Group Message Authentication*

Bartosz Przydatek
Google Switzerland
przydatek@google.com

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

November 18, 2010

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.

*Work done in part at ETH Zurich. This paper was presented at Conference on Security and Cryptography for Networks 2010.

Contents

1 Introduction

A typical sequence of events in an offline credit card purchase is the following. The 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 [11] 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 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 [4], but these constructions are prohibitively inefficient for practical purposes. There are efficient schemes, e.g., Ateniese *et al.* [1] or Boyen and Waters [6], 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 [7], but the random oracle model is not sound [9]. 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 to solve this problem:

- 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.

1.1 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. [29]), but we think our analysis is of independent interest.

In group signatures in the random oracle model (cf. [7]), the signer encrypts a token and gives a non-interactive Fiat-Shamir [18] 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 [4]) 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 when the group manager plays the role of the verifier, i.e., signatures forged due to the Fiat-Shamir heuristic failing can be detected by the group manager.

1.2 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 [24], multi-group and subgroup signatures [2, 27], or hierarchical group signatures [35]. In contrast to these various extensions of group signatures, group message authentication is actually a relaxation of group signatures, as pointed out above.

Another related primitive is identity escrow [26], 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 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.

Other notions related to group message authentication include designated verifier signatures [23] and designated confirmer signatures [10]. Recently, Kiayias *et al.* [25] and Qin *et al.* [32] proposed a group-oriented notion of cryptosystems.

1.3 Outline of Paper

We first give notation in Section 2. Then we define the notion of group message authentication and capture its security in Section 3. In Section 4 we recall the definitions of standard notions and provide definitions of relaxed notions that are used in the remainder of the paper, and give the new proof of security of the Cramer-Shoup cryptosystem. In Section 5 we present and prove the security of a generic construction of a group message authentication scheme. We then give a detailed description of an efficient instantiation of the generic scheme in Section 6.

2 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

[1] In independent work, Laur and Pasini [28] use the term "group message authentication" in a different context.

time algorithms, PPT the set of probabilistic polynomial time algorithms, and IPPT the set of interactive, probabilistic polynomial time algorithms (sometimes with oracles). Given $P, V \in \text{IPPT}$, 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.

3 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. We introduce the following formal definition.

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 $\mathsf{RKg} \in \text{PPT}$, that on input 1^n outputs a public key pk and a secret key sk.
2. An authentication key generation algorithm $\mathsf{AKg} \in \text{PPT}$, 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 $\mathsf{Aut} \in \text{PPT}$, 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 $\mathsf{Check} \in \text{PT}$, 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 $\pi_a = (P_a, V_a) \in \text{IPPT}^2$, 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 $r \in \{0,1\}^*$: if $\sigma = \mathsf{Aut}_{ak_i,r}(pk, m)$, then $\mathsf{Check}_{sk}(m, \sigma) = i$ and $\Pr[\langle P_a(ak_i, r), V_a\rangle(pk, m, \sigma) = 1]$ is overwhelming.

3.1 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 [4] for group signatures.

3.1.1 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]

Experiment 2 (Anonymity, $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n)$)**.**

$$
\begin{aligned}
(pk, sk) &\leftarrow \mathsf{RKg}(1^n) && \textit{// Receiver key}\\
ak_i &\leftarrow \mathsf{AKg}_{sk}(1^n, i) \text{ for } i \in [1, \ell(n)] && \textit{// Auth. keys}\\
(m, i_0, i_1, \text{state}) &\leftarrow A^{\mathsf{Check}_{sk}(\cdot,\cdot)}(pk, ak_1, \ldots, ak_{\ell(n)}) && \textit{// Choose ids}\\
\sigma &\leftarrow \mathsf{Aut}_{ak_{i_b},r}(pk, m) \ , \quad \text{where } r \in \{0,1\}^* \text{ is random} && \textit{// Challenge}\\
d &\leftarrow \langle P_a(ak_{i_b}, r), A^{\mathsf{Check}_{sk}(\cdot,\cdot)}(\text{state})\rangle(pk, \sigma, m) && \textit{// Guess}
\end{aligned}
$$

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

Definition 3 (Anonymity)**.** A group message authentication scheme GMA is anonymous if for every adversary $A \in \text{IPPT}$: $\left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}-0}_{\mathsf{GMA},A}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{anon}-1}_{\mathsf{GMA},A}(n) = 1\right]\right|$ is negligible.

3.1.2 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

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

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 $P_a^+ \in$ 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.

Experiment 4 (Traceability, $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n)$)**.**

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

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

Definition 5 (Traceability)**.** A group message authentication scheme GMA is traceable if for for every adversary $A \in$ IPPT: $\Pr\left[\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n) = 1\right]$ is negligible.

Definition 6 (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 [24] or identity escrow [26] (cf. Section 1.2), group message authentication it is a *relaxation* of group signatures, aiming at typical applications, and improved efficiency and security.

[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.

4 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.

4.1 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 7 (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 $\mathsf{SKg} \in \text{PPT}$, that on input 1^n outputs a public key spk and a secret key ssk.
2. A signature algorithm $\mathsf{Sig} \in \text{PPT}$, that on input a secret key ssk and a message $m \in [1, \ell(n)]$ outputs a signature s.
3. A verification algorithm $\mathsf{Vf} \in \text{PT}$, 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) [22] is then directly applicable. The existence of an ordinary signature scheme clearly implies the existence of a bounded one.

4.2 Cryptosystems With Labels

Cryptosystems with labels were introduced by Shoup and Gennaro [33]. 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 8 (Public Key Cryptosystem With Labels [33])**.** A *public key cryptosystem with labels* consists of three algorithms $(\mathsf{CKg}, \mathsf{Enc}, \mathsf{Dec})$:

1. A key generation algorithm $\mathsf{CKg} \in \text{PPT}$, that on input 1^n outputs a public key cpk and a secret key csk.

2. An encryption algorithm $\mathsf{Enc} \in \text{PPT}$, 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 $\mathsf{Dec} \in \text{PT}$, 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 9 (CCA2-Security With Labels [33], $\mathsf{Exp}^{\mathsf{cca2}-b}_{\mathsf{CSL},A}(n)$)**.**

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

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

Definition 10 (CCA2-Security)**.** Let CSL denote a public key cryptosystem with labels. We say that the cryptosystem CSL is CCA2-secure if for every adversary $A \in \text{PPT}$: $\left| \Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathsf{CSL},A}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathsf{CSL},A}(n) = 1\right] \right|$ is negligible.

4.3 Cramer-Shoup Cryptosystems In Groups of Composite Orders

It not hard to prove, using the techniques in [13, 15, 8], that the Cramer-Shoup cryptosystem is secure in cyclic groups of composite order, but we use a different approach.

We reduce the security of a Cramer-Shoup cryptosystem over a cyclic group G of composite order to the security of Cramer-Shoup cryptosystems over the subgroups of G. More precisely, we show that a Cramer-Shoup cryptosystem with labels over a cyclic group G can be decomposed into cryptosystems with labels in its cyclic subgroups and that the original cryptosystem is secure if all parts in its decomposition are secure. To the best of our knowledge this approach is novel and of independent interest, since it says something about the structure of the Cramer-Shoup cryptosystem.

Before we start, we point out that although we consider the Cramer-Shoup cryptosystem over a group of unknown order, its security does not rely on the hardness of determining this order. Thus, in the reduction we assume that the factorization of the order is known.

We now define what we mean by a Cramer-Shoup scheme over a group of composite order for an arbitrary collision-free family of hash functions with appropriate range.

Construction 11 (Cramer-Shoup with labels)**.** Let G be a cyclic group of order q with minimal prime divisor q_{min} and let $\mathscr{H}$ be a family of hash functions such that if H is sampled from $\mathscr{H}(1^n)$, then $H : \{0,1\}^* \to [0, q_{min} - 1]$.

Key Generation. Choose generators g_1 and g_2 of G, and $z, x_1, x_2, y_1, y_2 \in [0, q-1]$ randomly, and compute $h = g_1^z$, $c = g_1^{x_1} g_2^{x_2}$, and $d = g_1^{y_1} g_2^{y_2}$. Sample H from $\mathscr{H}$ and output the keys

$$(cpk, csk) = ((H, g_1, g_2, h, c, d), (z, x_1, x_2, y_1, y_2)) \ .$$

Encryption. On input a public key cpk, a label $l \in \{0,1\}^*$ and a message $m \in G$, choose $r \in [0, q-1]$ randomly and output

$$(u_1, u_2, e, v) = (g_1^r, g_2^r, h^r m, c^r d^{rH(l,u_1,u_2,e)}) \ .$$

Decryption. On input a secret key csk, a label $l \in \{0,1\}^*$ and a ciphertext (u_1, u_2, e, v) check if $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^{H(l,u_1,u_2,e)} = v$ holds. If so, output $(l, e u_1^{-z})$, and otherwise output $\perp$.

Our starting point is the following theorem from [13], where we have taken the liberty to add labels as input to the hash function, since it is easy to see from their proof that the theorem remains true also in this case.

Theorem 12. *If $\mathscr{H}$ is collision-free, the decision Diffie-Hellman assumption holds in G, and G has prime order, then Construction 11 is CCA2-secure.*

We use the following obvious lemma to prove our main composition theorem below.

Lemma 13. *If $\mathscr{H}$ is a collision-free family of hash functions and we define a family of hash functions $\bar{\mathscr{H}}$ by sampling H from $\mathscr{H}$ and outputting $H \circ \bar{B}$, where $\bar{B} : \{0,1\}^* \to \{0,1\}^*$ is an efficiently computable bijection, then $\bar{\mathscr{H}}$ is collision-free.*

Theorem 14. *Let G, $\bar{G}$, and $\tilde{G}$ be cyclic groups with efficiently computable isomorphisms $G \to \bar{G} \times \tilde{G}$ and $G \leftarrow \bar{G} \times \tilde{G}$. If Construction 11 is CCA2-secure over $\bar{G}$ and over $\tilde{G}$ for every collision-free families of hash functions $\bar{\mathscr{H}}$ and $\tilde{\mathscr{H}}$ respectively, then it is CCA2-secure over G for every collision-free family of hash functions $\mathscr{H}$.*

The following is an immediate corollary of Theorem 12 and Theorem 14.

