Divisible On-line/Off-line Signatures*

Chong-zhi Gao[1], Baodian Wei[2], Dongqing Xie[1], and Chunming Tang[3]

[1] School of Computer Science, Guangzhou University,
Guangzhou 510006, China
czgao@gzhu.edu.cn, dongqing_xie@hotmail.com

[2] Department of Electronics and Communication Engineering,
Sun Yat-sen University, Guangzhou 510275, China
weibd@mail.sysu.edu.cn

[3] Institute of Information Security, Guangzhou University,
Guangzhou 510006, China
tangcm622@hotmail.com

Abstract. On-line/Off-line signatures are used in a particular scenario where the signer must respond quickly once the message to be signed is presented. The idea is to split the signing procedure into two phases: the off-line and on-line phases. The signer can do some pre-computations in off-line phase before he sees the message to be signed.

In most of these schemes, when signing a message m, a partial signature of m is computed in the off-line phase. We call this part of signature the off-line signature token of message m. In some special applications, the off-line signature tokens might be exposed in the off-line phase. For example, some signers might want to transmit off-line signature tokens in the off-line phase in order to save the on-line transmission bandwidth. Another example is in the case of on-line/off-line threshold signature schemes, where off-line signature tokens are unavoidably exposed to all the players in the off-line phase.

This paper discusses this exposure problem and introduces a new notion: divisible on-line/off-line signatures, in which exposure of off-line signature tokens in off-line phase is allowed. An efficient construction of this type of signatures is also proposed. Furthermore, we show an important application of divisible on-line/off-line signatures in the area of on-line/off-line threshold signatures.

Keywords: Signature Schemes, Divisible On-line/Off-line Signatures, On-line/Off-line Threshold Signatures.

1 Introduction

On-line/Off-line signatures are used in a particular scenario where the signer must respond quickly once the message to be signed is presented. This notion was first introduced by Even, Goldreich and Micali in 1990 [12]. The idea of on-line/off-line signatures is to split the signing procedure into two phases. The first

* An extended abstract of this paper appears in CT-RSA 2009, LNCS 5473, Springer-Verlag, 2009.

phase is off-line: in this phase, the signer does some preparing works before the message to be signed is presented. The second phase is on-line: once the message to be signed is known, the signer utilizes the result of the pre-computation and uses a very short time to accomplish the signing procedure.

As pointed out in [12], some signature schemes such as the Fiat-Shamir [13], Schnorr [22], El-Gamal [11] and DSS [19] signature schemes can be naturally viewed as on-line/off-line signature schemes since the first step of these schemes does not depend on the given message, and can thus be carried out off-line.

Up to now, there are two general paradigms to convert any signature scheme into an on-line/off-line signature scheme. They are Even et al.'s paradigm [12] based on one time signatures and Shamir and Tauman's paradigm [23] based on trapdoor hash functions. Even et al.'s concrete implementation in [12] has a very long signature length and thus is not practical. Shamir-Tauman paradigm greatly reduces the signature length, whilst the on-line computation is fast. In PKC08, Catalano et al. [6] unified Even et al.'s paradigm and Shamir-Tauman paradigm, in the sense that they both use an ordinary signature scheme and a (weak) one time signature scheme as components[4]. Here the trapdoor hash function in Shamir-Tauman paradigm is viewed as a weak one time signature scheme. However, these two paradigms truly have different security characterizations if we consider the partial signature exposure problem described in the next subsection. See next subsection for more details.

Some recent works in on-line/off-line signatures have also been done in [21, 24, 25, 7, 17, 8, 4, 3]. These schemes aim at some specific goals such as improving the efficiency [21, 3], eliminating the random oracle model assumption [17], constructing ID-based schemes [24], constructing threshold schemes [8, 4], avoiding key exposure [7], or avoiding trapdoor hash primitives [25].

1.1 Divisible On-line/Off-line signatures

In most of the on-line/off-line signature schemes([11, 19, 21, 24, 25, 7, 17, 8, 4, 3] and some variations of [13, 22]), when signing a message m, a partial signature of m is computed in the off-line phase. We call this part of signature the off-line signature token of message m. Although the signature generation is broken into two stages, the transmission of a signature is at one time, i.e., the whole signature of a message is transmitted to the recipient at the end of the on-line phase, while nothing is transmitted in the off-line phase.

