol for the above, but that would give an unnecessarily costly solution, involving an additional independently generated RSA-modulus

and generators. We exploit the relaxed notions of soundness and zero-knowledge to significantly reduce this cost. We use a joint RSA-modulus of the cryptosystem and signature scheme to avoid the need for additional RSA-parameters, i.e., soundness holds even given a signature oracle. Our simulation of an interaction is indistinguishable from a real interaction only over the randomness of the public keys of the cryptosystem, but even given a decryption oracle. We identify which exponents need to be extracted to prove soundness and settle for existence of the other exponents, i.e., the witness is only partly extractable using standard rewinding techniques. Finally, we use special tricks, e.g., we use exponents of the form $(\rho-3)/8$ to avoid proving that $|\rho| \geq 3$ using an interval proof [4].

Below we give an explicit protocol and state its security properties. Let n_r and n_b be additional security parameters such that 2^{-n_r} and 2^{-n_b} are negligible, and $2^{n_b} < \sqrt{N}/2$. Choose some n_p such that $n_p > n_b + n_r$. Denote by G_Q a subgroup of $\mathbb{Z}_P^*$ with generator G of prime order Q for some prime P, where $\log Q = n$.

Protocol 1 (Authentication Protocol).
common input: $cpk = (N, H, g_1, g_2, h, c, d)$, $g \in SQ_N$, $m \in \{0,1\}^*$, $u_1, u_2, e, v \in \mathbb{Z}_N^*$.
private input: $\rho > 2^{n_p}$ such that $\rho = 3 \bmod 8$, ω, and $r \in [0, N2^{n_r}]$ such that $\omega^{2\rho} = g \bmod N$ and $(u_1, u_2, e, v) = \mathsf{Enc}_{cpk}^{cs}(m, \omega, r)$.

Set $\hat{u}_1 = u_1^2$, $\hat{e} = e^2$, $\hat{u}_2 = u_2^2$, $\hat{v} = v^2$, and $f = (cd^{H(m,u_1,u_2,e)})^2$.

1. V_a picks a random $X \in \mathbb{Z}_Q$, hands $Y = G^X$ to the P_a, and proves the knowledge of X using the zero-knowledge proof of knowledge of a logarithm from [10].
2. P_a chooses $b_{P_a} \in [0, 2^{n_b} - 1]$, $R \in \mathbb{Z}_Q$, $s, k \in [0, 2^{n+n_r} - 1]$, $l_r, l_s, l_k \in [0, 2^{n+n_b+2n_r} - 1]$, $l_\rho \in [0, 2^{n_p+n_b+n_r} - 1]$, and $l_t \in [0, 2^{n+2n_b+3n_r} - 1]$ randomly and hands to V_a:
$$C = G^{b_{P_a}} Y^R, \quad (\alpha_1, \alpha_2, \beta) = (g_1^{l_r}, g_2^{l_r}, f^{l_r}), \quad (\delta_1, \delta_\rho) = (g_1^{l_t}, h^{l_t}),$$
$$(\gamma_1, \gamma_\rho, \gamma) = (g_1^{l_s} \hat{u}_1^{2l_\rho}, h^{l_s} \hat{e}^{l_\rho}, g_1^{l_k} g_2^{l_\rho}), \text{ and}$$
$$(\nu_1, \nu_\rho, \nu) = (g_1^s \hat{u}_1^{(\rho-3)/8}, h^s e^{(\rho-3)/8}, g_1^k g_2^{(\rho-3)/8}).$$
3. V_a chooses $b_{V_a} \in [0, 2^{n_b} - 1]$ randomly and hands it to P_a.
4. P_a sets $b = b_{P_a} \oplus b_{V_a}$, and hands $(b_{P_a}, R, a_r, a_s, a_k, a_\rho, a_t)$ to V_a, where
$$a_r = 2rb + l_r \qquad a_\rho = ((\rho-3)/8)b + l_\rho$$
$$(a_s, a_k) = (2sb + l_s, kb + l_k) \qquad a_t = (16s + 4r\rho)b + l_t \ .$$
5. V_a first checks if $C \stackrel{?}{=} G^{b_{P_a}} Y^R$, $b_{P_a} \in [0, 2^{n_b} - 1]$, and $a_\rho \in [0, 2^{n_p+n_b+n_r} - 1]$. Then it sets $\hat{\nu}_1 = \nu_1^2$ and $\hat{\nu}_\rho = \nu_\rho^2$, $\tilde{\nu}_1 = \hat{\nu}_1^8 \hat{u}_1^6$, $\tilde{\nu}_\rho = \hat{\nu}_\rho^8 \hat{e}^3$, and $b = b_{P_a} \oplus b_{V_a}$, and checks that
$$(\hat{u}_1^b \alpha_1, \hat{u}_2^b \alpha_2, \hat{v}^b \beta) \stackrel{?}{=} (g_1^{a_r}, g_2^{a_r}, f^{a_r})$$
$$(\hat{\nu}_1^b \gamma_1, \hat{\nu}_\rho^b \gamma_\rho, \nu^b \gamma) \stackrel{?}{=} (g_1^{a_s} \hat{u}_1^{2a_\rho}, h^{a_s} \hat{e}^{a_\rho}, g_1^{a_k} g_2^{a_\rho})$$
$$\left(\tilde{\nu}_1^b \delta_1, (\tilde{\nu}_\rho / g)^b \delta_\rho\right) \stackrel{?}{=} (g_1^{a_t}, h^{a_t}) \ .$$

