Short Group Signatures

Dan Boneh[1,⋆], Xavier Boyen[2], and Hovav Shacham[3]

[1] Stanford University, dabo@cs.stanford.edu
[2] Voltage Security, xb@boyen.org
[3] Stanford University, hovav@cs.stanford.edu

Abstract. We construct a short group signature scheme. Signatures in our scheme are approximately the size of a standard RSA signature with the same security. Security of our group signature is based on the Strong Diffie-Hellman assumption and a new assumption in bilinear groups called the Decision Linear assumption. We prove security of our system, in the random oracle model, using a variant of the security definition for group signatures recently given by Bellare, Micciancio, and Warinschi.

1 Introduction

Group signatures, introduced by Chaum and van Heyst [14], provide anonymity for signers. Any member of the group can sign messages, but the resulting signature keeps the identity of the signer secret. In some systems there is a third party that can trace the signature, or undo its anonymity, using a special trapdoor. Some systems support revocation [12, 4, 29, 15] where group membership can be selectively disabled without affecting the signing ability of unrevoked members. Currently, the most efficient constructions [2, 12, 4] are based on the Strong-RSA assumption introduced by Baric and Pfitzman [5].

In the last two years a number of projects have emerged that require the properties of group signatures. The first is the Trusted Computing effort [28] that, among other things, enables a desktop PC to prove to a remote party what software it is running via a process called *attestation*. Group signatures are needed for privacy-preserving attestation [17, Sect. 2.2]. Perhaps an even more relevant project is the Vehicle Safety Communications (VSC) system from the Department of Transportation in the U.S. [18]. The system embeds short-range transmitters in cars; these transmit status information to other cars in close proximity. For example, if a car executes an emergency brake, all cars in its vicinity are alerted. To prevent message spoofing, all messages in the system are signed by a tamper-resistant chip in each car. (MACs were ruled out for this many-to-many broadcast environment.) Since VSC messages reveal the speed and location of the car, there is a strong desire to provide user privacy so that the full identity of the car sending each message is kept private. Using group signatures, where the group is the set of all cars, we can maintain privacy while

⋆ Supported by NSF and the Packard Foundation.

still being able to revoke a signing key in case the tamper resistant chip in a car is compromised. Due to the number of cars transmitting concurrently there is a hard requirement that the length of each signature be under 250 bytes.

The two examples above illustrate the need for efficient group signatures. The second example also shows the need for short group signatures. Currently, group signatures based on Strong-RSA are too long for this application.

We construct short group signatures whose length is under 200 bytes that offer approximately the same level of security as a regular RSA signature of the same length. The security of our scheme is based on the Strong Diffie-Hellman (SDH) assumption [8] in groups with a bilinear map. We also introduce a new assumption in bilinear groups, called the Linear assumption, described in Sect. 3.2. The SDH assumption was recently used by Boneh and Boyen to construct short signatures without random oracles [8]. A closely related assumption was used by Mitsunari et al. [22] to construct a traitor-tracing system. The SDH assumption has similar properties to the Strong-RSA assumption. We use these properties to construct our short group signature scheme. Our results suggest that systems based on SDH are simpler and shorter than their Strong-RSA counterparts.

Our system is based on a new Zero-Knowledge Proof of Knowledge (ZKPK) of the solution to an SDH problem. We convert this ZKPK to a group signature via the Fiat-Shamir heuristic [16] and prove security in the random oracle model. Our security proofs use a variant of the security model for group signatures proposed by Bellare, Micciancio, and Warinschi [6].

Recently, Camenisch and Lysyanskaya [13] proposed a signature scheme with efficient protocols for obtaining and proving knowledge of signatures on committed values. They then derive a group signature scheme using these protocols as building blocks. Their signature scheme is based on the LRSW assumption [21], which, like SDH, is a discrete-logarithm-type assumption. Their methodology can also be applied to the SDH assumption, yielding a different SDH-based group signature.

The SDH group signature we construct is very flexible and we show how to add a number of features to it. In Sect. 7 we show how to apply the revocation mechanism of Camenisch and Lysyanskaya [12]. In Sect. 8 we briefly sketch how to add strong exculpability.

2 Bilinear Groups

We first review a few concepts related to bilinear maps. We follow the notation of Boneh, Lynn, and Shacham [9]:

1. G_1 and G_2 are two (multiplicative) cyclic groups of prime order p;
2. g_1 is a generator of G_1 and g_2 is a generator of G_2;
3. ψ is a computable isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$; and
4. e is a computable map $e : G_1 \times G_2 \to G_T$ with the following properties:
   - Bilinearity: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
   - Non-degeneracy: $e(g_1, g_2) \neq 1$.

Throughout the paper, we consider bilinear maps $e : G_1 \times G_2 \to G_T$ where all groups G_1, G_2, G_T are multiplicative and of prime order p. One could set $G_1 = G_2$. However, we allow for the more general case where $G_1 \neq G_2$ so that our constructions can make use of certain families of non-supersingular elliptic curves defined by Miyaji et al. [23]. In this paper we only use the fact that G_1 can be of size approximately 2^{170}, elements in G_1 are 171-bit strings, and that discrete log in G_1 is as hard as discrete log in $\mathbb{Z}_q^*$ where q is 1020 bits. We will use these groups to construct short group signatures. We note that the bilinear groups of Rubin and Silverberg [25] can also be used.

We say that two groups (G_1, G_2) as above are a bilinear group pair if the group action in G_1 and G_2, the map ψ, and the bilinear map e are all efficiently computable.