Corollary 15. *Let G be a cyclic group. If $\mathscr{H}$ is collision-free and the decision Diffie-Hellman assumption holds in G, then Construction 11 is CCA2-secure.*

Proof of Theorem 14. In the proof we use the fact that every element $x \in G$ can be represented as $(\bar{x}, \tilde{x}) \in \bar{G} \times \tilde{G}$. By assumption the conversion between representations can be computed efficiently: given $x \in G$ we denote by $x \to (\bar{x}, \tilde{x})$ the conversion of x to the representation in $\bar{G} \times \tilde{G}$, and given $(\bar{x}, \tilde{x}) \in \bar{G} \times \tilde{G}$ we denote by $x \leftarrow (\bar{x}, \tilde{x})$ the conversion to the representation in G.

Let $\mathscr{C}$ denote an instantiation of Construction 11 over the group G using a particular collision-free family of hash functions $\mathscr{H}$ and assume that an adversary A breaks the CCA2-security of $\mathscr{C}$. This means that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathscr{C},A}(n)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathscr{C},A}(n)=1\right]\right|$$

is not negligible. Denote by T the experiment $\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathscr{C},A}(n)$ except for the following modification. When the adversary A outputs a challenge label and messages (l, m_0, m_1) the experiment computes $m_0 \to (\bar{m}_0, \tilde{m}_0)$, $m_1 \to (\bar{m}_1, \tilde{m}_1)$, and $m' \leftarrow (\bar{m}_1, \tilde{m}_0)$ and returns a ciphertext (u_1, u_2, e, v) of m'. The triangle inequality implies that one of

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathscr{C},A}(n)=1\right]-\Pr\left[T=1\right]\right|$$

and

$$\left|\Pr\left[T=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathscr{C},A}(n)=1\right]\right|$$

is not negligible. In the remainder of the proof we show that the former case leads to a contradiction. The latter case then follows by symmetry.

Using A we construct an attacker $\bar{A}$ breaking the CCA2-security of an instantiation $\bar{\mathscr{C}}$ of Construction 11 over the group $\bar{G}$ using a collision-free family of hash functions $\bar{\mathscr{H}}$ related to $\mathscr{H}$. To sample a hash function $\bar{H}$ from the family $\bar{\mathscr{H}}$, a random sample H of $\mathscr{H}$ is generated and then the hash function $\bar{H} = H \circ \bar{B}$ is output, where $\bar{B}$ is a bijection. We represent the hash function $\bar{H}$ by a pair $(H, \bar{B})$ and define $\bar{B}$ below. From Lemma 13 we conclude that the resulting family of hash functions $\bar{H}$ is collision-free.

The bijection $\bar{B}$ takes as input $(\bar{l}, \bar{u}_1, \bar{u}_2, \bar{e})$, where $\bar{l} = (l, \tilde{u}_1, \tilde{u}_2, \tilde{e})$, $\bar{u}_1, \bar{u}_2, \bar{e} \in \bar{G}$, l is an arbitrary label, and $\tilde{u}_1, \tilde{u}_2, \tilde{e} \in \tilde{G}$. On such an input $\bar{B}$ outputs (l, u_1, u_2, e), where the values $u_1, u_2, e \in G$ are computed from the corresponding components in $\bar{G} \times \tilde{G}$, i.e., $u_1 \leftarrow (\bar{u}_1, \tilde{u}_1)$, $u_2 \leftarrow (\bar{u}_2, \tilde{u}_2)$ and $e \leftarrow (\bar{e}, \tilde{e})$. We define a corresponding bijection $\tilde{B}$ for the group $\tilde{G}$. It takes as input $(\tilde{l}, \tilde{u}_1, \tilde{u}_2, \tilde{e})$, with $\tilde{l} = (l, \bar{u}_1, \bar{u}_2, \bar{e})$, and outputs (l, u_1, u_2, e), where the values $u_1, u_2, e \in G$ are computed from the corresponding components in $\bar{G} \times \tilde{G}$. Note that $\bar{B}(\bar{l}, \bar{u}_1, \bar{u}_2, \bar{e}) = \tilde{B}(\tilde{l}, \tilde{u}_1, \tilde{u}_2, \tilde{e})$ holds. This implies that

$$\bar{H}(\bar{l}, \bar{u}_1, \bar{u}_2, \bar{e}) = \tilde{H}(\tilde{l}, \tilde{u}_1, \tilde{u}_2, \tilde{e}) = H(l, u_1, u_2, e)$$

for a given hash function H of $\mathscr{H}$, which plays a crucial role in the reduction.

The attacker $\bar{A}$ takes as input a public key $\bar{cpk} = (\bar{H}, \bar{g}_1, \bar{g}_2, \bar{h}, \bar{c}, \bar{d})$, and starts a simulation of the experiment $\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathscr{C},A}(n)$ except for the following modifications.

Key Generation. The attacker $\bar{A}$ computes a public key $cpk = (H, g_1, g_2, h, c, d)$ of $\mathscr{C}$ and gives it as input to A, where cpk is computed as follows.

1. The hash function H is taken without modification from the pair $(H, \bar{B})$ representing $\bar{H}$.

2. Generators $\tilde{g}_1$ and $\tilde{g}_2$ of $\tilde{G}$, and $\tilde{x}_1, \tilde{x}_2, \tilde{y}_1, \tilde{y}_2, \tilde{z} \in [0, \tilde{q}-1]$ are chosen randomly, and $\tilde{c} = \tilde{g}_1^{\tilde{x}_1}\tilde{g}_2^{\tilde{x}_2}$, $\tilde{d} = \tilde{g}_1^{\tilde{y}_1}\tilde{g}_2^{\tilde{y}_2}$, and $\tilde{h} = \tilde{g}_1^{\tilde{z}}$ computed.

3. The public key of $\mathscr{C}$ is defined by: $g_1 \leftarrow (\bar{g}_1, \tilde{g}_1)$, $g_2 \leftarrow (\bar{g}_2, \tilde{g}_2)$, $h \leftarrow (\bar{h}, \tilde{h})$, $c \leftarrow (\bar{c}, \tilde{c})$, and $d \leftarrow (\bar{d}, \tilde{d})$.

Note that the resulting public key is identically distributed to a public key output by the key generator of $\mathscr{C}$. Note also that this defines a related public key $c\tilde{p}k = (\tilde{H}, \tilde{g}_1, \tilde{g}_2, \tilde{h}, \tilde{c}, \tilde{d})$ and secret key $(\tilde{x}_1, \tilde{x}_2, \tilde{y}_1, \tilde{y}_2, \tilde{z})$ of a Cramer-Shoup cryptosystem $\tilde{\mathscr{C}}$ over $\tilde{G}$ with related hash function $\tilde{H} = H \circ \tilde{B}$. We stress that the hash functions of the public keys cpk, $c\bar{p}k$, and $c\tilde{p}k$ are not sampled independently.

Decryption Oracle. The decryption oracle is simulated by $\bar{A}$ using the decryption oracle of the CCA2-experiment it is attacking. Given a query $(l, (u_1, u_2, e, v))$ it computes the reply as follows.

1. The representations $u_1 \to (\bar{u}_1, \tilde{u}_1)$, $u_2 \to (\bar{u}_2, \tilde{u}_2)$, $e \to (\bar{e}, \tilde{e})$, and $v \to (\bar{v}, \tilde{v})$ are computed. This gives two ciphertexts: $(\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$ with label $\bar{l} = (l, \tilde{u}_1, \tilde{u}_2, \tilde{e})$ corresponding to the public key $c\bar{p}k$, and $(\tilde{u}_1, \tilde{u}_2, \tilde{e}, \tilde{v})$ of $\tilde{\mathscr{C}}$ with label $\tilde{l} = (l, \bar{u}_1, \bar{u}_2, \bar{e})$ corresponding to the public key $c\tilde{p}k$.

2. Using $c\tilde{s}k = (\tilde{x}_1, \tilde{x}_2, \tilde{y}_1, \tilde{y}_2, \tilde{z})$, the ciphertext $(\tilde{u}_1, \tilde{u}_2, \tilde{e}, \tilde{v})$ with label $\tilde{l}$ is decrypted to obtain the corresponding plaintext $\tilde{m} \in \tilde{G} \cup \{\perp\}$. The decryption oracle is queried on the ciphertext $(\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$ with label $\bar{l}$, to obtain $\bar{m} \in \bar{G} \cup \{\perp\}$.

3. If $\bar{m} = \perp$ or $\tilde{m} = \perp$, then the simulated decryption oracle replies $\perp$, and otherwise it computes $m \leftarrow (\bar{m}, \tilde{m})$ and replies m.

Note that this is a perfect simulation of the decryption oracle.

The Challenge Ciphertext. When A outputs a tuple (l, m_0, m_1) consisting of a challenge label and messages, $\bar{A}$ proceeds as follows.

1. The representations $m_0 \to (\bar{m}_0, \tilde{m}_0)$ and $m_1 \to (\bar{m}_1, \tilde{m}_1)$ of the challenge plaintexts are computed.

2. The encryption process of $\tilde{m}_0$ is started, but not completed. More precisely, $\tilde{r} \in [0, \tilde{q} - 1]$ is chosen randomly, $(\tilde{u}_1, \tilde{u}_2, \tilde{e}) = (\tilde{g}_1^{\tilde{r}}, \tilde{g}_2^{\tilde{r}}, \tilde{h}^{\tilde{r}} \tilde{m}_0)$ is computed, and a label $\bar{l} = (l, \tilde{u}_1, \tilde{u}_2, \tilde{e})$ defined.

3. The label and messages $(\bar{l}, \bar{m}_0, \bar{m}_1)$ are output and execution is interrupted until the CCA2-experiment returns a challenge ciphertext $(\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$.

4. The challenge ciphertext is then combined with the partial ciphertext $(\tilde{u}_1, \tilde{u}_2, \tilde{e})$ to form a ciphertext (u_1, u_2, e, v). More precisely, $(l, u_1, u_2, e) = \bar{B}(\bar{l}, \bar{u}_1, \bar{u}_2, \bar{e})$, $\tilde{v} = \tilde{c}^{\tilde{r}} \tilde{d}^{\tilde{r} H(l, u_1, u_2, e)}$, and $v \leftarrow (\bar{v}, \tilde{v})$, are computed and (u_1, u_2, e, v) is given to A.

Note that by the definition of the dependent public keys cpk, $c\bar{p}k$ and $c\tilde{p}k$, the resulting ciphertext (u_1, u_2, e, v) is a randomly distributed encryption of m_0 whenever $(\bar{u}_1, \bar{u}_2, \bar{e}, \bar{v})$ is an encryption of $\bar{m}_0$, and otherwise it is an encryption of m', where $m' \leftarrow (\bar{m}_1, \tilde{m}_0)$.

