Strong Forward Security in Identity-Based Signcryption

Madeline González Muñiz[1] and Peeter Laud[2]

[1] Cybernetica AS
Akadeemia tee 21, Tallinn, Estonia
[2] Cybernetica AS
Aleksandri 8a, Tartu, Estonia
{madeline,peeter}@cyber.ee

Abstract. Due to the possibility of key exposure, forward security in encryption and signing has been well studied, especially in the identity-based setting where an entity's public key is that entity's name. From an efficiency point of view, one would like to use the signcryption primitive and have the best of both worlds. However, *strong* forward security, where the adversary cannot signcrypt in sender's name nor designcrypt in receiver's name for past time periods even if it has the secrets of both, requires periodic updating of the secret keys of the users. This is an improvement over signcryption schemes that only protect against designcrypting in the past. In this paper, we propose the first ever strong forward secure identity-based signcryption scheme which has public ciphertext verifiability and a third-party verification protocol.

Keywords: forward security, signcryption, pairing-based cryptography, identity-based cryptography

1 Introduction

To achieve both authenticity and confidentiality, usually the sign-then-encrypt paradigm is employed. Proposed by Zheng [28], the aim of signcryption is to combine encryption and signing into one efficient primitive. As expected, the goals of signcryption are confidentiality and unforgeability. In addition, if the current state of the user is compromised, one would like to have assurances that this information cannot be used to break the security of the system for past periods. Several forms of forward security for signcryption schemes have been discussed in the literature so far. It is often defined to mean that messages signcrypted in the past cannot be designcrypted even if sender's private key is exposed; we will refer to this as *partial forward secrecy*. If past messages signcrypted cannot be designcrypted even with access to both sender's and receiver's private keys, then we say the signcryption scheme is *forward secret*. If in addition to being forward secret, the adversary cannot forge signcryptions in the past, we say that the signcryption scheme is *strong forward secure*.

Non-repudiation is another a desirable property of signcryption schemes, also coming in several flavors. If the origin of a signcryption can be checked

by everybody then we say that the scheme has *public ciphertext verifiability*. A stronger property is *third party verification*, where the receiver of a signcryption can convince everybody else that it came from a particular sender and contains a particular *plaintext*.

In identity-based cryptography, the names of the parties serve as their public keys. There is a trusted party, the key generation center (KGC), that is capable of computing the secret key corresponding to any public key and uses this capability to give to each party its secret key. Secret keys are bound to public keys through the *master public key* known to everybody. The master public key, together with the *master secret key* are generated by KGC.

Our contribution In this paper, we propose the first identity-based strongly forward secure signcryption scheme. We show the insider security [1] of our scheme in the random oracle model. Additionally, third party verifiability can be added to our scheme. The random oracle model [3], while less theoretically desirable, allows us to obtain efficient constructions.

In order to achieve strong forward security, one would like for the signcryption scheme to be unforgeable and retain message secrecy (past signcryptions cannot be designcrypted) for all periods prior to key exposure; thus, the secret keys must be updated. We use the mechanism of *binary tree encryption* [5], first adapted to signature schemes by Kang et al. [17] for updating the keys. In this paper, we begin in Section 2 by reviewing the existing signcryption schemes that purport to have forward security, as well as introducing the necessary preliminaries about the computationally hard problems our scheme's security will be based on. In Section 3, we propose a definition for signcryption schemes in the binary tree setting, together with definitions for their strong forward security. The definitions are natural adaptations of definitions from earlier work, thus they can be assumed to be the right ones. We propose a binary tree signcryption scheme and show that it is strongly forward secure. We show how to transform the binary tree based signcryption scheme to a plain strongly forward secure signcryption scheme with evolving secret keys in Sec. 4. Last, a third party message verification protocol is provided in Section 5.

Further related work. Using bilinear maps, Barreto et al. proposed efficient and provably-secure signatures and signcryptions [2]. An efficient signcryption scheme with public ciphertext verifiability was proposed by Chow et al. [7]. In [29], forward secrecy is addressed for low-power devices. Also related are identity-based key exchanged protocols, many of which have been proposed in the literature ([9,14,24,6] to name a few), and identity-based signing [21,15,19,22] and encryption [4]. Combined (identity-based) public key schemes in which parties use a public key encryption scheme and a signature scheme with a single public key/private key pair are also worth mentioning here [12]. The only proposed strong forward secure signcryption scheme that we are aware of is by Yin et al. [27].

In many signcryption schemes, there is a general pattern employed in three parts. First, a key transport protocol takes place via encrypting an ephemeral key

under the receiver's public key. Second, a symmetric encryption protocol using the ephemeral key takes place. Last, the ciphertext output by the symmetric encryption protocol is signed by the sender. In some cases, the signature on the ciphertext can be verified by a third party from the signcryption (we discuss the role of public verifiability). For a general discussion on the security of hybrid signcryption schemes see [8].

2 Preliminaries and Definitions

We write $x \xleftarrow{\$} X$ to denote that the value x is uniformly chosen from the set X. We also write $x \xleftarrow{\$} \mathcal{A}(\ldots)$ to denote that x is the result of running a probabilistic algorithm $\mathcal{A}$ with certain arguments. We write $\mathcal{A}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot),\cdots}$ to denote that the algorithm $\mathcal{A}$ has access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$.

Our constructions build on bilinear pairings commonly used in identity-based schemes [4,18][1]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q (represented additively and multiplicatively, respectively).

Definition 1 (Bilinear Map). *A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ have prime order q and satisfy the following properties:*

1. Bilinearity: $\forall P_1, P_2 \in \mathbb{G}_1$, $\forall \alpha, \beta \in \mathbb{Z}_q^*$, $\hat{e}(\alpha P_1, \beta P_2) = \hat{e}(P_1, P_2)^{\alpha\beta}$.
2. Non-Degeneracy: *For any $P_1 \in \mathbb{G}_1$, $\hat{e}(P_1, P_2) = 1$ for all $P_2 \in \mathbb{G}_1$ iff $P_1 = \mathcal{O}$.*
3. Computability: *There exists an efficient algorithm to compute $\hat{e}(P_1, P_2)$ from $P_1, P_2 \in \mathbb{G}_1$.*

The security proofs of our constructions depend on the hardness of certain computational problems involving bilinear pairings. Namely, the *bilinear decisional Diffie-Hellman* (BDDH) problem states that it is infeasible to distinguish tuples of the form $(P, \alpha P, \beta P, \gamma P, \hat{e}(P, P)^{\alpha\beta\gamma})$ from tuples $(P, \alpha P, \beta P, \gamma P, z)$. Here P is a generator of $\mathbb{G}_1$, α, β, γ are uniformly chosen from $\mathbb{Z}_q^*$, and z is uniformly chosen from $\mathbb{G}_2$. The hardness of BDDH problem implies that the computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ — given $P, \alpha P, \beta P \in \mathbb{G}_1$, find $\alpha\beta P$ — is hard, too.

In order to improve efficiency over "Sign-then-Encrypt" and "Encrypt-then-Sign" paradigms, hybrid signcryption schemes often involve a symmetric encryption scheme. An established notion of security, we recall that indistinguishability under chosen plaintext attack (IND-CPA) allows the adversary to get encryptions of adaptively chosen messages [10]. We will use a symmetric encryption scheme as part of our construction.

Definition 2 (Symmetric IND-CPA). *Let the triple $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ with security parameter μ be a symmetric key encryption scheme and $\mathcal{A}$ a probabilistic polynomial time algorithm. Consider the following game between the adversary $\mathcal{A}$ and the environment.*

[1] Although we do not focus on specific pairings, we assume that they are carefully chosen as to avoid known attacks such as [25].