The isomorphism ψ is only needed for the proofs of security. To keep the discussion general, we simply assume that ψ exists and is efficiently computable. (When G_1, G_2 are subgroups of the group of points of an elliptic curve $E/\mathbb{F}_q$, the trace map on the curve can be used as this isomorphism. In this case, $G_1 \subseteq E(\mathbb{F}_q)$ and $G_2 \subseteq E(\mathbb{F}_{q^r})$.)

3 Complexity Assumptions

3.1 The Strong Diffie-Hellman Assumption

Let G_1, G_2 be cyclic groups of prime order p, where possibly $G_1 = G_2$. Let g_1 be a generator of G_1 and g_2 a generator of G_2. Consider the following problem:

q-Strong Diffie-Hellman Problem. The q-SDH problem in (G_1, G_2) is defined as follows: given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)})$ as input, output a pair $(g_1^{1/(\gamma+x)}, x)$ where $x \in \mathbb{Z}_p^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH in (G_1, G_2) if

$$\Pr\Big[\mathcal{A}(g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma^q)}) = (g_1^{\frac{1}{\gamma+x}}, x)\Big] \geq \epsilon \ ,$$

where the probability is over the random choice of γ in $\mathbb{Z}_p^*$ and the random bits of $\mathcal{A}$.

Definition 1. *We say that the (q, t, ϵ)-SDH assumption holds in (G_1, G_2) if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in (G_1, G_2).*

Occasionally we drop the t and ϵ and refer to the q-SDH assumption rather than the (q, t, ϵ)-SDH assumption. The q-SDH assumption was recently used by Boneh and Boyen [8] to construct a short signature scheme without random oracles. To gain confidence in the assumption they prove that it holds in generic groups in the sense of Shoup [27]. The q-SDH assumption has similar properties to the Strong-RSA assumption [5]. We use these properties to construct our short group signature scheme.

3.2 The Linear Diffie-Hellman Assumption

With $g_1 \in G_1$ as above, along with arbitrary generators u, v, and h of G_1, consider the following problem:

Decision Linear Problem in G_1. Given $u, v, h, u^a, v^b, h^c \in G_1$ as input, output `yes` if $a + b = c$ and `no` otherwise.

One can easily show that an algorithm for solving Decision Linear in G_1 gives an algorithm for solving DDH in G_1. The converse is believed to be false. That is, it is believed that Decision Linear is a hard problem even in bilinear groups where DDH is easy. More precisely, we define the advantage of an algorithm $\mathcal{A}$ in deciding the Decision Linear problem in G_1 as

$$\mathsf{Adv\,Linear}_{\mathcal{A}} \stackrel{\text{def}}{=} \left| \begin{array}{l} \Pr\left[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = \texttt{yes} : u, v, h \stackrel{\text{R}}{\leftarrow} G_1, a, b \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p\right] \\ \quad - \Pr\left[\mathcal{A}(u, v, h, u^a, v^b, \eta) = \texttt{yes} : u, v, h, \eta \stackrel{\text{R}}{\leftarrow} G_1, a, b \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p\right] \end{array} \right| .$$

The probability is over the uniform random choice of the parameters to $\mathcal{A}$, and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}$ (t, ϵ)-decides Decision Linear in G_1 if $\mathcal{A}$ runs in time at most t, and $\mathsf{Adv\,Linear}_{\mathcal{A}}$ is at least ϵ.

Definition 2. *We say that the (t, ϵ)-Decision Linear Assumption (LA) holds in G_1 if no t-time algorithm has advantage at least ϵ in solving the Decision Linear problem in G_1.*

In the full version of the paper we show that the Decision Linear Assumption holds in generic bilinear groups.

Linear Encryption The Decision Linear problem gives rise to the Linear encryption (LE) scheme, a natural extension of ElGamal encryption. Unlike ElGamal encryption, Linear encryption can be secure even in groups where a DDH-deciding algorithm exists. In this scheme, a user's public key is a triple of generators $u, v, h \in G_1$; her private key is the exponents $x, y \in \mathbb{Z}_p$ such that $u^x = v^y = h$. To encrypt a message $M \in G_1$, choose random values $a, b \in \mathbb{Z}_p$, and output the triple $(u^a, v^b, m \cdot h^{a+b})$. To recover the message from an encryption (T_1, T_2, T_3), the user computes $T_3/(T_1^x \cdot T_2^y)$. By a natural extension of the proof of security of ElGamal, LE is semantically secure against a chosen-plaintext attack, assuming Decision-LA holds.

4 A Zero-Knowledge Protocol for SDH

We are now ready to present the underlying building block for our group signature scheme. We present a protocol for proving possession of a solution to an SDH problem. The public values are $g_1, u, v, h \in G_1$ and $g_2, w \in G_2$. Here $w = g_2^\gamma$ for some (secret) $\gamma \in \mathbb{Z}_p$. The protocol proves possession of a pair (A, x), where $A \in G_1$ and $x \in \mathbb{Z}_p$, such that $A^{x+\gamma} = g_1$. Such a pair satisfies $e(A, wg_2^x) = e(g_1, g_2)$. We use a standard generalization of Schnorr's protocol for proving knowledge of discrete logarithm in a group of prime order [26].

Protocol 1. Alice, the prover, selects exponents $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$, and computes a Linear encryption of A:

$$T_1 \leftarrow u^\alpha \qquad T_2 \leftarrow v^\beta \qquad T_3 \leftarrow A h^{\alpha+\beta} \ .$$

She also computes two helper values $\delta_1 \leftarrow x\alpha$ and $\delta_2 \leftarrow x\beta$.