A question thus naturally arises: can the off-line signature token be transmitted to the recipient off-line? An equivalent question is: is the signature scheme still secure if the adversary is allowed to query the signing oracle with a message depending on this message's off-line signature token? Addressing this question is meaningful because in some special applications, the off-line signature tokens might be exposed in the off-line phase. For example, some signers might want to

[4] Here the "weak" means the signature scheme is unforgeable only against generic chosen message attack [16].

transmit off-line signature tokens in the off-line phase in order to save the on-line transmission bandwidth. Another example is in the case of on-line/off-line threshold signature schemes [8, 4], where off-line signature tokens are unavoidably exposed to all the players in the off-line phase.

Unfortunately, most on-line/off-line signature schemes can not be proven to be secure if their off-line signature tokens are exposed in the off-line signing phase.[5] In this paper, we introduce a new notion called *divisible on-line/off-line signatures*, in which exposure of off-line signature tokens in off-line singing phase is allowed. To exemplify this new notion, we give in appendices some on-line/off-line signature schemes extracted from existing literatures, which satisfy the new property of divisibility. This paper also presents an efficient construction satisfying the new requirement.

An informal description. Let $\mathcal{OS}$ be an on-line/off-line signature scheme. When signing a message m submitted by a requester (or generated randomly), the signer uses the signing algorithm of $\mathcal{OS}$ to obtain a signature, say Σ. Informally, we say scheme $\mathcal{OS}$ is divisible if: i) Σ can be separated into two parts Σ^{off} and Σ^{on}, where Σ^{off} is computed before the message m is known by the signer. ii) Before the signer knows the message, he can send Σ^{off} to the recipient first. In other word, the message requested to be signed in the attack game can depend on the first part of the signature. A formal definition is presented in Section 3.

An on-line/off-line signature scheme is trivially divisible if its Σ^{off} is *null*. For this reason we restrict to non-trivial divisibility in this paper. In the rest of this paper, the word divisible/divisibility usually means a non-trivial case.

Existing Schemes with divisibility. Some existing on-line/off-line signature schemes are listed in Table 1 to show whether they can be proven divisible. We can see that some schemes are divisible such as Scheme Schnorr-OS and Even et al.'s scheme. However, most schemes like Shamir and Tauman's general paradigm can not be proven to have this property, at least using currently known methods.

It is worthwhile noting that Even et al.'s paradigm, which uses an one time signature scheme as a component, is divisible; whereas Shamir and Tauman's general paradigm cannot be proven divisible because it only uses a weak one time signature scheme.

Remark 1. We argue that El-Gamal signature scheme cannot be proven divisible using the technique in [11]. In short, the simulated hash oracle $H(\cdot)$ in security proof should not set the value of $H(m\|\Sigma^{\text{off}})$ to a value pre-determined in off-line phase, because that could lead to a hash collision if another $m'\|\Sigma^{\text{off}}$ is also requested to the hash oracle before Σ^{off} is used.

Motivations. Considering the exposure problem might be interesting by itself. Besides, there are two main reasons to consider the divisibility of an on-line/off-line signature scheme:

1. *To save the on-line bandwidth.* If an on-line/off-line scheme is divisible, the signer can send the off-line part of the signature in the off-line phase instead of

[5] However at present we also cannot present a substantial attack against these schemes when exposure problem exists.

Schemes	Divisible?	Note
Fiat-Shamir [13]	No	
El-Gamal [11]	No	
DSS [19]	No	
Boneh-Boyen [3]	No	
Shamir and Tauman's paradigm (general) [23]	No	Some specific constructions can be proven divisible. See Appendix A,B.
Xu et al.'s scheme [24]	No	It seems divisible. However a deeper analysis shows it is not.
Chen et al.'s scheme [7]	No	
Even et al.'s scheme [12]	Yes	It has a long signature length.
CMTW-OS	Yes	See Appendix A. It is extracted from [8].
BCG-OS	Yes	See Appendix B. It is extracted from [4].
Schnorr-OS	Yes	It's a variant of Schnorr signature scheme [22]. See Appendix C.

Table 1. Some on-line/off-line signature schemes. The second column shows whether they can be proved to be divisible using existing methods.

in the on-line phase. This reduces the on-line bandwidth of the communication channel.