By construction, the random variables $\mathsf{Exp}^{\mathsf{cca2}-0}_{\bar{\mathscr{E}},\bar{A}}(n)$ and $\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathscr{E},A}(n)$ representing experiments are identically distributed, and also $\mathsf{Exp}^{\mathsf{cca2}-1}_{\bar{\mathscr{E}},\bar{A}}(n)$ and T are identically distributed. This implies that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\bar{\mathscr{E}},\bar{A}}(1^n)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\bar{\mathscr{E}},\bar{A}}(1^n)=1\right]\right|$$

is not negligible. Therefore $\bar{\mathscr{E}}$ is not CCA2-secure, which contradicts Theorem 12. □

4.4 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 [21, 31], 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., [16]. In fact, non-interactive computationally sound proofs may be viewed as an instance of this notion.

Definition 16 ((T,O)-Soundness)**.** Let $T \in \mathrm{PPT}$ and let $O \in \mathrm{PT}$ be an oracle. Define a random variable $(t_1,t_2,t_3)=T(1^n)$.

A 2-party protocol $(P,V) \in \mathrm{IPPT} \times \mathrm{IPPT}$ is (T,O)-sound for language L if for every instance chooser $I \in \mathrm{PT}$ and prover $P^* \in \mathrm{PT}$ the following holds: If $(y,z) = I^{O(t_3,\cdot)}(t_1,t_2)$ and $x=(t_1,y)$, then

$$\Pr\left[x \notin L \wedge \langle P^{O(t_3,\cdot)*}(z), V\rangle(x)=1\right]$$

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 22). 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.

4.5 A Relaxed Notion of Computational Zero-Knowledge

Recall that a protocol is zero-knowledge if it can be simulated for *every* instance. Goldreich [20] 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 [38] 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 22).

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 [17]. We use the following experiment to define our notion formally.

Experiment 17 (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 18 ((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:

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

4.6 Sequential Composition Lemma

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. The following lemma establishes such a result.

Experiment 19 (Sequential Composition, $\mathsf{Exp}^{\mu(n)-(T,F,O)-\mathsf{zk}-b}_{\pi,R,I,V^*,M,D}(n)$)**.**

$$\begin{aligned}
(t_1, t_2) &\leftarrow T(1^n) \quad z'_0 \leftarrow t_1 \\
(y_j, z_j) &\leftarrow I^{O(t_2,\cdot)}(1^n, z'_{j-1}) \\
(x_j, w_j) &\leftarrow F(t_1, y_j) \\
z'_j &\leftarrow \begin{cases} \langle P(w_j), V^{*O(t_2,\cdot)}(z_j)\rangle(x_j) & \text{if b=0} \\ M^{O(t_2,\cdot)}(z_j, x_j) & \text{if b=1} \end{cases} \\
d &\leftarrow D^{O(t_2,\cdot)}(z'_{\mu(n)})
\end{aligned}$$

Return 0 if $R(x_j, w_j) = 0$ and d otherwise.

Lemma 20 (Sequential Composition)**.** *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.*

Then for every verifier $V^* \in$ PPT and every constant $c > 0$ the simulator $M \in$ PPT guaranteed to exist by the definition of (T, F, O)-zero-knowledge satisfies that for every instance chooser $I \in$ PPT and every distinguisher $D \in$ PPT:

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

Proof. Consider a verifier V^* and a constant $c > 0$, the corresponding simulator M guaranteed by the zero-knowledge property, an instance chooser I, and a distinguisher D, and write H^b for the algorithm that simulates the experiment $\mathsf{Exp}^{\mu(n)-(T,F)-\mathsf{zk}-b}_{\pi,R,I,V^*,M,D}(n)$.

Denote by H^0_l the machine H^0, except that if $j \leq l$, then the simulator behaves as H^1. Thus, H^0_0 is identical to H^0 and H^0_μ is identical to H^1. Denote by I_l the instance chooser that simulates H^0_{l-1} until (y_l, z_l) is output in the simulation. Denote by D_l the distinguisher that takes z'_{l+1} as input and computes z_μ exactly as is done in the simulation H^0, by simulating the honest prover. It follows that H^0_{l-1} is identically distributed to $\mathsf{Exp}^{(T,F)-\mathsf{zk}-0}_{\pi,R,I_l,V^*,M,D_l}(n)$. and H^0_l is identically distributed to $\mathsf{Exp}^{(T,F)-\mathsf{zk}-1}_{\pi,R,I_l,V^*,M,D_l}(n)$.

Suppose that the lemma is false. Then there is a verifier V^*, an instance chooser I, and a distinguisher D such that $|\Pr\left[H^0 = 1\right] - \Pr\left[H^1 = 1\right]| \geq \mu(n)n^{-c}$. A hybrid argument shows that there exists a fixed l such that $|\Pr\left[H^0_{l-1} = 1\right] - \Pr\left[H^0_l = 1\right]| \geq n^{-c}$, but this contradicts the (T, F, O)-zero-knowledge property and the lemma must be true. □

Remark 21. Although the proof of the lemma is straightforward, there are seemingly reasonable definitions of (T, F, O)-zero-knowledge for which it is false. For example, it

is false if the simulator M is given the secret part t_2 of the sample from T in cleartext. Thus, the lemma may be viewed as a sanity check of our definition.

5 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 $\sigma = \mathsf{Enc}_{cpk}(m, ak_i)$.

5.1 Details of the Construction

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 22 (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 of the form $((pk, m, \sigma), (ak_i, r))$ such that $\mathsf{Vf}_{spk}(i, ak_i) = 1$ and $\sigma = \mathsf{Enc}_{cpk}(m, ak_i, r)$, and let L_a be the language corresponding to R_a, i.e., $L_a = \{x : \exists y \text{ s.t. } (x, y) \in R_a\}$ is the honestly encrypted valid tags. Let $F_{\mathsf{Enc}} \in \mathrm{PT}$ take as input a tuple $(cpk, (s, m, i, s', r))$.

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

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 to be used by SKg_2 resp. Sig_{ssk}. Note that requiring explicit randomness as input ensures that the signature public key spk and the signature ak_i are correctly formed. 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 the relation R_a, and $(((cpk, spk), csk, ssk), \mathsf{Sig})$-sound for the language L_a.

Proposition 23. *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.*

5.2 Proof of Security (Proposition 23)

The functionality of GMA can be easily verified. To prove the security of GMA, we prove its anonymity and traceability.

5.2.1 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.

Step 1. Given adversary A, we construct an instance chooser I and a malicious verifier V_a^*. Then we describe a distinguisher D that will later be considered in the context of the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge experiment.

Denote by I the algorithm that given (cpk, csk) chooses $s \in \{0,1\}^*$ randomly and computes $(spk, ssk) = \mathsf{SKg}_{2,s}(spk_1)$, where spk_1 is the first part of cpk, and simulates the experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n)$ up to and including the challenge step. The machine I expects access to a $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle, where csk is the secret key corresponding to the public key cpk. Given such an oracle it can internally simulate a $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle as follows.

The jth query (m_j, σ_j) is simply forwarded to the $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle and it is verified that the reply s_j satisfies $\mathsf{Vf}_{spk}(i, s_j) = 1$ for some index $i \in [1, \ell(n)]$. The $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle returns i in that case, and $\perp$ otherwise. When the challenge step in the simulation is reached, I outputs $((s, m, i_b, s', r), z)$, where z is the state of I concatenated with the list of queries it made and $s' \in \{0,1\}^*$. Recall that by the definition of F_{Enc} this results in an instance on the form

$$\big((pk, m, \sigma), (ak_{i_b}, r)\big) \ ,$$

where

$$\begin{aligned} pk &= (cpk, spk) \ , \\ (spk, ssk) &= \mathsf{SKg}_{2,s}(spk_1) \ , \\ ak_{i_b} &= \mathsf{Sig}_{ssk,s'}(i_b) \ , \quad \text{and} \\ \sigma &= \mathsf{Enc}_{cpk}(m, ak_{i_b}, r) \ . \end{aligned}$$

Denote by V_a^* the verifier that takes z as private input and (pk, σ, m) as common input, and continues the simulation except that instead of running the prover P_a it expects to interact externally with this prover. The simulation continues until P_a outputs its last message at which point V_a^* outputs the concatenation z' of z, (pk, σ, m), its state, and the list of queries it made. The verifier V_a^* simulates the $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle exactly as I.

Denote by M the simulator guaranteed by the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge property of π_a to exist for a constant $c > 0$ (to be determined below) and denote by $T^b(c)$ the simulation of the experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n)$ where the interaction between A and the prover P_a in the guessing step is replaced by invoking the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge simulator M. More precisely, z' is defined as the output of $M((pk, m, \sigma), z)$. Recall that M has access to a decryption oracle and that its output contains the list of queries it asked as a postfix.

Denote by D the distinguisher that takes z' as input and completes the execution of one of the two experiments (with or without simulation) except that it outputs 0 if the challenge pair (m, σ) was handed to the $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle by V_a^* or the simulator M respectively. Note that this can be determined by D from z', since it contains lists of all queries asked.

Then $\mathsf{Exp}^{(\mathsf{CKg},F_{\mathsf{Enc}},\mathsf{Dec})-\mathsf{zk}-0}_{\pi,R,I,V^*,M,D}(n)$ is identically distributed to $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n)$, $T^b(c)$ is identically distributed to $\mathsf{Exp}^{(\mathsf{CKg},F_{\mathsf{Enc}},\mathsf{Dec})-\mathsf{zk}-1}_{\pi,R,I,V^*,M,D}(n)$, and the claim below follows immediately from the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge of π_a.

Claim 24. *For every constant $c > 0$ there exists $n_0 \in \mathbb{N}$, such that for every $n > n_0$:*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GMA},A}(n) = 1\right] - \Pr\left[T^b(c) = 1\right]\right| < n^{-c} \ .$$

Step 2. Suppose that the GMA scheme is not anonymous, i.e., there is an adversary $A \in \mathrm{PT}$ such that $\left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}-0}_{\mathsf{GMA},A}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{anon}-1}_{\mathsf{GMA},A}(n) = 1\right]\right|$ is not negligible. Then from the claim above we know that for a suitable choice of $c > 0$, $\left|\Pr\left[T^0(c) = 1\right] - \Pr\left[T^1(c) = 1\right]\right|$ is not negligible as well.

We construct an adversary A' that breaks the CCA2-security of CSL. It accepts cpk as input and computes $(spk, ssk) = \mathsf{SKg}_2(spk_1)$, where spk_1 is the first part of cpk. Then it simulates the experiment $T^0(c)$ to A except that whenever a ciphertext must be decrypted it is instead handed to the $\mathsf{Dec}_{sk}(\cdot,\cdot)$-oracle of the CCA2-experiment from which A' received cpk. At some point A outputs $(m, i_0, i_1, \text{state})$. Then A' forwards (m, ak_{i_0}, ak_{i_1}) to its CCA2-experiment, and receives a challenge $\sigma = \mathsf{Enc}_{cpk}(m, ak_{i_b})$ which it hands to A in the simulated experiment. The result of the simulated experiment is finally given as output by A'.