Alice and Bob then undertake a proof of knowledge of values $(\alpha, \beta, x, \delta_1, \delta_2)$ satisfying the following five relations:

$$\begin{gathered} u^\alpha = T_1 \qquad\qquad v^\beta = T_2 \\ e(T_3, g_2)^x \cdot e(h, w)^{-\alpha-\beta} \cdot e(h, g_2)^{-\delta_1-\delta_2} = e(g_1, g_2)/e(T_3, w) \\ T_1^x u^{-\delta_1} = 1 \qquad\qquad T_2^x v^{-\delta_2} = 1 \ . \end{gathered}$$

This proof proceeds as follows. Alice picks blinding values r_α, r_β, r_x, r_{δ_1}, and r_{δ_2} at random from $\mathbb{Z}_p$. She computes five values based on all these:

$$\begin{gathered} R_1 \leftarrow u^{r_\alpha} \qquad\qquad R_2 \leftarrow v^{r_\beta} \\ R_3 \leftarrow e(T_3, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}} \\ R_4 \leftarrow T_1^{r_x} \cdot u^{-r_{\delta_1}} \qquad\qquad R_5 \leftarrow T_2^{r_x} \cdot v^{-r_{\delta_2}} \ . \end{gathered}$$

She then sends $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$ to the verifier. Bob, the verifier, sends a challenge value c chosen uniformly at random from $\mathbb{Z}_p$. Alice computes and sends back $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_x = r_x + cx$, $s_{\delta_1} = r_{\delta_1} + c\delta_1$, and $s_{\delta_2} = r_{\delta_2} + c\delta_2$. Finally, Bob verifies the following five equations:

$$u^{s_\alpha} \stackrel{?}{=} T_1^c \cdot R_1 \tag{1}$$

$$v^{s_\beta} \stackrel{?}{=} T_2^c \cdot R_2 \tag{2}$$

$$e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \stackrel{?}{=} \big(e(g_1, g_2)/e(T_3, w)\big)^c \cdot R_3 \tag{3}$$

$$T_1^{s_x} u^{-s_{\delta_1}} \stackrel{?}{=} R_4 \tag{4}$$

$$T_2^{s_x} v^{-s_{\delta_2}} \stackrel{?}{=} R_5 \ . \tag{5}$$

Bob accepts if all five hold.

Theorem 1. *Protocol 1 is an honest-verifier zero-knowledge proof of knowledge of an SDH pair under the Decision Linear assumption.*

The proof of the theorem follows from the following lemmas that show that the protocol is (1) complete (the verifier always accepts an interaction with an honest prover), (2) zero-knowledge (can be simulated), and (3) a proof of knowledge (has an extractor).

Lemma 1. *Protocol 1 is complete.*

Proof. If Alice is an honest prover in possession of an SDH pair (A, x) she follows the computations specified for her in the protocol. In this case,

$$u^{s_\alpha} = u^{r_\alpha + c\alpha} = (u^\alpha)^c \cdot u^{r_\alpha} = T_1^c \cdot R_1 \quad ,$$

so (1) holds. For analogous reasons (2) holds. Further,

$$T_1^{s_x} u^{-s_{\delta_1}} = (u^\alpha)^{r_x + cx} u^{-r_{\delta_1} - cx\alpha} = (u^\alpha)^{r_x} u^{-r_{\delta_1}} = T_1^{r_x} \cdot R_4 \quad ,$$

so (4) holds. For analogous reasons (5) holds. Finally,

$$\begin{aligned}
&e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \\
&= e(T_3, g_2)^{r_x + cx} \cdot e(h, w)^{-r_\alpha - r_\beta - c\alpha - c\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2} - cx\alpha - cx\beta} \\
&= e(T_3, g_2^x)^c \cdot e(h^{-\alpha-\beta}, wg_2^x)^c \cdot \big(e(T_3, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}\big) \\
&= e(T_3 h^{-\alpha-\beta}, wg_2^x)^c \cdot e(T_3, w)^{-c} \cdot (R_3) \\
&= \big(e(A, wg_2^x)/e(T_3, w)\big)^c \cdot R_3 = \big(e(g_1, g_2)/e(T_3, w)\big)^c \cdot R_3 \quad .
\end{aligned}$$

so (3) holds. □

Lemma 2. *Transcripts of Protocol 1 can be simulated, under the Decision Linear assumption.*

Proof. We describe a simulator that outputs transcripts of Protocol 1.

Pick $A \xleftarrow{\mathrm{R}} \mathbb{G}_1$, and $\alpha, \beta \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. Set $T_1 \leftarrow u^\alpha$, $T_2 \leftarrow v^\beta$, and $T_3 \leftarrow Ah^{\alpha+\beta}$. Assuming the Decision Linear assumption holds on G_1, the tuples (T_1, T_2, T_3) generated by the simulator are drawn from a distribution that is indistinguishable from the distribution output by any particular prover.

The remainder of this simulator does not assume knowledge of A, x, α, or β, so it can also be used when T_1, T_2, and T_3 are pre-specified. When the pre-specified (T_1, T_2, T_3) are a random Linear encryption of some A, the remainder of the transcript is simulated perfectly.

Now choose a challenge $c \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. Select $s_\alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and set $R_1 \leftarrow T_1^c / u^{s_\alpha}$. Then (1) is satisfied. With α and c fixed, a choice for either of r_α or s_α determines the other, and a uniform random choice of one gives a uniform random choice of the other. Therefore s_α and R_1 are distributed as in a real transcript. Choose s_β and R_2 analogously.