Remark 2. For example, the signer can pre-compute a series of off-line signature tokens and sends these tokens when the communication channel is not busy. Alternatively, the signer may store these off-line tokens in the form of a DVD/CD and send the disk to the recipient directly since these off-line tokens do not depend on the messages to be later signed. At the same time, to ensure the one-to-one correspondence of the off-line tokens with the on-line ones, we can append a digital label to each off-line/on-line signature token. Note that doing this does not much increase the on-line signature length. For example, labels of 15 bit each can distinguish more than 32,000 signature tokens whereas each on-line signature token is at least 160 bits up to date. Furthermore, the index in the labels can be reused while the unused off-line signature tokens are exhausted.

2. *To construct on-line/off-line threshold signatures.* An on-line/off-line threshold signature ($\mathcal{OTS}$) scheme [8, 4] is a threshold signature scheme [9, 10] which can be partitioned into off-line and on-line phases. There are two main approaches to prove the unforgeability of a threshold signature scheme: the direct reduction approach (e.g., the part of reduction to the one-more-discrete-log assumption of Theorem 2 in [8], and the part of reduction to the discrete log assumption of Theorem 1 in [4]) and the simulation approach(e.g., [15, 20, 14]). In the direct reduction approach, the security of a threshold signature scheme is directly reduced to the hardness of an underlying hard problem such as the factoring problem or the discrete log problem. In the simulation approach, the security of a threshold signature scheme is reduced

to the unforgeability of its underlying signature scheme (This reduction approach is called *simulation*, and the property which guarantees the success of simulation is called the *simulatability* of a threshold signature scheme). In essence, the two approaches are the same, in the sense that the security is reduced to the hardness of an underlying hard problem in the end. However, if the underlying signature scheme is known to be unforgeable, the simulation approach will simplify the proof.
In Section 5 we prove that if an $\mathcal{OTS}$ scheme is simulatable from a divisible on-line/off-line signature scheme $\mathcal{DOS}$, then the unforgeability of $\mathcal{OTS}$ can be reduced to that of $\mathcal{DOS}$. This provides a theoretical basis for securely constructing an $\mathcal{OTS}$ scheme through the simulation approach.

Related work. The notion of divisible on-line/off-line signatures is first explicitly given in this paper, but the original idea goes back to [8, 4]. When proving the unforgeability of an on-line/off-line threshold signature scheme, the authors noticed that the off-line simulation of the scheme should not depend on the message to be signed. From [8, 4], we extract two on-line/off-line signature schemes(CMTW-OS and BCG-OS, see Appendix A,B), which can be proven divisible using the same proof techniques in [8, 4]. Besides, some existing schemes can also be proven divisible. They include Even et al.'s paradigm [12] and Scheme Schnorr-OS (a variant of Schnorr signature scheme, see Appendix C). Even et al.'s work has already contained a proof for their scheme's divisibility. By a new proof given in Appendix C, Scheme Schnorr-OS can also be proven divisible.

Scheme CMTW-OS and BCG-OS are both based on Shamir and Tauman's hash-sign-switch paradigm [23], which utilizes trapdoor hash functions. But Shamir-Tauman paradigm itself cannot be proven divisible. However, as in CMTW-OS and BCG-OS, if the specific trapdoor hash functions used can be viewed as a fully secure one time signature scheme, Shamir-Tauman paradigm can be unified again into Even et al.'s general paradigm, in the sense that these two paradigms both uses an one time signature scheme as a component and thus can be proven divisible.

1.2 Our Contribution

In this paper, we first explicitly give and exemplify the notion of divisible on-line/off-line signatures. Furthermore, without resorting to the random oracle model, we present an efficient divisible scheme, which is based on Boneh and Boyen(BB)'s signature scheme [3]. Compared to divisible schemes extracted from [8, 4], it does not rely on another signature scheme's security and is more efficient. Finally, an application to on-line/off-line threshold signatures is presented. We show that based on a divisible on-line/off-line signature scheme, an on-line/off-line threshold signature scheme can be proven unforgeable if it is simulatable.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we give some preliminaries. Section 3 gives the security model of divisible on-line/off-line signatures.

