Online/Offline Signatures for Low-Power Devices

Andrew Chi-Chih Yao and Yunlei Zhao

***Abstract*—When digital signature is applied on low-power devices, like smart cards, wireless sensors and RFID tags, some specific properties, e.g., better offline storage, more modular and flexible deployment, are desired. To meet these needs, a new variant of the Fiat–Shamir transformation for digital signatures, referred to as Γ-transformation, is introduced and formalized in this work. Following this new transformation approach, some new signature schemes (referred to as Γ-signatures) are presented and discussed. In particular, it is shown that the Γ-signatures for discrete logarithm problem (DLP) developed in this work combine, in essence, the advantages of both Schnorr's signature and the digital signature standard (DSS), while saving from the disadvantages of them both.**

***Index Terms*—Digital signatures, Fiat–Shamir transform, privacy preserving authentication.**

I. Introduction

DIGITAL signature is fundamental to information security, and the construction of digital signature schemes that can be provably secure is one focus of modern cryptography. A common paradigm of obtaining signatures, known as the Fiat–Shamir (FS) paradigm [14], is to collapse any Σ-protocol (which is a kind of 3-round public-coin honest verifier zero-knowledge protocol [11]) into a noninteractive scheme with hash functions that are modeled to be random oracles (RO) [8]. Roughly speaking, denote by a, e, z the first, the second and the third message of a Σ-protocol on a common input U respectively, where a and z are from the prover, e is a random challenge from the verifier, and the value a can be determined from (U, e, z). Given a message $m \in \{0,1\}^*$ to be signed, the FS-paradigm computes and outputs (e, z) as the signature, where $e = h(a, m)$ and h is a hash function. To improve the online/offline efficiency of digital signatures instantiated via the FS-paradigm, the signer can precompute and store a list of values a's.

Manuscript received July 06, 2012; revised November 07, 2012; accepted November 21, 2012. Date of publication December 11, 2012; date of current version January 03, 2013. The work of A. C.-C. Yao was supported by the National Basic Research Program of China (2007CB807900, 2007CB807901), and by the National Natural Science Foundation of China (61033001, 61061130540). The work of Y. Zhao was supported by National Natural Science Foundation of China (61070248, 61272012), and by the Innovation Project (12ZZ013) of Shanghai Municipal Education Commission. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Michel Abdalla. *(Corresponding author: Y. Zhao.)*

A. C.-C. Yao is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China (e-mail: andrewcyao@tsinghua.edu.cn).

Y. Zhao is with the Software School, Fudan University, Shanghai 200433, China, and also with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: ylzhao@fudan.edu.cn).

Digital Object Identifier 10.1109/TIFS.2012.2232653

The FS-paradigm is popular and successful in developing practical digital signatures in practice, and plays a fundamental and central role to most digital signature branches (e.g., blind signature [28], multisignature [7], forward-secure (threshold) signature [4], [5], identity-based signature [6], group and ring signature [23]). For presentation simplicity, we refer to signature schemes derived via the FS-paradigm as FS-signatures. One standing FS-signature scheme is the Schnorr's signature scheme based on the discrete logarithm problem (DLP) [30], which was identified as a provably secure one among the various variants of ElGamal scheme [20]. The formal analysis of digital signatures via the FS-paradigm in the random-oracle model, where the hash function h is modeled to be a RO, was conducted in [28] (we would also remind the reader of some critiques on security in the standard model [18]). At the core of the analysis in [28] is a forking lemma, which is further abstracted and generalized in [1].

Most FS-signature schemes enjoy very practical online efficiency. The notion of *online/offline signature* was introduced in [13]. The idea is to perform signature generation into two phases: the offline phase and the online phase. Online/offline signature schemes are useful, since in many applications the signer (e.g., a smart card or a wireless sensor) has a very limited response time once the message is presented (but it can carry out costly computations between consecutive signing requests). The online phase is typically very fast, and hence can be executed even on a weak processor. Online/offline signature schemes are particularly remarkable in applications based on storage or power limited devices (e.g., smart cards, wireless sensors, and RFID tags) [31]: the offline phase can be implemented either during the device manufacturing process or as a background computation whenever the device is connected to power. Some general transformations from any signature scheme to secure online/offline signature scheme are known (e.g., [13], [31]), but the resulting signature schemes are typically not as efficient as the signature schemes resulted directly via the FS-paradigm.

Motivation. In this work, we discuss some disadvantages of the FS-paradigm (for certain applications).

- Firstly, suppose the signer precomputes and stores a set of values $\bar{A} = \{a_1, \ldots, a_{q_s}\}$ (besides the preimages of them, e.g., the discrete logarithms for Schnorr's signatures) in the offline phase, where $q_s \geq 1$ and each a_i is of length len, $1 \leq i \leq q_s$. Then, the offline storage complexity for storing $\bar{A}$ is $q_s \cdot len$ which is quite large for most FS-signatures (e.g., for Schnorr's signature over Z_p^* for a 1024-bit prime p, $len = 1024$).
- Secondly, the set $\bar{A}$ cannot be stored in a public storage or in private at the verifier side (without sacrificing the provable security), in order to reduce the offline storage of the signer, to allow the verifier to precompute some interme-

diate values (derived from $\bar{A}$) to accelerate online signature verification, and to enable privacy-preserving authentication (for the case that $\bar{A}$ is kept in private by some designated verifier).

- Thirdly, the generation of $e = h(a, m)$ requires the knowledge of both a and m. This causes less modular and flexible deployment of FS-signatures within interactive protocols (e.g., the IKEv2 standard [21], [22]) in order to get better balanced communication flows and computational loads, where the message to be signed is generated and exchanged interactively and can be determined only in the last round. More detailed discussions are referred to Section IV-A.

Contribution. In this work, we first formalize a new family of protocols, called Γ-protocols. Γ-protocols are a special kind of Σ-protocols, but, briefly speaking, with the following main differences: (1) Γ-protocol has a special first-round message structure, which, besides the value a, additionally consists of another random value d. (2) Besides the special soundness as required for Σ-protocols, Γ-protocol additionally requires a new property, called knowledge extraction w.r.t. e-condition, which ensures knowledge extraction from any two different conversations $\{(a, d), e, z\}$ and $\{(a, d'), e', z'\}$, where $d \neq d'$, as long as (d, e, d', e') satisfy some predetermined relation called e-relation R_e. We then present and prove several practical Γ-protocols for some commonly used number-theoretic problems, say, the DLP problem and the q-th root problem (QRP).

Our new transformation method, referred to as Γ-transformation, transforms a Γ-protocol into a digital signature scheme (referred to as Γ-signature) as follows: Let f and h be two hash functions. On a message m to be signed, the signer computes $d = f(a)$, $e = h(m)$ and z, and outputs (d, z) as the signature on m. For the security of Γ-signatures, we define a new *stronger* security notion, called *strong existential unforgeability under concurrent interactive attacks*. In the security definitional game, the adversary is allowed to interact with a Γ-signature signer *concurrently and interactively*, which can in particular invoke the signer to get a list of values $\bar{D} = \{d_1, \ldots, d_{q_s}\}$ before presenting any actual message to be signed. This security captures strong unforgeability for interactive settings and for public or private precomputed $\bar{D}$. We then show that Γ-signatures are strongly unforgeable under concurrent interactive attacks in the random oracle, where only f is assumed to be random oracle *while h is still a real hash*. Specifically, we assume h to be any hash function that is collision-resistant (CR) and satisfies another property named *target one-way* (TOW). Roughly speaking, a hash function h is target one-way, if given a value taken randomly from the range of h no efficient algorithm can compute its preimage. We discuss that target one-wayness can be a quite reasonable assumption, and clarify the relationship between TOW and CR, which might be of independent interest and brings the TOW property for future investigations of hash functions. Following this new transformation approach, we present and discuss some new signature schemes referred to as Γ-signatures.

We show that Γ-transformation overcomes all the disadvantages of the FS-paradigm discussed above. We focus on the new Γ-signatures for DLP developed in this work, and compare them in detail with Schnorr's signature and the DSS scheme [15]. In essence, the Γ-signatures for DLP developed in this work combine the advantages of both Schnorr's signature and DSS, while saving from the disadvantages of them both. Finally, we conclude this work with discussions on some future research directions.

II. Preliminaries

We review preliminaries in this section, with more details contained in Appendix A.

If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; \rho)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins ρ. We let $y \leftarrow A(x_1, x_2, \ldots; \rho)$ denote the experiment of picking ρ at random and letting y be $A(x_1, x_2, \ldots; \rho)$. If $\mathcal{S}$ is a finite set then $|\mathcal{S}|$ is its cardinality, and $(x_1, \ldots, x_\nu) \leftarrow \mathcal{S}$, $\nu \geq 1$, is the operation of picking ν elements uniformly and independently from $\mathcal{S}$ (that is, each value $x_i, 1 \leq i \leq \nu$, is taken uniformly and independently from the set $\mathcal{S}$). If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. By $\Pr[E : R_1; \ldots; R_n]$ we denote the probability of event E, after the *ordered* execution of random processes $R_1, \ldots, R_n$. A string means a binary one, and for arbitrary strings e_1 and e_2, $e_1 \| e_2$ denotes the concatenation of e_1 and e_2. For presentation simplicity, we simply denote by $\{0,1\}^l \setminus \{0\}$ the set of all l-bit binary strings except 0^l. Throughout this work, the notations $\{(a, d), e, z\}$ and (a, d, e, z) are interchangeable. A function $\varepsilon(l)$ is *negligible* if for every $c > 0$ there exists an l_c such that $\varepsilon(l) < 1/l^c$ for all $l > l_c$. A hash function $h : \{0,1\}^* \rightarrow \{0,1\}^l$ is (t, ε_{cr})-collision resistant, if for any t-time algorithm A it holds that $\mathsf{Adv}^{cr}_{h,A}(1^l) = \Pr[h(x) = h(x') : (x, x') \leftarrow A(h)] < \varepsilon_{cr}$.[1]

A polynomial-time computable function $F : \{0,1\}^* \rightarrow \{0,1\}^*$ is called *one way*, if for any probabilistic polynomial-time (PPT) algorithm A there exists a negligible function $\varepsilon_A(\cdot)$ such that for all sufficiently large l, it holds that $\mathsf{Adv}^{ow}_{F,A}(1^l) = \Pr[y = f(x') : x \leftarrow \{0,1\}^l; y = f(x); x' = A(1^l, y)]] \leq \varepsilon_A(l)$. Alternatively speaking, $\mathsf{Adv}^{ow}_{F,A}(\cdot)$ is a negligible function for every PPT algorithm A. On the security parameter l we say an algorithm A (t, ε)-breaks the one-wayness of F, if A runs in time t and has $\mathsf{Adv}^{ow}_{F,A}(1^l) > \varepsilon$.