Proposition 4. *The authentication protocol (Protocol 1) is overwhelmingly complete,* $(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$*-zero-knowledge for the relation* R_a*, and* $(((cpk, spk), csk, ssk), \mathsf{Sig})$*-sound for the language* L_a*, under the strong RSA assumption and the decision Diffie-Hellman assumption.*

Proposition 4 is proved the full version of this paper [29]. It seems impossible to prove that the protocol is zero-knowledge, since the pair (ν_1, ν_ρ) may be viewed as an El Gamal ciphertext of part of the witness, namely $g^{\frac{1}{8}-\frac{3}{8\rho}}$, using a public key h which is part of the common input. Hence it is conceivable that the auxiliary input contains sufficient information to check if this is the case, without allowing any simulator to produce a correct view. The protocol is only sound as long as no adversary can reduce modulo the order of the group SQ_N.

5.4 Efficiency of the Concrete Scheme

An authentication tag requires 5 exponentiations to compute and 6 exponentiations to verify. Unique prefixes of authentication keys can be tabulated to speed up identification. The authentication protocol requires 7 rounds, since the subprotocol from [10] requires 4 rounds, but this can be reduced to 5 rounds by interlacing the last two rounds of the subprotocol with the first rounds of the main protocol. For practical parameters, $n = 1024$ and $n_r = 30$, $n_b = 50$, and $n_p = 85$ the complexity of the prover and verifier in the authentication protocol corresponds to 19 and 17 exponentiations [29].

Furthermore, the complexity of our scheme can be reduced by using standard techniques such as simultaneous exponentiation and fixed-base exponentiation [27], but for typical applications this is not practical for the sender. A simpler way to reduce complexity of a sender is to pre-compute most exponentiations in an offline phase. It is immediate that this reduces the complexity in the online phase to less than one exponentiation. This approach is feasible even on weak computational devices.

The protocol can also be simplified in an other direction by letting the receiver choose the commitment parameters G and Y to be used by all parties. This reduces the number of rounds to 3, and also decreases the number of exponentiations by 6 for the prover and 7 for the verifier. However, it seems hard to abstract this version in a natural way and keep the description of the generic scheme reasonably modular. Hence, to keep the exposition clear we have chosen not to present the most efficient solution.

6 Conclusion

We remind the reader that performing an exponentiation in a bilinear group used for the provably secure group signature schemes corresponds to roughly 6-8 modular exponentiations for comparable security levels. Thus, our scheme is in fact competitive with these schemes, but under a better understood assumption. The standard group signature schemes, analyzed in the random oracle model,

clearly out-perform our scheme, but the random oracle model is not sound [7]. This is sometimes considered a purely theoretical nuisance, but we think that the recent attacks on hashfunctions, e.g., the collision-attacks on SHA-1 of Wang [34], show that even in practice it is prudent not to model a hashfunction as a random function.