1. *The environment computes a key* $K \xleftarrow{\$} \mathcal{K}'(1^\mu)$.
2. $\mathcal{A}$ *can submit messages* m *to the environment. The environment answers with* $\mathcal{E}'_K(m)$.
3. *Eventually,* $\mathcal{A}$ *selects messages* m_0 *and* m_1 *of equal length and submits them to the environment.*
4. *The environment selects* $b \xleftarrow{\$} \{0,1\}$ *and returns* ${\mathcal{E}'}_K(m_b)$ *to* $\mathcal{A}$.

In the end, $\mathcal{A}$ *must guess* b*. Its advantage is the difference between its success probability and* $1/2$*. The symmetric encryption scheme is* IND-CPA *secure if no probabilistic polynomial-time adversary has non-negligible advantage.*

Identity-based *public key* cryptosystems enable the computation of a public key for a user given the scheme parameters and a string identifying the user. A key generation center (KGC) computes private keys from a master secret key and distributes these to the users in the scheme. Signcryption schemes in the identity-based setting consist of the four algorithms that follow. The first two of them are executed by KGC, the last two by the users of the system.

$\mathcal{G}(1^\eta)$ Upon input 1^k the KGC generates the system's master secret key msk and master public key mpk.

$\mathcal{K}(msk, \mathsf{ID})$ outputs the secret key sk^{ID} corresponding to the identity ID.

$\mathcal{S}^c(mpk, sk^{\mathsf{A}}, \mathsf{ID_B}, M)$ computes the signcryption of the message M from the party A to the party B.

$\mathcal{D}^c(mpk, sk^{\mathsf{B}}, \mathsf{ID_A}, \sigma)$ designcrypts the signcryption σ sent to party B, and verifies that it came from party A. It outputs the plaintext corresponding to σ or $\perp$ if an error occurred.

The consistency requirement states $m = \mathcal{D}^c(mpk, sk^{\mathsf{B}}, \mathsf{ID_A}, \mathcal{S}^c(mpk, sk^{\mathsf{A}}, \mathsf{B}, m))$, for any message m and identities A, B.

First proposed by An et al. [1], insider security assumes that the adversary $\mathcal{A}$ is a legal user of the system (either a sender or receiver). An attack against a signcryption scheme is either an attack against confidentiality or authenticity. When attacking authenticity, the adversary has access to all private keys except the sender's, and when attacking indistinguishability, the adversary has access to all private keys except the receiver's. Additionally, the adversary has access to signcryption oracles using any identity.

To achieve strong forward security, the private keys of the parties (but not the public keys) must be updated during the lifetime of the scheme. To be efficient, the signing and encryption must be meaningfully related. That is, one would like the ability to sign using the same setup from a forward-secure encryption scheme. This allows us to build a signcryption protocol that yields the forward security in the strong sense that is desired. In Table 1, we show a comparison of what previous schemes have achieved as compared to our signcryption scheme BTSC which will be introduced in Section 3.

We note that Toorani and Shirazi [26] point out some flaws in the scheme by Hwang et al. [16] mentioned in Table 1. As the secret key used to encrypt is never updated in the scheme by Yin et al. [27], the scheme does not have any

Table 1. Security comparison of signcryption schemes

Signcryption Scheme	BTSC	[7]	[16]	[20]	[27]
Partial Forward Secrecy	✓	✓	✓	✓	
Public Ciphertext Verifiability	✓	✓	✓	✓	
Forward Secrecy	✓				
Strong Forward Security	✓				
Third Party Verification	✓	✓			

forward secrecy. However, the scheme uses the forward-secure signature scheme by Kang et al. [17] to achieve unforgeability in past time periods. In addition, the table shows if an explicit algorithm for third party verification is given.

3 Forward Secure Signcryption in the Binary Tree Setting

In [5], Canetti et al. propose a forward-secure encryption scheme in the standard model; we assume familiarity with the security of their construction. The concept is similar to the forward-secrecy property in signcryption in that past encryptions cannot be decrypted with the compromised current state of the user (if one wishes to perform decryptions of past ciphertexts, the initial seed used to create the secret keys must be stored in a separate secure device). The proposed protocols use a binary tree and each node (not just the leaves) represents a time period and has a corresponding secret key. A binary tree scheme is transformed to a forward secure scheme by traversing the nodes of the tree in a certain order (e.g. depth-first). We adapt their binary tree construction in this section to signcryption. However, we use the identity-based setup in which a KGC creates the parameters and keys rather than each individual user.

In the following, we address the nodes of a binary tree at level $t \leq \ell$ with bit-strings of length t where ℓ is the depth of the tree. The address of the root node is the empty string ε. The *children* of the node (with the address) ω are the nodes $\omega 0, \omega 1$. For these nodes, ω is the *parent* node. For a bit-string ω of length t, we let $\omega|_i$ denote its i-bit prefix. The *sibling* of a non-root node ω is the other child of its parent; we denote it with $\overline{\omega}$. The bit-strings ω and $\overline{\omega}$ have the same length and differ only in their last bit.

An *identity-based binary tree signcryption scheme* has the same algorithms as identity-based signcryption schemes (Sec. 2). It has an additional algorithm $\mathcal{R}$ for updating the secret keys. This algorithm is executed by the users at the end of time periods.

Definition 3 (BTSC). *An identity-based binary tree digital signcryption scheme (BTSC) is a 5-tuple of probabilistic polynomial-time algorithms* $(\mathcal{G}, \mathcal{K}, \mathcal{R}, \mathcal{S}^c, \mathcal{D}^c)$*, where*

$\mathcal{G}(1^\eta, \ell, \mathsf{params})$ *generates the master secret key msk, the master public key mpk, and possibly other parameters.*

$\mathcal{K}(msk, \mathsf{ID})$ *outputs the secret key* $sk_{\varepsilon}^{\mathsf{ID}}$ *for the user* ID*, corresponding to the root of the binary tree.*

$\mathcal{R}(mpk, \omega, sk_{\omega}^{\mathsf{ID}})$ *receives the address of the node* ω *of length strictly less than* ℓ *and the secret key* $sk_{\omega}^{\mathsf{ID}}$ *corresponding to this node. It returns the secret keys* $sk_{\omega 0}^{\mathsf{ID}}$ *and* $sk_{\omega 1}^{\mathsf{ID}}$ *corresponding to the children of this node.*

$\mathcal{S}^c(mpk, \omega, sk_{\omega}^{\mathsf{A}}, \mathsf{B}, M)$ *computes the signcryption of the message* M *from the party* A *to the party* B*, corresponding to the "time period" (or node)* ω*.*

$\mathcal{D}^c(mpk, \omega, sk_{\omega}^{\mathsf{B}}, \mathsf{A}, \sigma)$ *designcrypts the signcryption* σ *sent to party* B*, and verifies that it came from party* A*. It outputs the plaintext corresponding to* σ *or* $\perp$ *if an error occurred.*

The algorithms must satisfy the consistency requirement: for any node ω *(of length polynomial in* η*), keys correctly generated for this node* sk_{ω}^{A} *and* sk_{ω}^{B} *output by* $\mathcal{K}(msk, \mathsf{A})$ *and* $\mathcal{K}(msk, \mathsf{B})$*, respectively, and any message* M*, we have the correct designcryption* $M = \mathcal{D}^c(mpk, \omega, sk_{\omega}^{\mathsf{B}}, \mathsf{A}, \mathcal{S}^c(mpk, \omega, sk_{\omega}^{\mathsf{A}}, \mathsf{B}, M))$*.*

A signcryption is secure if it provides confidentiality and authenticity. It is forward secure if an attacker with access to the secrets of current and future time periods cannot compromise the security of communications in the past. In the binary tree based setting, the time can be thought as flowing downwards from the root node, with different branches being independent of each other. When the adversary answers to a challenge at a certain node (time moment), it is allowed to access the secrets in every node except the one under attack and its ancestors. Similarly to [5], we force the adversary to first commit to the node where it will attempt the challenge. This will still allow us to obtain a forward secure scheme in the usual (non binary-tree) sense. The confidentiality and authenticity definitions for the BTSC are the following. The first of them is the adaptation of SN-CPA [5] to signcryptions and the identity-based setting. The second is the adaptation of CMA security to the identity- and binary tree based setting. Recall that the insider security of signcryption schemes was expressible in two separate definitions.

