Universally Convertible Directed Signatures

Fabien Laguillaumie[13], Pascal Paillier[2], and Damien Vergnaud[1]

[1] Laboratoire de Mathématiques Nicolas Oresme
Université de Caen, Campus II, B.P. 5186,
14032 Caen Cedex, France,
{laguillaumie,vergnaud}@math.unicaen.fr
[2] Gemplus Card International, Cryptography Group
34, rue Guynemer, 92447 Issy-les-Moulineaux Cedex, France
pascal.paillier@gemplus.com
[3] Projet TANC - INRIA Futurs
Laboratoire d'informatique (LIX)
École polytechnique, 91128 Palaiseau cedex, France

Abstract. Many variants of Chaum and van Antwerpen's undeniable signatures have been proposed to achieve specific properties desired in real-world applications of cryptography. Among them, *directed signatures* were introduced by Lim and Lee in 1993. Directed signatures differ from the well-known confirmer signatures in that the signer has the simultaneous abilities to confirm, deny and individually convert a signature. The *universal* conversion of these signatures has remained an open problem since their introduction in 1993. This paper provides a positive answer to this quest by showing a very efficient design for universally convertible directed signatures (UCDS) both in terms of computational complexity and signature size. Our construction relies on the so-called *xyz*-trick applicable to bilinear map groups. We define proper security notions for UCDS schemes and show that our construction is secure in the random oracle model, under computational assumptions close to the CDH and DDH assumptions. Finally, we introduce and realize *traceable* universally convertible directed signatures where a master tracing key allows to link signatures to their direction.

1 Introduction

Digital signatures were introduced to identify the source of digital data. In particular they are *non-repudiable* and *universally verifiable*. For centuries, seals and handwritten signatures were attached to documents to indicate the issuer's identity. To determinate the authenticity of this identity, the *original scripts* have to be validated in some sense. In the electronic world, however, the ease of recopy and thereby distribution of digital signatures associated to the *self-authenticating* property may pose a serious threat to the signer's privacy. The concept of *undeniable signature* was first addressed at Crypto'89 by Chaum and van Antwerpen [12]. These signatures have the appealing property that a purported signature cannot be checked without the cooperation of the signer and cannot be denied if the latter has indeed generated the signature. They have found numerous applications in applied cryptography, but the obvious problem with this idea is that in any setting where the signer becomes unavailable, nothing can be determined. Hence, Boyar, Chaum, Damgård and Pedersen [7] proposed *convertible undeniable signatures* which provide the additional feature of converting (individually or universally) the undeniable signatures to ordinary signatures. Another approach has produced various flavors of undeniable signatures which may also be verified by interacting with an entity which has been designated by the signer. *Directed signatures* introduced in 1993 by Lim and Lee [21], *(designated) confirmer signatures* [11], or *limited verifier signatures* [1] are among the best known examples. All of them, which we gather under the generic name of *delegated undeniable signatures*, guarantee to the recipient of a signature the ability to verify it, even when the signer cannot (or refuses to) do so. Directed signatures find a prominent application in the realization of complete peer-to-peer secure messaging systems and are a powerful tool to devise protocols for contract signing [2] or verifiable signature sharing [15]. They propose an individual conversion operation, but up to now *none of them* provides a mechanism for *universal* conversion[1].

From a formal point of view, directed signatures and confirmer signatures are quite similar, the only notable difference, apart from the signer's ability to convert signatures, being the real-world applications the authors had in mind. In brief, a universally convertible directed signature scheme enjoys the following properties. Assuming a signer A and a confirmer B, seen as registered users of the system, A produces signatures that only B (and A her/himself) can verify. Signatures of that type are called (A, B)-directed signatures. Now both A and B have the ability to

– prove in a non-transferable way the validity or invalidity of an (A, B)-directed signature to any other party.
– convert a given (A, B)-directed signature into a regular, universally verifiable signature. This operation does not affect other (A, B)-directed signatures and is carried out independently of the signed message.
– publish a universal trapdoor $\mathcal{T}$ by the means of which all (A, B)-directed signatures become universally verifiable. The trapdoor has no impact whatsoever on (A', B')-directed signatures for $(A', B') \neq (A, B)$.

These operations are independent and performed concurrently, meaning that A and B do not have to interact with each other to achieve either one of these operations.

The literature on confirmer signatures is inconsistent on whether the signer is able to confirm and/or deny signatures. In the recent formalization of confirmer signatures [9, 17], in order to protect the signer from a coercer, this ability is delegated only to the designated confirmer. However, the signer's ability to confirm, deny and sometimes convert signatures is requested or strongly desirable in

[1] The limited verifier signature scheme, introduced in 1999 by Araki *et al* [1], provides the universal conversion operation. However, this protocol was broken by Zhang and Kim [23].

many contexts, and this is supported by a number of schemes (*e.g.* [11, 12, 21]) including directed signatures. Again, none of these supports universal conversion of signatures.

Contributions of the paper. The main contribution of this paper is an efficient and secure directed signature scheme featuring for the first time the universal conversion property. Our design relies on a simple observation known as the xyz-trick [20] which applies to bilinear map groups and allows to realize new cryptographic protocols achieving tradeoffs between authenticity and privacy.

We propose a security model for universally convertible directed signatures that captures and extends the strongest notions of unforgeability and signature invisibility. We prove that our signatures are existentially unforgeable, in the random oracle model, under chosen-message attacks with respect to a new computational assumption closely related to the Diffie-Hellman assumption.

We also show that our signatures are invisible, in the random oracle model, in a weak sense assuming the Decisional Tripartite Diffie-Hellman (DTDH) problem is intractable, and in a strong sense under a non-standard yet well-defined assumption. The scheme supports many variations, and it is easy to achieve invisibility under the DTDH assumption.

In addition to that, we introduce *traceable universally convertible directed signatures* by which a (master) tracing key enables a Tracing Authority (TA) to link signatures to their direction i.e. their issuer and confirmer (we also use the term receiver). We realize the concept using an efficient variation of our basic scheme. We show that the obtained signature scheme inherits the security properties of the basic scheme and that the power conferred to the TA by the tracing key is computationally limited to the tracing procedure.

2 Preliminaries

2.1 Bilinear group systems

Recently, bilinear maps have allowed the opening of a new territory in cryptography, making possible the realization of protocols that were previously unknown or impractical. We now recall the definition of bilinear group systems. In the sequel, we make use of a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ for which there is an efficiently computable isomorphism ρ from $\mathbb{G}_2$ to $\mathbb{G}_1$.

Definition 1 (Bilinear group system). *A* bilinear group system *is a tuple* $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle \cdot, \cdot \rangle, \rho)$ *where* q *is a prime number,* $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ *are groups of order* q *with efficiently computable inner laws,* $\langle P_1 \rangle = \mathbb{G}_1$, $\langle P_2 \rangle = \mathbb{G}_2$, $\langle g_t \rangle = \mathbb{G}_t$, $\langle \cdot, \cdot \rangle : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ *is an efficiently computable map such that for all* $(x, y) \in \mathbb{Z}^2$, $\langle xP_1, yP_2 \rangle = \langle P_1, P_2 \rangle^{xy}$ *holds and* $\langle P_1, P_2 \rangle \neq 1$ *and* $\rho : \mathbb{G}_2 \to \mathbb{G}_1$ *is an efficiently computable isomorphism with* $\rho(P_2) = P_1$.

Definition 2 (Bilinear group system generator). *A* bilinear group system generator *is a probabilistic algorithm* Setup *that takes as input a security parameter k and outputs a bilinear group system $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho) \overset{\$}{\leftarrow}$ $\textsc{Setup}(k)$ such that q is a k-bit prime number.*

2.2 Computational Problems in Bilinear Group Systems

Depending on its practical embodiment, a bilinear group system may or may not provide an efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. In particular, ρ may not be efficiently invertible. In this case, there is a computational separation between problems defined over $\mathbb{G}_1$ and $\mathbb{G}_2$. For instance, the Decisional Diffie-Hellman problem DDH $[\mathbb{G}_2]$ on $\mathbb{G}_2$ is trivial since $\langle\rho(xP_2), yP_2\rangle = \langle P_1, xyP_2\rangle$ for any $x, y \in \mathbb{Z}_q^*$, but the same problem defined over $\mathbb{G}_1$ may remain somewhat intractable. Several new computational problems of various flavors have recently been defined over bilinear groups. We now give the definition of the complexity assumptions we will be using in this paper.

Tripartite Diffie-Hellman (TDH): Let $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ be a bilinear group system. Given group elements $(xP_1, yP_1, zP_2) \overset{\$}{\leftarrow} \mathbb{G}_1^2 \times \mathbb{G}_2$, compute $xyzP_1 \in \mathbb{G}_1$.

