Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100603

Pairing-Friendly Curves with Discrete Logarithm Trapdoor Could be Useful

Parshuram Budhathoki[1], *Thomas Eisenbarth*[2], *Rainer Steinwandt*[3,*] *and Adriana Suárez Corona*[4]

[1] Salt Lake Community College, Salt Lake City, UT 84123, USA
[2] Worcester Polytechnic Institute, Worcester, MA 01609-2280, USA
[3] Florida Atlantic University, Boca Raton, FL 33431, USA
[4] Universidad de León, 24004 León, Spain

Received: 14 Oct. 2014, Revised: 22 May 2016, Accepted: 13 Jun. 2016
Published online: 1 Nov. 2016

Abstract: Pairing-friendly curves and elliptic curves with a trapdoor for the discrete logarithm problem are versatile tools in the design of cryptographic protocols. We show that curves having both properties simultaneously enable a non-interactive protocol for identity-based 3-party key distribution and deterministic identity-based signing with "short" signatures. All our protocols are in the random oracle model.

Keywords: cryptography, identity-based signature, non-interactive identity-based key distribution, pairing, discrete logarithm trapdoor

1 Introduction

Pairing-friendly curves are a versatile tool in the design of cryptographic protocols, especially for identity-based solutions. As documented in a taxonomy by Freeman et al. [11], a number of constructions to obtain pairing-friendly elliptic curves are available. Another cryptographically useful family of elliptic curves comes with a trapdoor for the discrete logarithm problem. At ASIACRYPT 2000, Paillier proposed several encryption schemes invoking such curves [14], and more recently Teske [19] suggested an elliptic curve cryptosystem with a trapdoor for the discrete logarithm problem. Interestingly, no constructions for elliptic curves in the intersection of these two families appear to be available in the literature. Differing from a setting considered by Dent and Galbraith [10], we want the efficient evaluation of the pairing to be possible without invoking trapdoor information.

As demonstrated in Section 3, in the random oracle model a complete construction would yield a non-interactive solution for identity-based 3-party key distribution. Thereafter, in Section 4, we present a deterministic identity-based signature scheme in the random oracle model, which assumes the availablity of a pairing-friendly group with discrete logarithm trapdoor. This identity-based signature scheme affords "short" signatures in the sense that the signature consists of a single group element. While it is known how to create such a signature in the public key setting [5], no such construction is known in the identity-based setting. Moreover, the number of group and pairing operations in the described scheme compares favorably to existing schemes, and the verification cost can be reduced when verifying multiple messages that are presumably signed by the same identity. The security reduction relies on the strong Diffie-Hellman assumption, which comes at the usual cost: results by Brown and Gallant [6], Cheon [8], and by Jao and Yoshida [13] indicate that for many groups the strong Diffie-Hellman problem is easier to solve than the discrete logarithm problem.

2 Preliminaries

The main technical tools we need are admissible bilinear maps and two hardness assumptions inspired by the Diffie-Hellman problem. In this section we briefly fix the relevant terminology (cf. [4,2]). We will denote the security parameter by k and statements about polynomial time always refer to k.

* Corresponding author e-mail: rsteinwa@fau.edu

Definition 1(Admissible bilinear map). *Let* $(\mathbb{G}_1,+)$, $(\mathbb{G}_2,+)$, *and* $(\mathbb{G}_T,\cdot)$ *denote cyclic groups of prime order* $q \in [2^k, 2^{k+1}]$. *Then we refer to a map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ *as* admissible bilinear map *if all of the following conditions are satisfied:*

Bilinearity:For all $(P_1,P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ *and* $a,b \in \mathbb{Z}$ *we have*

$$e(aP_1,bP_2) = e(P_1,P_2)^{ab}.$$

Non-degeneracy:There exists $(Q_1,Q_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ *such that* $e(Q_1,Q_2) \neq 1$.

Efficiency:The map e can be evaluated in polynomial time.

All schemes described subsequently are formulated in the random oracle model. For the construction of the non-interactive identity-based 3-party key distribution scheme we assume the admissible bilinear map to be symmetric, i. e., $\mathbb{G}_1 = \mathbb{G}_2$. In this setting, we capture the bilinear Diffie-Hellman problem as follows:

Definition 2(BDH problem). *Let* $(\mathbb{G},+)$ *and* $(\mathbb{G}_T,\cdot)$ *denote cyclic groups of prime order q such that there is an admissible bilinear map* $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. *The* bilinear Diffie-Hellman (BDH) problem *for* $(\mathbb{G},\mathbb{G}_T,e)$ *is to find on input*

$$(G, a\cdot G, b\cdot G, c\cdot G) \in \mathbb{G}^4$$

with uniformly at random chosen generator $G \in \mathbb{G}$ *and uniformly at random chosen* $a,b,c \in \{0,\dots,q-1\}$, *the element* $e(G,G)^{abc}$.

In Section 3.2 we will show how a successful adversary against our non-interactive key distribution scheme can be used to construct an algorithm to solve the BDH problem in the underlying group. For the identity-based signature scheme that we propose, the underlying hardness assumption will be the strong Diffie-Hellman assumption. Following [2] we capture this problem as follows:

Definition 3(Strong Diffie-Hellman problem). *Let* $(\mathbb{G}_1,+)$ *and* $(\mathbb{G}_2,+)$ *denote cyclic groups of prime order q such that there is an admissible bilinear pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. *The* ℓ-Strong Diffie-Hellman (ℓ-SDH) problem *for* $(\mathbb{G}_1,\mathbb{G}_2)$ *is to find on input*

$$([r^i \cdot G_1]_{i=0}^{\ell}, [G_2, r\cdot G_2]) \in \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2^2$$

with uniformly at random chosen generators $G_1 \in \mathbb{G}_1$, $G_2 \in \mathbb{G}_2$ *and uniformly at random chosen* $r \in \{0,\dots,q-1\}$ *a pair* $(c, \frac{1}{r+c}\cdot G_1)$ *with* $c \in \{0,\dots,q-1\} \setminus \{-r\}$.

In Section 4.2 we will show how from an efficient algorithm $\mathscr{A}$ successfully forging a signature one can derive an algorithm to solve the ℓ-SDH problem in the underlying group pair.