On input of the form (p, q, g, X) such that $X = g^x \bmod p$, where p, q are primes, g is an element in Z_p^* of order q and x is taken uniformly at random from Z_q^*, the discrete logarithm problem (DLP) is to compute x, and the DLP assumption says that, for any PPT algorithm A there exists a negligible function $\varepsilon_A(\cdot)$ such that for all sufficiently large security parameter $l = |q|$, it holds that $\mathsf{Adv}^{ow}_{DL,A}(1^l) = \Pr[y = g^{x'} : x \leftarrow Z_q; y = g^x; A(p, q, g, y) = x'] \leq \varepsilon_A(l)$, where the probability is taken over the random choices of (p, q, g, x) and the coins of A. The DLP problem and DLP assumption can also be defined on elliptic curves over finite fields.

Let N be an RSA modulus (i.e., the product of two large primes), and let $q < N$ be a prime. Given a value $y \in Z_N$, the

[1]To be precise, h is a keyed cryptographic hash function: $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^l$, where k is polynomial in l. The probability $\mathsf{Adv}^{cr}_{h,A}(1^l)$ is taken over the random choice of $K \leftarrow \{0,1\}^k$ and the random coins of A. For presentation simplicity, we assume h is hardwired with a key K that is chosen uniformly at random from $\{0,1\}^k$ and serves as a public parameter.

q-th root problem (QRP) over Z_N^* is to find an element $x \in Z_N$ such that $y = x^q \bmod N$. The QRP problem is assumed to be one-way (without knowing the order of Z_N^*), where the probability is taken over the random choices of (N, x, y) and the random coins of the attacker. Note that, if $q = 2$, the QRP problem is just the square root problem (modulo N) whose hardness is computationally equivalent to that of factoring N. For $q > 2$, the QRP problem can be viewed as a special case of the RSA problem, which is conjectured to be easier than factoring [10].

III. Results

A. Γ-Protocols

Definition 3.1 (Γ-Protocol): We consider a 3-round protocol $\langle P, V\rangle$ for an $\mathcal{NP}$-relation $\mathcal{R}$. On the security parameter l in unary notation and a common input U, the private input of the prover P is a string w satisfying $(U, w) \in \mathcal{R}$, where both the length of U and that of w are polynomials in l. On the security parameter l and a common input U, the interaction between $P(w)$ and the verifier V consists of three rounds:

In the first round, the prover P computes $a = f_a(r_P, U)$ and chooses a value d uniformly at random from a set denoted $\mathcal{D}$, where f_a is a polynomial-time computable function, r_P is taken uniformly at random from a set denoted R_P. The prover P sends (a, d) to the verifier V. For presentation simplicity, we require the value a be distributed uniformly over a set denoted $\mathcal{A}$.

After receiving (a, d), the verifier responds back a random challenge e in the second round, where e is taken uniformly at random from a set $\mathcal{E}$. For presentation simplicity, we assume both e and d are of length l, which also serves as the security parameter.

In the third round, the prover sends back $z = f_z(r_P, w, d, e)$, where f_z is a polynomial-time computable function and z is uniquely determined by (U, a, d, e). The verifier V applies a deterministic predication procedure Ver on (U, a, d, e, z) and accepts if and only if $Ver(U, a, d, e, z) = 1$. *We require that, for any* (U, a, d, e, z)*satisfying* $Ver(U, a, d, e, z) = 1$*, the value* a*be computed (determined) from* (U, d, e, z).

The protocol $\langle P, V\rangle$ is called a Γ-protocol, if for any sufficiently large security parameter l, it satisfies:

- Completeness. If P, V follow the protocol, the verifier always accepts.
- Perfect/statistical SHVZK (special honest verifier zero-knowledge). There exists a probabilistic polynomial-time simulator S, which on input $U \in \{0,1\}^{poly(l)}$ (where there exists an $\mathcal{NP}$-witness w such that $(U, w) \in \mathcal{R}$) and a random string $\hat{d} \leftarrow \mathcal{D}$ and an *arbitrary* string $\hat{e} \in \mathcal{E}$, outputs an accepting conversation, such that the following two probability ensembles are identical or statistically indistinguishable:

$$\{S(U, \hat{d}, \hat{e})\}_{U,\hat{d},\hat{e}}$$
$$\{a = f_a(r_P, U), \hat{d}, \hat{e}, z = f_z(r_P, w, \hat{d}, \hat{e})\}_{U,\hat{d},\hat{e}}.$$

- Knowledge extraction w.r.t. e-condition. From any common input U of length n (that is polynomial in l) and any pair of accepting conversations on input U, (a, d, e, z) and (a, d', e', z') where $(d, e) \neq (d', e')$, one can efficiently compute w such that $(U, w) \in \mathcal{R}$ under a condition w.r.t. an $\mathcal{NP}$-relation R_e, referred to as e-condition, that $R_e(d, e, d', e') = 1$. In particular, for all Γ-protocols it holds that $R_e(d, e, d', e') = 1$ for all $d = d'$ but $e \neq e'$, which then implies the special soundness property of Σ-protocol (recalled in Appendix A).

Γ-protocol versus Σ-protocol. By definition, Γ-protocol is a special case of Σ-protocol (recalled in Appendix A), and thus all results (particularly, the OR-techniques [12]) that hold for Σ-protocols hold also for Γ-protocols. The major differences between Γ-protocol and Σ-protocol are: (1) Γ-protocol has a special first round message structure, which particularly includes a random value d sent by the prover. The reason that we require such a special structure of the first round message is: when the Γ-protocol is transformed into a signature scheme, the value d is set to be $f(a)$, where f is a hash function that is modeled to be a RO in security analysis. (2) The SHVZK simulator takes both d and e as input. In particular, the SHVZK property holds w.r.t. *arbitrary* input $e \in \mathcal{E}$. (3) The knowledge extraction property considers not only different verifier challenges e and e', but also different prover messages d and d'. In particular, the e-condition is required for the case of $d \neq d'$.

Remark: The above different features of Γ-protocols (in comparison with Σ-protocols) are important to reduce the security of the signature schemes derived from them, i.e., Γ-signatures, to the security of the underlying Γ-protocols (particularly, with only one of the two hash functions is modeled to be RO). Indeed, the introduction of Γ-protocols is tailored for demonstrating the underlying derivation mechanism of Γ-signatures and for security reductions. We remark that Γ-protocols do not necessarily enjoy advantages (in efficiency and security) over Σ-protocols. However, as we shall see, the signatures derived from Γ-protocols, as is the focus of this work, can overcome the mentioned disadvantages of signatures derived from Σ-protocols.

We note that many Σ-protocols can be modified into Γ-protocols. Based on the Σ-protocols [19], [30], we present and prove Γ-protocols for DLP and QRP below.

Γ-protocol for DLP. The common input is the same as that of Schnorr's protocol [30]: $U = (p, q, g, y)$ such that $y = g^{-w} \ mod \ p$, where g is of order q. The prover's private input is $w \in Z_q^*$.

- P chooses r uniformly at random in Z_q and d uniformly at random from $\{0,1\}^l \setminus \{0\}$, where l is fixed such that $2^l < q$. P sends $a = g^r \bmod p$ and d to V.
- V chooses a challenge e uniformly at random in $\{0,1\}^l \setminus \{0\}$, and sends e to P.
- P sends $z = dr + ew \bmod q$ (respectively, $z = r + dwe \bmod q$) to V, who checks that $d \in Z_q^*$ and $g^{zd^{-1}} y^{ed^{-1}} = a \bmod p$ (respectively, $g^z y^{de} = a \bmod p$), that p, q are primes and that g, y are of order q, and accepts iff this is the case.

Theorem 3.1: The above protocols are Γ-protocols for DLP under the e-condition that: $R_e(d, e, d', e') = 1$ iff $d^{-1}e \neq d'^{-1}e' \bmod q$ (respectively, $de \neq d'e' \bmod q$).

Proof: The completeness property is direct. Note also that the value z (respectively, a) is determined by (U, a, d, e) (respectively, (U, d, e, z)).

Perfect SHVZK. Let $\mathcal{D} = \mathcal{E} = \{0,1\}^l \setminus \{0\}$. On common input y and given (d,e), where $d \leftarrow \mathcal{D}$ while e is an arbitrary string in $\mathcal{E}$, the SHVZK simulator S works as follows: It first selects z uniformly at random from Z_q, computes $a = g^{zd^{-1}} y^{ed^{-1}} \bmod p$ (respectively, $a = g^z y^{de} \bmod p$), and outputs (a,d,e,z) as the simulated transcript. Note that $a = g^{(z-we)d^{-1}}$ (respectively, $a = g^{z-wde}$), and thus the value $r = (z-we)d^{-1} \bmod q$ (respectively, $r = z - wde \bmod q$) is distributed uniformly over Z_q, from which perfect SHVZK follows.

Knowledge extraction w.r.t. e-relation. Given two accepting conversations (a,d,e,z) and (a,d',e',z'), we have that $a = g^{zd^{-1}} y^{ed^{-1}} = g^{z'd'^{-1}} y^{e'd'^{-1}} \bmod p$ (respectively, $a = g^z y^{de} = g^{z'} y^{d'e'} \bmod p$), from which we can compute $w = zd^{-1} - z'd'^{-1}/ed^{-1} - e'd'^{-1} \bmod q$ (respectively, $w = z - z'/de - d'e' \bmod q$) under the e-condition that $d^{-1}e \neq d'^{-1}e' \bmod q$ (respectively, $de \neq d'e' \bmod q$). Note that if $d = d'$ but $e \neq e'$, the e-condition always holds (recall that both e and e' are in Z_q). □

Γ-protocol for the q-th root problem (QRP). Let N be an RSA modulus and q be a prime. The common input is $U = (N,q,y)$, and the private input is $w \in Z_N^*$ such that $y = w^q \bmod N$.