Select $s_x, s_{\delta_1}, s_{\delta_2} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and set $R_4 \leftarrow T_1^{s_x} u^{s_{\delta_1}}$ and $R_5 \leftarrow T_2^{s_x} v^{s_{\delta_2}}$. Again, all the computed values are distributed as in a real transcript. Finally set

$$R_3 \leftarrow e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \big(e(T_3, w)/e(g_1, g_2)\big)^c \quad .$$

This R_3 satisfies (3), and it, too, is properly distributed.

The transcript output is $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. As argued above, this transcript is distributed identically to transcripts of Protocol 1, assuming the Decision Linear assumption holds. □

Lemma 3. *There exists an extractor for Protocol 1.*

Proof. Suppose that an extractor can rewind a prover in the protocol above to the point just before the prover is given a challenge c. At the first step of the protocol, the prover sends T_1, T_2, T_3 and R_1, R_2, R_3, R_4, R_5. Then, to challenge value c, the prover responds with s_α, s_β, s_x, s_{δ_1}, and s_{δ_2}. To challenge value $c' \neq c$, the prover responds with s'_α, s'_β, s'_x, s'_{δ_1}, and s'_{δ_2}. If the prover is convincing, all five verification equations (1–5) hold for each set of values.

For brevity, let $\Delta c = c - c'$, $\Delta s_\alpha = s_\alpha - s'_\alpha$, and similarly for Δs_β, Δs_x, Δs_{δ_1}, and Δs_{δ_2}.

Now consider (1) above. Dividing the two instances of this equation, we obtain $u^{\Delta s_\alpha} = T_1^{\Delta c}$. The exponents are in a group of known prime order, so we can take roots; let $\tilde{\alpha} = \Delta s_\alpha / \Delta c$. Then $u^{\tilde{\alpha}} = T_1$. Similarly, from (2), we obtain $\tilde{\beta} = \Delta s_\beta / \Delta c$ such that $v^{\tilde{\beta}} = T_2$.

Consider (4) above. Dividing the two instances gives $T_1^{\Delta s_x} = u^{\Delta s_{\delta_1}}$. Substituting $T_1 = u^{\tilde{\alpha}}$ gives $u^{\tilde{\alpha}\Delta s_x} = u^{\Delta s_{\delta_1}}$, or $\Delta s_{\delta_1} = \tilde{\alpha}\Delta s_x$. Similarly, from (5) we deduce that $\Delta s_{\delta_2} = \tilde{\beta}\Delta s_x$.

Finally, dividing the two instances of (3), we obtain

$$\begin{aligned}\bigl(e(g_1, g_2)/e(T_3, w)\bigr)^{\Delta c} &= e(T_3, g_2)^{\Delta s_x} \cdot e(h, w)^{-\Delta s_\alpha - \Delta s_\beta} \cdot e(h, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}} \\ &= e(T_3, g_2)^{\Delta s_x} \cdot e(h, w)^{-\Delta s_\alpha - \Delta s_\beta} \cdot e(h, g_2)^{-\tilde{\alpha}\Delta s_x - \tilde{\beta}\Delta s_x} \enspace .\end{aligned}$$

Taking Δc-th roots, and letting $\tilde{x} = \Delta s_x / \Delta c$, we obtain

$$e(g_1, g_2)/e(T_3, w) = e(T_3, g_2)^{\tilde{x}} \cdot e(h, w)^{-\tilde{\alpha} - \tilde{\beta}} \cdot e(h, g_2)^{-\tilde{x}(\tilde{\alpha} + \tilde{\beta})} \enspace .$$

This can be rearranged as

$$e(g_1, g_2) = e(T_3 h^{-\tilde{\alpha} - \tilde{\beta}}, w g_2^{\tilde{x}}) \enspace ,$$

or, letting $\tilde{A} = T_3 h^{-\tilde{\alpha} - \tilde{\beta}}$,

$$e(\tilde{A}, w g_2^{\tilde{x}}) = e(g_1, g_2) \enspace .$$

Thus the extractor obtains an SDH tuple $(\tilde{A}, \tilde{x})$. Moreover, the $\tilde{A}$ in this SDH tuple is, perforce, the same as that in the Linear encryption (T_1, T_2, T_3). □

5 SDH Signatures of Knowledge

Armed with Theorem 1, we obtain from Protocol 1 a signature scheme secure in the random oracle model by applying the Fiat-Shamir heuristic [16]. Signatures obtained from a proof of knowledge via the Fiat-Shamir heuristic are often called signatures of knowledge. We use a variant of the Fiat-Shamir heuristic, used also by Ateniese et al. [2], where the challenge c rather than the values $R_1, \ldots, R_5$ is transmitted in the signature; the output of the random oracle acts as a checksum for those values not transmitted.

The signature scheme is defined as follows. The public key contains a hash function (viewed as a random oracle) $H : \{0,1\}^* \to \mathbb{Z}_p$, groups G_1 and G_2 with respective generators g_1 and g_2 as in Sect. 2, the random generators u, v, and h of G_1, and $w = g_2^\gamma \in G_2$, where γ is chosen at random in $\mathbb{Z}_p^*$. The private key is an SDH pair (A, x), i.e., a pair such that $A^{x+\gamma} = g_1$. Any such pair is a valid private key.