Definition 4 (IB-BT-CPA). *Let* $(\mathcal{G}, \mathcal{K}, \mathcal{R}, \mathcal{S}^c, \mathcal{D}^c)$ *be a BTSC. Consider the following game between the adversary* $\mathcal{A}$ *and the environment.*

1. *The adversary gives* $\omega^* \in \{0,1\}^*$ *to the environment.*
2. *The environment executes* $\mathcal{G}(1^\eta, \ell, \mathsf{params})$*, obtains msk and mpk, and gives the latter to the adversary. The environment also selects a random bit* b*.*
   The environment keeps a database mapping pairs (ID, ω) *to secret keys* $sk_{\omega}^{\mathsf{ID}}$*. Initially the database is empty. Whenever a key* $sk_{\omega}^{\mathsf{ID}}$ *is needed, it is taken from the database. If it is unavailable, and* $\omega = \varepsilon$*, the algorithm* $\mathcal{K}(msk, \mathsf{ID})$ *is invoked and the result is stored as* $sk_{\varepsilon}^{\mathsf{ID}}$*. If* $\omega \neq \varepsilon$ *and* $|\omega| < \ell$*, then the key* $sk_{\omega|_{|\omega|-1}}^{\mathsf{ID}}$ *is taken from the database, and* $\mathcal{R}$ *invoked to produce both* $sk_{\omega}^{\mathsf{ID}}$ *and* $sk_{\overline{\omega}}^{\mathsf{ID}}$*, which are then added to the database.*
3. *The adversary can perform queries* $\mathsf{breakin}(\mathsf{ID}, \omega)$*. In response, the environment answers with* $sk_{\omega}^{\mathsf{ID}}$*.*

4. *At a certain point, the adversary issues a query* $\mathsf{challenge}(\mathsf{A}, \mathsf{B}, M_0, M_1)$. *The query is accepted only if the adversary has made no queries* $\mathsf{breakin}(\mathsf{B}, \omega')$, *where* ω' *equals or is a prefix of* ω^*. *If it is accepted, it is answered with* $\mathcal{S}^c(mpk, \omega^*, sk^{\mathsf{A}}_{\omega^*}, \mathsf{B}, M_b)$.
5. *The adversary can continue issuing* breakin*-queries, but not for the identity* B *and the nodes* ω' *that are prefixes of* ω^* *in the* challenge*-query.*

In the end, the adversary must guess b. *Its advantage is the difference between its success probability and* $1/2$. *The BTSC is IB-BT-CPA-secure if no probabilistic polynomial-time adversary has non-negligible advantage.*

Definition 5 (IB-BT-CMA). *Let* $(\mathcal{G}, \mathcal{K}, \mathcal{R}, \mathcal{S}^c, \mathcal{D}^c)$ *be a BTSC. Consider the following game between the adversary* $\mathcal{A}$ *and the environment.*

1. *The adversary gives* $\omega^* \in \{0,1\}^*$ *to the environment.*
2. *The environment executes* $\mathcal{G}(1^\eta, \ell, \mathsf{params})$, *obtains msk and mpk, and gives the latter to the adversary. The environment also selects a random bit* b. *The environment keeps the same database of keys as in Def. 4.*
3. *The adversary can perform queries* $\mathsf{breakin}(\mathsf{ID}, \omega)$. *In response, the environment answers with* $sk^{\mathsf{ID}}_{\omega}$.
4. *The adversary can perform queries* $\mathsf{sc}(\omega, \mathsf{A}, \mathsf{B}, M)$. *The environment answers with* $\mathcal{S}^c(mpk, \omega, sk^{\mathsf{A}}_{\omega}, \mathsf{B}, M)$.

In the end, the adversary $\mathcal{A}$ *produces a message* M, *a signcryption* σ, *and two identities* A, B. *The adversary* wins *if*

- $\mathcal{D}^c(mpk, \omega^*, sk^{\mathsf{B}}_{\omega^*}, \mathsf{A}, \sigma) = M$;
- $\mathcal{A}$ *did not issue any query* $\mathsf{breakin}(\mathsf{A}, \omega')$, *where* ω' *is a prefix of* ω^*;
- $\mathcal{A}$ *did not issue the query* $\mathsf{sc}(\omega^*, \mathsf{A}, \mathsf{B}, M)$.

The BTSC is IB-BT-CMA-secure if no probabilistic polynomial-time adversary has non-negligible winning probability.

We say that a BTSC is **strongly BT-forward secure** if it is both IB-BT-CPA- and IB-BT-CMA-secure. Next, we present a scheme that achieves strong BT-forward security. Let $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ with security parameter μ be an IND-CPA-secure symmetric encryption scheme as defined in Section 2. The scheme is presented in Fig. 1.

A straightforward computation shows that the scheme in Fig. 1 satisfies the consistency requirements of Def. 3 as shown in Appendix A. Next, we show that it is strongly BT-forward secure.

Remark 1. Note in Figure 1 that $\hat{e}(P_{pub}, Q_{\mathsf{A}})$ and $\hat{e}(R_i, H(\omega|_i))$ for $i = 1, \ldots, t$ in algorithm $\mathcal{D}^c$ may be stored and reused by B to improve efficiency. Furthermore, the symmetric encryption scheme ensures efficient exchange of large amounts of data as opposed to simply using "Encrypt-then-Sign."

Theorem 1. *If the symmetric encryption scheme* $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ *is secure in the sense of symmetric* IND-CPA *and the BDDH assumption in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *holds, then the BTSC scheme in Figure 1 is insider-secure in the sense of IB-BT-CPA.*

$\mathcal{G}(1^\eta, \ell, 1^\mu)$ chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, a generator P of $\mathbb{G}_1$, and $\alpha \in \mathbb{Z}_q$. It computes $P_{\text{pub}} = \alpha P$ and chooses the hash functions $H, \bar{H} : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : (\{0,1\}^*)^3 \times \mathbb{G}_1^* \times \mathbb{G}_2^2 \to \mathbb{Z}_q$, and $H_2 : \mathbb{G}_2 \to \{0,1\}^\mu$. (considered as random oracles). The *master public key mpk* is (the descriptions of) $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{\text{pub}}, H, \bar{H}, H_1, H_2)$. The *master secret key* is $msk = (mpk, \alpha)$. The set of possible messages in the scheme is $\{0,1\}^*$.

$\mathcal{K}(msk, \mathsf{ID})$ computes $sk_\varepsilon^{\mathsf{ID}} = (D_\varepsilon^{\mathsf{ID}}) = \alpha Q_{\mathsf{ID}}$ where $Q_{\mathsf{ID}} = \bar{H}(\mathsf{ID})$.