By construction, $T^b(c)$ is identically distributed to $\mathsf{Exp}^{\mathsf{cca2}-b}_{\mathsf{CSL},A}(n)$, which implies that $\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathsf{CSL},A}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathsf{CSL},A}(n) = 1\right]\right|$ is not negligible. This contradicts the CCA2-security of CSL and we conclude that our group message authentication scheme is anonymous.

5.2.2 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 five steps:

1. We pick a fixed index i of an authentication key randomly and argue that with good probability this is the index of the attacked authentication key ak_i.

2. We replace each invokation of P_a using ak_i by a simulation. Due to the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge of π_a and the sequential composition lemma (Lemma 20) this changes the view of the adversary by an arbitrarily small amount.

3. We replace the authentication tags computed by P_a^+ using ak_i 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.

4. 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. Thus, assuming that it is never valid changes the view of the adversary only negligibly.

5. We show that the adversary in the modified traceability experiment can be used to break the CMA security of the bounded signature scheme. Note that in the modified experiment, we may postpone the computation of the 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 needed to extract it from the ciphertext.

For completeness we recall the definition of security against chosen message attacks (CMA) for signature schemes adapted to bounded signature schemes.

Experiment 25 (CMA-Security, $\mathsf{Exp}^{\mathsf{cma}}_{\mathsf{BSS},A}(n)$)**.**

$$\begin{aligned}(spk, ssk) &\leftarrow \mathsf{SKg}(1^n)\\ (m, s) &\leftarrow A^{\mathsf{Sig}_{ssk}(\cdot)}(\mathsf{guess}, spk)\end{aligned}$$

If $\mathsf{Vf}_{spk}(m, s) = 1$ and A did not ask for a signature of m return 1, else return 0.

Definition 26 (CMA-Security)**.** A signature scheme BSS is secure against chosen message attacks (CMA) if for every adversary $A \in$ PPT: $\Pr\left[\mathsf{Exp}^{\mathsf{cma}}_{\mathsf{BSS},A}(n) = 1\right]$ is negligible.

Step 1. Consider an execution of Experiment 4 with output 1. If in this execution $\mathsf{Check}_{sk}(m, \sigma) \in [1, \ell(n)] \setminus \mathcal{C}$ and P^+_a has never output σ, then due to our random (and independent) choice of i we have $\mathsf{Check}_{sk}(m, \sigma) = i$ with probability at least $1/\ell(n)$. Thus, without loss of generality, we assume from now on that $\mathsf{Check}_{sk}(m, \sigma) = \{i, \perp\}$ in any execution of Experiment 4 with output 1 (formally, we define a modified experiment with this requirement).

Step 2. Denote the original experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n)$ by T_0. Denote the ith copy of P^+_a, which runs on input (pk, ak_i) by $P^+_{a,i}$, and recall that each such copy only invokes P_a sequentially. We replace all invocations of P_a made by $P^+_{a,i}$ by simulations and apply Lemma 20 as follows.

Denote by I_i the algorithm that given $z'_0 = cpk$ chooses $s \in \{0,1\}^*$ randomly and computes $(spk, ssk) = \mathsf{SKg}_{2,s}(spk_1)$, where spk_1 is the first part of cpk, and simulates the experiment except that the simulation is interrupted before the first invocation of P_a made by $P^+_{a,i}$. As in the proof of anonymity, the machine I_i expects access to a $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle, where csk is the secret key corresponding to the public key cpk. Given such an oracle it internally simulates a $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle as follows. The jth query (m_j, σ_j) is simply forwarded to the $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle and it is verified that the result s_j satisfies $\mathsf{Vf}_{spk}(i, s_j) = 1$ for some index $i \in [1, \ell(n)]$. When the simulation is interrupted I_i outputs $((s, m_1, i, a_j, r_1), z_1)$, where $a_j \in \{0,1\}^*$ is random. Recall from the definition of F_{Enc} that this gives an instance

$$\big((pk, m_1, \sigma_1), (ak_i, r_1)\big) \ ,$$

where

$$\begin{aligned}pk &= (cpk, spk) \ ,\\ (spk, ssk) &= \mathsf{SKg}_{2,s}(spk_1) \ ,\\ ak_i &= \mathsf{Sig}_{ssk,a_j}(i) \ ,\\ \sigma_1 &= \mathsf{Enc}_{cpk}(m_1, ak_i, r_1) \ ,\end{aligned}$$

and z_1 is the concatenation of z'_0, (pk, m_1, σ_1), its state, and the the list of queries it made to its decryption oracle. This defines how I_i chooses the instance on which $P^+_{a,i}$ first invokes P_a.

Below we define how I_i takes z'_{j-1} as input and outputs $((s, m_j, ak_i, r_j), z_j)$, but before we complete the description of how this is done we must define the malicious verifier. Denote by V_i^* the verifier that on secret input z_j and common input (pk, m_j, σ_j) (this is output by the instance compiler F_{Enc}), continues the simulation except that $P_{a,i}^+$ instead of running P_a, expects to interact externally with this prover. The verifier has access to a $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle and simulates its $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle exactly as I_i does. The simulation continues until the current invocation of P_a of $P_{a,i}^+$ outputs its last message at which point V_i^* outputs the concatenation z'_j of z_j, (pk, m_j, σ_j), its state, and the the list of queries it made to its decryption oracle.

We now define I_i such that it on input z'_{j-1} continues the simulation. The simulation is interrupted before the jth invocation of P_a by $P_{a,i}^+$ and $((s, m_j, i, a_j, r_j), z_j)$ is output, where z_j is the state of I_i.

Denote by M_i the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge simulator for V_i^* and some constant $c > 0$ (to be determined later) of the protocol π_a. The distinguisher D is defined to output the result of the experiment, i.e., a single bit.

We then define $T_1(c)$ as the simulation, where the output z'_j of V_i^* with secret input z_{j-1} from an interaction with P_a with secret input (ak_i, r_j) on common input (pk, m_j, σ_j) are replaced by the output of $M_i((pk, m_j, \sigma_j), z_j)$ for all j. Then the first step in the outline corresponds to the following claim.

Claim 27. *For every constant $c > 0$: $|\Pr[T_0 = 1] - \Pr[T_1(c) = 1]| < \mu(n)n^{-c}$, where $\mu(n)$ is the number of invokations of the authentication protocol that are simulated.*

Proof. The execution performed by the instance chooser, the verifier, and the simulator above corresponds directly to the sequential composition of the zero-knowledge experiment formalized in Lemma 20. More precisely, T_b is identically distributed to $\mathsf{Exp}^{\mu(n)-(\mathsf{CKg},F_{\mathsf{Enc}},\mathsf{Dec})-\mathsf{zk}-b}_{\pi,R,I_i,V_i^*,M_i,D}(n)$, where this experiment is defined in Section 4.6. The claim now follows from the $(\mathsf{CKg}, F_{\mathsf{Enc}}, \mathsf{Dec})$-zero-knowledge of π_a and the sequential composition lemma. □

Step 3. We now consider a simulation where the authentication tags computed by $P_{a,i}^+$ are replaced by encryptions of 0. We begin with a conceptual modification of $T_1(c)$. When $P_{a,i}^+$ computes an authentication tag σ as a result of a request (Aut, m), the triple (m, σ, i) is stored. Furthermore, when the simulated $\mathsf{Check}_{sk}(\cdot,\cdot)$-oracle is queried on a pair (m, σ), either by A or by one of the simulators, the oracle first checks if (m, σ, i) is stored. If so, it returns i, and otherwise it computes the result as before. We stress that this does not change the distribution of the output of the simulation.

Consider now the following actual modification. We denote by $T_2(c)$ the simulation $T_1(c)$ except that when $P_{a,i}^+$ computes an authentication tag σ as a result of a request (Aut, m), it is defined by $\sigma = \mathsf{Enc}_{cpk}(m, 0, r)$. Thus, the authentication tags computed by the experiment no longer contain any signature ak_i of i of the bounded signature scheme BSS.

Claim 28. $|\Pr[T_1(c) = 1] - \Pr[T_2(c) = 1]|$ *is negligible.*

Proof. This follows from the CCA2-security of CSL. If the claim is false then define hybrid simulations $T_1^{(l)}$ for $l = 1, \ldots, Q_A$, where Q_A is the total number of requests A makes to the modified $P_{a,i}^+$ (it is modified since each invocation of the protocol π_a is simulated). We let $T_1^{(l)}$ be identical to $T_1(c)$ except that for the jth request with $j < l$ it behaves as $T_2(c)$. It follows that $T_1^{(0)}$ is identical to $T_1(c)$ and $T_1^{(Q_A)}$ is identical to $T_2(c)$. A hybrid argument then implies that $|\Pr\left[T_1^{(j-1)} = 1\right] - \Pr\left[T_1^{(j)} = 1\right]|$ is not negligible for some fixed $1 \leq j \leq Q_A$, since Q_A is polynomially bounded.

We now define A' as the CCA2-adversary that given input cpk computes $(spk, ssk) = \mathsf{SKg}_2(spk_1)$, where spk_1 is the first part of cpk, and simulates $T_1^{(j-1)}$ until the jth request (Aut, m_j) to $P_{a,i}^+$. Whenever a ciphertext must be decrypted, it is instead forwarded to the external $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle in the CCA2-experiment. When the simulation is interrupted, it outputs $(m_j, 0, ak_i)$ to the CCA2-experiment and uses the reply σ_j in the continued simulation.

Note that by the construction of $T_1(c)$, the adversary A' never queries the $\mathsf{Dec}_{csk}(\cdot,\cdot)$-oracle on (m_j, σ_j). This means that $\mathsf{Exp}^{\mathsf{cca2}-b}_{\mathsf{CSL},A'}(n)$ and $T_1^{(j-1+b)}$ are identically distributed. This implies that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-0}_{\mathsf{CSL},A'}(n) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cca2}-1}_{\mathsf{CSL},A'}(n) = 1\right]\right|$$

is not negligible and the CCA2-security of CSL is broken. This is a contradiction and the claim must be true. □

Step 4. We change the experiment such that the adversary no longer can win by convincing the honest verifier V_a that an invalid authentication tag is valid. Denote by $T_3(c)$ the simulation $T_2(c)$ except that it outputs 0 if $\mathsf{Vf}_{spk}(i, \mathsf{Dec}_{csk}(m, \sigma)) = 0$ for all $i \in [1, \ell(n)]$ regardless of if the honest verifier V_a is convinced of the validity of σ or not. Then the third step of the outline corresponds to the following claim.