This computational problem is at least as difficult as the computational bilinear Diffie-Hellman problem [6]. Similarly, Decisional Tripartite Diffie-Hellman is defined as the problem of distinguishing the distribution of (co-)Diffie-Hellman tuples $\{(xP_1, yP_1, zP_2, uP_1) \mid x, y, z \overset{\$}{\leftarrow} \mathbb{Z}_q^*\}$ from the uniform distribution over $\mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{G}_1$:

Decisional Tripartite Diffie-Hellman (DTDH): Let $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ be a bilinear group system. Given group elements $(xP_1, yP_1, zP_2, uP_1) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{G}_1$, decide whether $u \equiv xyz \pmod q$.

The security of our signatures also relies on the following new computational problem:

Flexible Square Diffie-Hellman (FSDH): Let $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ be a bilinear group system. Given $xP_2 \in \mathbb{G}_2$, output a tuple $(Q, xQ, x^2Q) \in \mathbb{G}_1^3$ for some freely chosen $Q \in \mathbb{G}_1$.

Remark 1. Even though not really considered as classical, the KEA1 assumption[2] was introduced in 1991 by Damgård [14]. Roughly speaking, KEA1 captures the intuition that any algorithm which, given a pair $(P, xP) \in \mathbb{G}^2$, computes a pair $(Q, xQ) \in \mathbb{G}^2$ must "know" $\log_{P_2} Q$. It is easily seen that under KEA1, the FSDH assumption is equivalent to a co-Diffie-Hellman assumption defined over $\mathbb{G}_1$ and $\mathbb{G}_2$.

[2] This assumption and some variants are formally analyzed in [3] and have been used to prove that 3-round protocols were zero-knowledge.

2.3 Designated-Verifier Proofs of equality of two discrete logarithms

To make our security reductions complete, the executions of the confirming/denying protocols have to be simulated in the random oracle model. Therefore we rely in the design of our scheme on a procedure allowing to prove in a non-transferable way the equality (or the inequality) of two discrete logarithms without revealing information on their value. We make use of non-interactive designated-verifier zero-knowledge proofs [18] of equality of discrete logarithms $\log_\alpha \beta = \log_g y$. The notation is $\text{DVPK}\left[x : \beta = \alpha^x \wedge y = g^x\right]$, where α and g are two elements of same prime order in their respective groups. We use the notation $\text{DVPK}\left[x : \beta \neq \alpha^x \wedge y = g^x\right]$ for the dual proof of inequality. Designated-verifier proofs form the basis of denying and confirmation protocols in many undeniable and confirmer signature schemes in the literature. We refer the reader to [18] for further details.

3 Universally Convertible Directed Signatures

3.1 Definition

Given an integer k, a *universally convertible directed signature scheme* DS with security parameter k is formally defined by the following:

- **generation of public parameters:** $\mathsf{DS.Setup}$ is a probabilistic algorithm which takes as input k and outputs public parameters (which include a description of the signature space);
- **key generation for signer A and confirmer B:** $\mathsf{DS.Signer.KeyGen}$ is a probabilistic algorithm which takes as input the public parameters and outputs a signing key pair (pk_A, sk_A). $\mathsf{DS.Confirmer.KeyGen}$ is a probabilistic algorithm which takes as input the public parameters and outputs a confirmer key pair (pk_B, sk_B);
- **key-registration:** $\mathsf{DS.Register}$ is a protocol between a user and a "Key Registration Authority" which takes as input the public parameters and the user's public key pk, and outputs a pair (pk, notif) where $\mathsf{notif} \in \{\texttt{accept}, \texttt{reject}\}$ is the registration authorization decision. The fact that a given public key has been properly registered with the authority, is guaranteed by a signature of it by the authority.
- **signature generation:** $\mathsf{DS.Sign}$ is a probabilistic algorithm which takes as input a bitstring $m \in \{0,1\}^\star$, the signer's private key sk_A, the confirmer's public key pk_B and the public parameters. The output bitstring σ is called *an (A, B)-directed signature on m*;
- **signature verification by confirmer B (resp. signer A):** $\mathsf{DS.Confirmer.Verify}$ (resp. $\mathsf{DS.Signer.Verify}$) is a deterministic algorithm which takes as input two bitstrings m and σ, the signer's public key pk_A, the confirmer's private key sk_B (resp. the signer's private key sk_A, the confirmer's public key pk_B) and the public parameters and checks whether σ is a valid (A, B)-directed signature on m;

– **confirming/denying protocols with confirmer B (resp. signer A):** $\mathsf{DS.Confirmer.}\{\mathsf{Confirm}, \mathsf{Deny}\}$ (resp. $\mathsf{DS.Signer.}\{\mathsf{Confirm}, \mathsf{Deny}\}$) are protocols between a confirmer (resp. a signer) and a third party which takes as input two bitstrings m and σ, the signer's public key pk_A, the confirmer's private key sk_B (resp. the signer's private key sk_A, the confirmer's public key pk_B) and the public parameters. The output is a non-transferable proof that σ is a valid or an invalid (A, B)-directed signature on m;
– **individual conversion by confirmer B (resp. signer A):** $\mathsf{DS.Confirmer.Convert}$ (resp. $\mathsf{DS.Signer.Convert}$) is a deterministic algorithm which takes as input a bitstring σ, the signer's public key pk_A, the confirmer's private key sk_B (resp. the signer's private key sk_A, the confirmer's public key pk_B) and the public parameters, and outputs a bitstring $\tilde{\sigma}_B$ called a *B-converted signature* (resp. $\tilde{\sigma}_A$ called an *A-converted signature*);
– **verification of a B-(resp. A-)converted signature:** $\mathsf{DS.User.VerifyConfirmer}$ (resp. $\mathsf{DS.User.VerifySigner}$) is a deterministic algorithm which takes as input two bitstrings m and $\tilde{\sigma}_B$ (resp. $\tilde{\sigma}_A$), the signer's public key pk_A, the confirmer's public key pk_B and the public parameters and checks whether $\tilde{\sigma}_B$ (resp. $\tilde{\sigma}_A$) is a valid B-converted (resp. A-converted) signature on m;
– **generation of a universal trapdoor by confirmer B (resp. signer A):** $\mathsf{DS.Confirmer.Trapdoor}$ (resp. $\mathsf{DS.Signer.Trapdoor}$) is a deterministic algorithm which takes as input the signer's public key pk_A, the confirmer's private key sk_B (resp. the signer's private key sk_A, the confirmer's public key pk_B), the public parameters and outputs a universal trapdoor $\mathcal{T}_{A,B}$ which makes it possible to universally verify all (A, B)-directed signatures;
– **universal signature verification:** $\mathsf{DS.User.Verify.}$ is a deterministic algorithm which takes as input three bitstrings m, σ and $\mathcal{T}$, the signer's public key pk_A, the confirmer's public key pk_B and the public parameters, and tells whether σ is a valid (A, B)-directed signature on m.

Moreover, a universally convertible directed signature scheme must satisfy the following (informally defined, precisely detailed in the next section) properties:

1. **correctness:** properly formed (A, B)-directed, A-converted and B-converted signatures must be accepted by the verification algorithms;
2. **unforgeability:** it is computationally infeasible, without the knowledge of the signer's private key, to produce a directed signature that is accepted by the verification algorithms or by the confirming protocols;
3. **completeness and soundness:** the verification protocols are complete and sound, where completeness means that valid (invalid) signatures can always be proven valid (invalid) and soundness means that no valid (invalid) signature can be proven invalid (valid).
4. **invisibility:** given a message m and a purported (A, B)-directed signature σ on m, it is computationally infeasible, without the knowledge of the confirmer's or the signer's private key, to ascertain that σ is a valid (A, B)-directed signature of m.
5. **non-transferability:** a user participating in an execution of the confirming/denying protocols does not obtain information that could be used to convince a third party about the validity/invalidity of a signature.

3.2 Security Notions for Universally Convertible Directed Signatures

Unforgeability against adaptive chosen message attacks. The *de facto* standard notion of security for digital signatures was formalized by Goldwasser, Micali and Rivest [16] as existential unforgeability under adaptive chosen message attacks (EF-CMA). For universally convertible directed signatures, the unforgeability security is defined along the same lines, with the notable difference that the adversary can be any of the confirmers chosen by the signer. Therefore, in the attack scenario, the forger $\mathcal{A}$ is allowed to request signatures directed to any registered user of her choice (whose secret key might be known to her). Besides, signer individual/universal conversion algorithms might also leak information to the adversary. We therefore suppose that the adversary knows the confirmers' private keys, the associated signer-generated universal trapdoors, and we allow her to request the individual conversion of any signature of her choice. As usual, the forger has the natural restriction that the returned forgery (including a message, a directed signature and a confirmer's identity) has not been returned by the signing oracle during the game.

Invisibility of signatures. The strongest security notion for undeniable and confirmer signatures is the one of invisibility introduced by Chaum, van Heijst and Pfitzmann in [13]. We precisely define the notion of signature *invisibility* under adaptive chosen message attacks in our context, introducing two flavors of invisibility, weak-Inv-CMA and Inv-CMA.