The groups needed for our construction can, unfortunately, not be chosen as arbitrary elliptic curves as used in (pairing-based) cryptography. We require a trapdoor for the discrete logarithm problem, as described in [17]:

Definition 4(Trapdoor discrete logarithm group). *A* trapdoor discrete log group *(TDL group) is defined by a pair of algorithms TDLGen and SolveDL as follows:*

TDLGen:This algorithm takes a security parameter 1^k *as input to generate a (description of a) cyclic group* $\mathbb{G}$ *of some order q with generator G and trapdoor information T.*

SolveDL:This polynomial-time algorithm takes as input 1^k, $(\mathbb{G},q,G,T)$ *and a group element H to produce a discrete logarithm* $a \in \mathbb{Z}_q$ *such that* $H = aG$.

To be usable in our protocols, we have to assume that the TDL groups output by the generator in this definition are pairing-friendly (see [11]). As shown in the next sections, having both a trapdoor for the discrete logarithm and a pairing available, enables interesting cryptographic applications.

3 Non-interactive identity-based 3-party key distribution

Keeping with the notation from Section 2, let $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be an admissible bilinear map, where $\mathbb{G} = \langle G \rangle$ is cyclic of prime order q such that a discrete logarithm trapdoor is available. We also make use of random oracles $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}$ mapping user identities to an element of $\mathbb{G}$ and $H_2 : \mathbb{G}_T \longrightarrow \{0,1\}^k$, mapping elements in $\mathbb{G}_T$ to session keys.

3.1 Description of the proposed scheme

Building on Paterson and Srinivasan's definition of a 2-party identity-based non-interactive key distribution scheme [17], we consider a 3-party version of this task. A *3-party identity-based non-interactive key distribution scheme* (3-ID-NIKD) is specified by a tuple of polynomial time algorithms:

Setup:This probabilistic algorithm is run by the trusted authority. Given the security parameter 1^k, Setup generates a secret master key along with the public system parameters. The public system parameters include the description of the private key space and the shared key space. We choose the latter as $\{0,1\}^k$.

KeyExtract:This probabilistic algorithm is run by the trusted authority to generate a secret user key from an identity.

SharedKey:This deterministic algorithm is run by a user to generate a shared key, using its private user key and two other users' identities.

We require that for any three pairwise different identities $\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C$ and corresponding private keys $S_{\mathrm{id}_A}, S_{\mathrm{id}_B}, S_{\mathrm{id}_C}$, SharedKey satisfies the constraint

$$\mathrm{SharedKey}(S_{\mathrm{id}_A}, \mathrm{id}_B, \mathrm{id}_C) =$$

Setup:Sets as secret master key the discrete logarithm trapdoor T for $\mathbb{G} = \langle G \rangle$ and outputs the public parameters $(\mathbb{G}, G, \mathbb{G}_T, e, H_1, H_2)$.
KeyExtract:On input an identity $\mathrm{id} \in \{0,1\}^*$ and master key T, the algorithm returns $S_{id} = \log_G(H_1(\mathrm{id}))$.
SharedKey:On input a private key S_{id_A} and two distinct identifiers $\mathrm{id}_B, \mathrm{id}_C \in \{0,1\}^* \setminus \{\mathrm{id}_A\}$, this algorithm outputs

$$K_{A,B,C} := H_2\left(e(H_1(\mathrm{id}_B), H_1(\mathrm{id}_C))^{S_{\mathrm{id}_A}}\right).$$

Fig. 1: a 3-party identity-based non-interactive key distribution scheme

$$\mathrm{SharedKey}(S_{\mathrm{id}_B}, \mathrm{id}_A, \mathrm{id}_C) = \mathrm{SharedKey}(S_{\mathrm{id}_C}, \mathrm{id}_A, \mathrm{id}_B).$$

This ensures that all the users corresponding to identities $\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C$ can compute the same session key without any interaction. Figure 1 describes our proposed 3-party identity-based non-interactive key distribution scheme.

Since $\mathbb{G}$ is cyclic, it is not difficult to see that the scheme in Figure 1 is correct; all the users with identities id_A, id_B and id_C will obtain identical keys when executing SharedKey with their respective private key and the other users' identities.

3.2 Security analysis

To capture the security of a 3-ID-NIKD scheme, we build on the security model used by Paterson and Srinivasan in [17]. It has to be infeasible to distinguish efficiently between a shared key established among three users and a random element from $\{0,1\}^k$—even knowing session keys established by a proper subset of these users with other identities and knowing private keys of users not involved. The adversary is modeled as a probabilistic algorithm $\mathscr{A}$ which obtains the public parameters produced by Setup as input. In addition to the random oracles H_1 and H_2, the algorithm $\mathscr{A}$ has access to the following oracles:

Key extraction oracle $\mathscr{E}$:On input an identity $\mathrm{id} \in \{0,1\}^*$, the corresponding secret key S_{id} is returned.
Reveal oracle $\mathscr{R}$:On input of three pairwise distinct user identities $\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C \in \{0,1\}^*$, this oracle returns the output of SharedKey when being executed with input $(S_{\mathrm{id}_A}, \mathrm{id}_B, \mathrm{id}_C)$.
Test oracle $\mathscr{T}$:The key $K_{A,B,C}$ is computed in the same way as for answering a Reveal query. Moreover, a random bit $b \xleftarrow{\$} \{0,1\}$ is chosen. If $b = 0$, the oracle returns $K_{A,B,C}$. Otherwise ($b = 1$), the oracle returns a uniformly at random chosen element from the session key space $\{0,1\}^k$.

The algorithm $\mathscr{A}$ outputs a value $b' \in \{0,1\}$ and wins if and only if $b = b'$ and the following restrictions hold:

– The test oracle has been queried only once; let $(\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C)$ be this query.
– The identities id_A, id_B, id_C are pairwise distinct.
– None of the identities $\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C$ has been queried to the key extraction oracle.
– Neither the triple $(\mathrm{id}_A, \mathrm{id}_B, \mathrm{id}_C)$ nor any permutation of it has been queried to the reveal oracle.