$\mathcal{R}(mpk, \omega, sk_\omega^{\mathsf{ID}})$ works as follows:

1. Parse sk_ω^{ID} as $(R_{\omega|_1}^{\mathsf{ID}}, R_{\omega|_2}^{\mathsf{ID}}, \ldots, R_{\omega|_t}^{\mathsf{ID}}, D_\omega^{\mathsf{ID}})$, where $t = |\omega|$.
2. Choose $\rho_{\omega b}^{\mathsf{ID}} \xleftarrow{\$} \mathbb{Z}_q$, set $R_{\omega b}^{\mathsf{ID}} = \rho_{\omega b}^{\mathsf{ID}} P$, and $D_{\omega b}^{\mathsf{ID}} = D_\omega^{\mathsf{ID}} + \rho_{\omega b}^{\mathsf{ID}} H(\omega b)$ for $b \in \{0,1\}$.
3. Output $sk_{\omega b}^{\mathsf{ID}} = (R_{\omega|_1}^{\mathsf{ID}}, R_{\omega|_2}^{\mathsf{ID}}, \ldots, R_{\omega|_t}^{\mathsf{ID}}, R_{\omega b}^{\mathsf{ID}}, D_{\omega b}^{\mathsf{ID}})$ for $b \in \{0,1\}$.

$\mathcal{S}^c(mpk, \omega, sk_\omega^{\mathsf{A}}, \mathsf{B}, M)$ works as follows:

1. Compute $Q_{\mathsf{B}} = \bar{H}(\mathsf{B}) \in \mathbb{G}_1$.
2. Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$, $P_1 \xleftarrow{\$} \mathbb{G}_1 \backslash \{0\}$ and compute $K := H_2\left(\hat{e}(P_{pub}, Q_{\mathsf{B}})^{x^{-1}}\right)$.
3. Set $R := (R_{\omega|_1}^{\mathsf{A}}, \ldots, R_{\omega|_t}^{\mathsf{A}})$ and $T := \hat{e}(P_1, P)^x$.
4. Set $U := (x^{-1}P, x^{-1}H(\omega|_1), \ldots, x^{-1}H(\omega))$.
5. Compute $c := {\mathcal{E}'}_K(M)$, $r := H_1(\omega, \mathsf{A}, \mathsf{B}, R, T, c)$, and $S := rD_\omega^{\mathsf{A}} + xP_1$.
6. The signcryption is $\sigma = (R, S, r, U, c)$.

$\mathcal{D}^c(mpk, \omega, sk_\omega^{\mathsf{B}}, \mathsf{A}, (R, S, r, U, c))$ works as follows:

1. Parse R as $(R_1, \ldots, R_t)$ and U as $(U_0, \ldots, U_t)$.
2. Compute $K := H_2\left(\dfrac{\hat{e}(U_0, D_\omega^{\mathsf{B}})}{\prod_{i=1}^t \hat{e}(R_{\omega|_i}^{\mathsf{B}}, U_i)}\right)$.
3. Let $Q_{\mathsf{A}} = \bar{H}(\mathsf{A})$. Set $T = \hat{e}(S, P)\left(\hat{e}(P_{pub}, Q_{\mathsf{A}}) \prod_{i=1}^t \hat{e}(R_i, H(\omega|_i))\right)^{-r}$. If $r = H_1(c, \omega, \mathsf{A}, \mathsf{B}, R, T)$, then set $M := {\mathcal{D}'}_K(c)$. Otherwise set $M = \perp$.

Fig. 1. Binary Tree Signcryption Scheme (BTSC)

Proof. Let $\mathcal{A}'$ be a distinguisher for the BTSC scheme that has a non-negligible advantage $p(\eta)$ in the IB-BT-CPA game. $\mathcal{A}$ simulates the KGC and all random oracle queries. Since $\mathcal{A}'$ has access to $\mathsf{breakin}(\mathsf{ID}, \omega)$, it can signcrypt messages on its own. At some point, $\mathcal{A}'$ outputs two messages m_0 and m_1 and identities A and B, where one of the messages will be signcrypted uniformly at random from A to B, and $\mathsf{breakin}(\mathsf{B}, \omega)$ has not been queried. Distinguisher $\mathcal{A}$ attacking the symmetric encryption scheme selects the same two messages. Once $\mathcal{A}$ receives ${\mathcal{E}'}_K(m_b)$, it sets $c := {\mathcal{E}'}_K(m_b)$, and computes S, r, U using a randomly generated x. Then, $\mathcal{A}$ forwards (R, S, r, U, c) to $\mathcal{A}'$. As $\mathcal{A}$ outputs the same bit returned by $\mathcal{A}'$, they have the same advantage $p(\eta)$.

We note that if $\mathcal{A}'$ can distinguish between the key K generated in the symmetric $\mathsf{IND\text{-}CPA}$ challenge from $H_2(\hat{e}(P_{pub}, Q_{\mathsf{B}})^{x^{-1}})$ using the x chosen by $\mathcal{A}$, then this would contradict the security of the Binary Tree Encryption scheme [5] because $\mathcal{A}$ would break the BDDH assumption as we now prove.

Let $\mathcal{A}'$ be a distinguisher for the BTSC scheme that has a non-negligible advantage $p(\eta)$ in the IB-BT-CPA game. Suppose that it makes at most $q_I(\eta)$ queries to the oracle $\bar{H}$ (also including the invocations of $\bar{H}$ through $\mathsf{breakin}$- and

challenge-queries). We show how to construct a distinguisher $\mathcal{A}$ for the BDDH problem. Let an instance $(P, \alpha P, \beta P, \gamma P, z)$ of the BDDH problem be given. $\mathcal{A}$ must guess whether $z = \hat{e}(P,P)^{\alpha\beta\gamma}$. The algorithm $\mathcal{A}$ will act as an environment for $\mathcal{A}'$ in the following manner.