The signer signs a message $M \in \{0,1\}^*$ using the private key (A, x) as follows. She first undertakes the computation specified in the first round of Protocol 1 to obtain $T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5$. She obtains the challenge c by giving M and her first-round values to the random oracle:

$$c \leftarrow H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p \quad . \tag{6}$$

She then undertakes the computation specified in the third round of the protocol using the challenge value c to obtain $s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}$. Finally, she outputs the signature σ, computed as

$$\sigma \leftarrow (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}) \quad . \tag{7}$$

The verifier uses equations (1–5) to re-derive R_1, R_2, R_3, R_4, and R_5:

$$\tilde{R}_1 \leftarrow u^{s_\alpha}/T_1^c \qquad \tilde{R}_2 \leftarrow v^{s_\beta}/T_2^c \qquad \tilde{R}_4 \leftarrow T_1^{s_x}/u^{s_{\delta_1}} \qquad \tilde{R}_5 \leftarrow T_2^{s_x}/v^{s_{\delta_2}}$$
$$\tilde{R}_3 \leftarrow e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \big(e(T_3, w)/e(g_1, g_2)\big)^c \quad .$$

He then checks that these, along with the other first-round messages included in σ, give the challenge c, i.e., that

$$c \stackrel{?}{=} H(M, T_1, T_2, T_3, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, \tilde{R}_5) \quad . \tag{8}$$

He accepts if this check succeeds.

The Fiat-Shamir heuristic shows that this signature scheme is secure against existential forgery in the random oracle model [1]. Note that a signature comprises three elements of G_1 and six of $\mathbb{Z}_p$.

6 Short Group Signatures from SDH

The signature scheme presented in Sect. 5 is, in fact, also a group signature scheme. In describing the scheme, we follow the definitions given by Bellare et al. [6].

Consider bilinear groups G_1 and G_2 with respective generators g_1 and g_2, as in Sect. 2. Suppose further that the SDH assumption holds on (G_1, G_2), and the Linear assumption holds on G_1. The scheme employs a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, treated as a random oracle in the proof of security.

***KeyGen*(n).** This randomized algorithm takes as input a parameter n, the number of members of the group, and proceeds as follows. Select $h \stackrel{\mathrm{R}}{\leftarrow} G_1 \setminus \{1_{G_1}\}$ and $\xi_1, \xi_2 \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p^*$, and set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select $\gamma \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p^*$, and set $w = g_2^\gamma$. Using γ, generate for each user i, $1 \le i \le n$, an SDH tuple

(A_i, x_i): select $x_i \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p^*$, and set $A_i \leftarrow g_1^{1/(\gamma+x_i)}$. The group public key is $gpk = (g_1, g_2, h, u, v, w)$. The private key of the group manager (the party able to trace signatures) is $gmsk = (\xi_1, \xi_2)$. Each user's private key is her tuple $\boldsymbol{gsk}[i] = (A_i, x_i)$. No party is allowed to possess γ; it is only known to the private-key issuer.

Sign$(gpk, \boldsymbol{gsk}[i], M)$**.** Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a user's key $\boldsymbol{gsk}[i] = (A_i, x_i)$, and a message $M \in \{0,1\}^*$, compute and output a signature of knowledge $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ as in the scheme of Sect. 5 (Equation (7)).

Verify(gpk, M, σ)**.** Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a message M, and a group signature σ, verify that σ is a valid signature of knowledge in the scheme of Sect. 5 (Equation (8)).

Open$(gpk, gmsk, M, \sigma)$**.** This algorithm is used for tracing a signature to a signer. It takes as input a group public key $gpk = (g_1, g_2, h, u, v, w)$ and the corresponding group manager's private key $gmsk = (\xi_1, \xi_2)$, together with a message M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ to trace, and proceeds as follows. First, verify that σ is a valid signature on M. Second, consider the first three elements (T_1, T_2, T_3) as a Linear encryption, and recover the user's A as $A \leftarrow T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$, following the decryption algorithm given at the end of Sect. 3.2. If the group manager is given the elements $\{A_i\}$ of the users' private keys, he can look up the user index corresponding to the identity A recovered from the signature.

Signature Length. A group signature in the system above comprises three elements of G_1 and six elements of $\mathbb{Z}_p$. Using any of the families of curves described in [9], one can take p to be a 170-bit prime and use a group G_1 where each element is 171 bits. Thus, the total group signature length is 1533 bits or 192 bytes. With these parameters, security is approximately the same as a standard 1024-bit RSA signature, which is 128 bytes.

Performance. The pairings $e(h, w)$, $e(h, g_2)$, and $e(g_1, g_2)$ can be precomputed and cached by both signers and verifiers. The signer can cache $e(A, g_2)$, and, when signing, compute $e(T_3, g_2)$ without evaluating a pairing. Accordingly, creating a group signature requires eight exponentiations (or multi-exponentiations) and no pairing computations. The verifier can derive $\tilde{R}_3$ efficiently by collapsing the $e(T_3, g_2)^{s_x}$ and $e(T_3, w)^c$ pairings into a single $e(T_3, w^c g_2^{s_x})$ term. Thus verifying a group signature requires six multi-exponentiations and one pairing computation. With parameters selected as above, the exponents are in every case 170-bit numbers. For the signer, all bases for exponentiation are fixed, which allows further speedup by precomputation.

6.1 Group Signature Security

We now turn to proving security of the system. Bellare et al. [6] give three properties that a group signature scheme must satisfy:

– correctness, which ensures that honestly-generated signatures verify and trace correctly;
– full-anonymity, which ensures that signatures do not reveal their signer's identity; and
– full-traceability, which ensures that all signatures, even those created by the collusion of multiple users and the group manager, trace to a member of the forging coalition.