The advantage of such an adversary is defined as

$$\mathrm{Adv}_{\mathscr{A}}^{\text{3-id-nikd}}(k) := \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

Theorem 1.*Assume there is a polynomial time algorithm $\mathscr{A}$ such that the advantage $\mathrm{Adv}_{\mathscr{A}}^{\text{3-id-nikd}}$ is non-negligible. Then there is a polynomial time algorithm $\mathscr{B}$ that solves the BDH problem in the underlying group with non-negligible success probability.*

*Proof.*For $\ell = 1, 2$ denote by $q_\ell \geq 1$ a polynomial upper bound on the number of $\mathscr{A}$'s queries to H_ℓ—including implicit queries through $\mathscr{E}$ and $\mathscr{R}$—and let $(G, a \cdot G, b \cdot G, c \cdot G)$ be the input for the BDH solver $\mathscr{B}$ that we want to derive. The task $\mathscr{B}$ faces is to compute $e(G,G)^{abc}$, and to do so, $\mathscr{B}$ will simulate all oracles for $\mathscr{A}$ and provide all its inputs. As public parameters for $\mathscr{A}$, the algorithm $\mathscr{B}$ uses $(\mathbb{G}, G, \mathbb{G}_T, e, H_1, H_2)$. To establish the theorem, we use "game hopping", letting the adversary $\mathscr{A}$ interact with $\mathscr{B}$ as simulator. The advantage of $\mathscr{A}$ in Game i will be denoted by $\mathrm{Adv}_{\mathscr{A}}^{\text{Game } i}$, and we assume without loss of generality that $\mathscr{A}$ submits the three challenge identities to H_1 before submitting them to the test oracle $\mathscr{T}$.

Game 0: This game is identical to the original attack game, with all oracles of $\mathscr{A}$ being simulated faithfully. Consequently,

$$\mathrm{Adv}_{\mathscr{A}}^{\text{3-id-nikd}} = \mathrm{Adv}_{\mathscr{A}}^{\text{Game } 0}.$$

Game 1: In this game we modify the simulation in such a way that at the beginning the simulator guesses uniformly at random which identities $\mathrm{id}_I, \mathrm{id}_J, \mathrm{id}_L$ will be queried to the test oracle $\mathscr{T}$. Whenever this guess turns out to be wrong, we abort the simulation and consider the adversary to be at loss. Otherwise the game is identical with Game 0. Consequently,

$$\mathrm{Adv}_{\mathscr{A}}^{\text{Game } 0} \leq q_1^3 \cdot \mathrm{Adv}_{\mathscr{A}}^{\text{Game } 1},$$

and since q_1 is polynomial in k, it suffices to recognize $\mathrm{Adv}_{\mathscr{A}}^{\text{Game } 1}$ as negligible.

Game 2: In this game we change the simulation of $\mathscr{A}$'s queries as follows:

H_1: The algorithm $\mathscr{B}$ keeps an initially empty list to answer H_1-queries. If a queried identity id already appears in an entry (id, d, h) of the list, then $\mathscr{B}$'s answer to the query will be h. Otherwise, if the i-th H_1 query is on identity id_i, then $\mathscr{B}$ proceeds as follows:

– If $i = I$, then $\mathscr{B}$ adds an entry $(\mathrm{id}_I, \perp, a \cdot G)$ to the H_1-list and returns $a \cdot G$ as answer to the query.

– If $i = J$, then $\mathscr{B}$ adds an entry $(\mathrm{id}_J, \perp, b \cdot G)$ to the H_1-list and returns $b \cdot G$ as answer to the query.

– If $i = L$, then $\mathscr{B}$ adds an entry $(\mathrm{id}_L, \perp, c \cdot G)$ to the H_1-list and returns $c \cdot G$ as answer to the query.

– Otherwise, $\mathscr{B}$ selects $d_i \in \{0, \ldots, q-1\}$ uniformly at random, adds an entry $(\mathrm{id}_i, d_i, d_i \cdot G)$ to the list and returns $d_i \cdot G$ as answer to the query.

So $\mathscr{B}$'s responses are uniformly and independently generated.

H_2: The algorithm $\mathscr{B}$ keeps an initially empty list to answer H_2-queries. If a query s already appears in an entry (s, N) of the list, then $\mathscr{B}$'s answer to the query will be N. Otherwise, if the i-th H_2-query is on s_i, then $\mathscr{B}$ selects a random element N_i from $\{0, \ldots, q-1\}$, adds the entry (s_i, N_i) to the list and N_i is the answer to the query. So $\mathscr{B}$'s responses are uniformly and independently generated.

$\mathscr{E}$: When $\mathscr{A}$ wants to extract the private key for identity id, then $\mathscr{B}$ first queries H_1 with id, if this has not already been done. Notice that a query on $\mathrm{id} \in \{\mathrm{id}_I, \mathrm{id}_J, \mathrm{id}_L\}$, is not allowed. For any other identity, $\mathscr{B}$ finds the entry (id, d, h) of the list corresponding to id and outputs d.

$\mathscr{R}$: When $\mathscr{A}$ wants to reveal the session key for three pairwise distinct identities $\mathrm{id}_i, \mathrm{id}_j, \mathrm{id}_l$ the simulator $\mathscr{B}$ queries H_1 on $\mathrm{id}_i, \mathrm{id}_j, \mathrm{id}_l$ if this has not already been done.

– Next, $\mathscr{B}$ looks up the entries $(\mathrm{id}_m, d_m, h_m)$ for $m \in \{i, j, l\}$ in the H_1-list. Notice there is at least one $d_{t^*} \neq \perp$, since otherwise the query would not be allowed. Thus, $\mathscr{B}$ can execute SharedKey. E.g., if we assume $t^* = i$, then $\mathscr{B}$ answers with $H_2(e(H(\mathrm{id}_j), H(\mathrm{id}_k))^{d_{t^*}})$, first making the H_2-query if necessary.

$\mathscr{T}$: At some point during the simulation $\mathscr{A}$ queries the tuple of identities $\mathrm{id}_I, \mathrm{id}_J, \mathrm{id}_L$ to the test oracle. As reply, $\mathscr{B}$ returns a randomly generated $K \xleftarrow{\$} \{0,1\}^k$. (Notice that because of the way in which the simulation is set up, the "correct" key is equal to $H_2(e(G,G)^{abc})$).

This completes our description of $\mathscr{B}$'s simulation.