- P chooses r at random in Z_N^*, and d at random from $\{0,1\}^l$ where l is fixed such that $2^l < q$, and sends $a = r^q \bmod N$ and d to V.
- V chooses a challenge e uniformly at random in $\{0,1\}^l$ and sends e to P.
- P computes $[de]_q = de \bmod q$, and sends $z = rw^{[de]_q} \bmod N$ to V, who checks that $a = z^q / y^{[de]_q} \bmod N$, that q is a prime, that $gcd(a,N) = gcd(y,N) = 1$, and accepts iff this is the case.

Theorem 3.2: The above protocol is a Γ-protocol for QRP under the e-condition that: $R_e(d,e,d',e') = 1$ iff $de \neq d'e' \bmod q$.

Proof: The completeness property is direct. Note also that the value z (respectively, a) is determined by (U,a,d,e) (respectively, (U,d,e,z)).

Perfect SHVZK. Given (U,d,e), the SHVZK simulator chooses z uniformly at random from Z_N^*, computes $a = z^q / y^{[de]_q} \bmod N$, and outputs (a,d,e,z) as the simulated transcript. Note that $a = z^q / y^{[de]_q} = (z/w^{[de]_q})^q$. Let $\hat{r} = z/w^{[de]_q} \bmod N$, we have: $a = \hat{r}^q$ and $z = \hat{r} w^{[de]_q}$. As z is taken uniformly at random from Z_N^*, $\hat{r}$ is distributed uniformly over Z_N^*, from which perfect SHVZK follows.

Knowledge-extraction w.r.t. e-condition. Given (a,d,e,z) and (a,d',e',z') such that $z^q = a y^{[de]_q} \bmod N$ and $z'^q = a y^{[d'e']_q} \bmod N$ (w.l.o.g., we can assume $[de]_q > [d'e']_q$). By letting $\hat{z} = z/z' \bmod N$ and $\Delta = [de]_q - [d'e']_q$, we have: $\hat{z}^q = y^\Delta \bmod N$. The e-condition is defined to be $R_e(d,e,d',e') = 1$ iff $\Delta \neq 0$. Note that the e-condition always holds for $d = d'$ but $e \neq e'$ (recall that both e and e' are in Z_q). Under the e-condition, we have that $gcd(\Delta, q) = 1$, and thus there exist integers α and β such that $\alpha\Delta + \beta q = 1$. From this fact, we have that $\hat{z}^{\alpha q} = y^{\alpha\Delta} = y^{1-\beta q} \bmod N$, from which we get $y = \hat{z}^{\alpha q} y^{\beta q} = (\hat{z}^\alpha y^\beta)^q \bmod N$. Thus, from (a,d,e,z) and (a,d',e',z') under the e-condition, we get $w = \hat{z}^\alpha y^\beta \bmod N$. □

B. Γ-Transformation and Γ-Signatures

Let $\mathcal{R}_F$ be a one-way $\mathcal{NP}$-relation such that $(U,w) \in \mathcal{R}_F$ if and only if $U = F(w)$, where F is a one-way function (OWF). Given a Γ-protocol for a one-way $\mathcal{NP}$-relation $\mathcal{R}_F$, the Γ-transformation conveys it into a signature scheme as follows:

- KEY GENERATION:$KeyGen$. On the security parameter l, the key-generation algorithm $KeyGen$ selects an element w of length n (that is polynomial in l) uniformly at random from the domain of the OWF F. Compute $U = F(w)$. U serves as the public-key, and w serves as the secret-key.
- SIGNATURE GENERATION:$Sign$. Let $f : \mathcal{A} \rightarrow \mathcal{D}$ and $h : \{0,1\}^* \rightarrow \mathcal{E}$ be two cryptographic hash functions.[2] On a message $m \in \{0,1\}^*$ to be signed, the signer computes $a = f_a(r_P, U)$, $d = f(a)$, $e = h(m)$, $z = f_z(r_P, w, d, e)$, and outputs (d,z) as the signature, where, as specified in Definition 3.1, r_P is taken uniformly at random from a set denoted R_P.
- SIGNATURE VERIFICATION:$Verify$. Given U, m and (d,z), the verifier first computes $e = h(m)$ and then computes a from (U,d,e,z). The verifier accepts if both $Ver(U,a,d,e,z) = 1$ and $d = f(a)$, where Ver is the verification procedure for the underlying Γ-protocol.

We refer to the signature schemes, derived by applying Γ-transformation on Γ-protocols, as Γ-signatures.

1) Online/Offline Γ-Signatures: As the value d can be computed without knowing the actual message to be signed, it can be sent to the verifier before knowing the message to be signed. This eases deployment of Γ-signatures in interactive settings, e.g., by allowing the verifier to precompute some intermediate values related to d to accelerate online signature verification, and by balancing traffic flows and computational loads. Moreover, the signer can offline precompute and store a set of values $\bar{D} = \{d_1, \ldots, d_{q_s}\}$, where $d_i = f(a_i)$, and any intermediate values that can be computed from (d_i, r_P^i, w) where $a_i = f_a(r_P^i, U)$, $1 \leq i \leq q_s$.

Furthermore, if the signer does not need to use the value d_i, but only some intermediate values precomputed from (d_i, r_P^i, w), for signature generation *in the online phase* upon receiving the actual messages to be signed, the set $\bar{D}$ can be stored in some public storage, or in private at the side of verifier (for application scenarios like RFID or wireless sensor authentications, where the verifier is a server or base station and the signer only signs messages to this designated verifier). In this case, the prover does not need to store and send the component d in signature generation, which further reduces the offline storage complexity and is important for certain application scenarios (e.g., the signer is an RFID tag, a wireless sensor, or a smart-card, etc., with very limited storage capability). Rather, the signer runs a counter ctr. For each signature query on a message m, the signer sets $ctr = ctr + 1$, $m' = m \| ctr$ and signs the message m' to get z, and sends (ctr, z) to the verifier. With ctr the verifier retrieves d_{ctr} from $\bar{D}$ and then

[2]The reason we differentiate the two hash functions f and h is that, as we shall see, we only require f to be random oracle in the security analysis. In practice, f and h can be the same.

checks (d_{ctr}, z) is a valid signature on $m\|ctr$.[3]In addition, for Γ-signatures with precomputed private $\bar{D}$, a signature of the form (ctr, z) can only be verified by the designated verifier who keeps $\bar{D}$ in private, which is useful for many application scenarios (e.g., authentications based on RFID, wireless sensors, and smart-cards) where privacy-preserving authentication is desired.

2) Γ-Signatures for DLP: Next, we present the Γ-signatures for DLP (derived by applying Γ-transformation on the above Γ-protocols for DLP). Actually, based on the special structure of the Γ-protocols for DLP, we directly describe the online/offline version of the resultant Γ-signatures. For presentation simplicity, we often omit the operations of "$\bmod q$" (for operations over Z_q^*) and "$\bmod p$" (for operations over Z_p^*).

- Public-key:$U = (p, q, y = g^{-w})$, where $w \in Z_q^*$.
- Secret-key: w.
- Offline precomputation: The signer precomputes and stores (d, dr) (respectively, (r, d, dw)), where r is taken randomly by the signer from Z_q, $a = g^r$, and $d = f(a)$. In general, the signer can offline precompute a list of values $\{(d_1, d_1 r_1), \ldots, (d_{q_s}, d_{q_s} r_{q_s})\}$ (respectively, $\{(r_1, d_1, d_1 w), \ldots, (r_{q_s}, d_{q_s}, d_{q_s} w)\}$), where $d_i = f(a_i)$ and $a_i = g^{r_i}$, $1 \le i \le q_s$. Moreover, the set of values $\bar{D} = \{d_1, \ldots, d_{q_s}\}$ can be stored in a public storage, or stored in private at the side of the verifier.
- Online signature generation: After receiving the message m to be signed, the signer computes $e = h(m)$, retrieves the prestored value (d, dr) (respectively, (r, dw)), and computes $z = dr + ew$ (respectively, $z = r + dwe$). The signer outputs (d, z) as the signature on m.
- Signature verification: Given a signature (d, z) on a message m, the verifier computes $e = h(m)$ and accepts iff $d \in Z_q^*, z \in Z_q$ and $f(g^{zd^{-1}} y^{ed^{-1}}) = d$ (respectively, $f(g^z y^{de}) = d$).

For presentation simplicity, we denote by $\Gamma - 1$ (respectively, $\Gamma - 2$) signature the resultant Γ-signature under DLP for the version of $z = dr + ew$ (respectively, $z = r + dwe$).

C. Random e-Condition and Target One-Way Hash

Recall that the knowledge-extraction property of Γ-protocol is defined w.r.t. the e-condition R_e, which is defined for arbitrary (d, e, d', e') where $(d, e) \neq (d', e')$. However, the actual security analysis of Γ-signatures (via the above Γ-transformation) will be conducted w.r.t. a random version of the e-condition, where d and d' are taken randomly from $\mathcal{D}$ while e and e' are outputs of a hash function $h : \{0,1\}^* \to \mathcal{E}$.

Definition 3.2 (Target One-Way Hash): A hash function $h : \{0,1\}^* \to \mathcal{E} \subseteq \{0,1\}^l$ is (t, ε_e)-*target one-way (TOW)* w.r.t. an e-condition R_e (and a set $\mathcal{D} \subseteq \{0,1\}^l$), if for any t-time algorithm $A = (A_1, A_2)$ it holds that $\mathsf{Adv}_{h,A}^{\mathrm{tow}}(1^l) = \Pr[R_e(d, e = h(m), d', e' = h(m')) = 0 : d \leftarrow \mathcal{D}; (m, s) \leftarrow A_1(h, d); d' \leftarrow \mathcal{D}; m' = A_2(h, d, m, d', s)] \le \varepsilon_e$, where s is some state information passed from A_1 to A_2. The function h is said to be target one-way, if $\mathsf{Adv}_{h,A}^{\mathrm{tow}}(\cdot)$ is a negligible function for every PPT adversary A.