Section 4 presents an efficient construction whose security is proven in the standard model. An important application to on-line/off-line threshold signature schemes is introduced in Section 5. Section 6 concludes the paper with some discussions.

2 Preliminaries

2.1 Notations and Definitions

We denote by $\mathbb{N}$ the set of natural numbers, and by $\mathbb{Z}$ the set of integers. If $k \in \mathbb{N}$, we denote by 1^k the concatenation of k ones and by $\{0,1\}^k$ the set of bitstrings of bitlength k. By $\{0,1\}^*$, we denote the set of bitstrings of arbitrary bitlength. "PPT" is an abbreviation for "probabilistic polynomial-time" and "$\|$" represents the concatenation operation.

If S is a set, then the notation $x \stackrel{\text{R}}{\leftarrow} S$ denotes that x is selected randomly from the set S. Similarly, $x \in_R S$ denotes x is a random element of S. If $\mathcal{A}$ is an algorithm, by $\mathcal{A}(\cdot)$ we denote that $\mathcal{A}$ receives only one input. If $\mathcal{A}$ receives two inputs we write $\mathcal{A}(\cdot,\cdot)$ and so on. If $\mathcal{A}(\cdot)$ is a probabilistic algorithm, $y \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\ldots}(x_1,x_2,\ldots)$ means that on input $x_1,x_2,\ldots$ and with access to oracles $\mathcal{O}_1,\mathcal{O}_2,\ldots$, $\mathcal{A}$'s output is y. When a oracle $\mathcal{O}$ is written as $\mathcal{O}(x_1,..,x_t,\cdot,..)$, we denote that the oracle's first t inputs are implicitly given by previously executed algorithms and the other inputs should be explicitly given by the algorithm which queries the oracle. If $p(\cdot,\cdot,\ldots)$ is a predicate, the notation $\Pr[p(x,y,\ldots) : x \stackrel{\text{R}}{\leftarrow} S; y \stackrel{\text{R}}{\leftarrow} T;\ldots]$ denotes the probability that $p(x,y,\ldots)$ will be true after the ordered execution of the algorithms $x \stackrel{\text{R}}{\leftarrow} S, y \stackrel{\text{R}}{\leftarrow} T,\ldots$, etc.

Definition 1 (Negligible Function). *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for all $c > 0$, $\epsilon(k) < 1/k^c$ for all sufficiently large k.*

Definition 2 (Discrete Logarithm Assumption). *Let $p \in \{0,1\}^k$ be a prime. Let $\mathbb{G}$ be a group of order p and let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$. Solving the discrete logarithm problem in $\mathbb{G}$ is to compute x, given $h = g^x \in \mathbb{G}$ where x is randomly selected in $\mathbb{Z}_p$. The discrete logarithm assumption in $\mathbb{G}$ states that the discrete logarithm problem is hard to solve, i.e., for any PPT algorithm $\mathcal{A}$, the following probability is negligible in k.*

$$\epsilon(k) = \Pr[\mathcal{A}(\mathsf{descr}(\mathbb{G}), g, h) = x : x \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p; h \leftarrow g^x]$$

where $\mathsf{descr}(\mathbb{G})$ is a description of $\mathbb{G}$ which contains the value of p and other group parameters.

Definition 3 (One-More-Discrete-Log Assumption [1]). *Let $p \in \{0,1\}^k$ be a prime. Let $\mathbb{G}$ be a group of order p and let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$. Define $DL_g(\cdot)$ as a oracle which on input $h \in \mathbb{G}$, returns $\mathrm{dl}_g h \in \mathbb{Z}_p$ (the discrete logarithm of h to the base of g).*

Solving the n-DL problem is to compute $x_1, x_2, \ldots, x_{n+1}$, with access to the oracle $DL_g(\cdot)$ at most n times, given $h_1 = g^{x_1}, \ldots, h_{n+1} = g^{x_{n+1}} \in \mathbb{G}$ where $x_i (i = 1, \ldots, n+1)$ are randomly selected in $\mathbb{Z}_p$. The one-more-discrete-log assumption in $\mathbb{G}$ states that n-DL problem is hard to solve for any $n \in \mathbb{N}$, i.e., for any $n \in \mathbb{N}$ and any PPT algorithm $\mathcal{A}$ with access to oracle $DL_g(\cdot)$ at most n times, the following probability is negligible in k.