Game 2 and Game 1 are identical unless $\mathscr{A}$ queries $e(G,G)^{abc}$ to H_2, and we denote the event that $e(G,G)^{abc}$ is queried to H_2 by Q. Using the difference lemma,

$$|\mathrm{Adv}_{\mathscr{A}}^{\text{Game 1}} - \mathrm{Adv}_{\mathscr{A}}^{\text{Game 2}}| \le \Pr[\mathsf{Q}].$$

We prove now that $\Pr[\mathsf{Q}] \le q_2 \cdot \mathrm{Adv}_{\mathscr{B}}^{\mathrm{bdh}}$, where $\mathrm{Adv}_{\mathscr{B}}^{\mathrm{bdh}} = \Pr[\mathsf{Succ}_{\mathscr{B}}^{\mathrm{bdh}}]$ denotes the success probability of $\mathscr{B}$ for solving the BDH problem. If $\mathscr{A}$ terminates by outputting a bit b' (or if $\mathscr{A}$ exceeds its normal running time), then $\mathscr{B}$ outputs the value s_R held in the R-th entry of the H_2 list, with R being chosen uniformly at random. If event Q occurs, the value $e(G,G)^{abc}$ is in the H_2-list, so $\mathscr{B}$ will succeed with probability at least $1/q_2$: $\mathrm{Adv}_{\mathscr{B}}^{\mathrm{bdh}} =$

$$\Pr[\mathsf{Succ}_{\mathscr{B}}^{\mathrm{bdh}}|\mathsf{Q}] \cdot \Pr[\mathsf{Q}] + \Pr[\mathsf{Succ}_{\mathscr{B}}^{\mathrm{bdh}}|\neg\mathsf{Q}] \cdot \Pr[\neg\mathsf{Q}] \ge \frac{1}{q_2} \cdot \Pr[\mathsf{Q}].$$

Hence, $\Pr[\mathsf{Q}] \le q_2 \cdot \mathrm{Adv}_{\mathscr{B}}^{\mathrm{bdh}}$, and we finish the proof by discussing $\mathrm{Adv}_{\mathscr{A}}^{\text{Game 2}}$: $\mathrm{Adv}_{\mathscr{A}}^{\text{Game 2}} =$

$$\begin{aligned}
&\left|\Pr[b' = b] - \frac{1}{2}\right| \\
= &\left|\Pr[b' = b|\mathsf{Q}] \cdot \Pr[\mathsf{Q}] + \Pr[b' = b|\neg\mathsf{Q}] \cdot \Pr[\neg\mathsf{Q}] - \frac{1}{2}\right| \\
= &\left|(\Pr[b' = b|\mathsf{Q}] - \Pr[b' = b|\neg\mathsf{Q}]) \cdot \Pr[\mathsf{Q}] + \right. \\
&\left. \Pr[b' = b|\neg\mathsf{Q}] - \frac{1}{2}\right| \le \Pr[\mathsf{Q}]
\end{aligned}$$

This last inequality holds since, if Q does not occur, $\mathscr{A}$ never queried H_2 on $e(G,G)^{abc}$. Thus, $\mathscr{A}$'s view is independent of the value $H_2(e(G,G)^{abc})$ and $|\Pr[b' = b|\neg\mathsf{Q}] - \frac{1}{2}|$ is zero. Also, $|\Pr[b' = b|\mathsf{Q}] - \Pr[b' = b|\neg\mathsf{Q}]| \le 1$.

Putting all the probabilities together, we get:

$$\mathrm{Adv}_{\mathscr{A}}^{\text{3-id-nikd}} \le 2 \cdot q_1^3 \cdot q_2 \cdot \mathrm{Adv}_{\mathscr{B}}^{\mathrm{bdh}}$$

□

4 A "short" identity-based signature

For $i = 1, 2$, denote by $\mathbb{G}_i$ an additively written cyclic group of prime order $q \in [2^k, 2^{k+1}]$ with uniformly at random chosen public generator G_i. We also assume that a trapdoor for the discrete logarithm problem in $\mathbb{G}_2$ and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ are available for some group $\mathbb{G}_T$ of order q. As indicated already, the identity-based signature scheme we propose is formulated in the random oracle model: let $h : \{0,1\}^* \longrightarrow \{0, \ldots, q-1\}$ be a random oracle that maps a message into an integer modulo q and $H : \{0,1\}^* \longrightarrow \mathbb{G}_1 \times \mathbb{G}_2$ a random oracle that maps user identities to an element of $\mathbb{G}_1 \times \mathbb{G}_2$. To refer to the two components of a value $H(\mathrm{id})$ we will use the notation $H(\mathrm{id}) = (\underbrace{H_{\mathrm{id},1}}_{\in \mathbb{G}_1}, \underbrace{H_{\mathrm{id},2}}_{\in \mathbb{G}_2})$.

4.1 Description of the identity-based signature scheme

To specify an identity-based signature scheme we have to provide four (polynomial-time) algorithms:

Setup:This algorithm is run by the trusted authority to generate a secret master key and public system parameters.

Extract:This algorithm is run by the trusted authority to extract a user-specific secret signing key from an identity.

Sign:This algorithm enables a user to create a signature for a message, using its secret user key (extracted by the trusted authority).

Verify:Given a message, a candidate signature, and the identity of the potential signer, this algorithm allows to decide if this signature is valid.

Figure 2 shows how each of these algorithms is realized in our proposal.

The signature computation fails if $r_{\mathrm{id}} + h(m) = 0 \pmod q$, i. e., if the required inversion modulo q cannot be performed. As h is a random oracle, this happens with (negligible) probability $1/q$ only. Otherwise the correctness of our scheme follows immediately from the equality

$$e(\sigma, r_{\mathrm{id}} \cdot G_2 + h(m) \cdot G_2) = e(S_{\mathrm{id}}, G_2) = e(H_{\mathrm{id},1}, P_{\mathrm{pub}}).$$

Section 4.3 offers a more detailed performance discussion, but one immediately observes the following:

–The right-hand side of the verification equation is message-independent and can be reused by the verifier.

–The signing algorithm is deterministic and can be implemented on devices which do not provide a (pseudo)random number generator.