We consider an adversary $\mathcal{A}$ that runs in two stages: in the `find` stage, $\mathcal{A}$ takes as input the public keys pk_A and $pk_B^\star$, and outputs a message $m^\star$ together with some state information $\mathcal{I}^\star$. In the `guess` stage, $\mathcal{A}$ gets as input $\mathcal{I}^\star$ and a challenge signature $\sigma^\star$ either formed by signing the message $m^\star$ or chosen at random in the signature space. Then $\mathcal{A}$ returns her guess as to whether $\sigma^\star$ is a valid (A, B)-directed signature on $m^\star$ or not.

In the weak-Inv-CMA-model, the adversary has access in both stages to the signing oracle Sign and to the confirming/denying oracle Confirm and Deny. In the Inv-CMA-model, $\mathcal{A}$ is also given access to the individual conversion oracle Convert, and to the universal trapdoor generation oracle Trapdoor. In both cases, she is allowed to invoke these oracles on any message and any confirmer of her choice with the restriction of not sending $(m^\star, \sigma^\star, pk_B^\star)$ to the oracles Confirm, Deny and Convert in the second stage and not sending $pk_B^\star$ to the oracle Trapdoor at any stage.

Let $t \in \mathbb{N}^{\mathbb{N}}$, $\boldsymbol{q} = (q_{\text{Sign}}, q_{\text{Confirm}}, q_{\text{Deny}}, q_{\text{Conv}}, q_{\text{Trap}}, q_{\text{Reg}}) \in [\mathbb{N}^{\mathbb{N}}]^6$ and $\varepsilon \in [0,1]^{\mathbb{N}}$. An algorithm $\mathcal{A}$ is a $(k, t, \boldsymbol{q}, \varepsilon)$-forger (*resp.* a $(k, t, \boldsymbol{q}, \varepsilon)$-distinguisher) against DS if for all integer k, it runs in time at most $t(k)$, makes at most $q_{\text{Sign}}(k), q_{\text{Confirm}}(k), q_{\text{Deny}}(k), q_{\text{Conv}}(k), q_{\text{Trap}}(k), q_{\text{Reg}}(k)$ queries to the given oracles, and has forgery success (*resp.* distinguishing advantage) $\geq \varepsilon(k)$ against DS with security parameter k.

4 Efficient Universally Convertible Directed Signatures

We now describe our first universally convertible directed signature scheme which for readability is denoted again by DS.

Generation of public parameters
$\mathsf{DS.Setup}$: Given a security parameter k, the public parameters are $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle \cdot, \cdot \rangle, \rho) \xleftarrow{\$} \textsc{Setup}(k)$ as well as a hash function H mapping arbitrary bit strings to $\mathbb{Z}_q^*$.

Key generation
$\mathsf{DS.Signer.KeyGen}$: Signer A picks random $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $X_1 = x_1 P_1$ and $X_2 = x_2 P_2$. A's public key is $(X_1, X_2) \in \mathbb{G}_1 \times \mathbb{G}_2$. A's private key is (x_1, x_2).

$\mathsf{DS.Confirmer.KeyGen}$: Confirmer B picks a random $y \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $Y = yP_1$. B's public key is $Y \in \mathbb{G}_1$. B's private key is y.

$\mathsf{DS.\{Signer, Confirmer\}.Register}$: A confirmer public key $pk_B = Y = yP_1$ is registered by letting B prove (possibly non interactively) the knowledge of y to the registration authority by engaging in $\textsc{dvpk}\,[y : Y = yP_1]$. Similarly, a user registers his signing public key $pk_A = (X_1, X_2) = (x_1 P_1, x_2 P_2)$ by proving in zero-knowledge her/his knowledge of x_1 and x_2. The fact that a given public key has been properly registered with the authority, is guaranteed by a signature of the it by the authority.

Signature generation
$\mathsf{DS.Sign}$: Given a message $m \in \{0,1\}^*$ and the public key Y of a confirmer, A picks a random $r \xleftarrow{\$} \mathbb{Z}_q^*$, and computes $U = rP_2$ and $V = (rx_1)(x_2 + H(m, U, Y))^{-1} Y$. In case $x_2 + H(m, U, Y) \equiv 0 \pmod q$, A restarts the signing procedure with a new value for r. The signature is $\sigma = (U, V)$.

Verification by confirmer/signer
$\mathsf{DS.Confirmer.Verify}$: Given a message $m \in \{0,1\}^*$ and a signature $\sigma = (U, V)$, B checks whether $\sigma \in \mathbb{G}_2 \times \mathbb{G}_1$ and $\langle V, X_2 + H(m, U, Y) P_2 \rangle = \langle X_1, U \rangle^y$.

$\mathsf{DS.Signer.Verify}$: Given a message $m \in \{0,1\}^*$ and a signature $\sigma = (U, V)$, A checks whether $\sigma \in \mathbb{G}_2 \times \mathbb{G}_1$ and $\langle V, X_2 + H(m, U, Y) P_2 \rangle = \langle Y, U \rangle^{x_1}$.

Confirmation and disavowal protocols
$\mathsf{DS.Signer.\{Confirm, Deny\}}$: Given a message $m \in \{0,1\}^*$ and a signature $\sigma = (U, V)$, A proves to any third party that

$$\textsc{dvpk}\,[x_1 : \langle V, X_2 + H(m, U, Y) P_2 \rangle = \langle Y, U \rangle^{x_1} \wedge X_1 = x_1 P_1]$$
$$\text{or } \textsc{dvpk}\,[x_1 : \langle V, X_2 + H(m, U, Y) P_2 \rangle \neq \langle Y, U \rangle^{x_1} \wedge X_1 = x_1 P_1]\,.$$

$\mathsf{DS.Confirmer.\{Confirm, Deny\}}$: Given a message $m \in \{0,1\}^*$ and a signature $\sigma = (U, V)$, B proves to any third party that

$$\textsc{dvpk}\,[y : \langle V, X_2 + H(m, U, Y) P_2 \rangle = \langle X_1, U \rangle^{y} \wedge Y = y P_1]$$
$$\text{or } \textsc{dvpk}\,[y : \langle V, X_2 + H(m, U, Y) P_2 \rangle \neq \langle X_1, U \rangle^{y} \wedge Y = y P_1]\,.$$

Individual conversion and verification algorithms

$\mathsf{DS.Signer.Convert}$: Given a purported (A,B)-directed signature $\sigma = (U,V)$, A computes $W = x_1 U \in \mathbb{G}_2$ and outputs $\tilde{\sigma}_A = (U,V,W) \in \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2$ as an A-converted signature on m.

$\mathsf{DS.User.VerifySigner}$: Given a message $m \in \{0,1\}^*$ and a converted signature $\tilde{\sigma}_A = (U,V,W)$, any user checks whether $\langle X_1, U\rangle = \langle P_1, W\rangle$ and $\langle V, X_2 + H(m,U,Y)P_2\rangle = \langle Y, W\rangle$.

$\mathsf{DS.Confirmer.Convert}$: Given a purported (A,B)-directed signature $\sigma = (U,V)$, B computes $W = yU \in \mathbb{G}_2$ and outputs $\tilde{\sigma}_B = (U,V,W) \in \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2$ as a B-converted signature on m.

$\mathsf{DS.User.VerifyConfirmer}$: Given a message $m \in \{0,1\}^*$ and a converted signature $\tilde{\sigma}_B = (U,V,W)$, any user checks whether $\langle Y, U\rangle = \langle P_1, W\rangle$ and $\langle V, X_2 + H(m,U,Y)P_2\rangle = \langle X_1, W\rangle$.

Universal trapdoor generation and verification algorithms

$\mathsf{DS.\{Signer, Confirmer\}.Trapdoor}$: A or B computes $\mathcal{T} = yX_1 = x_1 Y = x_1 y P_1$ and makes $\mathcal{T}$ public.

$\mathsf{DS.User.Verify}$: Given a message $m \in \{0,1\}^*$ and a signature $\sigma = (U,V)$, any user uses the trapdoor $\mathcal{T}$ to check whether $\langle V, X_2 + H(m,U,Y)P_2\rangle = \langle \mathcal{T}, U\rangle$.

The correctness of DS is obvious, and the completeness and soundness of all protocols are classical results [10]. We now discuss a few facts about our scheme.

Efficiency. An (A,B)-directed signature σ is a pair of elements in $\mathbb{G}_2 \times \mathbb{G}_1$, being in that comparable to Boneh and Boyen's recent signature scheme [5]. Signature generation requires an inversion modulo q followed by one exponentiation in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_2$. Therefore no pairing is required. Signature verification by the confirmer is a bit more demanding as a couple of pairings have to be computed. We note that conversion procedures, as well as the generation of a universal trapdoor require a single exponentiation in $\mathbb{G}_1$ or $\mathbb{G}_2$ and are therefore pairing-free.