For the details of the definitions, see Bellare et al. [6]. We prove the security of our scheme using a variation of these properties. In our proofs, we relax the full-anonymity requirement. As presented [6, Sect. 2], the full-anonymity experiment allows the adversary to query the opening (tracing) oracle before and after receiving the challenge σ. In this respect, the experiment mirrors the indistinguishability experiment against an adaptive CCA2 adversary. We therefore rename this experiment CCA2-full-anonymity. We define a corresponding experiment, CPA-full-anonymity, in which the adversary cannot query the opening oracle. We prove privacy in this slightly weaker model.

Access to the tracing functionality will likely be carefully controlled when group signatures are deployed, so CPA-full-anonymity is a reasonable model to consider. In any case, anonymity and unlinkability, the two traditional group signature security requirements implied by full anonymity [6, Sect. 3], also follow from CPA-full-anonymity. Thus a fully-traceable and CPA-fully-anonymous group signature scheme is still secure in the traditional sense.

In the statements of the theorem, we use big-O notation to elide the specifics of additive terms in time bounds, noting that, for given groups G_1 and G_2, operations such as sampling, exponentiation, and bilinear map evaluation are all constant-time.

Theorem 2. *The SDH group signature scheme is correct.*

Proof. For any group public key $gpk = (g_1, g_2, h, u, v, w)$, and for any user with key $\boldsymbol{gsk}[i] = (A_i, x_i)$, the key generation algorithm guarantees that $A_i^{\gamma+x_i} = g_1$, so (A_i, x_i) is an SDH tuple for $w = g_2^{\gamma}$. A correct group signature σ is a proof of knowledge, which is itself a transcript of the SDH protocol given in Sect. 4. Verifying the signature entails verifying that the transcript is correct; thus Lemma 1 shows that σ will always be accepted by the verifier.

Moreover, an honest signer outputs, as the first three components of any signature σ, values $(T_1, T_2, T_3) = (u^{\alpha}, v^{\beta}, A_i \cdot h^{\alpha+\beta})$ for some $\alpha, \beta \in \mathbb{Z}_p$. These values form a Linear encryption of A_i under public key (u, v, h), which the group manager, possessing the corresponding private key (ξ_1, ξ_2), can always recover. Therefore any valid signature will always be opened correctly. $\square$

Theorem 3. *If Linear encryption is (t', ϵ')-semantically secure on G_1 then the SDH group signature scheme is (t, q_H, ϵ)-CPA-fully-anonymous, where $\epsilon = \epsilon'$ and $t = t' - q_H O(1)$. Here q_H is the number of hash function queries made by the adversary and n is the number of members of the group.*

Proof. Suppose $\mathcal{A}$ is an algorithm that (t, q_H, ϵ)-breaks the anonymity of the group signature scheme. We show how to construct a $t+q_H O(1)$-time algorithm $\mathcal{B}$ that breaks the semantic security of Linear encryption (Sect. 3.2) with advantage at least ϵ.

Algorithm $\mathcal{B}$ is given a Linear encryption public key (u, v, h). It generates the remaining components of the group signature public key by following the group signature's key generation algorithm. It then provides to $\mathcal{A}$ the group public key (g_1, g_2, h, u, v, w), and the users' private keys (A_i, x_i).

At any time, $\mathcal{A}$ can query the random oracle H. Algorithm $\mathcal{B}$ responds with elements selected uniformly at random from $\mathbb{Z}_p$, making sure to respond identically to repeated queries.

Algorithm $\mathcal{A}$ requests its full-anonymity challenge by providing two indices, i_0 and i_1, and a message M. Algorithm $\mathcal{B}$, in turn, requests its indistinguishability challenge by providing the two user private keys A_{i_0} and A_{i_1} as the messages whose Linear encryption it must distinguish. It is given a Linear encryption (T_1, T_2, T_3) of A_{i_b}, where bit b is chosen by the Linear encryption challenger.

Algorithm $\mathcal{B}$ generates from this Linear encryption a protocol transcript $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ by means of the simulator of Lemma 2. This simulator can generate a trace given (T_1, T_2, T_3), even though $\mathcal{B}$ does not know α, β, or x. Since (T_1, T_2, T_3) is a random Linear encryption of A_{i_b}, the remainder of the transcript is distributed exactly as in a real protocol with a prover whose secret A is A_{i_b}.

Algorithm $\mathcal{B}$ then patches H at $(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$ to equal c. It encounters a collision only with negligible probability. In case of a collision, $\mathcal{B}$ declares failure and exits. Otherwise, it returns the valid group signature $\sigma \leftarrow (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a bit b'. Algorithm $\mathcal{B}$ returns b' as the answer to its own challenge. Since the encryption of A_{i_b} is turned by $\mathcal{B}$ into a group signature by user i_b, $\mathcal{B}$ answers its challenge correctly whenever $\mathcal{A}$ does.

The keys given to $\mathcal{A}$, and the answers to $\mathcal{A}$'s queries, are all valid and properly distributed. Therefore $\mathcal{A}$ succeeds in breaking the anonymity of the group signature σ with advantage ϵ, and $\mathcal{B}$ succeeds in distinguishing the Linear encryption (T_1, T_2, T_3) with the same advantage.

Algorithm $\mathcal{B}$'s running time exceeds $\mathcal{A}$'s by the amount it takes to answer $\mathcal{A}$'s queries. Each hash query can be answered in constant time, and there are at most q_H of them. Algorithm $\mathcal{B}$ can also create the challenge group signature σ in constant time. If $\mathcal{A}$ runs in time t, $\mathcal{B}$ runs in time $t + q_H O(1)$. $\square$

The following theorem proves full traceability of our system. The proof is based on the forking lemma [24] and is given in the full version of the paper.