4.2 Security analysis

To analyze the security of our scheme, we prove that from an algorithm $\mathscr{A}$ that produces an existential forgery, we can derive an algorithm $\mathscr{C}$ with comparable resource requirements that, for a suitable ℓ, solves the ℓ-SDH problem in the underlying group pair. The proof we give is an adaption of an analysis by Boneh and Boyen [2] and by Cha and Cheon [7] to our situation. The adversary is modeled as a probabilistic algorithm $\mathscr{A}$ which obtains the public parameters produced by Setup as input. Following [7], the algorithm $\mathscr{A}$ has access to the random oracles h and H and to two more oracles:

Key extraction oracle $\mathscr{E}$:On input an identity $\mathrm{id} \in \{0,1\}^*$, the corresponding secret key $(r_{\mathrm{id}}, S_{\mathrm{id}})$ is returned.

Signature oracle $\mathscr{S}$:On input a user identity $\mathrm{id} \in \{0,1\}^*$ and a message m, this oracle returns the output of Sign when being executed with input $(r_{\mathrm{id}}, S_{\mathrm{id}})$ and m.

The algorithm $\mathscr{A}$ outputs a user identity id_0, a message m_0, and a signature for m_0. If this signature is valid and neither id_0 has been queried to the key extraction oracle nor (id_0, m_0) has been queried to the signature oracle, then $\mathscr{A}$ succeeded in creating an existential forgery.

For the proof that an efficient algorithm to create existential forgeries in our scheme can be turned into an efficient algorithm to solve the ℓ-SDH problem for the group pair $(\mathbb{G}_1, \mathbb{G}_2)$ and a suitable ℓ, we make use of (the proof of) a lemma by Cha and Cheon [7, Lemma 1] and a lemma by Boneh and Boyen [2, Lemma 9].

Lemma 1.*Let $\mathscr{A}$ be an algorithm that in polynomial time with probability $\varepsilon_{\mathscr{A}}$ outputs an existential forgery for the scheme in Figure 2 for some identity* id_0. *Morever, let* $\mathrm{id}_1 \in \{0,1\}^k$ *be chosen uniformly at random and $q_H \geq 1$ an upper bound on the number of queries to H by $\mathscr{A}$. Then there is an algorithm $\mathscr{A}_1$ which outputs in polynomial time an existential forgery for* id_1 *with probability*

$$\varepsilon_{\mathscr{A}_1} \geq \frac{1}{q_H + q_{\mathscr{E}} + q_{\mathscr{S}}} \cdot \left(1 - \frac{1}{q}\right) \cdot \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_{\mathscr{A}}.$$

The number of extraction and signature queries made by $\mathscr{A}_1$ is the same as for $\mathscr{A}$.

*Proof.*Without loss of generality we may assume that $\mathscr{A}$ does not repeatedly send the same query to H—the algorithm $\mathscr{A}$ can simpliy maintain a list with already queried values and received responses. The algorithm $\mathscr{A}_1$ runs a simulation of $\mathscr{A}$, using the public key P_{pub} faced by $\mathscr{A}_1$ as input to $\mathscr{A}$ and simulating all oracles for the latter. Before starting the simulation, $\mathscr{A}_1$ chooses a value $t \in \{1, \ldots, q_H + q_{\mathscr{E}} + q_{\mathscr{S}}\}$ uniformly at random, where $q_{\cdot}$ denotes a polynomial upper bound on the number of queries that $\mathscr{A}$ submits to the respective oracle. The simulation of the individual oracles is almost completely faithful:

h:This is the trivial simulation—$\mathscr{A}_1$ forwards the query to its own h-oracle and returns the answer of that oracle.

H:For the t-th query, return $H(\mathrm{id}_1) = (H_{\mathrm{id}_1,1}, H_{\mathrm{id}_1,2})$, for all other queries simply forward the query to $\mathscr{A}$'s own H-oracle.

$\mathscr{E}$:If the queried identity id has not been queried to H yet, submit id to the simulation of H and then forward id to $\mathscr{A}$'s extraction oracle $\mathscr{E}$. Otherwise forward id to $\mathscr{E}$ directly. The answer of $\mathscr{E}$ is given to $\mathscr{A}$.

$\mathscr{S}$:If the identity id in a signature query (id, m) has not been submitted to H yet, submit id to the simulation of H and then forward id to $\mathscr{A}$'s signing oracle $\mathscr{S}$. Otherwise forward id to $\mathscr{S}$ directly. The answer of $\mathscr{S}$ is given to $\mathscr{A}$.

Let $(\mathrm{id}_0, m_0, \sigma_0)$ be the output of $\mathscr{A}$, interacting with the simulated oracles. If $H(\mathrm{id}_0) = H(\mathrm{id}_1)$ and the signature created by $\mathscr{A}$ is valid, then $\mathscr{A}_1$ outputs the valid signature $(\mathrm{id}_1, m_0, \sigma_0)$. In all other cases, $\mathscr{A}_1$'s strategy failed and

Setup:Selects a secret master key $s \in \{0,\dots,q-1\}$ uniformly at random and outputs the public key $P_{\text{pub}} = s \cdot G_2 \in \mathbb{G}_2$.
Extract:On input an identity id $\in \{0,1\}^*$ and master key s, the values $r_{\text{id}} = \log_{G_2}(H_{\text{id},2}) \in \{0,\dots,q-1\}$, and $S_{\text{id}} = s \cdot H_{\text{id},1} \in \mathbb{G}_1$ are returned.
Sign:To sign $m \in \{0,1\}^*$ with secret user key $(r_{\text{id}}, S_{\text{id}})$, compute

$$\sigma = \frac{1}{r_{\text{id}} + h(m)} \cdot S_{\text{id}},$$

and output the signature $\sigma \in \mathbb{G}_1$.
Verify:A signature $\sigma \in \mathbb{G}_1$ for $m \in \{0,1\}^*$ and identity id is accepted if and only if $e(\sigma, H_{\text{id},2} + h(m) \cdot G_2) = e(H_{\text{id},1}, P_{\text{pub}})$.

Fig. 2: an identity-based signature scheme

it outputs a random guess $(\text{id}_1, m_1, \sigma_1)$ with $m_1 = 0$ and $\sigma_1 \in \mathbb{G}_1$ chosen uniformly at random.