Verifiability Properties. We note that our scheme is fully verifiable in the sense that all private operations are independently verifiable by third parties. These properties are desirable even though not requested in our definitions. If our system serves as a basic primitive in a cryptographic protocol typically, full verifiability may allow early detection of cheating behaviors and localization of malicious parties.

Security. We note first that the property of non-transferability is fulfilled by our scheme as a direct consequence of the use of designated-verifier proofs in confirmation/disavowal protocols. Further, we state that our scheme resists existential forgeries and that signatures are invisible. Both security reductions stand in the random oracle model.

Theorem 1 (Unforgeability of DS). *Let $t, q_H \in \mathbb{N}^{\mathbb{N}}$, $\boldsymbol{q} = (q_{\text{Sign}}, q_{\text{Confirm}}, q_{\text{Deny}}, q_{\text{Conv}}, q_{\text{Trap}}, q_{\text{Reg}}) \in [\mathbb{N}^{\mathbb{N}}]^6$ and $\varepsilon \in [0,1]^{\mathbb{N}}$. Assume there exists a $(k, t, \boldsymbol{q}, \varepsilon)$-forger $\mathcal{A}$ against DS, in the random oracle model.*

Further assume that $\mathcal{A}$ is limited to q_H executions of H. Then there is an algorithm that solves the FSDH problem in the bilinear group system generator SETUP *with probability $\varepsilon'(k) \geq 1 - 1/2^k$ within time*

$$t' \leq t \cdot \frac{(q_H + q_{\text{Confirm}} + q_{\text{Deny}} + q_{\text{Convert}} + 2)}{\varepsilon} + (\|\boldsymbol{q}\| + q_H) \cdot p_1,$$

where p_1 is a explicit polynomial and $\|\boldsymbol{q}\| = q_{\text{Sign}} + q_{\text{Confirm}} + q_{\text{Deny}} + q_{\text{Conv}} + q_{\text{Trap}} + q_{\text{Reg}}$.

Proof. The proof relies on the Forking Lemma [22] and is in spirit rather similar to the security proof of known discrete-log-based signature schemes such as Schnorr. Assume $\mathcal{A}$ is a forger that $(k, t, \boldsymbol{q}, \varepsilon)$-breaks DS. Here, q_H stands for the number of queries submitted by $\mathcal{A}$ to H since H is viewed as a random oracle. We construct a reduction algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the FSDH problem with time bound and success probability as claimed in Theorem 1. Algorithm $\mathcal{B}$ is given bilinear map parameters $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ generated by SETUP(k) and an instance xP_2 of the FSDH problem. $\mathcal{B}$'s goal is to produce a tuple (Q, xQ, x^2Q) for some $Q \in \mathbb{G}_1$. $\mathcal{B}$ does so by interacting with the forger $\mathcal{A}$ as follows. First, $\mathcal{B}$ picks a random $x_1 \xleftarrow{\$} \mathbb{Z}_q^*$ and sets $X_1 = x_1P_1$ and $X_2 = xP_2$. The knowledge of x_1 in the simulation will be used intensively in the simulation of DS.Signer.Confirm and DS.Signer.Deny}

FIND STAGE. We define a probabilistic subroutine $\mathcal{B}_0(\varpi)$ of $\mathcal{B}$. Given an arbitrary input ϖ, $\mathcal{B}_0(\varpi)$ runs $\mathcal{A}$ with random tape ϖ, transmits $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ as public parameters to $\mathcal{A}$, as well as the public key $pk_A = (X_1, X_2)$. Then $\mathcal{B}_0(\varpi)$ simulates the scheme's oracles H, DS.Sign, DS.Signer.{Confirm, Deny}, DS.Signer.Convert and DS.Signer.Trapdoor as follows.

Simulation of $\boldsymbol{H}$. Given $m \in \{0,1\}^*$ and $(U, Y) \in \mathbb{G}_2 \times \mathbb{G}_1$, if $H(m, U, Y)$ is defined, output its value. Otherwise, pick a random $h \xleftarrow{\$} \mathbb{Z}_q^*$, define $H(m, U, Y) = h$ and output h.

Simulation of DS.Sign. Given $m \in \{0,1\}^*$ and a confirmer's public key $pk_B = Y$, pick a random $r, h \xleftarrow{\$} \mathbb{Z}_q^*$. Set $V = rx_1Y$ and $U = rX_2 + rhP_2$. If $H(m, U, Y)$ is defined and is $\neq h$, $\mathcal{B}_0(\varpi)$ aborts. Otherwise $\mathcal{B}_0(\varpi)$ defines $H(m, U, Y) = h$ and outputs $\sigma = (U, V)$.

Simulation of DS.Signer.{Confirm, Deny}. Since $\mathcal{B}_0(\varpi)$ knows x_1, $\mathcal{B}_0(\varpi)$ is able to verify any given directed signature and consequently, to engage successfully in one of the two protocols DVPK $[x_1 : \langle V, X_2 + H(m, U, Y)P_2\rangle = \langle Y, U\rangle^{x_1} \wedge X_1 = x_1P_1]$ or DVPK $[x_1 : \langle V, X_2 + H(m, U, Y)P_2\rangle \neq \langle Y, U\rangle^{x_1} \wedge X_1 = x_1P_1]$ for any given $m \in \{0,1\}^*$, $\sigma = (U, V) \in \mathbb{G}_2 \times \mathbb{G}_1$ and $Y \in \mathbb{G}_1$. Note that a simulation of H is required in either case.

Simulation of DS.Signer.Convert. Given $\sigma = (U, V)$, output $\tilde{\sigma}_A = (U, V, x_1U)$.

Simulation of DS.Signer.Trapdoor. Given $Y \in \mathbb{G}_1$, output $\mathcal{T} = x_1Y$.

If $\mathcal{A}$ returns a forgery $(Y, m, \sigma = (U, V))$, $\mathcal{B}_0(\varpi)$ simulates H once again to get $h = H(m, U, Y)$ and checks whether $\langle V, X_2 + H(m, U, Y)P_2\rangle = \langle Y, U\rangle^{x_1}$. If the equality holds, and if σ does not appear in the transcript of $\mathsf{DS.Sign}$, $\mathcal{B}_0(\varpi)$ is said to *succeed*.

Algorithm $\mathcal{B}$ restarts $\mathcal{B}_0(\varpi)$ for random values of $\varpi \xleftarrow{\$} \{0,1\}^*$ until $\mathcal{B}_0(\varpi)$ succeeds. Let $(Y, m, \sigma = (U, V))$ be the last output of $\mathcal{A}$. Then $\mathcal{B}$ memorizes ϖ, the index j of $(m, U, Y) \mapsto h$ in H's transcript (sorted in chronological order), and the first j outputs of H noted $h_1, \ldots, h_j$. If ℓ denotes the index in the transcript of $\mathsf{DS.Sign}$ of the last signature output *before* H returns h_j, $\mathcal{B}$ also memorizes $\ell, \sigma_1, \ldots, \sigma_\ell$.

Replay Stage. As is classical with forking-based reductions, we define a second probabilistic subroutine $\mathcal{B}_1(\varpi)$ of $\mathcal{B}$ which role is essentially to replay the last and successful execution of $\mathcal{B}_0(\varpi)$ until the moment when H is about to output h_j, and then simulate all oracles with fresh random values from that moment on. The tape ϖ being given by the find stage, $\mathcal{B}_1(\varpi)$ runs $\mathcal{A}$ with random tape ϖ, transmits the same public parameters $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ to $\mathcal{A}$, as well as $pk_A = (X_1, X_2)$. Then $\mathcal{B}_1(\varpi)$ simulates the oracles as follows, using its own random tape π.

Simulation of H. If the query index is $i < j$, output h_i. Otherwise, simulate as in the find stage with fresh random values.

Simulation of $\mathsf{DS.Sign}$. If the query index is $i' \leq \ell$, output $\sigma_{i'}$. Otherwise, simulate as in the find stage with fresh random values.

All other oracles are simulated exactly as in the find stage. If $\mathcal{A}$ returns a forgery $(Y', m', \sigma' = (U', V'))$, $\mathcal{B}_1(\varpi)$ queries its own simulation of H to verify σ' the same way $\mathcal{B}_0(\varpi)$ verified σ. If σ' is invalid or was output by the simulation of $\mathsf{DS.Sign}$ or if the index of $(m', U', Y') \mapsto h'$ in the transcript of H is $j' \neq j$, then $\mathcal{B}_1(\varpi)$ is said to fail.

Algorithm $\mathcal{B}$ restarts $\mathcal{B}_1(\varpi)$ with random values for π until $\mathcal{B}_1(\varpi)$ succeeds. Let then $(Y', m', \sigma' = (U', V'))$ be the last output of $\mathcal{A}$.