Theorem 4. *If SDH is (q, t', ϵ')-hard on (G_1, G_2), then the SDH group signature scheme is $(t, q_H, q_S, n, \epsilon)$-fully-traceable, where $n = q - 1$, $\epsilon = 4n\sqrt{2\epsilon' q_H} + n/p$, and $t = \Theta(1) \cdot t'$. Here q_H is the number of hash function queries made by the adversary, q_S is the number of signing queries made by the adversary, and n is the number of members of the group.*

7 Revocation

We now discuss how to revoke users in the SDH group signature scheme of Sect. 6. A number of revocation mechanisms for group signatures have been proposed [4, 12]. All these mechanisms can be applied to our system. Here we describe a revocation mechanism along the lines of [12].

Recall that the group's public key in our system is (g_1, g_2, h, u, v, w) where $w = g_2^{\gamma} \in G_2$ for random $\gamma \in \mathbb{Z}_p^*$ and random $h, u, v \in G_1$. User i's private key is a pair (A_i, x_i) where $A_i = g_1^{1/(\gamma+x_i)} \in G_1$.

Now, suppose we wish to revoke users $1, \ldots, r$ without affecting the signing capability of other users. To do so, the Revocation Authority (RA) publishes a Revocation List (RL) containing the private keys of all revoked users. More precisely, $\mathrm{RL} = \{(A_1^*, x_1), \ldots, (A_r^*, x_r)\}$, where $A_i^* = g_2^{1/(\gamma+x_i)} \in G_2$. Note that $A_i = \psi(A_i^*)$. Here the SDH secret γ is needed to compute the A_i^*'s. In the case where G_1 equals G_2 then $A_i = A_i^*$ and consequently the Revocation List can be derived directly from the private keys of revoked users without having to use γ.

The list RL is given to all signers and verifiers in the system. It is used to update the group public key used to verify signatures. Let $y = \prod_{i=1}^{r}(\gamma + x_i) \in \mathbb{Z}_p^*$. The new public key is $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$ where $\bar{g}_1 = g_1^{1/y}$, $\bar{g}_2 = g_2^{1/y}$, and $\bar{w} = (\bar{g}_2)^{\gamma}$. We show that, given RL, anyone can compute this new public key, and any unrevoked user can update her private key locally so that it is well formed with respect to this new public key. Revoked users are unable to do so.

We show how to revoke one private key at a time. By repeating the process r times (as the revocation list grows over time) we can revoke all private keys on the Revocation List. We first show how given the public key (g_1, g_2, h, u, v, w) and one revoked private key $(A_1^*, x_1) \in \mathrm{RL}$ anyone can construct the new public key $(\hat{g}_1, \hat{g}_2, h, u, v, \hat{w})$ where $\hat{g}_1 = g_1^{1/(\gamma+x_1)}$, $\hat{g}_2 = g_2^{1/(\gamma+x_1)}$, and $\hat{w} = (\hat{g}_2)^{\gamma}$. This new public key is constructed simply as:

$$\hat{g}_1 \leftarrow \psi(A_1^*) \qquad \hat{g}_2 \leftarrow A_1^* \qquad \text{and} \qquad \hat{w} \leftarrow g_2 \cdot (A_1^*)^{-x_1} \ ;$$

then $\hat{g}_1 = \psi(A_1)^* = g_1^{1/(\gamma+x_1)}$ and $\hat{w} = g_2 \cdot (A_1^*)^{-x_1} = g_2^{1-\frac{x_1}{\gamma+x_1}} = (A_1^*)^{\gamma} = (\hat{g}_2)^{\gamma}$, as required.

Next, we show how unrevoked users update their own private keys. Consider an unrevoked user whose private key is (A, x). Given a revoked private key, (A_1^*, x_1) the user computes $\hat{A} \leftarrow \psi(A_1^*)^{1/(x-x_1)}/A^{1/(x-x_1)}$ and sets his new private key to be $(\hat{A}, x)$. Then, indeed,

$$(\hat{A})^{\gamma+x} = \psi(A_1^*)^{\frac{\gamma+x}{x-x_1}}/A^{\frac{\gamma+x}{x-x_1}} = \psi(A_1^*)^{\frac{(\gamma+x_1)+(x-x_1)}{x-x_1}}/g_1^{\frac{1}{x-x_1}} = \psi(A_1^*) = \hat{g}_1 \ ,$$

as required. Hence, $(\hat{A}, x)$ is a valid private key with respect to $(\hat{g}_1, \hat{g}_2, h, u, v, \hat{w})$.

By repeating this process r times (once for each revoked key in RL) anyone can compute the updated public key $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$ defined above. Similarly, an unrevoked user with private key (A, x) can compute his updated private key $(\bar{A}, x)$ where $\bar{A} = (\bar{g}_1)^{1/(\gamma+x)}$. We note that it is possible to process the entire

RL at once (as opposed to one element at a time) and compute $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$ directly; however this is less efficient when keys are added to RL incrementally.

A revoked user cannot construct a private key for the new public key $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$. In fact, the proof of Theorem 4 shows that, if a revoked user can generate signatures for the new public key $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$, then that user can be used to break the SDH assumption. Very briefly, the reason is that given an SDH challenge one can easily generate a public key tuple $(\bar{g}_1, \bar{g}_2, h, u, v, \bar{w})$ along with the private key for a revoked user $(g_1^{1/(x+\gamma)}, x)$. Then an algorithm that can forge signatures given these two tuples can be used to solve the SDH challenge.