The above TOW definition is quite general. As a special case, suppose the value e' determined by (d, e, d') under the relation R_e is distributed uniformly over $\mathcal{E}$. Then, the target one-wayness ensures that any polynomial-time algorithm can inverse only a negligible portion of elements in $\mathcal{E}$. We suggest that the corresponding TOW assumption can be quite natural and reasonable, particularly for most existing cryptographical hash functions and for all Γ-protocols considered in this work. For example, consider the e-condition for the Γ-protocol for DLP presented in Section III-A: $R_e(d, e, d', e') = 1$ iff $d^{-1}e \neq d'^{-1}e' \bmod q$. That is, $R_e(d, e, d', e') = 0$ iff $e' = d^{-1}ed' \bmod q$. As $d' \leftarrow \{0,1\}^l \setminus \{0\}$, the value e' is distributed uniformly over a subset of cardinality $2^l - 1$ in Z_q^*. To break the random e-condition in this case, an adversary A has to find m' such that $h(m') = e'$. Recall that the range of the hash function h is also $\{0,1\}^l \setminus \{0\}$. We consider two cases:

- Case-1: The value e' is not in $\{0,1\}^l \setminus \{0\}$ (but in $Z_q^* \setminus \{1, \ldots, 2^l\}$). In this case, the random e-condition holds unconditionally, and any (even power unlimited) algorithm cannot break it.
- Case-2: The value $e' \in \{0,1\}^l \setminus \{0\}$. This case implies that the adversary A can compute the preimage of a random value e' determined by the values (d, e, d') under R_e.

In general, if we view both the ranges of d and e are defined to be Z_q^*, then breaking the above random e-condition corresponds to inversing a value taken uniformly at random from the range of the hash function h.

Target one-wayness versus preimage resistance. Let $h : Dom \to \mathcal{E}$ be a hash function. Informally speaking, h is one-way (a.k.a., preimage resistant) if given $y = h(x)$, where x is taken uniformly at random from the domain Dom, it is hard to find a preimage of y. In comparison, target one-wayness informally says that it is hard to find a preimage of y, *where y is taken randomly from the range $\mathcal{E}$*. It is well known that collision resistance implies traditional one-wayness as long as $|\mathcal{E}|/|Dom|$ is small (i.e., negligible in security parameter) [29]. However, as we shall see below, target one-wayness and collision resistance can be two fundamentally different notions.

Target one-wayness versus collision-resistance. Consider a special function $h : \{0,1\}^* \to \mathcal{E}$, where for all $m \in \{0,1\}^*$, $h(m)$ is a fixed value in $\mathcal{E}$. Clearly such a function is not collision-resistant, but it can still be target one-way. On the other hand, suppose $\mathcal{E}'$ is any subset of cardinality $|\mathcal{E}|/2$ in $\mathcal{E}$. Consider another function $\hat{h} : \{0,1\}^* \to \mathcal{E}$, where $\hat{h}(m) = m$ if $m \in \mathcal{E}'$, and $h(m) = f(m)$ for $m \notin \mathcal{E}'$ where $f : \{0,1\}^* \to \mathcal{E}$ is another collision-resistant and one-way hash function. The function $\hat{h}$ is not target one-way w.r.t. certain natural e-relations (e.g., the e-relation for DLP when $\mathcal{D}$ and $\mathcal{E}$ are defined to be Z_q^*), but can still be collision-resistant. This shows that target one-wayness and collision-resistance do not imply each other in general.

However, for most existing cryptographic hash functions and for most e-relations considered in this work, it may still be reasonable to assume that collision resistance implies TOW.

[3]In general, if the signer needs only to sign messages to several, say k, servers or organizations $\{V_1, \ldots, V_k\}$, the signer can offline precompute several sets of $\bar{D}_1, \ldots, \bar{D}_k$, and stores $\bar{D}_i$ at $V_i, 1 \le i \le k$. In this general case, the signer needs to run k counters $\{ctr_1, \ldots, ctr_k\}$, where ctr_i is for synchronism on the set D_i between the signer and $V_i, 1 \le i \le k$.

With the e-relation for DLP as an example, suppose there exists an efficient algorithm A that can violate TOW of the function $h : \{0,1\}^* \rightarrow \mathcal{E} = \{0,1\}^l \setminus \{0\}$ with nonnegligible probability. Then, there exists a subset denoted $\mathcal{E}'$ in $\mathcal{E}$ such that $|\mathcal{E}'|/|\mathcal{E}|$ is nonnegligible in l and for each value $e \in \mathcal{E}'$ A can inverse e with nonnegligible probability. Then, if with nonnegligible probability we can sample in polynomial-time a value m such that $h(m) \in \mathcal{E}'$ and $h(m)$ has at least two preimages, then we can use the ability of A to violate the collision-resistance of h. In particular, supposing the function h is regular, i.e., each value in the range of h has the same number of preimages, collision resistance implies TOW in this case.

Target one-wayness in the random-oracle model. For all Γ-protocols considered in this work, we have that for any (d, e, d') there exists at most one value $e' \in \mathcal{E}$ satisfying $R_e(d, e, d', e') = 0$. For these e-conditions, supposing h is modeled to be a random oracle, then h is always target one-way. Specifically, for any adversary A who makes at most q_h queries to the RO h, $\mathsf{Adv}^{\mathrm{tow}}_{h,A} \leq q_h/|\mathcal{E}|$.

D. Strong Existential Unforgeability Under Concurrent Interactive Attack

Motivated for deploying signatures in interactive protocols or w.r.t. public/private $\bar{D}$, in this section, we model the security of Γ-signatures in concurrent interactive settings. Strong existential unforgeability under concurrent interactive attacks for a signature scheme $\prod = (KeyGen, Sign, Verify)$, *where a signature can be divided into two parts* (d, z), is defined using the following game between a challenger and a forger adversary $\mathcal{F}$.

- Setup. On the security parameter l, the challenger runs $(PK, SK) \leftarrow KeyGen(1^l)$. The public-key PK is given to adversary $\mathcal{F}$ (while the secret-key SK is kept in private).
- Suppose $\mathcal{F}$ makes at most q_s signature queries. Each signature query consists of the following steps: (1) $\mathcal{F}$ sends "Initialize" to the signer. The i-th initialization query is denoted as I_i, $1 \leq i \leq q_s$. (2) Upon the i-th initialization query, the signer responds back d_i. (3) $\mathcal{F}$ adaptively chooses the message m_i to be signed, and sends m_i to the signer. (4) The signer sends back z_i, where (d_i, z_i) is the signature on message m_i. $\mathcal{F}$ is allowed to adaptively and concurrently interact with the signer in arbitrary interleaved order. As a special case, $\mathcal{F}$ can first make q_s initialization queries, and get all the values in $\bar{D} = \{d_1, \ldots, d_{q_s}\}$ before presenting any message to be signed. We refer to the interactions for each signature generation as a session. To distinguish different concurrent sessions, we assume that the signer assigns a unique session identifier sid to each session, and each message bears the corresponding sid. In particular, it is ensured that the same d_i will never be used in two sessions.
- Output. Finally, $\mathcal{F}$ outputs a pair of m and (d, z), and wins the game if (1) $Verify(PK, m, d, z) = 1$ and (2) $(m, d, z) \notin \{(m_1, d_1, z_1), \ldots, (m_{q_s}, d_{q_s}, z_{q_s})\}$.

We define $\mathsf{AdvSig}^{\mathrm{suf-cia}}_{\prod,\mathcal{F}}(1^l)$ to be the probability that $\mathcal{F}$ wins in the above game, taken over the coin tosses of $KeyGen$ and of $\mathcal{F}$ and the signer (and the random choice of the random oracle). Then, on the security parameter l, we say a forger $\mathcal{F}$ $(t, q_s, q_f, \varepsilon)$-breaks a Γ-signature scheme $\prod$ (with respect to a RO f), if it works in time t, makes at most q_s signature queries and q_f random-oracle queries to f (for security in the standard model, the parameter q_f will be omitted) and $\mathsf{AdvSig}^{\mathrm{suf-cia}}_{\Gamma,\mathcal{F}}(1^l) \geq \varepsilon$. We say the signature scheme $\prod$ is strongly existential unforgeable, if $\mathsf{AdvSig}^{\mathrm{suf-cia}}_{\prod,\mathcal{F}}(\cdot)$ is a negligible function for every PPT forger $\mathcal{F}$.

E. Security Analysis

Next, we present the security analysis of Γ-signatures. The analysis uses the general forking lemma [7].

Lemma 3.1 (General Forking Lemma [7]): Fix an integer $\hat{q} \geq 1$ and a set $\mathcal{O}$ of size λ. Let $\mathcal{C}$ be a randomized algorithm that on input $U, d_1, \ldots, d_{\hat{q}}$ returns a pair, the first element of which is an integer in the range $0, \ldots, \hat{q}$ and the second element of which we refer to as a *side output*. Let IG be a randomized algorithm that we call the input generator. The accepting probability of $\mathcal{C}$, denoted acc, is defined as

$$\Pr[J \geq 1 : U \leftarrow IG; d_1, \ldots, d_{\hat{q}} \leftarrow \mathcal{O}; (J, \sigma) \leftarrow \mathcal{C}(U, d_1, \ldots, d_{\hat{q}})].$$

The forking algorithm $F_{\mathcal{C}}$ associated with $\mathcal{C}$ is the randomized algorithm that takes input U and proceeds as follows:

Algorithm 1 The Forking Algorithm $F_c(U)$

```
Pick coins ρ for C at random
d_1, ..., d_q̂ ← O
(J, σ) ← C(U, d_1, ..., d_q̂; ρ)
if J = 0 then
    return (0, ⊥, ⊥)
end if
d'_J, ..., d'_q̂ ← O
(J', σ') ← C(U, d_1, ..., d_{J-1}, d'_J, ..., d'_q̂; ρ)
if J = J' ∧ d_J ≠ d'_J then
    return (1, σ, σ')
else
    return (0, ⊥, ⊥)
end if
```

Let

$$frk = \Pr\left[b = 1 : U \leftarrow \mathsf{IG}; (b, \sigma, \sigma') \leftarrow F_{\mathcal{C}}(U)\right].$$

Then

$$frk \geq acc\left(\frac{acc}{\hat{q}} - \frac{1}{\lambda}\right).$$

□

In the following analysis, for presentation simplicity, we assume the underlying Γ-protocols are of perfect SHVZK (which is particularly the case for all Γ-protocols considered in this work). The extension to statistical Γ-protocols is direct. In the following theorem, let t_{ex} denote the time to extract the secret-key SK from two conversations (a, d, e, z) and (a, d', e', z'), where $(d, e) \neq (d', e')$ and $R_e(d, e, d', e') = 1$. Let t_s denote the time to answer a signature query, and

Setup: $(PK, SK) \leftarrow KeyGen(1^l)$, $(d_1, \cdots, d_{q_s}, d_{q_s+1}, \cdots, d_{q_s+q_f}) \leftarrow (\mathcal{D})^{q_s+q_f}$. Let $\hat{q} = q_s + q_f$. The input to $\mathcal{C}$ is PK and $(d_1, \cdots, d_{\hat{q}})$. $\mathcal{C}$ provides $\mathcal{F}$ with a random tape, and runs $\mathcal{F}$ by playing the role of signer of public-key PK. $\mathcal{C}$ also maintains an array T that is initiated to be empty.