Key Retrieval Stage. We perform a specific stage that allows $\mathcal{B}$ to retrieve the confirmer private key y' associated to the find-stage forgery (Y', m', σ'), *i.e.* such that $Y' = y'P_1$. To this end, $\mathcal{B}$ replays $\mathcal{B}_1$ once with a slightly modified random tape $\pi' \approx \pi$ such that replaying the registration stage of Y' by $\mathcal{A}$ allows to extract y'. As registration is performed via a non-interactive DVPK of a discrete log, modifying the 'challenge' value returned by the internal hash function of the protocol yields y' by knowledge extraction[3]. $\mathcal{B}$ then stops $\mathcal{A}$ and memorizes y'.

Final Outcome. $\mathcal{B}$ disposes of two valid forgeries $(Y, m, \sigma = (U, V))$ and $(Y', m', \sigma' = (U', V'))$. Since $(m', U', Y') \mapsto h'$ and $(m, U, Y) \mapsto h$ have the same index in the transcript of H, we must have $(m', U', Y') = (m, U, Y)$ by a causality argument. In particular, $\mathcal{B}$ knows $y = y'$. $\mathcal{B}$ then computes $\Delta = x_1^{-1}y^{-1}V$ and $\Delta' = x_1^{-1}y^{-1}V'$. From the simulation, there exists $r \in \mathbb{Z}_{q-1}^*$ (unknown to $\mathcal{B}$) such that $\Delta = r(x+h)^{-1}P_1$ and $\Delta' = r(x+h')^{-1}P_1$. $\mathcal{B}$ poses $R = rP_1$ and $Q = (h'-h)^{-1}(\Delta - \Delta') = r[(x+h)(x+h')]^{-1}P_1 = [(x+h)(x+h')]^{-1}R$, or aborts if $h' - h \equiv 0 \pmod q$. Finally, one has $\Delta = (x+h')\,Q$ so that

[3] This technique is classical and we therefore do not enter into more details here.

$\varDelta - h'Q = xQ$. Since $R = (x+h)(x+h')Q$, we get $R-(h+h')xQ - hh'Q = x^2Q$ and $\mathcal{B}$ outputs (Q, xQ, x^2Q) to its own challenger.

Reduction cost. We start with a preliminary observation. The transcript of H contains exactly $q_{\mathrm{tot}}(k)$ hash definitions, where $q_{\mathrm{tot}} = q_H + q_{\mathrm{Sign}} + q_{\mathrm{Confirm}} + q_{\mathrm{Deny}} + 1$, since the simulation of H is invoked by the simulation of other oracles (the constant term 1 comes from the verification of the forgery). Among these hash values, exactly $q_{\mathrm{Sign}}(k)$ were defined by the simulation of $\mathsf{DS.Sign}$ and by construction, the j-th hash definition $H(m, U, Y)$ cannot be one of these.

Let us denote by $\mathcal{H}_j$ the set of vectors $(h_1, \ldots, h_{q_{\mathrm{tot}}(k)})$ leading to a forgery of index j and $\varepsilon_j = \Pr\left[(h_1, \ldots, h_{q_{\mathrm{tot}}(k)}) \in \mathcal{H}_j\right]$, the probability being taken over all the values of $h_1, \ldots, h_{q_{\mathrm{tot}}(k)}$ over $\mathbb{Z}_q^*$. Note that the ε_i's may depend on $\mathcal{A}$ and w but in any case $\sum_j \varepsilon_j = \varepsilon(k)$ must hold. Following our remark above, there must be at least $q_{\mathrm{Sign}}(k)$ values of j for which $\varepsilon_j = 0$. We now invoke the

Lemma 1 (Splitting Lemma [22]). *Noting $\mathcal{X}_j$ the set of vectors $(h_1, \ldots, h_{j-1})$ such that*

$$\Pr\left[(h_1, \ldots, h_{j-1}, h'_j, \ldots, h'_{q_{\mathrm{tot}}(k)}) \in \mathcal{H}_j\right] \geq \frac{\varepsilon_j}{2},$$

where the probability is taken over $h'_j, \ldots, h'_{q_{\mathrm{tot}}(k)} \xleftarrow{\$} \mathbb{Z}_q^*$, one has

$$\Pr\left[(h_1, \ldots, h_{j-1}) \in \mathcal{X}_j \mid (h_1, \ldots, h_{q_{\mathrm{tot}}(k)}) \in \mathcal{H}_j\right] \geq \frac{1}{2}\ .$$

where the probability is taken over $h_1, \ldots, h_{j-1} \xleftarrow{\$} \mathbb{Z}_q^*$.

We expect the find and key retrieval stages to require at most $\varepsilon(k)^{-1} + 1$ executions of $\mathcal{A}$. Suppose that the transcript of H when $\mathcal{B}_0(\varpi)$ succeeds is $(h_1, \ldots, h_{q_{\mathrm{tot}}(k)}) \in \mathcal{H}_j$ for some j. This event occurs with non-zero probability $\varepsilon_j/\varepsilon(k)$ as soon as $\varepsilon_j \neq 0$. Further assume that $(h_1, \ldots, h_{j-1}) \in \mathcal{X}_j$; the probability that this occurs is at least 1/2. Then the expected number of executions of $\mathcal{B}_1(\varpi)$ is $2/\varepsilon_j$. Putting it all together, and taking into account the abortion case $h' \equiv h \pmod q$, $\mathcal{B}$ succeeds with probability $\geq 1 - 1/2^{-k}$ after

$$\frac{1}{\varepsilon(k)} + 1 + \sum_{1 \leq j \leq q_{\mathrm{tot}}(k), \varepsilon_j \neq 0} \frac{\varepsilon_j}{\varepsilon(k)} \cdot \frac{1}{2} \cdot \frac{2}{\varepsilon_j} \leq \frac{q_{\mathrm{tot}}(k) - q_{\mathrm{Sign}}(k) + \varepsilon(k)}{\varepsilon(k)}$$

executions of $\mathcal{A}$, *i.e.* in time at most $[t \cdot (q_H + q_{\mathrm{Confirm}} + q_{\mathrm{Deny}} + q_{\mathrm{Convert}} + 2)/\varepsilon](k)$. The term $(\|\boldsymbol{q}\| + q_H)\, p_1$ comes from the time needed to simulate all oracles. $\square$

Remark 2. The simulation of the DVPKs imposes the random oracle model and we therefore must allow the adversary to query the internal oracles used to compute proofs. The simulation cost induced by these queries are included into q_H.

We also state that DS is weakly invisible under the assumption that the Decisional Tripartite Diffie-Hellman problem is intractable:

Theorem 2 (Weak Invisibility of DS). *Let $t, q_H \in \mathbb{N}^{\mathbb{N}}$, $\boldsymbol{q} = (q_{\mathrm{Sign}}, q_{\mathrm{Confirm}}, q_{\mathrm{Deny}}, 0, 0, q_{\mathrm{Reg}}) \in [\mathbb{N}^{\mathbb{N}}]^6$ and $\varepsilon \in [0,1]^{\mathbb{N}}$. Assume there exists a $(k, t, \boldsymbol{q}, \varepsilon)$-distinguisher $\mathcal{A}$ against DS, in the random oracle model. Then there exists an algorithm that solves the DTDH problem in the bilinear group system generator* Setup *with probability $\varepsilon' = \varepsilon/2 - o(1)$ within time $t' \leq q_{\mathrm{Reg}} \cdot t + (\|\boldsymbol{q}\| + q_H) \cdot p_2$, where p_2 is an explicit polynomial.*

Proof. We show that, assuming the hardness of the Decisional Tripartite Diffie-Hellman DTDH, DS is weakly invisible under an adaptive chosen-message attack. Our reduction is in essence similar to previously known reductions in the standard model, and we therefore skip minor details. Note that H needs not be seen as a random oracle. The fact that we require the random oracle model only stems from the need to simulate zero-knowledge proofs.

Assume $\mathcal{A}$ is an attacker that $(k, t, \boldsymbol{q}, \varepsilon)$-breaks the weak invisibility of DS as defined earlier. We construct a reduction algorithm $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves a DTDH instance with time bound and advantage as claimed in Theorem 2. The outline of the reduction is as follows. Algorithm $\mathcal{B}$ is given an bilinear group system $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ generated by Setup(k) and an instance $(\alpha P_1, \beta P_1, \gamma P_2, \delta P_1) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{G}_1$. $\mathcal{B}$'s goal is to decide whether $\delta \equiv \alpha\beta\gamma \pmod{q}$. $\mathcal{B}$ does so by interacting with the forger $\mathcal{A}$ as follows. First, $\mathcal{B}$ picks a random $x_2 \xleftarrow{\$} \mathbb{Z}_q^*$ and sets $X_1 = \alpha P_1$ and $X_2 = x_2 P_2$. Then $\mathcal{B}$ sends the public parameters to $\mathcal{A}$ as well as the signer public key (X_1, X_2) and the challenge confirmer public key $Y^\star = \beta P_1$. $\mathcal{B}$ attempts to simulate all oracles throughout the find stage, as shown later. $\mathcal{A}$ then returns a challenge message $m^\star$. $\mathcal{B}$ then picks a random bit $b \in \{0,1\}$ and sets $U = \gamma P_2$. If $b = 0$, $\mathcal{B}$ sets $h^\star = H(m^\star, U, Y)$ and $V = (x_2 + h^\star)^{-1} (\delta P_1)$. If $b = 1$, $\mathcal{B}$ initializes $V \xleftarrow{\$} \mathbb{G}_1$. Then $\mathcal{B}$ defines $\sigma^\star = (U, V)$ and $\sigma^\star$ is returned to $\mathcal{A}$ as the signature challenge. Throughout the guess stage, $\mathcal{B}$ simulates the oracles the same way it did in the find stage. $\mathcal{A}$ finally outputs a guess $b' \in \{0,1\}$ and $\mathcal{B}$ returns 1 to its own challenger if $b' = b$ or 0 otherwise.