1. $\mathcal{A}$ receives the value ω^* from $\mathcal{A}'$. Let $t = |\omega^*|$.
2. $\mathcal{A}$ selects $P_{\text{pub}} = \alpha P$ and sets $mpk = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{\text{pub}}, H, \bar{H}, H_1, H_2)$ where H, $\bar{H}$, H_1 and H_2 are random oracles also answered by $\mathcal{A}$. It generates a random number $\mathsf{idx} \xleftarrow{\$} \{1, \ldots, q_I\}$ and a random bit $b \xleftarrow{\$} \{0,1\}$.
3. $\mathcal{A}$ randomly chooses $\chi_1, \lambda_1, \varphi_1, \ldots, \chi_t, \lambda_t, \varphi_t, \lambda^0_{t+1}, \varphi^0_{t+1}, \lambda^1_{t+1}, \varphi^1_{t+1}$ from $\mathbb{Z}_q$. It sets $H(\omega^*|_i) = \chi_i P$ and $H(\overline{\omega^*|_i}) = \lambda_i P - \frac{1}{\varphi_i}\beta P$ for $i \in \{1, \ldots, t\}$. It also sets $H(\omega^* 0) = \lambda^0_{t+1} P - \frac{1}{\varphi^0_{t+1}}\beta P$ and $H(\omega^* 1) = \lambda^1_{t+1} P - \frac{1}{\varphi^1_{t+1}}\beta P$. Clearly, this means that the values of H in those points are chosen uniformly randomly from $\mathbb{G}_1$, because P is a generator. Also, the values are independent of each other because P has a different multiplier in each expression.
4. Whenever a H-query x is made and $H(x)$ has not yet been fixed, a random element of $\mathbb{G}_1$ is generated, stored, and returned.
5. Whenever (except for the next item) a $\bar{H}$-query x is made by $\mathcal{A}'$ or by $\mathcal{A}$ itself, and that query has not been made before, it generates a random exponent δ, stores (x, δ), and answers with δP. Denote δ with $\log \bar{H}(x)$.
6. Let B be the argument to the idx-th $\bar{H}$-query. The distinguisher $\mathcal{A}$ guesses that $\mathcal{A}'$ will attempt to guess b using a signcryption addressed to B. The query is answered with βP by $\mathcal{A}$.
7. After fixing $\bar{H}(\mathsf{B})$, the distinguisher $\mathcal{A}$ fixes the secret keys sk^{B}_{ω} for $\omega = \overline{\omega^*|_i}$ $(1 \le i \le t)$ and $\omega = \omega^* b^*$ $(b^* \in \{0,1\})$. It selects $R^{\mathsf{B}}_{\omega^*|_i} \xleftarrow{\$} \mathbb{G}_1$, sets $R^{\mathsf{B}}_{\overline{\omega^*|_i}} = \varphi_i \alpha P$, and $D^{\mathsf{B}}_{\overline{\omega^*|_i}} = \lambda_i \varphi_i \alpha P + \sum_{j=1}^{i-1} \chi_j R^{\mathsf{B}}_{\overline{\omega^*|_j}}$ for $1 \le i \le t$. It also sets $R^{\mathsf{B}}_{\omega^* b^*} = \varphi^{b^*}_{t+1} \alpha P$ and $D^{\mathsf{B}}_{\omega^* b^*} = \lambda^{b^*}_{t+1} \varphi^{b^*}_{t+1} \alpha P + \sum_{j=1}^{t} \chi_j R^{\mathsf{B}}_{\overline{\omega^*|_j}}$ for $b^* \in \{0,1\}$. By the arguments of Canetti et al. [5], these keys have the correct distribution.
8. Whenever $\mathcal{A}'$ makes a query $\mathsf{breakin}(\mathsf{ID}, \omega)$, these are reduced to a number of invocations of the algorithms $\mathcal{K}$ and $\mathcal{R}$ according to Def. 4. The result of the invocation $\mathcal{K}(msk, \mathsf{ID})$ can be simulated by returning $(\log \bar{H}(\mathsf{ID}))\alpha P$. The inputs to the algorithm $\mathcal{R}$ are available to $\mathcal{A}$. The query cannot be answered if $\mathsf{ID} = \mathsf{B}$ and ω is a prefix of ω^*. If this happens, $\mathcal{A}$ gives up — it stops and makes a random guess.
9. The query $\mathsf{challenge}(\mathsf{A}, \mathsf{B}, M_0, M_1)$ is answered as follows. Obtain $sk^{\mathsf{A}}_{\omega^*}$. Let $K = H_2(z)$. Set $R = (R^{\mathsf{A}}_{\omega|_1}, \ldots, R^{\mathsf{A}}_{\omega|_t})$. Pick $T_0 \xleftarrow{\$} \mathbb{G}_1$. Set $T = \hat{e}(T_0, P)$ and $U = (\gamma P, \chi_1 \gamma P, \ldots, \chi_t \gamma P)$. Compute $c = \mathcal{E}'_K(M_b)$, $r = H_1(\omega, \mathsf{A}, \mathsf{B}, R, T, c)$, $S = r D^{\mathsf{A}}_{\omega^*} + T_0$ and return (R, S, r, U, c). If the second argument of the query is not B, then give up.

We see that the secret γ is used as x^{-1} in answering the challenge-query. If $z = \hat{e}(P,P)^{\alpha\beta\gamma}$ then the symmetric key K is computed as in Fig. 1. If z is random then the key K is also random. In Fig. 1, the secret x is also used

to compute T and S, but always inside the subexpression xP_1, where P_1 is a random element of $\mathbb{G}_1$. The value xP_1 is also random and is simulated by T_0.

In the end, $\mathcal{A}'$ guesses the bit b. If $z = \hat{e}(P,P)^{\alpha\beta\gamma}$ then the advantage of $\mathcal{A}'$ is $p(\eta)$. If z is random, then c is also random and the view of $\mathcal{A}'$ contains no information about b. Hence $\mathcal{A}$ guesses that $z = \hat{e}(P,P)^{\alpha\beta\gamma}$, if $\mathcal{A}'$ guesses the bit b correctly, and guesses that z is random if $\mathcal{A}'$ is wrong. The advantage of $\mathcal{A}$ is $p(\eta)/2q_I(\eta)$. □

Remark 2. By multiplying each element of the set U by a fixed constant, the key K becomes malleable, and thus, chosen ciphertext security is not guaranteed. Canetti et al. sketched some ways that chosen ciphertext security may be achieved (in Section 3.2 of [5]).

Theorem 2. *If the CDH problem is hard in $\mathbb{G}_1$ then the BTSC scheme in Figure 1 is insider-secure in the sense of IB-BT-CMA.*

Proof. Let $\mathcal{A}'$ be a forger for the BTSC scheme that has a non-negligible chance in the IB-BT-CMA game. Suppose it makes at most $q_I(\eta)$ queries to the oracle $\hat{H}$. We show how to construct an algorithm $\mathcal{A}$ solving the CDH problem in $\mathbb{G}_1$. Let an instance $(P, \alpha P, \beta P)$ of the CDH problem be given. $\mathcal{A}$ must compute $\alpha\beta P$. It acts as an environment for $\mathcal{A}'$ in the following manner.

1.–5. These steps of the description of the behaviour of $\mathcal{A}$ are the same as in the proof of Thm.1.

6.–8. These steps of the description are also the same as in the proof of Thm. 1. But this time, let us denote the distinguished identity with A. The algorithm $\mathcal{A}$ guesses that $\mathcal{A}'$ attempts to forge a signature coming from A.

9. The sc-queries are answered by $\mathcal{A}$ as follows. If the sender is not A or the node is not a prefix of ω^* then $\mathcal{A}$ can obtain the sender's secret key in this node and construct the signcryption normally. If $\mathcal{A}'$ makes a query $\mathsf{sc}(\omega, \mathsf{A}, \mathsf{B}, M)$, where ω is a prefix of ω^* then $\mathcal{A}$ simulates the signcryption by computing K, R, U and c as in Fig. 1, selecting $r \xleftarrow{\$} \mathbb{Z}_q$ and $S \xleftarrow{\$} \mathbb{G}_1$, computing $T = \hat{e}(S,P)\left(\hat{e}(P_{pub}, Q_{\mathsf{A}})\prod_{i=1}^{t}\hat{e}(R_i, H(\omega|_i))\right)^{-r}$ and defining $H_1(\omega, \mathsf{A}, \mathsf{B}, R, T, c) := r$.

Eventually, $\mathcal{A}'$ will output a successful forgery $(M, (R, S, r, U, c), \mathsf{A}, \mathsf{B})$. With non-negligible probability, $\mathcal{A}$ managed to guess the sender A. We can assume that r has been output by $H_1(\omega^*, \mathsf{A}, \mathsf{B}, R, T, c)$ for a certain T, because otherwise the verification has only a negligible probability of succeeding. We can also assume that before making that H_1-query, the "public key" $Q_A = \bar{H}(\mathsf{A})$ had already been computed, because T depends on Q_A as shown in the 3rd step of the designcryption algorithm in Fig. 1 (except if $r = 0$; this has negligible probability because r is an output from H_1).

The algorithm $\mathcal{A}$ now reruns $\mathcal{A}'$ with the same random tape. It also repeats the answers to all oracle queries, up to, but excluding $H_1(\omega^*, \mathsf{A}, \mathsf{B}, R, T, c)$. This query is answered with $r' \neq r$ and the following queries are handled independently of the first run of $\mathcal{A}'$.

By the forking lemma [23], there is a non-negligible chance that the forgery output by $\mathcal{A}'$ in the second run uses the same query result of H_1. In this case the forged signcryption is (R, S', r', U', c). The first and last components are the same because they were contained in the argument of H_1. For the same reason, the corresponding values T, A and Q_A are also the same for both runs. We have

$$T = \hat{e}(S, P)\left(\hat{e}(P_{pub}, Q_{\mathsf{A}})\prod_{i=1}^{t}\hat{e}(R_i, H(\omega|_i))\right)^{-r}$$

$$= \hat{e}(S', P)\left(\hat{e}(P_{pub}, Q_{\mathsf{A}})\prod_{i=1}^{t}\hat{e}(R_i, H(\omega|_i))\right)^{-r'}$$

Recall that $P_{\text{pub}} = \alpha P$ and $Q_A = \beta P$. Also recall that $H(\omega|_i) = \chi_i P$ where χ_i was generated by $\mathcal{A}$. By comparing the exponents of $\hat{e}(P, P)$ in the above equation, we find that

$$S - r(\alpha\beta P + \sum_{i=1}^{t}\chi_i R_i) = S' - r'(\alpha\beta P + \sum_{i=1}^{t}\chi_i R_i) \ .$$

It is straightforward to find $\alpha\beta P$ from here. □

4 Strongly Forward Secure Signcryption

The binary tree setting can be easily modified into the usual setting of forward security with evolving keys.