Furthermore, the strong RSA assumption is arguably the most trusted assumption under which a provably secure ordinary signature scheme is known to exist with sufficiently low complexity for practical use, and the decision Diffie-Hellman assumption is the most studied assumption used in practice for public key cryptography.

In this work we have formalized the new notion of group message authentication, which relaxes some of the requirements of group signatures and has applications to anonymous credit cards, and we have constructed a provably secure scheme under the above two assumptions.

References

1. 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/.
2. G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Financial Cryptography '99*, pages 196–211, 1999.
3. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Eurocrypt 2003*, pages 614–629, 2003.
4. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Eurocrypt 2000*, pages 431–444, 2000.
5. X. Boyen and B. Waters. Compact group signatures without random oracles. In *Eurocrypt 2006*, pages 427–444, 2006.
6. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks 2004*, 2005.
7. R. Canetti, O. Goldreich, and S. Halevi. The random oracle model revisited. In *30th ACM STOC*, pages 209–218, 1998.
8. D. Chaum. Designated confirmer signatures. In *Eurocrypt '94*, pages 86–91, 1994.
9. D. Chaum and E. van Heyst. Group signatures. In *Eurocrypt '91*, pages 257–265, 1991.
10. R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography – PKC 2000*, pages 354–372, 2000.
11. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto '98*, pages 13–25, 1998.
12. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM CCS*, pages 46–51, 1999.
13. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Asiacrypt 2002*, pages 125–142, 2002.
14. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, 1998.

15. A. Fiat and A. Shamir. How to prove yourself. practical solutions to identification and signature problems. In *Crypto '86*, pages 186–189, 1986.
16. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Eurocrypt '99*, pages 123–139, 1999.
17. O. Goldreich. A uniform-complexity treatment of encryption and zeroknowledge. *Journal of Cryptology*, 6(1):21–53, 1993.
18. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
19. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
20. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Eurocrypt '96*, pages 143–154, 1996.
21. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Eurocrypt 2004*, 2004.
22. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. Cryptology ePrint Archive, Report 2007/015, 2007. http://eprint.iacr.org/2007/015.
23. J. Kilian and E. Petrank. Identity escrow. In *Crypto '98*, pages 169–185, 1998.
24. S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *Information Security Workshop – ISW '97*, pages 273–281, 1998.
25. S. Laur and S. Pasini. Sas-based group authentication and key agreement protocols. In *Public Key Cryptography – PKC 2008*, pages 197–213, 2008.
26. S. Lucks. A variant of the cramer-shoup cryptosystem for groups of unknown order. In *Asiacrypt '02*, pages 27–45, 2002.
27. A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
28. S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
29. B. Przydatek and D. Wikström. Group message authentication. Cryptology ePrint Archive, http://eprint.iacr.org/, 2010. The full version of this paper.
30. B. Qin, Q. Wu, W. Susilo, and Y. Mu. Group decryption. Cryptology ePrint Archive, Report 2007/017, 2007. http://eprint.iacr.org/2007/017.
31. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Eurocrypt '98*, pages 1–16, 1998.
32. M. Trolin and D. Wikström. Hierarchical group signatures. Cryptology ePrint Archive, Report 2004/311, 2004. http://eprint.iacr.org/.
33. M. Trolin and D. Wikström. Hierarchical group signatures. In *32nd ICALP*, pages 446–458, 2005. (Full version [32]).
34. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In *Crypto 2005*, pages 17–36, 2005.
35. D. Wikström. Designated confirmer signatures revisited. Cryptology ePrint Archive, Report 2006/123, 2006. http://eprint.iacr.org/2006/123.
36. D. Wikström. Designated confirmer signatures revisited. In *4th TCC*, pages 342–361, 2007. Full version [35].