The simulation for $\mathscr{A}$ is perfect, unless id_1 has been queried to H (explicitly or implicitly through a signature or extraction query) before the t-th query. Since id_1 is chosen uniformly at random from $\{0,1\}^k$, the probability for a simulation failure can be bounded by

$$\Pr[\mathsf{SimulationFail}] \leq \frac{q_H}{2^k}.$$

So with the simulated oracles, $\mathscr{A}$ outputs a valid forgery with probability at least

$$\Pr[(\text{id}_0, m_0, \sigma_0) \text{ is a valid signature}] \geq \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_{\mathscr{A}}. \quad (1)$$

Moreover, the probability that the output $(\text{id}_0, m_0, \sigma_0)$ of $\mathscr{A}$ is a valid signature without id_0 having been queried to H is $\leq 1/q$, as then the right-hand side of the verification equation is a random element from $\mathbb{G}_T$. In other words, we have

$$\Pr[\text{id}_0 \text{ was queried to } H | (\text{id}_0, m_0, \sigma_0) \text{ is a valid signature}] \geq 1 - \frac{1}{q}. \quad (2)$$

Since t is independently and randomly chosen, we also have

$$\Pr[\text{id}_0 \text{ was the } t\text{-th query to } H | \text{id}_0 \text{ was queried to } H \text{ and } (\text{id}_0, m_0, \sigma_0) \text{ is a valid signature}] \geq \frac{1}{q_H + q_{\mathscr{E}} + q_{\mathscr{S}}}. \quad (3)$$

Combining inequalities (1)–(3), we obtain the desired lower bound for the success probability ε_{A_1} of $\mathscr{A}_1$:

$$\varepsilon_{\mathscr{A}_1} \geq \frac{1}{q_H + q_{\mathscr{E}} + q_{\mathscr{S}}} \cdot \left(1 - \frac{1}{q}\right) \cdot \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_{\mathscr{A}}$$

□

Building on the adversary constructed in the proof of Lemma 1, one can derive an algorithm $\mathscr{A}_2$ which solves the strong Diffie-Hellman problem in the group pair underlying the proposed signature scheme.

Lemma 2.*Let $\mathscr{A}_1$ be a polynomial time adversary against the scheme in Figure 2 and* $\text{id}_1 \in \{0,1\}^k$ *chosen uniformly at random. Assume that $\mathscr{A}_1$ submits a total of at most $q_h \geq 1$ queries to h and that $\ell \geq q_h - 1$. If $\mathscr{A}_1$ succeeds with probability $\varepsilon_{\mathscr{A}_1}$ in creating a forgery for the identity* id_1, *then there is a polynomial time algorithm $\mathscr{C}$ which can solve the ℓ-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ with probability* $\varepsilon_{\mathscr{C}} \geq \frac{1}{q_h} \cdot \left(1 - \frac{1}{q}\right)^4 \cdot \left(1 - \frac{1}{q_h}\right) \cdot \varepsilon_{\mathscr{A}_1}$.

*Proof.*The algorithm $\mathscr{C}$ runs $\mathscr{A}_1$ as a subroutine, providing all inputs and simulating all oracles for $\mathscr{A}_1$. Let $([r^i \cdot G_1]_{i=0}^{\ell}, [G_2, r \cdot G_2]) \in \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2^2$ be the ℓ-SDH challenge faced by $\mathscr{C}$. To answer queries to h, the algorithm $\mathscr{C}$ chooses $(h_1,\dots,h_{q_h}) \in \{0,\dots,q-1\}^{q_h}$ uniformly at random. In addition, $\mathscr{C}$ selects $t \in \{1,\dots,q_h\}$ uniformly at random and defines the polynomial

$$f(y) = \prod_{\substack{i=1 \\ i \neq t}}^{q_h} (y + h_i) \in \mathbb{F}_q[y]$$

of degree $q_h - 1$. Note that by expanding this polynomial into standard distributive form and using the first component of the ℓ-SDH-challenge, $\mathscr{C}$ can compute $f(r) \cdot G_1$, provided that $\ell \geq q_h - 1$.

If $f(r) \cdot G_1$ happens to be $0 \in G_1$, the algorithm $\mathscr{C}$ can recover r as one of the h_i-values multiplied by $-1 \in \mathbb{F}_q$, making the ℓ-SDH problem trivial. So in the sequel we may assume that $f(r) \neq 0 \bmod q$. To create the public key, $\mathscr{C}$ chooses a master key $s \in \{0,\dots,q-1\}$ uniformly at random and hands $P_{\text{pub}} = s \cdot G_2$ to the adversary $\mathscr{A}_1$. The individual oracles for $\mathscr{A}_1$ are simulated as follows:

h:In response to the i-th new query, the value h_i is returned. By keeping track of received queries, repeated queries are answered consistently.

H:In response to a new query id $\in \{0,1\}^* \setminus \{\text{id}_1\}$, choose $\mu_{\text{id}}, r_{\text{id}} \in \{0,\dots,q-1\}$ uniformly at random and return

$$H(\text{id}) = \begin{cases} (\mu_{\text{id}} \cdot G_1, r \cdot G_2), & \text{, if id} \neq \text{id}_1 \\ (\mu_{\text{id}} \cdot f(r) \cdot G_1, r \cdot G_2) & \text{, if id} = \text{id}_1 \end{cases}.$$

(The value $r \cdot G_2$ is available as part of the ℓ-SDH-challenge.) By keeping track of received queries, repeated queries are answered consistently.

$\mathscr{E}$: The correct key extraction algorithm is executed, using the simulated h- and H-oracles.

$\mathscr{S}$:For identities other than id_1, the correct signing algorithm is executed, using the simulated h- and H-oracles. When being asked to sign a message m for id_1, then

$$s \cdot \underbrace{\mu_{\mathrm{id}_1} \cdot \frac{f(r)}{r+h(m)} \cdot G_1}_{=\frac{1}{r+h(m)} \cdot H_{\mathrm{id}_1,1}}$$

is returned, using the simulated h- and H-oracles. Unless $h(m)$ ends up being the t-th new query to h, knowing the polynomials $f(y)$ and $y+h(m)$ as well as s and μ_{id}, the algorithm $\mathscr{C}$ can compute this signature by means of the values in the ℓ-SDH-challenge.