Definition 6 (Signcryption with Evolving Keys). *An identity-based digital signcryption scheme with evolving keys is a 5-tuple of probabilistic polynomial-time algorithms $(\mathcal{G}, \mathcal{K}, \mathcal{R}, \mathcal{S}^c, \mathcal{D}^c)$, where*

$\mathcal{G}(1^\eta, \mathsf{params})$ *generates the master secret key msk and the master public key mpk. It fixes the maximum number of time periods for the scheme as τ.*

$\mathcal{K}(msk, \mathsf{ID})$ *outputs the secret key $sk_\varepsilon^{\mathsf{ID}}$ for the user ID, corresponding to the first time period.*

$\mathcal{R}(mpk, i, sk_i^{\mathsf{ID}})$ *receives the current time period $i < \tau$ and the secret key sk_i^{ID} in this period. It returns the secret key sk_{i+1}^{ID} for the next time period.*

$\mathcal{S}^c(mpk, i, sk_i^{\mathsf{A}}, \mathsf{B}, M)$ *computes the signcryption of the message M from the party A to the party B, corresponding to the time period i.*

$\mathcal{D}^c(mpk, i, sk_i^{\mathsf{B}}, \mathsf{A}, \sigma)$ *designcrypts the signcryption σ sent to party B, and verifies that it came from party A. It outputs the plaintext corresponding to σ or $\perp$ if an error occurred.*

Again, $\mathcal{D}^c(mpk, i, sk_i^{\mathsf{B}}, \mathsf{A}, \mathcal{S}^c(mpk, i, sk_i^{\mathsf{A}}, \mathsf{B}, M)) = M$ is the consistency requirement for all correctly generated mpk, sk_i^{A}, sk_i^{B}.

The confidentiality and authenticity definitions for forward secure signcryption with evolving keys can be straightforwardly adapted from the definitions of IB-BT-CPA and IB-BT-CMA. For the sake of space, we do not repeat them here. Compared to definitions in Sec. 3, there are following differences.

- Instead of node names $\omega \in \{0,1\}^*$, there are time periods $i \in \mathbb{N}$.
- The adversary does not have to commit to the time period of the attack (denoted with ω^* in Sec. 3) in the beginning. Instead, the time period is indicated in the challenge-query or in the produced forgery.
- For the identity that the adversary attacks, breakin-queries are allowed only for time periods later than the period of the attack.

Canetti et al. [5] have shown how to transform a binary tree based encryption scheme into a forward secure encryption scheme with evolving keys. Exactly the same construction works here. Let us recall a couple of main points of that construction and its security proof here (see [5] for full details).

The algorithm $\mathcal{G}(1^\eta, \mathsf{params})$ invokes $\mathcal{G}_{\mathbf{BTSC}}(1^\eta, \ell, \mathsf{params})$. Additionally, it fixes (in mpk) the maximum depth of the tree by defining ℓ as the smallest such value that a complete binary tree of depth ℓ has at least τ nodes (in our case $\tau = 2^{\ell+1} - 1$). The key extraction algorithm $\mathcal{K}$ works exactly as $\mathcal{K}_{\mathbf{BTSC}}$. The secret key in the scheme is a *stack* of secret keys of the BTSC scheme (initially containing the key produced by $\mathcal{K}$ as the only element). The algorithms $\mathcal{S}^c$ and $\mathcal{D}^c$ invoke $\mathcal{S}^c_{\mathbf{BTSC}}$ and $\mathcal{D}^c_{\mathbf{BTSC}}$, passing the key at the top of the stack as the secret key.

The key update algorithm $\mathcal{R}(mpk, i, [sk^{\mathsf{ID}}_{\omega_1}, \ldots, sk^{\mathsf{ID}}_{\omega_k}])$ is the most interesting one. The topmost key $sk^{\mathsf{ID}}_{\omega_k}$ is popped off of the stack. If it does not correspond to a leaf node, then the algorithm invokes $\mathcal{R}_{\mathbf{BTSC}}(mpk, \omega_k, sk^{\mathsf{ID}}_{\omega_k})$, obtains $sk^{\mathsf{ID}}_{\omega_k 1}$ and $sk^{\mathsf{ID}}_{\omega_k 0}$ and pushes them (in that order) to the stack. If ω_k corresponds to a leaf node then nothing more is done.

The security of this scheme is reduced to the security of the BTSC scheme. The simulator guesses the time period where the adversary attempts the attack (the guess is correct with non-negligible probability), translates it into ω^* of the topmost key in the stack in that period, and hands it over to the environment defining IB-BT-CPA or IB-BT-CMA security.

5 Public Ciphertext Verifiability and Third Party Verification

With signcryption, B can be sure that the signcryption received came from A. Public verifiability constitutes to ciphertext verifiability which can be done with signcryption σ and the public parameters by any third party. In Figure 1, since computing T and r may be done by any party, this signcryption scheme has public ciphertext verifiability.

Remark 3. In the indistinguishability game, suppose that $\mathcal{A}$ selects messages M_1 and M_2. $\mathcal{A}$ gets a signcryption of M_1 or M_2. With public *message* verifiability,

$\mathcal{A}$ can check whether M_1 or M_2 was signcrypted, thus distinguishing. Hence, when the authors cited refer to public verifiability, they are typically referring to public ciphertext verifiability which can be done without the aid of B.

For third party verification, we denote the prover by B and the third party verifier by T. The verifier knows the plaintext M as well as the key K, such that $M = \mathcal{D}'_K(c)$, but wants to get proof that these indeed were the plaintext and the key used by the sender. Knowing both c and K amounts to knowing M. Hence B will provide proof that K was the correct key.

We use a hash function $H_3 : \mathbb{G}_1^4 \times \mathbb{G}_2^3 \to \mathbb{Z}_q$ in the random oracle model. Non-interactive proofs in the bilinear group setting have been proposed in [13]. Our proof is standard and uses the Fiat-Shamir heuristic to convert a ZK proof into a NIZK proof. The construction for the BTSC scheme in Fig. 1 is given in Fig. 2.

B The prover does the following on signcryption (R, S, r, U, c), message M and key K, where $M = \mathcal{D}'_K(c)$:

1. Let $\widetilde{K} \in \mathbb{G}_2$ be the argument to H_2 in step 2 of the algorithm $\mathcal{D}^c$ in Fig. 1. B recomputes it.
2. Choose $y \xleftarrow{\$} \mathbb{Z}_q$.
3. Set $Y = \hat{e}(P, P_{\text{pub}})^y$ and $K' = \hat{e}(P_{pub}, U_0)^y$.
4. Set $z = H_3(P, P_{\text{pub}}, Q_{\mathsf{B}}, U_0, \widetilde{K}, Y, K')$.
5. Set $Z = yP_{\text{pub}} + zD^{\mathsf{B}}_{\omega}$.
6. Return $\pi = (\widetilde{K}, Y, K', Z)$.