RO queries: Upon the i-th RO query denoted Q_i from $\mathcal{F}$, $1 \le i \le q_f$, $\mathcal{C}$ first checks whether $f(Q_i)$ has been defined by checking whether there exists a record of the form (j, Q_i, α) in T where $1 \le j \le \hat{q}$. If yes, $\mathcal{C}$ just sends back the already defined value $f(Q_i) = \alpha$ to $\mathcal{F}$. Otherwise, $\mathcal{C}$ defines $f(Q_i) = d_{q_s+i}$, stores the record $(j = q_s + i, Q_i, f(Q_i) = d_{q_s+i})$ in T, and sends d_{q_s+i} to $\mathcal{F}$.

Signature query simulation: Upon the i-th initialization query I_i, $1 \le i \le q_s$, $\mathcal{C}$ responds back d_i. Upon receiving the i-th message m_i to be signed, $\mathcal{C}$ computes $e_i = h(m_i)$, and runs the SHVZK simulator $S(PK, d_i, e_i)$ (of the underlying Γ-protocol) to get a simulated transcript (a_i, d_i, e_i, z_i). Note that, due to the perfect SHVZK property, we have that the value a_i is distributed uniformly over the set $\mathcal{A}$. Then, $\mathcal{C}$ checks whether $f(a_i)$ has been defined, by checking whether there exists a record of the form $(j, a_i, \cdot)$ in T. If yes, $\mathcal{C}$ halts and outputs $(0, \perp)$ (this event is referred to as Bad-1); otherwise, $\mathcal{C}$ defines $f(a_i) = d_i$, stores the record (i, a_i, d_i) in T, and sends back z_i to $\mathcal{F}$.

When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs (m, d, z). Letting $e = h(m)$ and letting a be the value computed from (PK, d, e, z), we have (1) (a, d, e, z) is an accepting conversation on PK and (2) there exists a record of the form (J, a, d) in T, i.e, $d = f(a)$, where $1 \le J \le \hat{q}$.

F2. $(m, d, z) \notin \{(m_1, d_1, z_1), \cdots, (m_{q_s}, d_{q_s}, z_{q_s})\}$.

F3. For any $m_i \ne m$, $1 \le i \le q_s$, it holds that $e = h(m) \ne h(m_i) = e_i$.

If the above conditions do not hold, $\mathcal{C}$ halts and outputs $(0, \perp)$. Otherwise, $\mathcal{C}$ further checks whether $J \in \{1, \cdots, q_s\}$. If $J \in \{1, \cdots, q_s\}$, $\mathcal{C}$ outputs $(J, (a, d, e, z), (a, d, e_J, z_J))$, where $e = h(m)$ and $e_J = h(m_J)$. If $J \notin \{1, \cdots, q_s\}$ (but in $\{q_s + 1, \cdots, \hat{q}\}$), $\mathcal{C}$ outputs $(J, (a, d, e, z))$.

Fig. 1. Reduction from OWF to Γ-signature forgeries.

$\hat{q} = q_s + q_f$. We assume the time in answering a RO query is $O(1)$.

Theorem 3.3: Let $\prod = (KeyGen, Sign, Verify)$ be a Γ-signature scheme derived by applying the Γ-transformation on a Γ-protocol w.r.t. an e-relation R_e and two hash functions $h : \{0,1\}^* \rightarrow \mathcal{E}$ and $f : \mathcal{A} \rightarrow \mathcal{D}$, and let F be a OWF that dominates key generation, i.e., for any (PK, SK) output by $KeyGen$ it holds that $PK = F(SK)$. Suppose f is a random oracle, and assume h is (t, ε_{cr})-collision resistant and (t, ε_{tow})-target one-way, and assume there exists a forger $\mathcal{F}$ who can $(t, q_s, q_f, \varepsilon)$-break the strong existential unforgeability of $\prod$ under concurrent interactive attack. Then, there exists an algorithm that can (t', ε')-break the one-wayness of F, where $t' = 2t + 2q_s t_s + t_{ex} + O(q_f)$ and

$$\varepsilon' \ge acc \left(\frac{acc}{\hat{q}} - \frac{1}{|\mathcal{D}|} \right) - \varepsilon_{tow}, \tag{1}$$

where,

$$acc \ge \left(1 - \frac{q_s(2q_f + q_s - 1)}{2|\mathcal{A}|} \right) \left(\varepsilon - \frac{1}{|\mathcal{D}|} - \varepsilon_{cr} \right). \tag{2}$$

Proof (of Theorem 3.3): On the security parameter l, supposing a forger $\mathcal{F}$ can $(t, q_s, q_f, \varepsilon)$-break the Γ-signature scheme $\prod$, we build an efficient solver $\mathcal{B}$ for the one-way function F. Namely, run $(PK, SK) \leftarrow KeyGen(1^l)$ and give PK to $\mathcal{B}$, whose task is then to compute SK. To apply the general forking lemma [7], we first build an algorithm $\mathcal{C}$, which is presented in Fig. 1 (page 8).

Denote by acc the probability that $\mathcal{C}$ outputs (J, σ) for $J \ge 1$, where $\sigma = ((a, d, e, z), (a, d, e_J, z_J))$ (respectively, $\sigma = (a, d, e, z)$) for $J \in \{1, \ldots, q_s\}$ (respectively, $J > q_s$).

Lemma 3.2: $acc \ge (1 - q_s(2q_f + q_s - 1)/2|\mathcal{A}|)(\varepsilon - 1/|\mathcal{D}| - \varepsilon_{cr})$.

Proof (of Lemma 3.2): We first bound the probability that the event Bad-1 occurs. Note that for each a_i generated by $\mathcal{C}$ in answering the i-th signature queries, it is distributed uniformly over the set $\mathcal{A}$. In the RO model, there are two cases to cause Bad-1 to occur:

- Case-1: For some i, $1 \le i \le q_s$, $\mathcal{F}$ ever successfully guessed the value a_i in one of its q_f random oracle queries. Thus, the probability that $\mathcal{C}$ fails in Case-1 is at most $q_f q_s / |\mathcal{A}|$.
- Case-2: For some i, $1 \le i \le q_s$, the value a_i has ever been generated in dealing with the j-th signing oracle query, $j < i$. By the birthday paradox, the probability that $\mathcal{C}$ fails in Case-2 is at most $q_s(q_s - 1)/2|\mathcal{A}|$.

Thus, we have that the event Bad-1 occurs with probability at most $\varepsilon_{bad1} \le q_s(2q_f + q_s - 1)/2|\mathcal{A}|$. Conditioned on the event Bad-1 does not occur, which is denoted as $\neg$ Bad-1 , the signature simulation by $\mathcal{C}$ is perfect. Here, a point worthy of noting is that Γ-protocol is defined w.r.t. standalone (perfect) SHVZK, but in our analysis the forger $\mathcal{F}$ concurrently interacts with the signer (or the simulator $\mathcal{C}$). As the simulation by $\mathcal{C}$ is straight-line perfect zero-knowledge in the RO model, this does not pose a problem in the concurrent setting. Specifically, by a simple hybrid argument, concurrent perfect zero-knowledge can be reduced to the standalone straight-line perfect zero-knowledge in the random-oracle model.

Suppose with probability ε, $\mathcal{F}$ outputs a successful forgery (m, d, z) in its real interactions with the signer. Let a be the value determined from $(d, e = h(m), z)$. With probability at most $1/|\mathcal{D}|$, the random oracle f has not been defined over a (i.e., there exists no a record of the form (J, a, d) in T). Also, the probability that a collision occurs (i.e., $h(m) = h(m_i)$ but $m \ne m_i$ for some i, $1 \le i \le q_s$) is at most ε_{cr}. We conclude that conditioned on $\neg$Bad-1, with probability at most $\varepsilon - 1/|\mathcal{D}| - \varepsilon_{cr}$, $\mathcal{F}$ will output (m, d, z) satisfying all the three conditions F1, F2 and F3. Thus we have $acc \ge (1 - q_s(2q_f + q_s - 1)/2|\mathcal{A}|)(\varepsilon - 1/|\mathcal{D}| - \varepsilon_{cr})$. □

Note that if $\mathcal{F}$ did not abort and $J \in \{1, \ldots, q_s\}$, as the record of (J, a, d) is put into T (i.e., $\mathcal{C}$ has defined $f(a) = d$)

only if $\mathcal{C}$ has successfully answered the J-th signature query made by $\mathcal{F}$ on a message m_J, it implies that the tuple $(m_J, a, d, e_J = h(m_J), z_J)$ is well defined. We further consider two cases: (1)$m = m_J$ and thus $(a, d, e) = (a, d, e_J)$. In this case, by the definition of Γ-protocol the last-round value z is determined by $(a, d, e) = (a, d, e_J)$, we have that $(m, a, d, e, z) = (m, a, d, e_J, z_J)$. This contradicts the fact that the forged signature (m, d, z) is valid, i.e., $(m, d, z) \notin \{(m_1, d_1, z_1), \ldots, (m_{q_s}, d_{q_s}, z_{q_s})\}$. Equation (2)$m \neq m_J$. For this case, by the condition F3 that $e = h(m) \neq h(m_J)$, we have that if $\mathcal{F}$ did not abort and $J \in \{1, \ldots, q_s\}$ we will get two conversations (a, d, e, z) and (a, d, e_J, z_J) for $e \neq e_J$ on the same public-key PK and the same first round message (a, d), from which the secret-key SK can be computed. That is, if $\mathcal{F}$ did not abort and $J \in \{1, \ldots, q_s\}$, we can directly compute the secret-key SK, and we do not need to run the forking algorithm $F_{\mathcal{C}}(PK)$ associated to $\mathcal{C}$ in this case.