The above simulation is perfect, provided that $\mathscr{A}_1$ does not submit an extraction query for id_1 or the t-th new query to h results from a query to $\mathscr{S}$ with identity id_1. The former is not allowed for a successful forgery for the identity id_1, and so the success probability of $\mathscr{A}_1$ in creating a forgery for id_1 when interacting with the simulated oracles is at least

$$\left(1-\frac{1}{q_h}\right) \cdot \varepsilon_{\mathscr{A}_1}.$$

Let the forgery output by $\mathscr{A}_1$ be for some message m^* with corresponding signature σ^*. If this forgery is not valid for id_1, or if m^* has not been the new query no. t to h, then $\mathscr{C}$'s strategy failed and $\mathscr{C}$ simply outputs a random guess (c, G_1) with $c \in \{0,\ldots,q-1\}$ chosen uniformly at random. To resist some trivial attacks, we assume that the master key $s \neq 0 \mod q$ and $H_{\mathrm{id}_1,1} \neq 0$, which is true with probability $(1-\frac{1}{q})^2$.

If m^* has never been queried to h, then the left-hand side of the verification equation is a random element in G_T, and we see that

$$\Pr[m^* \text{ was queried to } h | (m^*, \sigma^*) \text{ is a valid forgery for } \mathrm{id}_1] \geq 1-\frac{1}{q}.$$

Since t is independently and randomly chosen, if m^* has been queried to h, it was the new query no. t to h with probability

$$\Pr[h(m^*) = h_t | m^* \text{ was queried to } h \text{ and } (m^*, \sigma^*) \text{ is a valid forgery for } \mathrm{id}_1] \geq \frac{1}{q_h}.$$

Thus, algorithm $\mathscr{A}_1$ returns a valid forgery (m^*, σ^*) for id_1 such that $h(m^*) = h_t$ with probability at least

$$\frac{1}{q_h} \cdot \left(1-\frac{1}{q}\right)^3 \cdot \left(1-\frac{1}{q_h}\right) \cdot \varepsilon_{\mathscr{A}_1}.$$

Since $\sigma^* \in \mathbb{G}_1$, we can write $\sigma^* = d \cdot G_1$ for some value $d \in \{0,\ldots,q-1\}$, and we claim that $d = \mu_{\mathrm{id}_1} \cdot s \cdot f(r)/(r+h_t) \mod q$. From the verification condition we obtain

$$\begin{aligned} & e(d \cdot G_1, H_{\mathrm{id}_1,2} + h_t \cdot G_2) = e(H_{\mathrm{id}_1,1}, P_{\mathrm{pub}}) \\ \iff & e(G_1,G_2)^{d\cdot(r+h_t)} = e(\mu_{\mathrm{id}_1} \cdot f(r) \cdot G_1, s \cdot G_2), \\ \iff & e(G_1,G_2)^{d\cdot(r+h_t)} = e(G_1,G_2)^{\mu_{\mathrm{id}_1} \cdot s \cdot f(r)} \end{aligned}$$

and we may conclude that indeed

$$d = \frac{\mu_{\mathrm{id}_1} \cdot s \cdot f(r)}{r+h_t} \mod q.$$

With probability $1-1/q$ the condition $\mu_{\mathrm{id}_1} \cdot s \neq 0 \mod q$ is satisfied and we can write

$$(\mu_{\mathrm{id}_1} \cdot s)^{-1} \sigma_* = \frac{f(r)}{r+h_t} \cdot G_1. \tag{4}$$

By construction $y+h_t$ does not divide $f(y)$, so there exists a non-zero constant $\gamma_0 \in \mathbb{F}_q^*$ and a polynomial $\gamma(y) \in \mathbb{F}_q[y]$ of degree $\leq q_h - 1$ such that $f(y) = g(y) \cdot (y+h_t) + \gamma_0$. Consequently,

$$\frac{f(r)}{r+h_t} = \gamma(r) + \frac{\gamma_0}{(r+h_t)},$$

from which we obtain the relation

$$\frac{f(r)}{r+h_t} \cdot G_1 - \gamma(r) \cdot G_1 = \frac{\gamma_0}{r+h_t} \cdot G_1.$$

By means of Equation (4) and using the values from the ℓ-SDH challenge, $\mathscr{C}$ can evaluate the left-hand side of this equation. A final division by γ_0 yields

$$\frac{1}{r+h_t} \cdot G_1 = \gamma_0^{-1} \cdot \left((\mu_{\mathrm{id}_1} \cdot s)^{-1} \sigma_* - \gamma(r) G_1\right),$$

i. e., a solution for the ℓ-SDH problem. □

From Lemma 1 and Lemma 2 we immediately obtain the desired security reduction.[1]

Theorem 2.*Assume there is a polynomial time algorithm $\mathscr{A}$ creating an existential forgery against the scheme in Figure 2 with non-negligible probability. If $\mathscr{A}$ queries h no more than $q_h \geq 1$ times and $\ell \geq q_h - 1$, then there is a polynomial time algorithm that solves the ℓ-SDH problem in the underlying group pair with non-negligible success probability.*

4.3 Comparison with other identity-based signature schemes

In this section we compare our identity-based signature scheme with related schemes from a performance

[1] For the sake of readability we do not explicitly state the success probability and running times, which follow immediately from these lemmas and their proofs.

perspective. Table 1 compares the performance of selected schemes [1,7,9,12,16,18] in terms of computations required for signature generation and verification. Computation complexity for signing and signature verification is given in number of pairing evaluations, scalar multiplications in $\mathbb{G}_i$, and exponentiations in $\mathbb{G}_T$ (denoted as P, M, and E, respectively). Since several schemes require one less pairing for subsequent verifications after the first, we distinguish first verifications for previously unknown identities (Verify once) from cases where the result of the constant pairing evaluation of that signer has already been stored (Verify subseq.). Operations such as point additions or multiplications in $\mathbb{G}_T$ are assumed to be of negligible cost and therefore do not appear in Table 1. This also applies to the inversion necessary during the signature generation in our scheme. This inversion is in $\mathbb{Z}_q^*$ and hence considered negligible compared to the other arithmetic operations.