Claim 29. $|\Pr[T_2(c) = 1] - \Pr[T_3(c) = 1]|$ *is negligible.*

Proof. This follows from $(((cpk, spk), csk, ssk), \mathsf{Sig})$-soundness. Denote by I the instance chooser that on input $((cpk, spk), csk)$ simulates $T_2(c)$ except that before the last step of experiment $T_2(c)$ it halts and outputs $((m, \sigma), z)$, where (m, σ) is the instance chosen by A and z is the complete state of I. Denote by P_a^* the prover that takes z as auxiliary input and (m, σ) as common input and completes the simulation of the experiment except that instead of simulating the honest verifier V_a it simply forwards messages to and from the external verifier of the $(((cpk, spk), csk, ssk), \mathsf{Sig})$-soundness experiment. Both I and P_a^* forward all requests for authentication keys to the $\mathsf{Sig}_{ssk}(\cdot)$-oracle of the soundness experiment. The $(((cpk, spk), csk, ssk), \mathsf{Sig})$-soundness of π_a implies that the probability that P_a^* convinces the honest verifier when $\mathsf{Check}_{sk}(m, \sigma) = \perp$ is negligible. The union bound then gives our claim. □

Step 5. We begin with a conceptual modification. We assume that the authentication key ak_i is not generated at the beginning of the simulation $T_3(c)$, but only when

requested by A. Note that this does not change the distribution of $T_3(c)$, since an authentication key no longer influence the distribution of $T_3(c)$ until requested by A.

Assume now that there is an adversary $A \in \mathrm{PT}$ such that $\Pr\left[\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GMA},A}(n) = 1\right]$ is not negligible. It follows from the claims above that $\Pr\left[T_3(c) = 1\right]$ is not negligible for a suitable choice of $c > 0$. Let us recall our modifications: (1) we only consider an execution of Experiment 4 to be successful if $\mathsf{Check}_{sk}(m, \sigma) = \{i, \perp\}$, (2) all invocations of $P^+_{a,i}$ of π_a are simulated, (3) $P^+_{a,i}$ never uses its authentication key ak_i to compute an authentication tag, (4), we only consider an execution of (the already modified) Experiment 4 to be successful if $\mathsf{Check}_{csk}(m, \sigma) \in \{1, \ldots, \ell(n)\} \setminus \mathcal{C}$, and (5) ak_i is only computed if ak_i is requested by A.

We construct an adversary A' against the CMA-security of the bounded signature scheme. It accepts spk as input, computes $(cpk, csk) = \mathsf{CKg}_2(spk_1)$, where spk_1 is the first part of spk, and then simulates $T_3(c)$ except that when A corrupts i and requests its authentication key ak_i, it computes this by querying its $\mathsf{Sig}_{ssk}(\cdot)$-oracle on i. Finally, when A outputs (m, σ) it computes $s = \mathsf{Dec}_{csk}(m, \sigma)$ and if possible it identifies i such that $\mathsf{Vf}_{spk}(i, s) = 1$ and outputs (i, s). It follows that the advantage of A' in the CMA-experiment is not negligible. This contradicts the CMA-security of the bounded signature scheme BSS and our group message authentication scheme is traceable. ■

6 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 assumption. 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 $\log q = \Omega(\log N)$, 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. Appendix B for formal definitions).

6.1 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 30 (Bounded Signature Scheme $\mathsf{BSS}^{\mathrm{rsa}}$)**.**

Key Generation. On input 1^n the algorithm $\mathsf{SKg}^{\mathrm{rsa}}_1$, 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 $\mathsf{SKg}^{\mathrm{rsa}}_2$ 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 $\mathsf{SKg}_2^{\text{rsa}}$ 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 $\mathsf{Sig}^{\text{rsa}}$ 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 Vf^{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 $\mathsf{BSS}^{\text{rsa}}$ directly 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[6], 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 [14, 19], is given in Appendix A.

Proposition 31. *For every polynomial $\ell(n)$, the scheme $\mathsf{BSS}^{\text{rsa}}$ is a CMA-secure bounded signature scheme under the strong RSA assumption.*

6.2 A Cramer-Shoup Cryptosystem

The original cryptosystem of Cramer and Shoup [13] 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 [13] using these parameters. Denote by n_r an additional security parameter such that 2^{-n_r} is negligible.

Construction 32 (Cramer-Shoup Cryptosystem CSL^{cs} in SQ_N)**.**

Key Generation. On input 1^n, the first key generator CKg_1^{cs} runs exactly as $\mathsf{SKg}^{\text{rsa}}(1^n)$, 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 $(u_1, u_2, e, v) = (g_1^r, g_2^r, h^{2r} m, c^r d^{rH(l,u_1,u_2,e)})$.

[6] If the reader is worried about large gaps between primes in the definition of the ρ_i, note that we can define ρ_i to be the ith "random" prime with the above properties derived from a pseudorandom source based on a seed output as part of the public key. The important property is that we can compute ρ_i deterministically from i.

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 below.

Proposition 33. *The cryptosystem CSL^{cs} is CCA2-secure under the decision Diffie-Hellman assumption.*

Proof. We must show that the security of our particular cryptosystem CSL^{cs} follows from Theorem 14.

First note that choosing the exponents in the key generation and encryption algorithm in $\mathbb{Z}_{|SQ_N|}$ instead of in $[0, N2^{n_r}]$ only changes the distributions negligibly. Recall that u_1 and u_2 are squared before doing the validity check. Let us write $\bar{u}_1$ and $\bar{u}_2$ for the squared values. Thus, if the validity check holds, then $\bar{u}_1$, $\bar{u}_1$, and v are squares. It is tempting to conclude that the scheme is secure from Theorem 14 and the structure of the group SQ_N, but unfortunately $e \in \mathbb{Z}_N$ may not be a square.

However, we may write $\mathbb{Z}_N^* = SQ_N \times V$, for a group V of order 4 in which each element has order at most 2, and given the factorization of N, the corresponding isomorphisms are easily computed. We now apply a labeling trick similar to the one used in the proof of Theorem 14. Any element $m \in \mathbb{Z}_N^*$ can be represented as a pair (m_{SQ}, m_V), where $m_{SQ} \in SQ_N$ and $m_V \in V$. We simply view m as one particular way to concatenate the label m_V with the message m_{SQ}. Note that a collision-free hash function may take any representation of the pair (m_{SQ}, m_V) as input, so the factorization is not needed to satisfy the functional properties of a cryptosystem with labels if we accept the slightly modified definition we use in this paper. Thus, we may conclude that our variation of the Cramer-Shoup cryptosystem with labels CSL^{cs} is CCA2-secure. □

6.3 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 the inequality $u_1^{2x_1} u_2^{2x_2} (u_1^{y_1} u_2^{y_2})^{2H(m,u_1,u_2,e)} = v^2$, 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. [5], 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 [5].

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 34 (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 [12].

2. P_a chooses $b_{P_a} \in [0, 2^{n_b} - 1]$, $R \in \mathbb{Z}_Q$, $s, k \in [0, 2^{n+n_r} - 1]$, $l_r, l_s, l_k \in \left[0, 2^{n+n_b+2n_r} - 1\right]$, $l_\rho \in [0, 2^{n_p+n_b+n_r} - 1]$, and $l_t \in \left[0, 2^{n+2n_b+3n_r} - 1\right]$ randomly and hands to V_a:

$$\begin{aligned} C &= G^{b_{P_a}} Y^R \quad , \\ (\alpha_1, \alpha_2, \beta) &= (g_1^{l_r}, g_2^{l_r}, f^{l_r}) \quad , \\ (\gamma_1, \gamma_\rho, \gamma) &= (g_1^{l_s} \hat{u}_1^{2l_\rho}, h^{l_s} \hat{e}^{l_\rho}, g_1^{l_k} g_2^{l_\rho}) \quad , \\ (\nu_1, \nu_\rho, \nu) &= (g_1^{s} \hat{u}_1^{(\rho-3)/8}, h^{s} e^{(\rho-3)/8}, g_1^{k} g_2^{(\rho-3)/8}) \quad , \text{ and} \\ (\delta_1, \delta_\rho) &= (g_1^{l_t}, h^{l_t}) \ . \end{aligned}$$

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 the verifier, where

$$\begin{aligned} a_r &= 2rb + l_r & a_\rho &= ((\rho-3)/8)b + l_\rho \\ (a_s, a_k) &= (2sb + l_s, kb + l_k) & a_t &= (16s + 4r\rho)b + l_t \ . \end{aligned}$$

5. V_a first checks that $C \stackrel{?}{=} G^{b_{P_a}} Y^R$, $b_{P_a} \in [0, 2^{n_b} - 1]$, 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

$$\begin{aligned} (\hat{u}_1^b \alpha_1, \hat{u}_2^b \alpha_2, \hat{v}^b \beta) &\stackrel{?}{=} (g_1^{a_r}, g_2^{a_r}, f^{a_r}) \quad , \\ (\hat{\nu}_1^b \gamma_1, \hat{\nu}_\rho^b \gamma_\rho, \nu^b \gamma) &\stackrel{?}{=} (g_1^{a_s} \hat{u}_1^{2a_\rho}, h^{a_s} \hat{e}^{a_\rho}, g_1^{a_k} g_2^{a_\rho}) \ , \quad \text{and} \\ \left(\tilde{\nu}_1^b \delta_1, (\tilde{\nu}_\rho / g)^b \delta_\rho\right) &\stackrel{?}{=} (g_1^{a_t}, h^{a_t}) \ . \end{aligned}$$

Proposition 35. *Protocol 34 is overwhelmingly complete and* $(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$*-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 DDH assumption.*

Proposition 35 is proved below. 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.

6.4 Proof of Proposition 35

Overwhelming completeness follows by inspection. We concentrate on proving the soundness and zero-knowledge properties of the protocol.