On the other hand, if $\mathcal{F}$ did not abort and $J \notin \{1, \ldots, q_s\}$, the algorithm $\mathcal{B}$ runs the forking algorithm $F_{\mathcal{C}}(PK)$ associated to $\mathcal{C}$ on PK (by reviewing the key generation algorithm $KeyGen$ as the generator IG in the definition of general forking lemma). By the general forking lemma, with probability at least $frk \geq acc\,(acc/\hat{q} - 1/|\mathcal{D}|)$, $F_{\mathcal{C}}$ will return back two conversations of the form (a, d, e, z) and (a, d', e', z') on PK, where $d \neq d'$ or $e \neq e'$, from which the secret-key SK can be computed in time t_{ex} conditioned on (d, e, d', e') satisfy the random e-condition. As the hash function h is (t, ε_{tow})-target one-way, we have that with probability at least $\varepsilon' = frk - \varepsilon_{tow}$, $\mathcal{B}$ will output the secret-key SK and thus break the one-wayness of the OWF F. The running time t' of $\mathcal{B}$ is $2t + 2q_s t_s + t_{ex} + O(q_f)$. □

We did not make a quantitative comparison between the analysis in [28] for Schnorr's signature and our analysis for Γ-signatures, for the following reasons: (1) they use different forking lemmas, and (2) they proved w.r.t. different security definitions. We note that if we base the analysis of [28] (for standard existential unforgeability [17]) also on the general forking lemma [7], unforgeability (under concurrent interactive attacks) using as far as we can see, the major difference between the analyses will be the quantity ε_{tow} (that is at most $q_h/|\mathcal{E}|$ assuming h is also a RO as discussed in Section III-C) introduced on the security of the real hash function h for Γ-signatures.

A note on the RO f. Recall that, for presentation simplicity, we have assumed that the length of d and that of e are identical, i.e., both of them are l (as in practice the hash functions h and f can be the same). However, according to the above (1) and (2) specified in Theorem 3.3, supposing ε is nonnegligible, in order to conclude ε' is also nonnegligible it is actually sufficient to assume the value $1/|\mathcal{D}|$ (as well as the values ε_{cr} and ε_{tow}) is negligible in l.

With the Γ-signatures for DLP as an example, we can set $\mathcal{D} = \{0,1\}^{\kappa} \setminus \{0\}$, where $\kappa = \omega(\log l)$; that is, the output of the RO f can be set to be $\omega(\log l)$ in general, conditioned on that ε_{cr} and ε_{tow} are negligible in l. We remark that all the Γ-protocols developed in Section III-A work also for such a smaller $\mathcal{D} = \{0,1\}^{\omega(\log l)} \setminus \{0\}$. As clarified in Section III-C (page 7) on "target one-wayness in the random-oracle model", if the hash function h is also assumed to be a random oracle, target one-wayness and collision-resistance of the RO h hold automatically. Thus, supposing h is also a RO, it is okay to just set the output length of f to be $\omega(\log l)$. On the other hand, as clarified in Section III-C (page 7) on the target one-wayness implication of collision resistance (relative to $\mathcal{D}$) in the plain model, we may want $|\mathcal{D}|/|\mathcal{E}| \approx 2^{\kappa}/2^{l}$ to be nonnegligible in l (e.g., $l - \kappa = O(\log l)$). Supposing $l = 160$, we may suggest that the output length of f is set to be 128 bits in practice.

Note that, for the security of Schnorr's signature, the RO h needs to be resistant to a special kind of collision attack, referred to as prefix-fixed collision attack. Specifically, given $e = h(a = g^r, m)$, no efficient algorithm can (with nonnegligible probability) output a collision of the form (a, m') such that $h(a, m) = h(a, m')$ but $m \neq m'$; otherwise, the security of Schnorr's signature will be totally broken. However, the Γ-signatures developed in this work do not suffer from such a prefix-fixed collision attack on the underlying RO f, as only the value a is put into the input of f. the analysis of Γ-signatures for DLP employs a weaker random oracle than that of Schnorr's.

IV. Discussions, and Future Investigation

A. Γ-Signatures versus Schnorr's Signature

We make comparisons between Schnorr's signature and the Γ-signatures for DLP (namely, the $\Gamma - 1$ and $\Gamma - 2$ schemes) presented in Section III-B2.

Offline storage complexity. For implementation of Schnorr' signature over (the subgroup of order q of) Z_p^*, where typically p is of 1024 bits and q is of 160 bits, and supposing the signer precomputes q_s values of a's, the offline space complexity of Schnorr's signature is $(|p| + |q|)q_s$, which is significantly larger than that of $\Gamma - 1$ (i.e., $2|q|q_s$) and that of $\Gamma - 2$ (i.e., $3|q|q_s$). For implementations of Schnorr's signature and Γ-signatures on certain elliptic curves over finite field F_q, the value a is an elliptic curve point that is typically represented by a pair of coordinates $(x, y) \in F_q^2$. In this case, the offline space complexity of Schnorr's signature is $3|q|q_s$ that is still much larger than the complexity $2|q|q_s$ of $\Gamma - 1$.[4]

Support public or private $\bar{D}$. As discussed in Section III-B2, for $\Gamma - 1$ (respectively, $\Gamma - 2$) scheme, the signer can offline precompute $\{(d_1, d_1 r_1), \ldots, (d_{q_s}, d_{q_s} r_{q_s})\}$ (respectively, $\{(d_1, d_1 w), \ldots, (d_{q_s}, d_{q_s} w)\}$, where $d_i = f(a_i)$ and $a_i = g^{r_i}$, $1 \leq i \leq q_s$. Moreover, the set of values $\bar{D} = \{d_1, \ldots, d_{q_s}\}$ can be stored in a public storage, or stored in private at the side of the verifier. This further reduces the offline storage complexity of the signer (e.g., the offline storage complexity of $\Gamma - 1$ is reduced to be only $|q|q_s$ in this case), and allows the verifier to offline precompute some intermediate values (e.g., d^{-1}) enabling more efficient online signature verification. In particular, if the message to be signed is prepared by the verifier, the verifier can also precompute the value $y^{ed^{-1}}$ (respectively, y^{ed}). In this case, the online

[4]We also note that the presentation of an elliptic curve point can be made more concise, e.g., by just using its x-coordinate [9]. However, with such a condensed presentation of elliptic curve point, the signer needs to recover the y-coordinate when generating signature (i.e., the value $e = h(a, m)$) *in the online phase*, which incurs much additional online computational complexity and may violate the spirit of online/offline signatures. In this sense, $\Gamma - 1$ signature still enjoys better online/offline efficiency than Schnorr's signature does in this case.

signature verification (upon receiving the component z) needs only about one modular exponentiation. In addition, similar to designated verifier signatures, Γ-signatures with private $\bar{D}$ kept by the verifier also enable privacy-preserving authentication, which are useful and desirable for certain application scenarios (e.g., RFID or wireless sensor authentication). All these advantageous features of $\Gamma-1$ and $\Gamma-2$ schemes make them much preferable for deployments by power and storage limited signers (e.g., RFID tags, wireless sensors, and smart-cards).

We note that Schnorr's signature does not support these advantageous features. One may suggest for the Schnorr's signature signer to offline precompute $\bar{A}=\{a_1=g^{r_1},\ldots,a_{q_s}=g^{r_{q_s}}\}$, and store $\bar{A}$ in public (or send $\bar{A}$ to the verifier in the offline stage). However, such a solution has the following disadvantages: (1) The signer still needs to store $\bar{A}$ locally. Otherwise, when the i-th message m_i to be signed is presented, the signer needs to recover the corresponding a_i (from the public storage or from the verifier), which can cause much communication overload. (2) We do not know how to prove the security of such a solution in the random-oracle model.

Ease of deployment for interactive protocols. For Schnorr's signature, the value $e=h(a,m)$ can only be generated and sent after the message to be signed is presented. However, for Γ-signatures, the value $d=f(a)$ can be sent to the verifier before the message to be signed is known. This much eases the deployment of Γ-signatures within interactive protocols. Consider the Γ-signature based implementation of the (simplified) IKEv2 protocol [21], [22]: (1) In Round-1, player $\hat{A}$ sends $(X=g^x,d_{\hat{A}})$. (2) In Round-2, player $\hat{B}$ sends $(Y=g^y,d_{\hat{B}})$. (3) In Round-3, $\hat{A}$ sends $(\hat{A},z_{\hat{A}})$, where $(d_{\hat{A}},z_{\hat{A}})$ is $\hat{A}$'s signature on the message $MAC_K(\hat{A},X,Y)$ where K is derived from g^{xy}. (4) In Round-4, $\hat{B}$ sends $(\hat{B},z_{\hat{B}})$, where $(d_{\hat{B}},z_{\hat{B}})$ is $\hat{B}$'s signature on the message $MAC_K(\hat{B},Y,X)$. For such a Γ-signature based IKEv2 implementation, after receiving $d_{\hat{A}}$ in Round-1, $\hat{B}$ can also precompute $y^{e_{\hat{A}}d_{\hat{A}}^{-1}}$ (respectively, $y^{e_{\hat{A}}d_{\hat{A}}}$) for $\Gamma-1$ (respectively, $\Gamma-2$), where $e_{\hat{A}}=h(MAC_K(\hat{B},Y,X))$. In Round-3, after receiving $z_{\hat{A}}$, $\hat{B}$ computes $g^{z_{\hat{A}}d_{\hat{A}}^{-1}}$ (respectively, $g^{z_{\hat{A}}}$) to finish the signature verification for $\Gamma-1$ (respectively, $\Gamma-2$). The same holds for player $\hat{A}$. In comparison with the FS-signature based IKEv2 implementation, this Γ-signature aided implementation of IKEv2 protocol enjoys better balanced communication flows and computational loads.