The proposed scheme would have several advantages if the required type of groups is available:

– The proposed scheme is the first identity-based short signature scheme: Unlike the other schemes, the signature is a single group element $\sigma \in \mathbb{G}$.
– Our scheme does not require any pairing evaluation in the signing phase, meaning that no pairing implementation is required for signature generation. In fact, only a single scalar multiplication and the inversion in $\mathbb{G}_T$ are needed, giving it a more efficient signing operation than all of the compared schemes.

Unfortunately, our scheme relies on the ℓ-SDH problem, which poses an additional hurdle for the curve selection [6,8,13]. Suitable parameters for BN curves have been proposed in [15].

Compared to prior work, the proposed scheme still appears quite attractive: The only category where our scheme is outperformed is verification for an unknown identity, where [1] needs one less pairing. However, it needs an additional exponentiation in $\mathbb{G}_T$ for any (i. e. also for subsequent) verification. In fact, only [7] and our scheme do not need to compute exponentiation in $\mathbb{G}_T$ at all. But compared to [7], our verification requires one less pairing evaluation for known identities.

Compared to [9], all exponentiations in $\mathbb{G}_T$ are replaced by scalar multiplications in G_i, with $i \in \{1,2\}$ in our scheme. Scalar multiplications for elliptic curves are usually more efficient than the exponentiation in $\mathbb{G}_T$. This is due to the fact that the extension field $\mathbb{G}_T$ has to be chosen significantly larger than the field the elliptic curve is defined on [11], especially for larger embedding degrees.

5 Conclusion

Assuming the availability of pairing-friendly groups with a discrete logarithm trapdoor, this paper shows how non-interactive identity-based three-party key establishment can be achieved. Moreover, we show how an identity-based signature scheme with short signatures can be obtained from such curves.

Acknowledgment

The authors thank Gabriel Gauthier-Shalom and David Jao for helpful discussions. Most of this work was done while PB was with Florida Atlantic University. RS was supported by the Spanish *Ministerio de Economía y Competitividad* through the project grant MTM2013-41426-R. ASC was supported by the Spanish *Ministerio de Economía y Competitividad* through the project grant MTM2013-45588-C3-1-P.

References

[1] Paulo S.L.M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, Lecture Notes in Computer Science, pages 515–532. Springer-Verlag, 2005.

[2] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21:149–177, 2008.

[3] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[4] Dan Boneh and Matthew Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003. Extended abstract appears in [3].

[5] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[6] Daniel R. L. Brown and Robert P. Gallant. The Static Diffie-Hellman Problem. Cryptology ePrint Archive: Report 2004/306, June 2005. http://eprint.iacr.org/2004/306.

[7] Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In Yvo G. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2003.

[8] Jung Hee Cheon. Security Analysis of the Strong Diffie-Hellman Problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[9] Sherman S.M. Chow, Siu-Ming Yiu, Lucas C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In Jong-In Lim and Dong-Hoon Lee, editors, *Information Security and Cryptology –*

Table 1: Performance comparison of popular identity-based signature schemes, where P denotes the number of pairing evaluations, M denotes scalar multiplications in $\mathbb{G}_1$ or $\mathbb{G}_2$, and E denotes exponentiations in $\mathbb{G}_T$

	Paterson [16]	Cha Cheon [7]	CYHC [9]	BLM [1]	Hess [12]	here
Sign	3M	2M	1E+1M	1E+1M	1M+1E	1M
Verify once	2E+2P	1M+2P	1E+2P	1E+1M+1P	1E+2P	1M+2P
Verify subseq.	2E+1P	1M+2P	1E+1P	1E+1M+1P	1E+1P	1M+1P
Signature	$\mathbb{G}\times\mathbb{G}$	$\mathbb{G}\times\mathbb{G}$	$\mathbb{F}_q^*\times\mathbb{G}$	$\mathbb{F}_q^*\times\mathbb{G}_1$	$\mathbb{F}_q^*\times\mathbb{G}$	$\mathbb{G}_1$

ICISC 2003, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer-Verlag, 2004.

[10] Alexander W. Dent and Steven D. Galbraith. Hidden Pairings and Trapdoor DDH Groups. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 436–451. Springer-Verlag, 2006.

[11] David Freeman, Michael Scott, and Edlyn Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[12] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2003.

[13] David Jao and Kayo Yoshida. Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem. In Hovav Shacham and Brent Waters, editors, *Paring-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2009.

[14] Pascal Paillier. Trapdooring Discrete Logarithms on Elliptic Curves over Rings. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 573–584. Springer-Verlag, 2000.

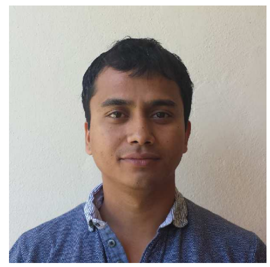
[15] Cheol-Min Park and Hyang-Sook Lee. Pairing-Friendly Curves with Minimal Security Loss by Cheon's Algorithm. *ETRI Journal*, 33(4):656–659, August 2011.

[16] Kenneth G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[17] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, 52:219–241, 2009.

[18] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems Based on Pairing. In *The 2000 Symposium on Cryptography and Information Security (SCIS)*, volume C20, 2000.

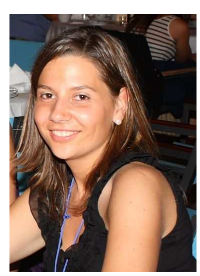
[19] Edlyn Teske. An Elliptic Curve Trapdoor System. *Journal of Cryptology*, 19(1):115–133, 2006.



Parshuram Budhathoki is a Faculty at the Department of Mathematics of Salt Lake Community College (SLCC). He received his doctoral degree from Florida Atlantic University (FAU), USA.

Thomas Eisenbarth is assistant professor at the Department of Electrical & Computer Engineering at WPI. His research interests are in applied cryptography and physical attacks. He received his doctoral degree from Ruhr-Universität Bochum, Germany.

Rainer Steinwandt is professor and chair at the Department of Mathematical Sciences of Florida Atlantic University (FAU) and Director of FAU's Center for Cryptology and Information Security. He received his doctoral degree from Univ. Karlsruhe, Germany.



Adriana Suárez Corona is assistant professor at the Department of Mathematics and a member of the Research Institute of Applied Sciences in Cybersecurity (RIASC) of Universidad de León, Spain. She received her doctoral degree from Universidad de Oviedo, Spain.