$(((cpk, spk), csk, ssk), \mathsf{Sig})$-Soundness. Suppose that the proposition is false, i.e., the authentication protocol is not sound in the sense of $(((cpk, spk), csk, ssk), \mathsf{Sig})$-soundness. Then there exists an instance chooser I and a prover P_a^* such that when the tuple $((cpk, spk), csk)$ consists of randomly chosen keys and

$$((m, (u_1, u_2, v, e)), \zeta) = I^{\mathsf{Sig}_{ssk}(\cdot)}((cpk, spk), csk) \ ,$$

then

$$\Pr\left[(pk, m, (u_1, u_2, v, e)) \notin R_a \wedge \langle P_a^*(\zeta), V_a\rangle(pk, m, (u_1, u_2, v, e)) = 1\right] \geq n^{-\psi}$$

with $pk = (cpk, spk)$, for some constant ψ and n in an infinite index set $\mathcal{N}$. Denote by

$$\Lambda = ((pk, m, (u_1, u_2, v, e)), \Gamma, C, \alpha_1, \alpha_2, \beta, \nu_1, \nu_\rho, \nu, \gamma_1, \gamma_\rho, \gamma, \delta_1, \delta_\rho)$$

the view of the verifier after the second step of the protocol, where Γ is the view of the subprotocol that is executed in the first step, and $(C, \alpha_1, \alpha_2, \beta, \nu_1, \nu_\rho, \nu, \gamma_1, \gamma_\rho, \gamma, \delta_1, \delta_\rho)$ is the message of the prover in the second step of the main protocol. Denote by E_{acc} the event that the chosen instance does not belong to the relation and the verifier accepts, i.e., the event

$$(pk, m, (u_1, u_2, v, e)) \notin R_a \wedge \langle P_a^*(\zeta), V_a\rangle(pk, m, (u_1, u_2, v, e)) = 1 \ .$$

Then define S_{good} as the set of (csk, Λ) such that

$$\Pr\left[E_{acc} \mid (csk, \Lambda) \in S_{good}\right] \geq n^{-\psi}/2 \ ,$$

and denote the event $(csk, \Lambda) \in S_{good}$ by E_{good}. Then the following claim follows by a simple averaging argument.

Claim 36. $\Pr\left[E_{good}\right] \geq n^{-\psi}/2$.

Denote by Ext the extraction algorithm, that given $((cpk, spk), csk, ssk)$ simply runs the soundness experiment with P_a^* to get a view $(\Lambda, b_{V_a}, (b_{P_a}, R, a_r, a_s, a_k, a_\rho, a_t))$. Then Ext rewinds to the challenge step and completes the experiment again using an independently chosen challenge b'_{V_a}, to produce another view $(\Lambda, b'_{V_a}, (b'_{P_a}, R', a'_r, a'_s, a'_k, a'_\rho, a'_t))$ with identical beginning Λ. Finally, Ext outputs these views. Denote by E_{fork} the event that Ext outputs two accepting transcripts.

Claim 37. $\Pr[E_{fork} \wedge b \neq b'] \geq n^{-3\psi}/8 - \epsilon(n)$ *for some negligible function* $\epsilon(n)$ *under the DDH assumption.*

Proof. The quantity $\Pr\left[E_{fork} \wedge b_{V_a} \neq b'_{V_a}\right]$ can be bounded by

$$\Pr[E_{acc} \mid E_{good}]^2 \Pr[E_{good}] - \Pr\left[b_{V_a} = b'_{V_a}\right] \geq n^{-3\psi}/8 - \epsilon(n)$$

for some negligible function $\epsilon(n)$ using independence and the union bound, since clearly $\Pr\left[b_{V_a} = b'_{V_a}\right]$ is negligible. Since $b \neq b'$ whenever $b_{V_a} \neq b'_{V_a}$ and $b_{P_a} = b'_{P_a}$, it suffices to show that $\Pr\left[E_{fork} \wedge b_{P_a} \neq b'_{P_a}\right]$ is negligible.

Suppose it is not. Then denote by A' the discrete logarithm algorithm that accepts (G, Y) as input and simulates Ext, except that it uses (G, Y) instead of generating these parameters, and it invokes the perfect zero-knowledge simulator instead of executing the proof of knowledge of the logarithm $\log_G Y$. By assumption, Ext outputs accepting transcripts such that $b_{P_a} \neq b'_{P_a}$ with probability that is not negligible. This implies that $G^{b_{P_a}} Y^R = C = G^{b'_{P_a}} Y^{R'}$, from which the discrete logarithm $\log_G Y = (b'_{P_a} - b_{P_a})/(R - R')$ can be extracted and output by A'. The claim follows, since this contradicts the DDH-assumption. □

Recall that $|SQ_N|$ is of the form $p'q'$, where $p = 2p' + 1$ and $q = 2q' + 1$ are the two safe prime divisors in the modulus N. Consider now two transcripts corresponding to the event $E_{acc} \wedge b \neq b'$. From the choice of the challenge length n_b we know that $|b - b'| < p', q'$, so $b - b'$ is invertible modulo $|SQ_N|$. Define $\rho^* = (b - b')^{-1}(a_\rho - a'_\rho)$ and $\rho = 8\rho^* + 3$.

Claim 38. $\Pr\left[E_{fork} \wedge b \neq b' \wedge (b - b') \mid (a_\rho - a'_\rho) \wedge \gcd(\rho, p'q') = 1\right] \geq n^{-3\psi}/8 - \epsilon'(n)$, *where* $\epsilon'(n)$ *is a negligible function, under the DDH assumption and the strong RSA assumption.*

Proof. Consider two accepting transcripts as above. We have $\nu^{b-b'} = g_1^{a_k - a'_k} g_2^{a_\rho - a'_\rho}$. If $b - b'$ does not divide both $a_k - a'_k$ and $a_\rho - a'_\rho$, then we conclude from Lemma 41 in Appendix B that the strong RSA assumption is broken. The reader may object that we need to implement the signature oracle, but this is easily even given (N, g_1, g_2) as input. Simply define $g = g_1^{2\prod_{i \in [1, \ell(n)]} \rho_i}$, and note this allows us to compute a signature $g^{1/(2\rho_m)}$ of m as $g = g_1^{\prod_{i \in [1, \ell(n)] \setminus \{m\}} \rho_i}$. Thus, Lemma 41 is applicable and the event $E_{fork} \wedge b \neq b' \wedge (b - b') \nmid (a_\rho - a'_\rho)$ occurs with negligible probability.

Then note that if ρ is not invertible modulo $p'q'$, then neither is $(b - b')\rho$. Consider the machine A' that given (N, g) uses these values to compute (cpk, csk) and execute

Ext. If Ext outputs two accepting transcripts with $b \neq b'$, then A' outputs $(b-b')\rho = 8(a_\rho - a'_\rho) + 3(b-b')$.

If the claim is false, then A' outputs an integer k with $\gcd(k, p'q') \neq 1$ with probability that is not not negligible. If $p'q'$ divides k, then $g_1^{k+1} = g_1$ and we have a non-trivial root of g_1. If only one of the primes, say p', divides k, then $g_1^k = 1 \bmod p$ holds, and $\gcd(g_1^k - 1, N) = p$ allows us to factor N. Therefore, the strong RSA assumption is broken, so the claim must hold. □

We now consider an output of Ext conditioned on the event

$$E_{fork} \wedge b \neq b' \wedge (b-b') | (a_\rho - a'_\rho) \wedge \gcd(\rho, N) = 1 \ .$$

In this case, we can define $\rho^* = (a_\rho - a'_\rho)/(b-b')$ by *integer division* and set $\rho = 8\rho^* + 3$. By construction $a_\rho, a'_\rho \in [0, 2^{n_p+n_b+n_r} - 1]$ and $n_p > n_b + n_r$, i.e., $\rho^* < \rho_i \rho_j$ for all $1 \leq i, j \leq \ell$. In other words, the values ρ^* and ρ not only exists, but we extract them explicitly.

We have $(\hat{u}_1^{b-b'}, \hat{u}_2^{b-b'}, \hat{v}^{b-b'}) = (g_1^{a_r - a'_r}, g_2^{a_r - a'_r}, f^{a_r - a'_r})$, Since $\hat{u}_1, \hat{u}_2$ and $\hat{v}$ are squares, we can define $r = (b-b')^{-1}(a_r - a'_r)$, and conclude that

$$(\hat{u}_1, \hat{u}_2, \hat{v}) = (g_1^r, g_2^r, f^r) \ . \tag{1}$$

This implies that the input ciphertext is valid and does not decrypt to $\perp$.

By construction, $\hat{e}$, $\hat{\nu}_1$, $\hat{\nu}_\rho$, $\tilde{\nu}_1$ and $\tilde{\nu}_\rho$ are squares, which also implies that $(\hat{\nu}_\rho / g)$ is a square. We have $(\hat{\nu}_1^{b-b'}, \hat{\nu}_\rho^{b-b'}) = (g_1^{a_s - a'_s} \hat{u}_1^{2(a_\rho - a'_\rho)}, h^{a_s - a'_s} \hat{e}^{a_\rho - a'_\rho})$. Therefore, we can define $s = (b-b')^{-1}(a_s - a'_s)$ and conclude that

$$(\hat{\nu}_1, \hat{\nu}_\rho) = (g_1^s \hat{u}_1^{2\rho^*}, h^s \hat{e}^{\rho^*}) \ , \tag{2}$$

where we defined $\rho^* = (a_\rho - a'_\rho)/(b-b')$ above. Finally, we know that $(\tilde{\nu}_1^{b-b'}, (\tilde{\nu}_\rho / g)^{b-b'}) = (g_1^{a_t - a'_t}, h^{a_t - a'_t})$. We define $t = (b-b')^{-1}(a_t - a'_t)$ and conclude that

$$(\tilde{\nu}_1, \tilde{\nu}_\rho / g) = (g_1^t, h^t) \ . \tag{3}$$

Recall that we defined $\rho = 8\rho^* + 3$. Then if we combine Equations (1)-(3) we have

$$\begin{aligned}
\big(\hat{u}_1^{2\rho}, \hat{e}^\rho / g\big) &= \big(\hat{u}_1^{16\rho^*+6}, \hat{e}^{8\rho^*+3} / g\big) = \big((\hat{u}_1^{2\rho^*})^8 \hat{u}_1^6, (\hat{e}^{\rho^*})^8 \hat{e}^3 / g\big) \\
&= \big((\hat{\nu}_1 g_1^{-s})^8 \hat{u}_1^6, (\hat{\nu}_\rho h^{-s})^8 \hat{e}^3 / g\big) = (g_1^{-8s} \tilde{\nu}_1, h^{-8s} \tilde{\nu}_\rho / g) \\
&= (g_1^{-8s} g_1^t, h^{-8s} h^t) = (g_1^{r''}, h^{r''}) \ ,
\end{aligned}$$

for some r'', or differently written: $(\hat{u}_1^{2\rho}, \hat{e}^\rho) = (g_1^{r''}, h^{r''} g)$. We define $\omega = e u_1^{-2z}$, which can be computed using *csk*, and note that $\omega^{2\rho} = g$. Recall that $\rho_i \rho_j > |\rho| \geq 3$ for all $1 \leq i, j \leq \ell$.