When $\delta \equiv \alpha\beta\gamma \pmod{q}$, signature simulations will all be correct and the advantage of $\mathcal{A}$ in guessing b is at least $\varepsilon(k)$. In the case $\delta \not\equiv \alpha\beta\gamma \pmod{q}$, the signatures output by $\mathcal{B}$ are simply invalid and $\mathcal{A}$ may then behave arbitrarily. Overall, $\mathcal{B}$ correctly guesses its own challenge with probability negligibly close to $\varepsilon(k)/2$.

Key Retrieval for $Y \neq Y^\star$. When $Y \neq Y^\star$, $\mathcal{A}$ must have registered the public key $Y = yP_1$ prior to requesting any signature of type (m, Y), so that $\mathcal{B}$ recovers y via registration replay as in the proof of unforgeability. This requires to reboot and replay A with the same random tape up to the point in executing the dvpk when bringing fresh randomness in the 'challenge' hash value allows to extract the discrete log y. Knowing y, $\mathcal{B}$ continues the second execution of A until a new confirmer key Y' is registered, and so forth. This strategy ensures that $\mathcal{B}$ can actually recover all the confirmer private keys matching the public keys registered by $\mathcal{A}$. The price to pay is a factor $q_{\mathrm{Reg}}(k)$ in the number of times $\mathcal{A}$ has to be executed. Note that, since we make use of non-interactive dvpk of a discrete logarithm, there is no concurrent interleving of registrations and therefore the “reboot and replay” technique applies readily.

Simulation of Signatures for $Y \neq Y^\star$. Signatures are simulated in the following way. Given (m, Y), $\mathcal{B}$ first recovers $y = \log_P Y$ in its transcript. Then $\mathcal{B}$ picks a random $r \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and sets $U = rP_2$ and $V = ry(x_2 + h)^{-1}(\alpha P_1)$, where $h = H(m, U, Y)$. $\mathcal{B}$ memorizes (σ, m, Y, r) in its transcript and returns $\sigma = (U, V)$.

Simulation of Other Operations Involving $B \neq B^\star$. As $y = \log_P Y$ is known to $\mathcal{B}$ whenever $Y \neq Y^\star$, $\mathcal{B}$ can individually convert any (A, B)-directed signature $\sigma = (U, V)$ given by $\mathcal{B}$ to $\mathcal{A}$ by simply computing $W = rX_1$ where r is the randomness used to construct σ. Similarly, $\mathcal{B}$ can generate universal trapdoors $\mathcal{T} = yX_1$.

Simulation of Confirmation/Denial Protocols. $\mathcal{B}$ simulates the DVPK of $x_1 = \log_{P_1} X_1$ (that $\mathcal{B}$ does not know) or $y = \log_{P_1} Y$ (that $\mathcal{B}$ knows from the key retrieval stage). This requires to simulate the internal random oracle of DVPK in the first case. The proof is then returned to (the user corrupted by) $\mathcal{A}$.

Simulation of Signatures for $Y = Y^\star$. Given m, $\mathcal{B}$ picks a random $r \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, sets $U = r(\gamma P_2)$ and $V = \frac{r}{x_2+h}(\delta P_1)$ where $h = H(m, U, Y^\star)$. $\mathcal{B}$ returns $\sigma = (U, V)$.

Reduction Cost. As discussed above, $\mathcal{B}$'s own challenge is solved with probability $\varepsilon'(k) \geq \varepsilon(k)/2 - \Pr[\text{DVPK fails}]$ within time bound $q_{\text{Reg}}(k) \cdot t(k) + \|\boldsymbol{q}\|(k) \cdot p_2(k)$ where p_2 is an explicit polynomiaml and the second term comes from the simulations of all oracles. This is as claimed in Theorem 2. $\square$

Invisible Universally Convertible Directed Signatures. We refer the reader to Appendix A for a proof that DS is invisible under a non-standard complexity assumption referred to as the q_s-Tripartite-DCAA problem.

Because of its simplicity, however, our scheme admits many variations. A possible direction to reach invisibility under a weaker assumption consists in replacing the individual conversion algorithms by standard non-interactive zero-knowledge (NIZK) proofs of knowledge of equality/inequality of discrete logarithms. The NIZK proof is then appended to the directed signature as a replacement of the third signature part W. It is then possible to obtain invisibility under the Decisional Tripartite Diffie-Hellman assumption. The proof is very similar to the one of Theorem 2 except that signature conversions are provided to the adversary by simulating the corresponding NIZK proofs for signatures issued by the reduction. The other cases are upper bounded in probability by the unforgeability property of our scheme.

5 Universally Convertible Directed Signatures with Traceability

Directed signatures find a prominent application in the realization of complete peer-to-peer secure messaging systems. In such a system, users have a unique key pair $\{pk = (X_1, X_2), sk = (x_1, x_2)\}$ where $X_1 = x_1P_1, X_2 = x_2P_2$ and x_1 plays simultaneously the role of a signing and of a confirming key. By misuse of language, we sometimes call x_1 the *anonymity key* and x_2 the *signing key* for reasons that will appear clearly in what follows. We view a confirmer more like a regular receiver of a signed message and preferably adopt this term in the sequel. In authenticated messaging systems, putting a restriction on the ability to verify signatures is of a certain interest towards the users' privacy. The property of invisibility guarantees this privacy until one of the two parties wishes to end it.

There are real-life contexts, however, in which conferring this ability to a trusted authority acting in extreme circumstances is desirable. One may think of private contract signing for instance, where criminals make use of the system to sign illegal contracts that are not publicly verifiable. What is really desired is a traceability mechanism[4] enabling a tracing authority (TA) to link upon request directed signatures to their direction i.e. the identities of their signer and receiver. We now introduce an extension of our scheme that supports signature tracing.

5.1 Description of the Scheme DST

Setup and Key generation

The generation of public parameters and keys in the system is essentially the same as above, except that we include $Z = zP_2 \in \mathbb{G}_2$ into the system public parameters. The tracing key is $z \in \mathbb{Z}_q^*$. Moreover, we require users to (securely) submit their anonymity keys x_1 to the Key Registration Authority (KRA). A receipt is returned to the registering user after that, under the form of a NIZK proof $\psi(X_1)$ that the KRA knows x_1.

Signature generation

Now, given a message $m \in \{0,1\}^*$ and the public key Y of the receiver, signer A picks random $r, s \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, and computes $U = rP_2$, $W = rZ + sP_2$, $T = s^{-1}Y$, and $V = rx_1(x_2 + H(m, U, W, T, Y))^{-1}Y$. Again, when $x_2 + H(m, U, W, T, Y) \equiv 0 \pmod q$, A restarts the signing procedure with new values for r, s. Next, signer A computes a NIZK proof of consistency $\pi = \text{NIZK}[(r, s, h, X_1, X_2, Y) : \psi(X_1) \wedge \psi(Y) \wedge \langle P_1, W\rangle\langle\psi(Z), U\rangle^{-1} = g_t^s \wedge Y = sT \wedge \langle\psi(V), X_2 + hP_2\rangle = \langle\psi(X_1), T\rangle^{rs}]$. The signature is $\sigma = (U, W, T, V, \pi)$.

Other operations

The verification procedure and the confirming/denying protocols are unchanged, except that the non-interactive proof π is verified. Signature conversions are done the same way i.e. by appending x_1U or yU, or a NIZK proof of knowledge of x_1 or y to the signature. The generation of trapdoors is unchanged. Universal verification requires the additional check that π is correct.

[4] In [19], Kiayias, Tsiounis and Yung propose a similar traceability mechanism in the context of group signatures.