T The verifier does the following on this signcryption and proof:

1. Verify that A signed c using algorithm $\mathcal{D}^c$ from Figure 1.
2. Verify that $K = H_2(\widetilde{K})$.
3. Set $z' = H_3(P, P_{\text{pub}}, Q_{\mathsf{B}}, U_0, \widetilde{K}, Y, K')$.
4. If the following holds, then accept π as proof of signcryption σ from A to B on M for time period ω.
   (a) $\hat{e}(Z, U_0) = K' \left(\widetilde{K} \prod_{i=1}^{t} (U_i, R^{\mathsf{B}}_{\omega|_i})\right)^{z'}$
   (b) $\hat{e}(Z, P) = Y \left(\hat{e}(P_{\text{pub}}, Q_{\mathsf{B}}) \prod_{i=1}^{t} \hat{e}(H(\omega|_i), R^{\mathsf{B}}_{\omega|_i})\right)^{z'}$

Fig. 2. Third Party Verification

Theorem 3. *The third party verification protocol* (B, T) *in Figure 2 is secure.*

Proof. Completeness — the property that if B and T are honest then T accepts — can be verified by a straightforward but tedious computation; see Appendix B. Soundness — if T accepts then the signcryption (R, S, r, U, c) sent from A to B actually contained the message M — follows from the fact that when computing π, the party B has no foreknowledge on the value of z he obtains in step 4. Hence, after choosing the values y, Y and K' in steps 2 and 3, the party B must be

capable of constructing a valid Z for several possible values of z. For example, let z_0 and z_1 be two possible values for which B is capable of constructing Z_0 and Z_1, such that both $(\widetilde{K}, Y, K', Z_0)$ and $(\widetilde{K}, Y, K', Z_1)$ are accepted as proofs if the hash value $H_3(P, P_{\text{pub}}, Q_B, U_0, \widetilde{K}, Y, K')$ equals z_0 or z_1, respectively. In this case, $\widetilde{K}$ must be equal to the argument of H_2 as computed in the step 2 of the designcryption algorithm in Fig. 1. Indeed, the equalities (a) resp. (b) checked by T give us

$$\widetilde{K}^{z_0-z_1} = \frac{\hat{e}(Z_0 - Z_1, U_0)}{\left(\prod_{i=1}^{t} \hat{e}(U_i, R^{\mathsf{B}}_{\omega|_i})\right)^{z_0-z_1}}$$

$$\hat{e}(P_{pub}, Q_{\mathsf{B}})^{z_0-z_1} = \frac{\hat{e}(Z_0 - Z_1, P)}{\left(\prod_{i=1}^{t} \hat{e}(H(\omega|_i), R^{\mathsf{B}}_{\omega|_i})\right)^{z_0-z_1}} \ .$$

Thus, $(\hat{e}(P_{pub}, Q_{\mathsf{B}})^{z_0-z_1})^{-x} = \widetilde{K}^{z_0-z_1}$ implies that $\widetilde{K} = \hat{e}(P_{pub}, Q_{\mathsf{B}})^{-x}$. □

In the insider-security model from Section 3, we need to consider the implication of allowing an adversary to obtain third party verification proofs as described in Figure 2. The definitions of IB-BT-CPA and IB-BT-CMA need to be extended with queries allowing the adversary to obtain proofs that a certain signcryption corresponds to a certain plaintext and symmetric key. The adversary is allowed to make these queries only if it actually knows the plaintext corresponding to the signcryption. These queries must be handled by the simulator reducing the security of the BTSC scheme to the hardness of a Diffie-Hellman related problem. Given a signcryption $\sigma = (R, S, r, U, c)$ from A to B, a message M, and a key K, such that σ corresponds to M, it is possible to simulate the third-party verification proof in the random oracle model, without the simulator knowing the secret key of B. If the signcryption has been correctly constructed, then the oracle H_2 has been used to generate the key K. The simulator locates the query $\widetilde{K}$ to H_2 that resulted in K. It will then

1. select $z \xleftarrow{\$} \mathbb{Z}_q$ and $Z \xleftarrow{\$} \mathbb{G}_1$;
2. set $Y = \dfrac{\left(\hat{e}(P_{pub}, Q_{\mathsf{B}}) \prod_{i=1}^{t} \hat{e}(H(\omega|_i), R^{\mathsf{B}}_{\omega|_i})\right)^{z}}{\hat{e}(Z, P)}$ and $K' = \dfrac{\left(\widetilde{K} \prod_{i=1}^{t} \hat{e}(U_i, R^{\mathsf{B}}_{\omega|_i})\right)^{z}}{\hat{e}(Z, U_0)}$;
3. set $H_3(P, P_{\text{pub}}, Q_{\mathsf{B}}, U_0, \widetilde{K}, Y, K') = z$.

Obviously, if M actually is the plaintext corresponding to σ, then this produces the same distribution of $(\widetilde{K}, Y, K', Z)$ as the prover's algorithm in Fig. 2. Indeed, z is uniformly randomly distributed because it is output by the random oracle. Z is also distributed uniformly randomly and independently from z because of the component yP_{pub}. The values of Y and K' are determined by z and Z.

6 Conclusions and Future Work

We have proposed an identity-based signcryption scheme that is forward secure in a stronger sense than what has been proposed before. The scheme protects

not only the confidentiality, but also the authenticity of the messages against a break-in by the adversary.

Now that a forward-secure signcryption scheme has been proposed, one may explore the connections with key dependent cryptography. As shown in [11], allowing key-dependent signature queries along with a public verification oracle means that the secret key must necessarily be updated. However, in the signcryption setting here, the public ciphertext verifiability would not help an adversary that can guess which message was signcrypted because it is encrypted. This brings up the question as to what additional conditions need to be met in order for a signcryption scheme to withstand an adversary that can query key-dependent signcryptions. For example, if we impose that the symmetric scheme (in the scheme proposed here) have the indistinguishability property despite an adversary getting encryptions of functions of the secret key, would this suffice? Also, leakage-resilient signcryption schemes may be constructed from recent schemes proposed in signing and encryption, the connection being that all of these notions involve the user's state being exposed to some extent.

Acknowledgment

We thank Ahto Buldas and Ronald C. Mullin for helpful discussions. This research was supported by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

References

1. J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2001*, Lecture Notes in Computer Science, pages 83–107, London, UK, 2002. Springer-Verlag.
2. P.S.L.M. Barreto, B. Libert, N. McCullagh, and J.J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In B. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer Berlin / Heidelberg, 2005.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
4. D. Boneh and M.K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
5. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, 2007.
6. L. Chen, Z. Cheng, and N.P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007.
7. S.S.M. Chow, S.M. Yiu, L.C.K. Hui, and K.P. Chow. Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. In J.I. Lim and D.H. Lee, editors, *Information Security and Cryptology - ICISC 2003*, Lecture Notes in Computer Science, pages 352–369. Springer Berlin / Heidelberg, 2004.