$$\epsilon(k) = \Pr\left[\begin{array}{l}\mathcal{A}^{DL_g(\cdot)}(\mathsf{descr}(\mathbb{G}), g, h_1, \ldots, h_{n+1}) = (x_1, \ldots, x_{n+1}) : \\ x_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_p, \ldots, x_{n+1} \xleftarrow{\mathrm{R}} \mathbb{Z}_p; h_1 \leftarrow g^{x_1}, \ldots, h_{n+1} \leftarrow g^{x_{n+1}}\end{array}\right]$$

where $\mathsf{descr}(\mathbb{G})$ is a description of $\mathbb{G}$ which contains the value of p and other group parameters.

Definition 4 (Bilinear Paring). *Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic group of prime order p. A bilinear pairing on $(\mathbb{G}, \mathbb{G}_T)$ is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which has the following properties:*

1. *Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$.*
2. *Non-degenerate: $e(u, v) \neq 1$ for some $u, v \in \mathbb{G}$. Here 1 denotes the identity element in $\mathbb{G}_T$.*
3. *Computable: paring $e(u, v)$ can be efficiently computed for all $u, v \in \mathbb{G}$.*

A group $\mathbb{G}$ satisfying above definition is called a bilinear group.

For generality, one can set $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where $\mathbb{G}_1 \neq \mathbb{G}_2$. An efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ can convert this general case to the simple case where $\mathbb{G}_1 = \mathbb{G}_2$.

Definition 5 (SDH Assumption [3]). *Let $\mathbb{G}$ be a group of prime order p where $p \in \{0, 1\}^k$. Let g be a generator of $\mathbb{G}$ and let x be a random element in $\mathbb{Z}_p^*$. Suppose $q \in \mathbb{N}$ is polynomial bounded in the security parameter k. Solving the q-SDH problem in $\mathbb{G}$ is to find a pair $(c, g^{1/(x+c)})$ where $c \in \mathbb{Z}_p \backslash \{-x\}$, given a $(q+1)$-tuple $(g, g^x, g^{(x^2)}, \ldots, g^{(x^q)})$. The q-SDH assumption in group $\mathbb{G}$ states that the q-SDH problem in $\mathbb{G}$ is hard to solve, i.e., for any PPT algorithm $\mathcal{A}$, the following probability is negligible in k.*

$$\epsilon(k) = \Pr[\mathcal{A}(g, g^x, g^{(x^2)}, \ldots, g^{(x^q)}) = (c, g^{1/(x+c)}) : x \xleftarrow{\mathrm{R}} \mathbb{Z}_p^*]$$

The SDH assumption in group $\mathbb{G}$ states that q-SDH problem is hard to solve for any polynomial bounded $q \in \mathbb{N}$.

The following lemma states that given a q-SDH problem instance $(g, g^x, \ldots, g^{(x^q)})$, we can construct a new 1-SDH problem instance (h, h^x) with $q-1$ known solutions $(c_i, s_i = h^{1/(x+c_i)})$ where any new solution reveals a solution to the original problem instance. Using the same technique in Lemma 9 of [3], this lemma can be easily proved.

Lemma 1. *There exists a PPT algorithm Γ which satisfies:*

– *Its inputs are:*
  1. $\mathsf{descr}(\mathbb{G})$. *A description of a group* $\mathbb{G}$ *with prime order* p.
  2. $(g, g^x, \ldots, g^{(x^q)})$. *A q-SDH problem instance in Group* $\mathbb{G}$ *where* $q \in \mathbb{N}$.
  3. $c_1, \ldots, c_{q-1} \in \mathbb{Z}_p \backslash \{-x\}$.
– *It outputs a PPT algorithm* Δ *and a tuple* $(h, u, s_1, \ldots, s_{q-1}) \in (\mathbb{G}\backslash\{1\})^{q+1}$ *which satisfy:*
  1. $u = h^x$.
  2. $s_i = h^{1/(x+c_i)}$, *i.e.,* (c_i, s_i) $(1 \leq i \leq q-1)$ *are solutions of the 1-SDH problem instance* (h, h^x).
  3. *By using Algorithm* Δ, *any new solution* $(c^*, s^*) \neq (c_i, s_i)$ *for the 1-SDH problem instance* (h, h^x) *reveals a solution to the original instance,i.e., on inputs* $(c^*, s^*) \in (\mathbb{Z}_p\backslash\{-x\}) \times (\mathbb{G}\backslash\{1\})$ *where* $(c^*, s^*) \neq (c_i, s_i)$ *for all* $i \in \{1, \ldots, q-1\}$ *and* $s^* = h^{1/(x+c^*)}$, Δ *can output a pair* $(c, g^{1/(x+c)}) \in (\mathbb{Z}_p\backslash\{-x\}) \times (\mathbb{G}\backslash\{1\})$ *in polynomial time.*