Signature Tracing

A prerequisite for signature tracing is the recovery of the anonymity key x_1 of the suspected user. This is done by the Key Registration Authority upon judicial request. Now the TA is given a signature $\sigma = (U, W, T, V, \pi)$ of some message m and is asked to decide whether σ was issued by the given user and if so, to whom the signature was directed. The TA first ascertains that π is correct and searches in the public key database a key Y for which $\langle T, W - zU \rangle = \langle Y, P_2 \rangle$. The search is always successful, because a proof that Y lies in the set of registered keys is included in π and is known to be correct. Now given Y and x_1, the TA checks whether $\langle V, X_2 + H(m, U, W, T, Y)P_2 \rangle = \langle Y, U \rangle^{x_1}$. We note that in case of mismatch, the TA is left with anonymous material meaning that if the signature was issued by some user A' then the identity of A' is preserved. This property is in fact computationally guaranteed, as stated later.

5.2 Security Analysis

We state that (A, B)-directed signatures are existentially unforgeable and invisible under adaptive chosen-message attack for any user $\neq A, B,$ TA. We rely again on the FSDH and the DTDH assumptions in the random oracle model.

Theorem 3 (Unforgeability and Weak Invisibility). *Let* $t, q_H \in \mathbb{N}^{\mathbb{N}}$, $\boldsymbol{q} = (q_{\text{Sign}}, q_{\text{Confirm}}, q_{\text{Deny}}, q_{\text{Conv}}, q_{\text{Trap}}, q_{\text{Reg}}) \in [\mathbb{N}^{\mathbb{N}}]^6$ *and* $\varepsilon \in [0,1]^{\mathbb{N}}$.

1. *Assume there exists a* $(k, t, \boldsymbol{q}, \varepsilon)$*-forger* $\mathcal{A}$ *against* DST*, in the random oracle model. Further assume that* $\mathcal{A}$ *is limited to* q_H *executions of* H*. Then there is an algorithm that solves the FSDH problem in the bilinear group system generator* Setup *with probability* $\varepsilon'(k) \geq 1 - 1/2^k$ *within time*
$$t' \leq t \cdot \frac{(q_H + q_{\text{Confirm}} + q_{\text{Deny}} + q_{\text{Convert}} + 2)}{\varepsilon} + (\|\boldsymbol{q}\| + q_H) \cdot p_3,$$
where p_3 *is a explicit polynomial.*
2. *Assume there exists a* $(k, t, (q_{\text{Sign}}, q_{\text{Confirm}}, q_{\text{Deny}}, 0, 0, q_{\text{Reg}}), \varepsilon)$*-distinguisher* $\mathcal{A}$ *against* DST*, in the random oracle model. Then there exists an algorithm that solves the DTDH problem in the bilinear group system generator* Setup *with probability* $\varepsilon' = \varepsilon/2 - o(1)$ *within time* $t' \leq q_{\text{Reg}} \cdot t + (\|\boldsymbol{q}\| + q_H) \cdot p_4$*, where* p_4 *is an explicit polynomial.*

Proof. The proof is similar to those of the security of the scheme DS and will be given in the full version of the paper. □

We also state two important properties fulfilled by the tracing mechanism. They tell us in essence that beyond traceability, the Tracing Authority has no 'hidden powers' over standard users of the system.

Theorem 4 (Abuse-free Traceability). *Signatures issued by user* A *remain invisible to the tracing authority itself as long as the anonymity key* x_1 *of user* A *is undisclosed to the TA.*

Theorem 5 (Tracing-Proof Unforgeability). *After x_1 is disclosed to the TA to enable tracing, the tracing authority is still unable to existentially forge signatures on behalf of A.*

We argue that these properties come from the computational separation between the anonymity key x_1 and the signing key x_2. In fact, after the x_1-part of the secret key of a traced user has been revoked, signatures from that user remain unforgeable because x_2 has not been compromised. The revoked user could even be rehabilitated and a new anonymity key generated to replace the revoked one. This allows a clear separation of powers invested in users and authorities of the system. Due to lack of space, the complete proofs will be given in the full version of the paper.

5.3 Technical Considerations

Implementation of the nizk proof π. The NIZK proof π is implemented as a Fiat-Shamir-transformed conjunction of interactive proofs of the predicates forming π. We rely on prior art [8] to provide an efficient procedure to generate π in practice.

Performances. Signature generation requires 2 exponentiations over group $\mathbb{G}_1$ and 2 over group $\mathbb{G}_2$, and no pairing. Here too, off-line/online signature generation trade-offs are possible by appending a third key part $X_3 = x_3 P_2$ in the user key. Signature conversions and the generation of trapdoors require a single exponentiation over $\mathbb{G}_2$ or $\mathbb{G}_1$ respectively. All verification algorithms require at least two evaluations of the bilinear map. We note that the tracing procedure requires $O(N)$ bilinear map evaluations where N is the number of registered (non-revoked) users. We leave as an open problem to find similar schemes admitting a tracing procedure in polylog complexity in all parameters.

Extensions. Invisibility under the Decisional Tripartite Diffie-Hellman assumption is obtained by replacing the individual conversion procedures by standard NIZK proofs. All operations within the scheme (signature conversion, trapdoor generation) are easily adapted to be verifiable. Among other possible extensions, we cite multi-receiver directed signatures.

6 Conclusion

We properly defined security notions for directed signatures that support the additional property of universal conversion. Using the xyz-trick, we realized the first scheme featuring both individual and universal conversion of signatures, thereby addressing a problem left open since 1993. The new scheme offers attractive practical advantages in terms of signature length and performances. In comparison with previous works, the computational costs for the signer in the signature generation, the confirmation/disavowal protocols and the conversion algorithms, are among the smallest of all delegated undeniable signature schemes. We have proved the security of our scheme in the random oracle model under computational assumptions closely related to the Diffie-Hellman and Decision Diffie-Hellman assumptions on bilinear map groups.

Finally, we introduced traceable directed signatures as a powerful extension to allow a Tracing Authority within the system to link signatures to their direction i.e. issuer and receiver. We believe that our signature schemes are simultaneously efficient and customizable, and we expect to see new cryptographic applications of our work in the future. The *xyz*-trick will certainly have other applications in future works as well. For example, our scheme is easily extended to achieve the time-selective conversion property as in [20].

References

1. S. Araki, S. Uehara, K. Imamura: The Limited Verifier Signature and Its Application. IEICE Trans. Fundamentals, Vol. E82-A (1), 63–68 (1999)
2. N. Asokan, V. Shoup, M. Waidner: Optimistic Fair Exchange of Digital Signatures. Proc. of Eurocrypt'98, Springer LNCS Vol. 1403, 591–606 (1998)
3. M. Bellare, A. Palacio: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. Proc. of Crypto'04, Springer LNCS Vol. 3152, 273–289 (2004)
4. M. Bellare, P. Rogaway: Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. Proc. of 1st ACM Conference on Computer and Communications Security. 62–73 (1993)
5. D. Boneh, X. Boyen: Short Signatures Without Random Oracles. Proc. of Eurocrypt'04, Springer LNCS Vol. 3027, 56–73 (2004)
6. D. Boneh, M. Franklin: Identity-based Encryption from the Weil Pairing. SIAM J. Computing, 32 (3), 586–615 (2003)
7. J. Boyar, D. Chaum, I. B. Damgård, T.P. Pedersen: Convertible Undeniable Signatures. Proc. of Crypto'90, Springer Vol. LNCS 537, 189–205 (1991)
8. E. Bresson, J. Stern: Proofs of Knowledge for Non-Monotone Discrete-Log Formulae and Applications. Proc. of ISC'02, Springer LNCS Vol. 2433, 272–288 (2002)
9. J. Camenisch, M. Michels: Confirmer Signature Schemes Secure against Adaptive Adversaries. Proc. of Eurocrypt'00, Springer LNCS Vol. 1807, 243–258 (2000)
10. J. Camenisch, M. Stadler: Efficient Group Signature Schemes for Large Groups. Proc. of Crypto'97, Springer LNCS Vol. 1296, 410–424 (1997)
11. D. Chaum: Designated Confirmer Signatures. Proc. of Eurocrypt'94, Springer LNCS Vol. 950, 86–91 (1995)
12. D. Chaum, H. van Antwerpen: Undeniable Signatures. Proc. of Crypto'89, Springer LNCS Vol. 435, 212–216 (1989)
13. D. Chaum, E. van Heijst, B. Pfitzmann: Cryptographically Strong Undeniable Signatures Unconditionally Secure for the Signer. Proc. of Crypto'91, Springer LNCS Vol. 576, 470–484 (1992)
14. I. Damgård: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. Proc. of Crypto'91, Springer LNCS Vol. 576, 445–456 (1991)
15. M. K. Franklin, M. K. Reiter: Verifiable Signature Sharing. Proc. of Eurocrypt'95, Springer LNCS Vol. 921, 50–63 (1995)
16. S. Goldwasser, S. Micali, R. Rivest: A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. SIAM J. Computing, 17 (2), 281–308 (1988)
17. S. Goldwasser, E. Waisbard: Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. Proc. of TCC'04, Springer LNCS Vol. 2951, 77–100 (2004)