8. A.W. Dent. Hybrid cryptography. Cryptology ePrint Archive, Report 2004/210, 2004. http://eprint.iacr.org/.
9. M. Girault and J.C. Pailles. An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. In *European Symposium on Research in Computer Security - ESORICS 1990*, pages 173–184, 1990.
10. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
11. M. González Muñiz and R. Steinwandt. Security of signature schemes in the presence of key-dependent messages. In *Central European Conference on Cryptography - CECC 2009*, 2009.
12. M.I. González-Vasco, F. Hess, and R. Steinwandt. Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466, 2008. http://eprint.iacr.org/.
13. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer Berlin / Heidelberg, 2008.
14. C.G. Günther. An identity-based key-exchange protocol. In *Advances in Cryptology - EUROCRYPT 1989*, pages 29–37, New York, NY, USA, 1990. Springer-Verlag New York, Inc.
15. F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer Berlin / Heidelberg, 2003.
16. R.J. Hwang, C.H. Lai, and F.F. Su. An efficient signcryption scheme with forward secrecy based on elliptic curve. *Applied Mathematics and Computation*, 167(2):870 – 881, 2005.
17. B.G. Kang, J.H. Park, and S.G. Hahn. A new forward secure signature scheme. Cryptology ePrint Archive, Report 2004/183, 2004. http://eprint.iacr.org/.
18. B. Libert and J.J. Quisquater. A new identity based signcryption scheme from pairings. *Information Theory Workshop, 2003. Proceedings*, pages 155 – 158, 2003.
19. B. Libert and J.J. Quisquater. The exact security of an identity based signature and its applications. Cryptology ePrint Archive, Report 2004/102, 2004. http://eprint.iacr.org/.
20. N. Mccullagh and P.S.L.M. Barreto. Efficient and forward-secure identity-based signcryption. Cryptology ePrint Archive, Report 2004/117, 2004. http://eprint.iacr.org/.
21. K.G. Paterson. Id-based signatures from pairings on elliptic curves. *Electronics Letters*, 38:1025–1026, 2002.
22. K.G. Paterson and J.C.N. Schuldt. Efficient identity-based signatures secure in the standard model. In L. Batten and R. Safavi-Naini, editors, *Information Security and Privacy*, volume 4058 of *Lecture Notes in Computer Science*, pages 207–222. Springer Berlin / Heidelberg, 2006.
23. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
24. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security - SCIS 2000*, pages 26–28, 2000.
25. N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12(3):193–196, 1999.

26. M. Toorani and A.A.B. Shirazi. Cryptanalysis of an efficient signcryption scheme with forward secrecy based on elliptic curve. *International Conference on Computer and Electrical Engineering*, 0:428–432, 2008.
27. X.C. Yin, J.W. Chen, and C.M. Wang. A new forward-secure signcryption scheme. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1615–1617, 6 2006.
28. Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In B. S. Jr. Kaliski, editor, *Advances in Cryptology - CRYPTO 1997*, Lecture Notes in Computer Science, pages 165–179, London, UK, 1997. Springer-Verlag.
29. R.W. Zhu, G. Yang, and D.S. Wong. An efficient identity-based key exchange protocol with kgs forward secrecy for low-power devices. *Theoretical Computer Science*, 378(2):198–207, 2007.

A Consistency of BTSC Scheme

We first show that $\dfrac{\hat{e}(U_0, D_\omega^{\mathsf{B}})}{\prod_{i=1}^{t} \hat{e}(R_{\omega|_i}^{\mathsf{B}}, U_i)} = \hat{e}(P_{pub}, Q_{\mathsf{B}})^{x^{-1}}$. Indeed,

$$\begin{aligned}
\frac{\hat{e}(U_0, D_\omega^{\mathsf{B}})}{\prod_{i=1}^{t} \hat{e}(R_{\omega|_i}^{\mathsf{B}}, U_i)} &= \frac{\hat{e}(x^{-1}P, \alpha Q_{\mathsf{B}} + \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i))}{\prod_{i=1}^{t} \hat{e}(\rho_{\omega|_i}^{\mathsf{B}} P, x^{-1} H(\omega|_i))} \\
&= \frac{\hat{e}(x^{-1}P, \alpha Q_{\mathsf{B}}) \prod_{i=1}^{t} \hat{e}(x^{-1}P, \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i))}{\prod_{i=1}^{t} \hat{e}(\rho_{\omega|_i}^{\mathsf{B}} P, x^{-1} H(\omega|_i))} \\
&= \frac{\hat{e}(P_{pub}, Q_{\mathsf{B}})^{x^{-1}} \prod_{i=1}^{t} \hat{e}(\rho_{\omega|_i}^{\mathsf{B}} P, x^{-1} H(\omega|_i))}{\prod_{i=1}^{t} \hat{e}(\rho_{\omega|_i}^{\mathsf{B}} P, x^{-1} H(\omega|_i))} = \hat{e}(P_{pub}, Q_{\mathsf{B}})^{x^{-1}} \quad .
\end{aligned}$$

Next, we show the correctness of the ciphertext verification. For $S := rD_\omega^{\mathsf{A}} + xP_1$, and $D_\omega^{\mathsf{A}} = sQ_{\mathsf{A}} + \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{A}} H(\omega|_i)$ in Figure 1, we have the following consistency check:

$$\begin{aligned}
&\hat{e}(P, S) = \hat{e}(P, rD_\omega^{\mathsf{A}} + xP_1) = \hat{e}(P, rD_\omega^{\mathsf{A}}) \hat{e}(P, xP_1) \\
&= \hat{e}(rP, sQ_{\mathsf{A}} + \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{A}} H(\omega|_i)) \hat{e}(xP, P_1) = \hat{e}(rP, sQ_{\mathsf{A}}) \hat{e}(rP, \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{A}} H(\omega|_i)) T \\
&= \left(\hat{e}(P, sQ_{\mathsf{A}}) \prod_{i=1}^{t} \hat{e}(\rho_{\omega|_i}^{\mathsf{A}} P, H(\omega|_i)) \right)^r T = \left(\hat{e}(P_{pub}, Q_{\mathsf{A}}) \prod_{i=1}^{t} \hat{e}(R_i, H(\omega|_i)) \right)^r T,
\end{aligned}$$

meaning that in step 3 of algorithm $\mathcal{D}^c$ in Fig. 1, we recompute the same T as in step 3 of the algorithm $\mathcal{S}^c$.

B Completeness of Third Party Verification Protocol

For $z = H_3(P, P_{pub}, Q_{\mathsf{B}}, U_0, \widetilde{K}, Y, K')$, we have the following:

$$\hat{e}(Z, U_0) = \hat{e}(yP_{pub} + zD_{\omega}^{\mathsf{B}}, U_0) = \hat{e}(yP_{pub}, U_0)\hat{e}(zD_{\omega}^{\mathsf{B}}, U_0) = \hat{e}(P_{pub}, U_0)^y \hat{e}(D_{\omega}^{\mathsf{B}}, U_0)^z$$
$$= K'\hat{e}(\alpha Q_{\mathsf{B}} + \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i), U_0)^z = K' \left(\hat{e}(\alpha Q_{\mathsf{B}}, U_0)\hat{e}(\sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i), U_0) \right)^z$$
$$= K' \left(\widetilde{K} \prod_{i=1}^{t} \hat{e}(x^{-1}H(\omega|_i), \rho_{\omega|_i}^{\mathsf{B}} P) \right)^z = K' \left(\widetilde{K} \prod_{i=1}^{t} \hat{e}(U_i, R_{\omega|_i}^{\mathsf{B}}) \right)^z$$

and

$$\hat{e}(Z, P) = \hat{e}(yP_{pub} + zD_{\omega}^{\mathsf{B}}, P) = \hat{e}(yP_{pub}, P)\hat{e}(zD_{\omega}^{\mathsf{B}}, P) = \hat{e}(P_{pub}, P)^y \hat{e}(D_{\omega}^{\mathsf{B}}, P)^z$$
$$= Y\hat{e}(\alpha Q_{\mathsf{B}} + \sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i), P)^z = Y \left(\hat{e}(\alpha Q_{\mathsf{B}}, P)\hat{e}(\sum_{i=1}^{t} \rho_{\omega|_i}^{\mathsf{B}} H(\omega|_i), P) \right)^z$$
$$= Y \left(\hat{e}(P_{pub}, Q_{\mathsf{B}}) \prod_{i=1}^{t} \hat{e}(H(\omega|_i), \rho_{\omega|_i}^{\mathsf{B}} P) \right)^z = Y \left(\hat{e}(P_{pub}, Q_{\mathsf{B}}) \prod_{i=1}^{t} \hat{e}(H(\omega|_i), R_{\omega|_i}^{\mathsf{B}}) \right)^z .$$