To summarize, given the instance chooser I and the prover P_a^* assumed to exist at the beginning of the proof, we can construct a PPT algorithm B that given (N, g) and

$$(g^{1/(2\rho_1)}, 2\rho_1), \ldots, (g^{1/(2\rho_\ell)}, 2\rho_\ell)$$

as input, outputs with probability that is not negligible, a pair $(\omega, 2\rho)$, where $\omega^{2\rho} = g$, $\rho = 3 \bmod 8$, and $\rho_i\rho_j > |\rho| \geq 3$ for all $1 \leq i, j \leq \ell$. By assumption we also have $\rho \neq \pm\rho_i$. Without loss of generality we may assume that $\rho \neq \pm 1$ and $\gcd(\rho, 2\prod_{i=1}^{\ell}\rho_i) = 1$, since we can always strip off any ρ_i-factors of ρ and modify the non-trivial root ω correspondingly.

We conclude the proof of soundness by showing how this contradicts the strong RSA assumption. Given an instance (N, g') of the strong RSA problem we run B on input (N, g) and the list of non-trivial roots $(g^{1/(2\rho_1)}, 2\rho_1), \ldots, (g^{1/(2\rho_\ell)}, 2\rho_\ell)$, where $g = (g')^{2\prod_{i=1}^{\ell}\rho_i}$. When B outputs $(\omega, 2\rho)$ such that $\omega^{2\rho} = g$ and $\gcd(\rho, 2\prod_{i=1}^{\ell}\rho_i) = 1$, we compute integers a and b such that $a \cdot 2\prod_{i=1}^{\ell}\rho_i + b\rho = 1$, and then output $(\omega^{2a}(g')^b, \rho)$ which is a non-trivial root of g'. More precisely,

$$(\omega^{2a}(g')^b)^\rho = g^a(g')^{b\rho} = (g')^{a2\prod_{i=1}^{\ell}\rho_i + b\rho} = g' \ .$$

This concludes the proof of soundness, since it contradicts the strong RSA assumption.

$(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$-Zero-Knowledge. The simulator is defined as follows. First it starts the execution of the verifier. If the simulator accepts the proof of knowledge of the logarithm $\log_G Y$ by the verifier, then the simulator invokes the knowledge extractor of this proof of knowledge. Let $X \in \mathbb{Z}_Q$ be the extracted value such that $Y = G^X$. Strictly speaking we can for every fixed $c > 0$ turn the *expected* polynomial time extractor into a *strict* polynomial time extractor with advantage bounded by $n^{-c}/2$ by bounding the running time of the expected time extractor. This follows immediately from Markov's inequality.

Given X such that $Y = G^X$, the simulator continues the simulation except that it chooses $s, k \in [0, 2^{n+n_r} - 1]$ randomly and defines

$$(\nu_1, \nu_\rho, \nu) = (g_1^s, h^s, g_1^k) \ .$$

Then it chooses random $b \in [0, 2^{n_b} - 1]$, $a_r, a_s, a_k \in \left[0, 2^{n+n_b+2n_r}\right]$, $a_\rho \in [0, 2^{n_p+n_b+n_r}]$, and $a_t \in \left[0, 2^{n+2n_b+3n_r}\right]$, and defines

$$\begin{aligned}
(\alpha_1, \alpha_2, \beta) &= (g_1^{a_r}/\hat{u}_1^b, g_2^{a_r}/\hat{u}_2^b, f^{a_r}/\hat{v}^b) \\
(\gamma_1, \gamma_\rho, \gamma) &= (g_1^{a_s}\hat{u}_1^{a_\rho}/\hat{\nu}_1^b, h^{a_s}\hat{e}^{a_\rho}/\hat{\nu}_\rho^b, h^{a_k}g_1^{a_\rho}/\nu^b) \\
(\delta_1, \delta_\rho) &= (g_1^{a_t}/\tilde{\nu}_1^b, h^{a_t}/(\tilde{\nu}_\rho/g)^b) \ .
\end{aligned}$$

Finally, it defines $b'_{P_a} = b \oplus b_{V_a}$ and $R' = (b_{P_a} + XR - b'_{P_a})/X$ and continues the simulation using the above values except that (b_{P_a}, R) is replaced by (b'_{P_a}, R').

A standard honest verifier zero-knowledge argument gives that if we could use correctly distributed (ν_1, ν_ρ), then the resulting simulated view would be statistically close to the view of the verifier in the protocol. It remains to show that it is infeasible to distinguish the distribution of our simulated (ν_1, ν_ρ) from the correct distribution, but this follows readily from the CCA2-security of the cryptosystem CSL^{cs}.

Suppose that the protocol is not $(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$-zero-knowledge. Then there exists a verifier $V_a^* \in$ PT a constant $c > 0$, an instance chooser $I \in$ PPT, and a

distinguisher $D \in \mathrm{PPT}$ such that

$$\left|\Pr\left[\mathsf{Exp}^{(\mathsf{CKg}^{cs},F_{\mathsf{Enc}^{cs}},\mathsf{Dec}^{cs})-\mathsf{zk}-0}_{\pi_a,R_a,I,V_a^*,M,D}(n)=1\right]-\Pr\left[\mathsf{Exp}^{(\mathsf{CKg}^{cs},F_{\mathsf{Enc}^{cs}},\mathsf{Dec}^{cs})-\mathsf{zk}-1}_{\pi_a,R_a,I,V_a^*,M,D}(n)=1\right]\right| \geq \delta(n)$$

for some $\delta(n) \geq n^{-c}$, where M is the simulator described above.

Denote by A' an adversary to the CCA2-security of CSL^{cs} defined as follows. It accepts as input a public key cpk and starts a simulation of the $(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$-zero-knowledge experiment $\mathsf{Exp}^{(\mathsf{CKg}^{cs},F_{\mathsf{Enc}^{cs}},\mathsf{Dec}^{cs})-\mathsf{zk}-1}_{\pi_a,R_a,I,V_a^*,M,D}(n)$ using the verifier V_a^*, the instance chooser I, the distinguisher D, and the simulator M described above. When the instance compiler outputs $((pk, m, (u_1, u_2, e, v)), (\omega, r))$, A' hands $(\omega^{(\rho-3)/8}, 1)$ to the CCA2-experiment and waits for a Cramer-Shoup ciphertext (u_1^*, u_2^*, e^*, v^*) in return, where ρ is the prime ρ_i such that $\omega^{2\rho_i} = g$. Then it replaces (ν_1, ν_ρ) by $((u_1^*)^2, e^*)$, continues the simulation and outputs its result. Note that by definition of $F_{\mathsf{Enc}^{cs}}$ we necessarily have $\omega^{2\rho} = g$ for some such prime, and this also implies that $\omega \in SQ_N$. Note that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{cca2}-b}_{\mathsf{CSL}^{cs},A'}(n)=1\right]-\Pr\left[\mathsf{Exp}^{(\mathsf{CKg}^{cs},F_{\mathsf{Enc}^{cs}},\mathsf{Dec}^{cs})-\mathsf{zk}-b}_{\pi_a,R_a,I,V_a^*,M,D}(n)=1\right]\right| < \frac{3}{4}n^{-c} \ ,$$

since the difference is bounded by the sum of $n^{-c}/2$ (due to the knowledge error of the extractor) and some negligible quantity (due to the statistically small error in our honest verifier simulation).

Thus, for a suitable choice of $c > 0$, A' breaks the CCA2-security of CSL^{cs}. This is a contradiction and the protocol must be $(\mathsf{CKg}^{cs}, F_{\mathsf{Enc}^{cs}}, \mathsf{Dec}^{cs})$-zero-knowledge.

6.5 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 [12] 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 (see Appendix C for how we estimate this).

Furthermore, the complexity of our scheme can be reduced by using standard techniques such as simultaneous exponentiation and fixed-base exponentiation [30], 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.

7 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 [9]. 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 [36], 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*, volume 1648 of *LNCS*, pages 196–211. Springer Verlag, 1999.

[3] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature scheme. In *Advances in Cryptology – Eurocrypt '97*, volume 1233 of *LNCS*, pages 480–494. Springer Verlag, 1997.

[4] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology – Eurocrypt 2003*, volume 2656 of *LNCS*, pages 614–629. Springer Verlag, 2003.

[5] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *LNCS*, pages 431–444. Springer Verlag, 2000.

[6] X. Boyen and B. Waters. Compact group signatures without random oracles. In *Advances in Cryptology – Eurocrypt 2006*, volume 4004 of *LNCS*, pages 427–444. Springer Verlag, 2006.

[7] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks 2004*, volume 3352 of *LNCS*. Springer Verlag, 2005.

[8] J. Camensisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – Crypto 2003*, volume 2729 of *LNCS*, pages 126–144. Springer Verlag, 2003.

[9] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model revisited. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 209–218. ACM Press, 1998.

[10] D. Chaum. Designated confirmer signatures. In *Advances in Cryptology – Eurocrypt '94*, volume 950 of *LNCS*, pages 86–91. Springer Verlag, 1994.

[11] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – Eurocrypt '91*, volume 547 of *LNCS*, pages 257–265. Springer Verlag, 1991.

[12] R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography – PKC 2000*, volume 1751, pages 354–372. Springer Verlag, 2000.

[13] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – Crypto '98*, volume 1462 of *LNCS*, pages 13–25. Springer Verlag, 1998.

[14] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51. ACM Press, 1999.

[15] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology – Eurocrypt '02*, volume 2332 of *LNCS*, pages 45–64. Springer Verlag, 2002.

[16] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *LNCS*, pages 125–142. Springer Verlag, 2002.

[17] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 409–418. ACM Press, 1998.

[18] A. Fiat and A. Shamir. How to prove yourself. practical solutions to identification and signature problems. In *Advances in Cryptology – Crypto '86*, volume 263 of *LNCS*, pages 186–189. Springer Verlag, 1986.

[19] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *LNCS*, pages 123–139. Springer Verlag, 1999.

[20] O. Goldreich. A uniform-complexity treatment of encryption and zeroknowledge. *Journal of Cryptology*, 6(1):21–53, 1993.

[21] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[22] 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, 1988.

[23] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology – Eurocrypt '96*, volume 1070 of *LNCS*, pages 143–154. Springer Verlag, 1996.

[24] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology – Eurocrypt 2004*, volume 3027 of *LNCS*. Springer Verlag, 2004.

[25] A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. Cryptology ePrint Archive, Report 2007/015, 2007. `http://eprint.iacr.org/2007/015`.