On another variant of Schnorr's signature. In this work, we also consider another FS-paradigm variant, which transforms a 3-round public-coin protocol (a,e,z) into a signature as follows: Letting f and h be two hash functions, on a message m to be signed, it computes $d=f(a),e=h(d,m)$ and outputs (e,z) as the signature. This variant can enjoy the same offline storage complexity as $\Gamma-1$ signature, but $\Gamma-1$ signature with public or private $\bar{D}$ (as clarified above) still enjoys much better offline storage complexity. We do not focus on this variant in this work, for the following reasons: (1) We do not know how to prove its standard existential unforgeability [17] assuming one of f,h is a real hash (and only one of them is assumed to be RO). (2) Assuming both f and h to be ROs (or, alternatively, assuming to use a single RO twice with each signature generation), we note that the security reduction for this variant may be much looser than that of the original Schnorr scheme analyzed in [28], due to the birthday paradox caused by the issue of $e=h(f(a),m)$ versus $e=h(a,m)$. (3) This variant does not well support deployments in concurrent interactive settings. Specifically, the signer cannot first send e before knowing the actual message to be signed.[5] (4) This variant still does not support public or private $\bar{D}=\{d_1,\ldots,d_{q_s}\}$, as when the message m to be signed is presented the signer still needs to get back the corresponding value in $\bar{D}$.

B. Γ-Signatures versus DSS

We note all performance advantages of DSS are essentially preserved with $\Gamma-1$ and $\Gamma-2$ schemes. We also note that the techniques proposed in [27] for improving the performance of DSS in certain scenarios, e.g., signature batch verification and compression, are also applicable to our Γ-signatures for DLP. In addition, Γ-signatures have the following advantages over DSS.

Provable security. We have shown that the Γ-signatures are strongly existential unforgeable under concurrent interactive attacks in the RO model, assuming only f is a RO. The provable security of DSS (for the standard existential unforgeability under adaptive chosen message attack [17]) is still unknown, even if both f and h are assumed to be ROs.

Offline storage complexity. Supposing q_s values of a's are precomputed, the offline space complexity of $\Gamma-1$ (respectively, $\Gamma-2$) is $2|q|q_s$ (respectively, $3|q|q_s$), while that of DSS is $3|q|q_s$.

Computational efficiency in total.[6] The computational complexity of signature generation in total for both $\Gamma-1$ and $\Gamma-2$ is: one modular exponentiation, two modular multiplications, one modular addition and two hash operations. Besides essentially the same operations needed for $\Gamma-1$ and $\Gamma-2$ signature generation, the DSS signature generation additionally needs to perform one modular inversion (i.e., r^{-1}). We remark that modular inverse is a relatively expensive operation (which is typically performed by the Euclid algorithm), and is thus much preferable to dispense with (particularly for deployments with low-power device, e.g., RFID tags or wireless sensors).

The computational complexity of signature verification for $\Gamma-2$ is: one simultaneous exponentiations (which amounts to about 1.3 exponentiations [26]), one modular multiplication and two hash operations. Besides essentially the same operations, the signature verification for DSS needs to additionally perform one modular inversion and one modular multiplication.

Online signature verification. We consider the online efficiency of signature verification for the case when the value d is known to the verifier in advance, e.g., with public or private $\bar{D}$

[5]To support interactive deployments, one way is to include d also into the signature. That is, the signer can first send d, and then send (e,z) only after the message to be signed is presented. However, this way, the signature length is significantly increased (from $2|q|$ to $3|q|$).

[6]In the efficiency comparisons between Γ-signatures for DLP versus DSS, for presentation simplicity, we assume the computation of $d=f(a)$ in DSS is counted as computing a hash function. Recall that, for DSS over elliptic curves $f(a)$ simply equals the x-coordinate of a, while for DSS over Z_p^* $f(a)=a \bmod q$ which is less efficient than that of elliptic curve based DSS implementation. Nevertheless, in any case, this is not a dominant factor in the efficiency comparison.

TABLE I
COMPARISONS AMONG $\Gamma-1$, $\Gamma-2$, DSS AND SCHNORR'S

		Γ-1	Γ-2	DSS	Schnorr
Provable security	standard	$\surd$	$\surd$	$\perp$	$\surd$
	concurrent interactive	$\surd$	$\surd$	$\perp$	$\perp$
Support public or private d		$\surd$	$\surd$	$\surd$	$\times$
Support interactive deployment		$\surd$	$\surd$	$\surd$	$\times$
Offline storage	w/o. public/private d	$2\|q\| \cdot q_s$	$3\|q\| \cdot q_s$	$3\|q\| \cdot q_s$	$(\|p\|+\|q\|) \cdot q_s$
	w. public/private d	$\|q\| \cdot q_s$	$2\|q\| \cdot q_s$	$2\|q\| \cdot q_s$	$\perp$
Sign	Offline	$1t_e+1t_m+1t_H$	$1t_e+1t_m+1t_H$	$1t_e+1t_I+2t_m+1t_H$	$1t_e$
	Online	$1t_m+1t_a+1t_H$	$1t_m+1t_a+1t_H$	$1t_m+1t_a+1t_H$	$1t_m+1t_a+1t_H$
Verification		$1t_{se}+1t_I+2t_m+2t_H$	$1t_{se}+1t_m+2t_H$	$1t_{se}+1t_I+2t_m+2t_H$	$1t_{se}+1t_H$
Online verification (d and m known)		$1t_e+1t_m+1t_H$	$1t_e+1t_H$	$1t_{se}+1t_I+2t_m+1t_H$	$\perp$

as discussed above. In this case, for verifying a DSS signature (d, z), the verifier has to *online* compute $\hat{z} = z^{-1}$ (which is a relatively expensive operation as mentioned above), two modular multiplications of $e\hat{z}$ and $d\hat{z}$, and the full (hashed) simultaneous-exponentiation $f(g^{e\hat{z}}y^{d\hat{z}})$, as the value z is known to the verifier *only when the signature comes to it*. That is, previously knowing *both* the value d and e does not accelerate the online signature verification for DSS. In comparison, the knowledge of d allows the $\Gamma-1$ signature verifier to offline precompute the value d^{-1}, thus avoiding online modular inversion, *while* $\Gamma-2$*signature verification does not need to compute* d^{-1}*at all*. Moreover, suppose the message m to be signed (and thus $e = h(m)$) is also known to, or prepared by, the verifier beforehand (which can be quite common in certain scenarios). Then, the verifier of $\Gamma-1$ (respectively, $\Gamma-2$) can also offline precompute the value $y^{ed^{-1}}$ (respectively, y^{ed}), and in this case the online signature verification essentially needs only about one modular exponentiation.

Offline signature generation. Besides essentially the same other precomputations, the DSS signer needs to perform one modular inverse r^{-1} and two modular multiplications for computing dwr^{-1}, but the signer of $\Gamma-1$ (respectively, $\Gamma-2$) needs to offline perform only one multiplication dr (respectively, dw).

The comparisons among $\Gamma-1$, $\Gamma-2$, DSS and Schnorr's are also briefly summarized in Table I (page 10). In the table, denote by t_{se} the time needed for performing one simultaneous exponentiation (which amounts to about 1.3 exponentiations [26]), by t_e the time needed for performing one modular exponentiation, by t_I the time needed for performing one modular inversion, by t_m the time needed for performing one modular multiplication, by t_a the time needed for performing one modular addition, and by t_H the time needed for performing one hashing. For provable security, by "standard" we refer to the standard existential unforgeability under adaptive chosen message attack [17], while "concurrent interactive" refers to our strong existential unforgeability under concurrent interactive attacks (defined in Section III-D) assuming only one of f and h is a RO. Offline storage is for that of q_s precomputed a's. The symbol "$\perp$" stands for "unknown" or "unapplicable". From the comparisons, we can see that the Γ-signatures combine, in essence, the advantages of both Schnorr's and DSS, while saving from the disadvantages of them both.

C. Future Directions

We conclude this work by discussing some directions for future explorations.

In the security analysis for Γ-signatures, we used a real hash function h that is both CR- and TOW-secure, and a function f that is modeled to be a random oracle. A first question is to show whether Γ-signatures are still provably secure without relying the random oracle but under some reasonable (even nonstandard) assumptions on the hash function f. Also, it is interesting to investigate the target one-wayness property for existing cryptographic (collision-resistant) hash functions.

As the FS-paradigm is fundamental and central to the literature of digital signature (and its various advanced variants), another future direction is to find more applications of the Γ-transformation in other areas, e.g., blind signatures, multisignatures, and forward-secure signatures.

Abdalla, *et al.* [1] showed the minimal conditions on any 3-round public-coin protocol in order to ensure the security of the signature scheme derived via the FS-paradigm in the random-oracle model. Can we show a similar result for the Γ-transformation? Towards this goal, we have observed that the hash function h may also have to be modeled to be a random oracle (and thus, in general, the potential results in this direction may be incomparable with the results in the current work).

Finally, we notice that there exist several other variants of the FS-paradigm in the literature [2], [24], [25], for tighter security reductions and/or for obtaining lattice-based signature schemes, etc. Another interesting question is to investigate the relationship among the existing FS-paradigm variants and our proposal, and to examine whether some approaches used in [2], [24], [25] can also be applied or adapted to the transformation method proposed in this work.

APPENDIX
BASICS

Σ-Protocols:

Definition A.1 (Σ-Protocol [11]): A 3-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ-protocol for an $\mathcal{NP}$-relation $\mathcal{R}$ if the followings hold:

- Completeness. If P, V follow the protocol, the verifier always accepts.
- Special soundness. From any common input U of length n and any pair of accepting conversations on input U, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(U, w) \in \mathcal{R}$ with overwhelming probability. Here a, e, z stand for the first, the second and the third message respectively and e is assumed to be a string of length l (that is polynomially related to n) selected uniformly at random in $\mathcal{E} \subseteq \{0,1\}^l$. It is required

that the value z (respectively, a) be uniquely determined by (U, a, e) (respectively, (U, e, z)).

- Statistical/perfect special honest verifier zero-knowledge (SHVZK). There exists a probabilistic polynomial-time simulator S, which on input U (where there exists an $\mathcal{NP}$-witness w such that $(U, w) \in \mathcal{R}$) and a random challenge string $\hat{e}$, outputs an accepting conversation of the form $(\hat{a}, \hat{e}, \hat{z})$, with the distribution identical or statistically close to that of the real conversation (a, e, z) between the honest $P(w)$, V on input U.