Brickell [11] proposes an alternate mechanism where revocation messages are only sent to signature verifiers, so that there is no need for unrevoked signers to update their keys. Similar mechanisms were also considered by Ateniese et al. [4] and Kiayias et al. [19]. We refer to this as Verifier-Local Revocation (VLR) group signatures. Boneh and Shacham [10] show how to modify our group signature scheme to support this VLR revocation mechanism.

8 Exculpability

In Bellare et al. [6], exculpability (introduced by Ateniese and Tsudik [3]) is informally defined as follows: No member of the group and not even the group manager – the entity that is given the tracing key – can produce signatures on behalf of other users. Thus, no user can be framed for producing a signature he did not produce. They argue that a group signature secure in the sense of full-traceability also has the exculpability property. Thus, in the terminology of Bellare et al. [6], our group signature has the exculpability property.

A stronger notion of exculpability is considered in Ateniese et al. [2], where one requires that even the entity that *issues* user keys cannot forge signatures on behalf of users. Formalizations of strong exculpability have recently been proposed by Kiayias and Yung [20] and by Bellare, Shi, and Zhang [7].

To achieve this stronger property the system of Ateniese et al. [2] uses a protocol (called JOIN) to issue a key to a new user. At the end of the protocol, the key issuer does not know the full private key given to the user and therefore cannot forge signatures under the user's key.

Our group signature scheme can be extended to provide strong exculpability using a similar mechanism. Instead of simply giving user i the private key $(g_1^{1/(\gamma+x_i)}, x_i)$, the user and key issuer engage in a JOIN protocol where at the end of the protocol user i has a triple (A_i, x_i, y_i) such that $A_i^{\gamma+x_i} h_1^{y_i} = g_1$ for some public parameter h_1. The value y_i is chosen by the user and is kept secret from the key issuer. The ZKPK of Sect. 4 can be modified to prove knowledge of such a triple. The resulting system is a short group signature with strong exculpability.

9 Conclusions

We presented a group signature scheme based on the Strong Diffie-Hellman (SDH) and Linear assumptions. The signature makes use of a bilinear map

$e : G_1 \times G_2 \to G_T$. When any of the curves described in [9] are used, the group G_1 has a short representation and consequently we get a group signature whose length is under 200 bytes – less than twice the length of an ordinary RSA signature (128 bytes) with comparable security. Signature generation requires no pairing computations, and verification requires a single pairing; both also require a few exponentiations with short exponents.

Acknowledgments

The authors thank the anonymous referees for their valuable feedback.

References

[1] M. Abdalla, J. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 418–33. Springer-Verlag, May 2002.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *LNCS*, pages 255–70. Springer-Verlag, Aug. 2000.

[3] G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Proceedings of Financial Cryptography 1999*, volume 1648, pages 196–211. Springer-Verlag, Feb. 1999.

[4] G. Ateniese, G. Tsudik, and D. Song. Quasi-efficient revocation of group signatures. In M. Blaze, editor, *Proceedings of Financial Cryptography 2002*, Mar. 2002.

[5] N. Baric and B. Pfitzman. Collision-free accumulators and fail-stop signature schemes without trees. In *Proceedings of Eurocrypt 1997*, pages 480–494. Springer-Verlag, May 1997.

[6] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 614–29. Springer-Verlag, May 2003.

[7] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. `http://eprint.iacr.org/`.

[8] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, LNCS, pages 56–73. Springer-Verlag, May 2004.

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–32. Springer-Verlag, Dec. 2001. Full paper: `http://crypto.stanford.edu/~dabo/pubs.html`.

[10] D. Boneh and H. Shacham. Group signatures with verifier-local revocation, 2004. Manuscript.

[11] E. Brickell. An efficient protocol for anonymously providing assurance of the container of a private key, Apr. 2003. Submitted to the Trusted Computing Group.

[12] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 61–76. Springer-Verlag, Aug. 2002.

[13] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Proceedings of Crypto 2004*, LNCS. Springer-Verlag, Aug. 2004.

[14] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 257–65. Springer-Verlag, 1991.

[15] X. Ding, G. Tsudik, and S. Xu. Leak-free group signatures with immediate revocation. In T. Lai and K. Okada, editors, *Proceedings of ICDCS 2004*, Mar. 2004.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, Aug. 1986.

[17] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of SOSP 2003*, pages 193–206, Oct. 2003.

[18] IEEE P1556 Working Group, VSC Project. Dedicated short range communications (DSRC), 2003.

[19] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 571–89. Springer-Verlag, May 2004.

[20] A. Kiayias and M. Yung. Group signatures: Efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.

[21] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Proceedings of SAC 1999*, volume 1758 of *LNCS*, pages 184–99. Springer-Verlag, Aug. 1999.

[22] S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–4, Feb. 2002.

[23] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, May 2001.

[24] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–96, 2000.

[25] K. Rubin and A. Silverberg. Supersingular Abelian varieties in cryptology. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 336–53. Springer-Verlag, Aug. 2002.

[26] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[27] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–66. Springer-Verlag, May 1997.

[28] Trusted Computing Group. Trusted Computing Platform Alliance (TCPA) Main Specification, 2003. Online: `www.trustedcomputinggroup.org`.

[29] G. Tsudik and S. Xu. Accumulating composites and improved group signing. In C. S. Laih, editor, *Proceedings of Asiacrypt 2003*, volume 2894 of *LNCS*, pages 269–86. Springer-Verlag, Dec. 2003.