[26] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology – Crypto '98*, pages 169–185, 1998.

[27] S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *Information Security Workshop – ISW '97*, volume 1396 of *LNCS*, pages 273–281. Springer Verlag, 1998.

[28] S. Laur and S. Pasini. Sas-based group authentication and key agreement protocols. In *Public Key Cryptography – PKC 2008*, volume 4939 of *LNCS*, pages 197–213. Springer Verlag, 2008.

[29] S. Lucks. A variant of the cramer-shoup cryptosystem for groups of unknown order. In *Advances in Cryptology – Asiacrypt '02*, pages 27–45, 2002.

[30] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[31] S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[32] B. Qin, Q. Wu, W. Susilo, and Y. Mu. Group decryption. Cryptology ePrint Archive, Report 2007/017, 2007. `http://eprint.iacr.org/2007/017`.

[33] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *LNCS*, pages 1–16. Springer Verlag, 1998.

[34] M. Trolin and D. Wikström. Hierarchical group signatures. Cryptology ePrint Archive, Report 2004/311, 2004. `http://eprint.iacr.org/`.

[35] M. Trolin and D. Wikström. Hierarchical group signatures. In *32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *LNCS*, pages 446–458. Springer Verlag, 2005. (Full version [34]).

[36] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In *Advances in Cryptology – Crypto 2005*, volume 3621 of *LNCS*, pages 17–36. Springer Verlag, 2005.

[37] D. Wikström. Designated confirmer signatures revisited. Cryptology ePrint Archive, Report 2006/123, 2006. `http://eprint.iacr.org/2006/123`.

[38] D. Wikström. Designated confirmer signatures revisited. In *4th Theory of Cryptography Conference (TCC)*, volume 4392 of *LNCS*, pages 342–361. Springer Verlag, 2007. Full version [37].

A Proof of Proposition 31

Suppose that there is a forger A with a success probability in the CMA-experiment that is not negligible. Denote by $m_1, \ldots, m_{t(n)}$ the queries A hands to the signature oracle $\mathsf{Sig}^{\text{rsa}}_{ssk}(\cdot)$, and let (m, ω) be the output of A. Denote by E the event that A outputs a pair (m, ω) such that $\mathsf{Vf}^{\text{rsa}}_{spk}(m, \omega) = 1$ and $\forall i : m \neq m_i$. By assumption $\Pr[E]$ is not negligible. Since $\ell(n)$ is polynomial, an averaging argument implies that there exists a fixed $\bar{m} \in [1, \ell(n)]$ such that $\Pr[E \wedge m = \bar{m}]$ is not negligible.

Denote by A' the adversary defined as follows. It chooses $\bar{m} \in [1, \ell(n)]$ randomly and defines $S = [1, \ell(n)] \backslash \{\bar{m}\}$ as the set of messages distinct from $\bar{m}$. The adversary A' takes a random strong RSA problem instance (N, h) as input and computes $g = h^{2\prod_{m \in S} \rho_m} \bmod N$. Note that g is identically distributed to h, since $2\prod_{m \in S} \rho_m$ is relatively prime to the order of SQ_N. Then it simulates the CMA-experiment to A. For all queries $m_l \neq \bar{m}$ the $\mathsf{Sig}^{\text{rsa}}_{ssk}(\cdot)$-oracle is easily simulated without using the secret key ssk, since $g^{1/(2\rho_{m_l})} = h^{\prod_{m \in S \backslash \{m_l\}} \rho_m} \bmod N$. If A asks for a signature of $\bar{m}$, then A' outputs $\perp$. Otherwise A eventually outputs (m, ω). If $m \neq \bar{m}$, then A' outputs $\perp$. If $m = \bar{m}$ then A' computes a and b such that $1 = a2\prod_{m \in S} \rho_m + b\rho_{\bar{m}}$, and outputs $(\omega^{2a} h^b, \rho_{\bar{m}})$. The integers a and b exist since $2\prod_{m \in S} \rho_m$ and $\rho_{\bar{m}}$ are relatively prime by construction. Note that if $\omega^{2\rho_{\bar{m}}} = g \bmod N$, then we have

$$(\omega^{2a} h^b)^{\rho_{\bar{m}}} = h^{a2\prod_{m \in S} \rho_m + b\rho_{\bar{m}}} = h \bmod N \ .$$

If $\omega^{-2\rho_{\bar{m}}} = g \bmod N$, then we change the sign of $\rho_{\bar{m}}$ and the above holds. We conclude that A' breaks the strong RSA-assumption, since conditioned on the event that the output of A' is not $\perp$, the event that $m = \bar{m}$ is independent of the event E and happens with probability $1/\ell(n)$, where $\ell(n)$ is polynomially bounded. ■

B Assumptions

B.1 Strong RSA-Assumption

In this paper we use random primes p and q such that both $(p-1)/2$ and $(q-1)/2$ are primes. Such primes are sometimes called safe primes. It is not known if there are infinitely many such primes, but in practice a random prime p has this property with probability roughly $1/\log p$. Formally we need an assumption.

Definition 39 (Safe Prime Assumption)**.** There exists an integer c such that a random n-bit prime p is safe with probability at least n^{-c}.

The strong RSA-assumption, introduced in [3], says that it is infeasible to compute any non-trivial root of a random element in SQ_N, where N is an RSA-modulus.

Definition 40 (Strong RSA-Assumption)**.** Let p and q be randomly chosen $n/2$-bit safe primes, define $N = pq$, and let $g \in SQ_N$ be randomly chosen. Then for every $A \in$ PPT: $\Pr\left[A(N,g) = (g', e) \wedge e \neq \pm 1 \wedge (g')^e = g \bmod N\right]$ is negligible in n.

Lemma 41. *Let p and q be randomly chosen $n/2$-bit safe primes, define $N = pq$, and let $g, h \in SQ_N$ be randomly chosen. Then for every $A \in$ PPT:*

$$\Pr\left[A(N,g) = (g', e_0, e_1, e_2) \wedge (g')^{e_0} = g^{e_1} h^{e_2} \bmod N \wedge e_0 \neq 0 \wedge (e_0 \nmid e_1 \vee e_0 \nmid e_2)\right]$$

is negligible in n.

A proof of this lemma is given, e.g., in Damgård and Fujisaki [16].

B.2 Decision Diffie-Hellman Assumption

The decision Diffie-Hellman problem can be considered over many different groups. Below we define the assumption over a large prime order subgroup of a modular multiplicative group.

Definition 42 (Decision Diffie-Hellman Assumption)**.** Let N be an integer and G_q a subgroup of $\mathbb{Z}_N^*$ of prime order q with $\log q = O(n)$. Let g be a generator of this group and let $a, b, c \in \mathbb{Z}_q$ be randomly chosen. Then for every adversary $A \in$ PPT the following quantity is negligible in n:

$$\left|\Pr\left[A(N,q,g,g^a,g^b,g^{ab}) = 1\right] - \Pr\left[A(N,q,g,g^a,g^b,g^c) = 1\right]\right| \ .$$

C Program Used To Estimate the Complexity

```
; (load "efficiency.scm")
; (complexity 1024 30 50 81)

(define (complexity secp secpR secpC secpP)

  ; Cost of an exponentiation as a function of number of bits in exponent
  (define (expos bits-in-exp)
    (/ bits-in-exp secp))

  ; Complexity of prover in terms of exponentiations
  (define (prover-complexity)
```

```
    (+ 1                                          ; Compute Y
       4                                          ; Cramer et al PoK of exponent
       1                                          ; Compute C
       (expos secpC)
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 2 alphas and beta
       (* 3 (expos (+ secp secpR)))               ; 3 nus
       (* 3 (expos secpP))
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 3 gammas
       (* 3 (expos (+ secpP secpC secpR)))
       (* 2 (expos (+ secp (* 2 secpC) (* 3 secpR)))) ; 2 deltas
       ))

  ; Complexity of verifier in terms of exponentiations
  (define (verifier-complexity)
    (+ 6                                          ; Cramer et al PoK of exponent
       1                                          ; Check opening of C
       (expos secpC)
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 2 alphas and beta
       (* 3 (expos secpC))
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 3 nus and gammas
       (* 3 (expos (+ secpP secpC secpR)))
       (* 3 (expos secpC))
       (* 2 (expos (+ secp (* 2 secpC) (* 3 secpR)))) ; 2 nus and 2 deltas
       (* 2 (expos secpC))
       ))

  ; Print the result nicely
  (define (print-complexities)
    (newline)
    (newline)
    (display "Complexity of transfer protocol:")
    (newline)
    (display "Prover ")
    (display (round (prover-complexity)))
    (display " exponentiations.")
    (newline)
    (display "Verifier ")
    (display (round (verifier-complexity)))
    (display " exponentiations.")
    (newline)
    (newline))

  (print-complexities)
); (load "efficiency.scm")
; (complexity 1024 30 50 81)

(define (complexity secp secpR secpC secpP)

  ; Cost of an exponentiation as a function of number of bits in exponent
  (define (expos bits-in-exp)
    (/ bits-in-exp secp))

  ; Complexity of prover in terms of exponentiations
  (define (prover-complexity)
    (+ 1                                          ; Compute Y
       4                                          ; Cramer et al PoK of exponent
       1                                          ; Compute C
       (expos secpC)
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 2 alphas and beta
       (* 3 (expos (+ secp secpR)))               ; 3 nus
       (* 3 (expos secpP))
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 3 gammas
       (* 3 (expos (+ secpP secpC secpR)))
       (* 2 (expos (+ secp (* 2 secpC) (* 3 secpR)))) ; 2 deltas
       ))

  ; Complexity of verifier in terms of exponentiations
  (define (verifier-complexity)
    (+ 6                                          ; Cramer et al PoK of exponent
       1                                          ; Check opening of C
       (expos secpC)
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 2 alphas and beta
       (* 3 (expos secpC))
       (* 3 (expos (+ secp secpC (* 2 secpR))))   ; 3 nus and gammas
       (* 3 (expos (+ secpP secpC secpR)))
       (* 3 (expos secpC))
       (* 2 (expos (+ secp (* 2 secpC) (* 3 secpR)))) ; 2 nus and 2 deltas
       (* 2 (expos secpC))
       ))

  ; Print the result nicely
  (define (print-complexities)
    (newline)
    (newline)
    (display "Complexity of transfer protocol:")
    (newline)
```

```
    (display "Prover ")
    (display (round (prover-complexity)))
    (display " exponentiations.")
    (newline)
    (display "Verifier ")
    (display (round (verifier-complexity)))
    (display " exponentiations.")
    (newline)
    (newline))

  (print-complexities)
)
```