The first Σ-protocol (for an $\mathcal{NP}$-language) in the literature can be traced back to the honest verifier zero-knowledge (HVZK) protocol for Graph Isomorphism [16] (but the name of Σ-protocol is adopted much later in [11]). A large number of practical Σ-protocols for various number-theoretic languages have been developed up to now (e.g., [19], [30]).

Σ-Protocol for DLP [30]. The following is a Σ-protocol $\langle P, V\rangle$ proposed by Schnorr [30] for proving the knowledge of discrete logarithm, $-w \in Z_q$, for a common input of the form (p, q, g, y) such that $y = g^{-w} \bmod p$, where p, q are primes and g is an element in Z_p^* of order q. Normally, the length of q, denoted $|q|$, is served as the security parameter.

- P chooses r at random in Z_q and sends $a = g^r \bmod p$ to V.
- V chooses a challenge e at random in $\{0,1\}^l$ and sends it to P. Here, l is fixed such that $2^l < q$.
- P sends $z = r + ew \bmod q$ to V, who checks that $g^z y^e = a \bmod p$, that p, q are prime and that g, y are of order q, and accepts iff this is the case.

The Fiat–Shamir Paradigm and Schnorr's Signature: Given *any* Σ-protocol (a, e, z) on common input U (which will be served as the public-key), the Fiat–Shamir paradigm collapses the Σ-protocol into a signature scheme as follows: $(a, e = h(a, m), z)$, where m is the message to be signed and h is a hash function. Note that, for actual signature schemes derived from the Fiat–Shamir paradigm, the generated signature only consists of (e, z) as the value a can be computed from (e, z). The provable security of the general Fiat–Shamir paradigm is shown by Pointcheval and Stern [28] in the random oracle model [8] (assuming h to be an idealized random oracle). At the core of the security arguments of [28] is a forking lemma, which is later abstracted and generalized in [1].

Schnorr's signature scheme. The signature scheme obtained by applying the Fiat–Shamir paradigm on the above Schnorr's Σ-protocol for DLP is referred to as Schnorr's signature scheme. Note that, for Schnorr's signature scheme, the signer can precompute and store a list of values $(a = g^r, r)$. Then, to sign a message m, it only online computes $e = h(a, m)$ and $z = r + ew \bmod q$. Upon (m, e, z) the verifier accepts if $e \in Z_q^*$ and $h(g^z y^e, m) = e$.

The Digital Signature Standard (DSS): The general structure of DSS [15] is as follows:

- Public-key: $U = (p, q, g, y)$, where $y = g^w \bmod p$ and $w \leftarrow\!\!\!- Z_q^*$.
- Secret-key: w.
- Signature generation: Let $m \in \{0,1\}^*$ be the message to be signed.
  1) Compute $a = g^r \bmod p$, where r is taken randomly from Z_q. Compute $d = f(a)$, where $f : \{0,1\}^* \rightarrow Z_q^*$ is a conversion function.[7]
  2) Compute z from the equation $h(m) = zr - dw \bmod q$, as follows:
     — Compute $\hat{r} = r^{-1}$.
     — Compute $z = (h(m) + dw)\hat{r}$, or $z = h(m)\hat{r} + dw\hat{r}$ with offline precomputed $dw\hat{r}$, where h is a hash function.
  3) Output (d, z) as the signature.
- Signature verification: Given (m, d, z) where $d, z \in Z_q^*$, the verifier verifies the signature roughly as follows:
  — Compute $e = h(m)$.
  — Compute $\hat{z} = z^{-1}$.
  — Verify $f(g^{e\hat{z}} y^{d\hat{z}}) = d$.

The DSS signer can offline precompute a list of values a's (just as in Schnorr's signature), but contrary to Schnorr's signature, the DSS signer does not need to store these precomputed values. Instead, for each precomputed value $a = g^r$, the DSS signer can offline compute $d = f(a)$, r^{-1} and dwr^{-1}, and only stores (d, r^{-1}, dwr^{-1}). *Actually, for smart-card based applications, the values (d, r^{-1}, dwr^{-1})'s can be stored during the card manufacturing process.* Note that $d, r^{-1}, dwr^{-1} \in Z_q$, while $a \in Z_p^*$. As p is typically of 1024 bits while q is of 160 bits, and assuming the signer offline precomputes q_s values of a's, then the offline storage complexity is reduced from $(|p| + |q|)q_s$ to $3|q|q_s$ in comparison with Schnorr's signature scheme.

ACKNOWLEDGMENT

The authors are much indebted to Prof. M. Bellare for many very helpful and insightful discussions and suggestions, and for permission to use them in this work. They thank the anonymous referees for their detailed and insightful comments and suggestions, which have much improved the presentation of this work. They thank B. Gong for many helpful discussions and editing assistance.

REFERENCES

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre, "From identification to signatures via the Fiat-Shamir transform: Necessary and sufficient conditions for security and forward-security," *IEEE Trans. Inf. Theory*, vol. IT-54, no. 8, pp. 3631–3646, Aug. 2008.

[2] M. Abdalla, P. Fouque, V. Lyubashevsky, and M. Tibouchi, "Tightly-secure signatures from lossy identification schemes," in *Proc. Eurocrypt, 2012 (LNCS)*, vol. 7237, pp. 572–590.

[3] M. Abdalla, S. K. Miner, and C. Namprempre, "Forward-secure threshold signature schemes," in *Proc. CT-RSA, 2001 (LNCS)*, vol. 2020, pp. 441–456.

[4] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Proc. Asiacrypt 2000 (LNCS)*, vol. 1976, pp. 116–129.

[5] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. CRYPTO 1999 (LNCS)*, vol. 1666, pp. 431–448.

[6] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," in *Proc. Eurocrypt, 2004 (LNCS)*, vol. 3027, pp. 268–286.

[7] Typically, f can simply be the "mod q" operation for implementations over Z_p^*, or the operation of just taking the x-coordinate for implementations over elliptic curves over finite field F_q.

[7] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. ACM Conf. Comput and Commun. Security*, 2006, pp. 390–399.
[8] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM Conf. Comput. and Commun. Security*, 1993, pp. 62–73.
[9] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Asiacrypt, 2001 (LNCS)*, vol. 2248, pp. 514–532.
[10] D. Boneh and R. Venkatesan, "Breaking RSA may not be equivalent to factoring," in *Proc. Eurocrypt, 1998 (LNCS)*, vol. 1403, pp. 59–71.
[11] R. Cramer, "Modular Design of Secure, yet Practical Cryptographic Protocols," Ph.D. thesis, University of Amsterdam, , 1996.
[12] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Proc. CRYPTO, 1994 (LNCS)*, vol. 839, pp. 174–187.
[13] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital sigantures," in *Proc. CRYPTO, 1989 (LNCS)*, vol. 435, pp. 263–277.
[14] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Proc. CRYPTO, 1986 (LNCS)*, vol. 263, pp. 186–194.
[15] *Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2*, FIPS Pub 186-2, U.S. Department of Commerce/National Institute of Standard and Technology, Jan. 27, 2000.
[16] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in $\mathcal{NP}$ have zero-knowledge proof systems," *J. ACM*, vol. 38, pp. 691–729, 1991.
[17] S. Goldwasser, S. Micali, and C. Rackoff, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.
[18] S. Goldwasser and Y. T. Kalai, "On the (in)security of the Fiat-Shamir paradigm," in *Proc. FOCS, 2003*, pp. 102–113.
[19] L. Guillou and J. J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," in *Proc. Eurocrypt, 1988 (LNCS)*, vol. 330, pp. 123–128.
[20] P. Horster, H. Petersen, and M. Michels, "Meta-elgamal signature schemes," in *Proc. ACM Conf. Comput. and Commun. Security, ACM Press*, 1994, pp. 96–107.
[21] C. Kaufman, Internet key Exchange (IKEv2) Protocol, The Internet Eng. Task Force: INTERNET-DRAFT, 2002.
[22] H. Krawczyk, "SIGMA: The sign-and-MAc approach to authenticated diffie-hellman protocols," in *Proc. CRYPTO, 2000 (LNCS)*, vol. 1880, pp. 546–566.
[23] M. F. Lee, N. P. Smart, and B. Warinschi, "The Fiat-Shamir transform for group and ring signature schemes," in *Proc. SCN, 2010 (LNCS)*, vol. 6280, pp. 363–380.
[24] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *Proc. Asiacrypt, 2009 (LNCS)*, vol. 5912, pp. 598–616.
[25] S. Micali and L. Reyzin, "Improving the exact security of digital signature schemes," *J. Cryptology*, vol. 15, no. 1, pp. 1–18, 2002.
[26] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1995.
[27] D. Naccache, D. M'Raihi, S. Vaudenay, and D. Raphaeli, "Can D.S.A be improved? complexity trade-offs with the digital signature standard," in *Proc. Eurocrypt, 1994 (LNCS950)*, pp. 77–85.
[28] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptology*, vol. 13, no. 2, pp. 361–396, 2000.
[29] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Proc. FSE , 2004 (LNCS)*, vol. 3017, pp. 371–388.
[30] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.
[31] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proc. CRYPTO, 1996 (LNCS)*, vol. 1109, pp. 355–367.

Andrew Chi-Chih Yao is the Dean of the Institute for Interdisciplinary Information Sciences, at Tsinghua University, Beijing. He received the B.S. degree in physics from National Taiwan University (1967), the Ph.D. degree in physics from Harvard University (1972), and the Ph.D. degree in computer science from the University of Illinois (1975).

After serving on the faculty at MIT, Stanford, U.C. Berkeley, and Princeton University, he left Princeton in 2004 to join Tsinghua University in Beijing. He is also a Distinguished Professor-at-Large at the Chinese University of Hong Kong.

Dr. Yao is recipient of the prestigious A.M. Turing Award in 2000 for his contributions to the theory of computation, including pseudorandom number generation, cryptography, and communication complexity. He is a member of the U.S. National Academy of Sciences and the Chinese Academy of Sciences.

Yunlei Zhao received the Ph.D. degree in computer science in 2004 from Fudan University, Shanghai, China.

In 2004, he joined Hewlett-Packard European Research Center, Bristol, U.K., as a Postdoc researcher. Since 2005, he worked at Fudan University, and is currently an associate professor at the Software School, Fudan University. His research interests include theory and applications of cryptography, information security, and the interplay between complexity theory and cryptography.