18. M. Jakobsson, K. Sako, R. Impagliazzo: Designated Verifier Proofs and their Applications. Proc.of Eurocrypt'96, Springer LNCS Vol. 1070, 142–154 (1996)
19. Aggelos Kiayias, Yiannis Tsiounis, Moti Yung: Traceable Signatures. Proc. of Eurocrypt'04, Springer LNCS Vol. 3027, 571–589 (2004)
20. F. Laguillaumie, D. Vergnaud: Time-Selective Convertible Undeniable Signatures. Proc. of CT-RSA'05, Springer LNCS Vol. 3376, 154-171 (2005)
21. C. H. Lim and P. J. Lee: Modified Maurer-Yacobi's Scheme and its Applications. Proc. of Auscrypt'92, Springer LNCS Vol. 718, 308–323 (1993)
22. D. Pointcheval, J. Stern: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptology, Vol. 13 (3), 361–396 (2000)
23. F. Zhang, K. Kim: A Universal Forgery on Araki *et al.*'s Convertible Limited Verifier Signature Scheme. IEICE Trans. Fundamentals, Vol. E86-A (2), 515–516 (2003)

A Invisibility of DS

The invisibility of DS relies on the difficulty of solving the following *ℓ-Tripartite-DCAA Problem* in connection to the xyz-trick. It is similar to a class of problems recently introduced by Laguillaumie and Vergnaud [20]:

ℓ-Tripartite-DCAA Problem: Let $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ be a bilinear group system. Given $(x_1P_1, x_2P_2, yP_1, zP_2, Q, h) \xleftarrow{\$} (\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_1 \times \mathbb{Z}_q^*$ and for some $\ell \geq 0$,

$$\big(h_i, x_1(x_2+h_i)^{-1}P_1, x_1y(x_2+h_i)^{-1}P_1\big)_{i\in[\![1,\ell]\!]} \in \big(\mathbb{Z}_q^* \times \mathbb{G}_1^2\big)^\ell$$

with $h \notin \{h_1, \ldots, h_\ell\}$, decide whether $Q = x_1yz(x_2+h)^{-1}P_1$.

We state that, assuming the hardness of the ℓ-Tripartite-DCAA problem and that of the Flexible Square Diffie-Hellman problem, the schemes DS and DST are invisible under chosen-message attack in the random oracle model.

Theorem 6 (Invisibility of DS and DST). *Let $t, q_H \in \mathbb{N}^\mathbb{N}$, $\boldsymbol{q} = (q_{\text{Sign}}, q_{\text{Confirm}}, q_{\text{Deny}}, q_{\text{Conv}}, q_{\text{Trap}}, q_{\text{Reg}}) \in [\mathbb{N}^\mathbb{N}]^6$ and $\varepsilon \in [0,1]^\mathbb{N}$. Assume there exists a $(k, t, \boldsymbol{q}, \varepsilon)$-distinguisher $\mathcal{A}$, in the random oracle model, against DS (or DST). Then there exists an algorithm $\mathcal{B}$ that solves the q_s-Tripartite-DCAA problem in the bilinear group generator* Setup *with advantage ε' and a $(k, t'', \boldsymbol{q}, \varepsilon'')$-forger $\mathcal{C}$ against DS such that*

$$\varepsilon' + (q_{\text{Confirm}} + q_{\text{Deny}} + q_{\text{Convert}}) \cdot \varepsilon'' \geq \varepsilon$$

where $\mathcal{B}$ runs in time at most $t' = (q_{\text{Reg}}+1)\cdot t$ and $\mathcal{C}$ runs in time $t'' = t + \mathcal{O}(1)$.

Proof. Assume $\mathcal{A}$ is an Inv-CMA-adversary that $(k, t, \boldsymbol{q}, \varepsilon)$-distinguishes the signatures of DS. As in the unforgeability proof, q_H represents the number of queries submitted by $\mathcal{A}$ to H since H is again viewed as a random oracle. We construct two reduction algorithms $\mathcal{B}$ and $\mathcal{C}$ that interact with $\mathcal{A}$ and respectively solve the q_s-Tripartite-DCAA problem and produce an existential forgery with time and success probability as claimed in Theorem 6.

Algorithm $\mathcal{B}$: Algorithm $\mathcal{B}$ is given public parameters $(q, P_1, P_2, g_t, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \langle\cdot,\cdot\rangle, \rho)$ generated by Setup(k) and an instance

$$\left((x_1P_1, x_2P_2, yP_1, zP_2, Q, h), \left(h_i, R_i = \frac{x_1}{x_2+h_i}P_1, S_i = \frac{x_1y}{x_2+h_i}P_1\right)_{i\in[\![1,q_s]\!]}\right)$$

in $(\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_1 \times \mathbb{Z}_q^* \times \left(\mathbb{Z}_q^* \times \mathbb{G}_1^2\right)^{q_s}$ of the q_s-Tripartite-DCAA problem. $\mathcal{B}$'s goal is to decide whether $Q = x_1yz(x_2+h)^{-1}P_1$ and $\mathcal{B}$ proceeds to use forger $\mathcal{A}$ to do so. $\mathcal{B}$ sets $X_1 = x_1P_1$, $X_2 = x_2P_2$, $Y^\star = yP_1$, initializes a counter $i = 1$ and simulates $\mathcal{A}$'s environment as follows:

Simulation of H. Same simulation as in the unforgeability proof.

Simulation of DS.Register. Each time the adversary registers a new public key $Y = y'P_1$, the reduction rewinds $\mathcal{A}$ from the beginning without changing anything but the challenge in the proof-of-knowledge of the discrete logarithm y' of Y in base P_2 (see the proof of unforgeability). Therefore, we can suppose wlog that the reduction knows the secret key of all the users registered by $\mathcal{A}$, at the expense of running $\mathcal{A}$ at most $q_{\text{Reg}}(k)$ times.

Simulation of DS.Signer.{Confirm, Deny}. If the signature has been produced by $\mathcal{B}$ in the simulation then use the same simulation as in the unforgeability proof. Otherwise, simulate a designated-verifier proof of invalidity.

Simulation of DS.Sign. Given $m \in \{0,1\}^*$ and a confirmer's public key Y, pick a random $r \xleftarrow{\$} \mathbb{Z}_q^*$. If $Y = Y^\star$ set $U = rP_2$ and $V = rS_i$. Otherwise $Y = y'P_1 \neq Y^\star$, and $\mathcal{B}$ sets $U = rP_2$ and $V = ry'R_i$. Now if $H(m, U, Y)$ is defined and is $\neq h_i$, the reduction restarts with a new value for r. Otherwise the reduction defines $H(m, U, Y) = h_i$, outputs $\sigma = (U, V)$ and increments i.

Simulation of DS.Convert.{Signer, Confirmer}. Given $Y \in \mathbb{G}$, $m \in \{0,1\}^*$ and $\sigma = (U, V) \in \mathbb{G}^2$, invoke the simulation of H on (m, U, Y). If σ has been obtained by the simulation of DS.Sign then retrieve the randomness r such that $U = rP_2$ and output $\tilde{\sigma}_A = (U, V, rX_1)$ or $\tilde{\sigma}_B = (U, V, rY)$. Otherwise, output `Invalid`.

Simulation of DS.Signer.Trapdoor. Given $Y \in \mathbb{G} \setminus \{Y^\star\}$, output $\mathcal{T} = y'X_1$ where $Y^\star = y'P_2$.

In this simulation $\mathcal{B}$ simulates perfectly $\mathcal{A}$'s environment unless at some point in time $\mathcal{A}$ queries a valid signature (U, V) not produced by $\mathcal{B}$ to the oracles DS.Signer.{Confirm, Deny} or DS.Convert.{Signer, Confirmer}. Let us denote Bad this event. We have $|\varepsilon'(k) - \varepsilon(k)| \leq \Pr(\mathsf{Bad})$ and the running time of $\mathcal{B}$ is at most $t'(k) = (q_{\text{Reg}}(k) + 1) \cdot t(k) + \mathcal{O}(1)$.

Algorithm $\mathcal{C}$: We claim that there exists an EF-CMA-adversary $\mathcal{C}$ which $(k, t'', \boldsymbol{q}, \varepsilon'')$-breaks DS, where $t'' = t + \mathcal{O}(1)$ and $\varepsilon'' \geq (q_{\text{Confirm}} + q_{\text{Deny}} + q_{\text{Convert}})^{-1}\Pr[\mathsf{Bad}]$. Basically, $\mathcal{C}$ runs $\mathcal{A}$ and outputs as a forgery one of the signatures (selected at random) queried by $\mathcal{A}$ during the Inv-CMA game, to one of the oracles DS.Signer.{Confirm, Deny} or DS.Convert.{Signer, Confirmer} which was not obtained by the oracle DS.Sign.

This directly leads to the above claims for the scheme DS and the proof extends readily to the invisibility of the scheme DST. □