3 Security Model

We give the security model of divisible online/offline signatures and some security notions.

3.1 Syntax

A divisible online/offline signature scheme ($\mathcal{DOS}$) is a tuple of algorithms (KeyGen, $\mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}, \mathsf{Ver}$).

– $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$. The Key generation algorithm, a PPT algorithm which on input a security parameter $k \in \mathbb{N}$, outputs a public/private key pair (pk, sk).
– $(\Sigma_i^{\text{off}}, St_i) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk)$. The i-th ($i \in \mathbb{N}$) execution of the off-line signing algorithm, a PPT algorithm which on input a private key, outputs a (public) off-line signature token Σ_i^{off} and a (secret) state information St_i. The state information is kept secret and will be passed to the i-th execution of the on-line signing algorithm.
– $\Sigma_i^{\text{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk, St_i, m_i)$. The i-th ($i \in \mathbb{N}$) execution of the on-line signing algorithm, a PPT algorithm which on input sk, a state information St_i and a message m_i, outputs an on-line signature token Σ_i^{on}. The signature for m_i is defined as $\Sigma_i = (\Sigma_i^{\text{off}}, \Sigma_i^{\text{on}})$.
– $0/1 \leftarrow \mathsf{Ver}(pk, m, \Sigma)$. The verification algorithm, a PPT algorithm which on input the public key pk, a message m and a signature Σ, outputs 0 or 1 for reject or accept respectively.

Completeness: It is required that if $(\Sigma^{\text{off}}, St) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk)$ and $\Sigma^{\text{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk, St, m)$, then $\mathsf{Ver}(pk, m, \Sigma) = 1$ for all (pk, sk) generated by $\mathsf{KeyGen}(1^k)$.

Remark 3. $\mathsf{Sign}^{\mathsf{off}}$ and $\mathsf{Sign}^{\mathsf{on}}$ can be viewed as sub-algorithms of a complete signing algorithm. For simplicity, we use the notation $(\Sigma^{\text{off}}, \Sigma^{\text{on}}) \leftarrow (\mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}})(sk, m)$ to denote such a complete signing process: $(\Sigma^{\text{off}}, St) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk)$ and $\Sigma^{\text{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk, St, m)$.

3.2 Security Notion

In the following, we define a security notion for a divisible on-line/off-line signature scheme, which is an extension of the standard security definition for ordinary signature schemes [16].

EU-CMA: For a divisible on-line/off-line signature scheme $\mathcal{DOS}$, existential unforgeability against adaptive chosen message attacks (EU-CMA) is defined in the following game. This game is carried out between a challenger and an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is allowed to make queries to an off-line signing oracle $\mathsf{Sign}^{\mathsf{off}}(sk)$ and an on-line signing oracle $\mathsf{Sign}^{\mathsf{on}}(sk, St, \cdot)$ defined in Section 3.1. We assume that if $\mathcal{A}$ makes the i-th on-line signature query then it has already made the i-th off-line signature query. This requirement is reasonable since the signer always executes his i-th off-line signature signing before his i-th on-line signing. If in the random oracle model, $\mathcal{A}$ is also allowed to make queries to a hash oracle $h(\cdot)$ which on input a message in $\{0,1\}^*$, outputs a hash value of this message. The attack game is as follows:

1. The challenger runs KeyGen on input 1^k to get (pk, sk). pk is sent to $\mathcal{A}$.
2. On input $(1^k, pk)$, $\mathcal{A}$ is allowed to query the oracles $\mathsf{Sign}^{\mathsf{off}}(sk)$, $\mathsf{Sign}^{\mathsf{on}}(sk, St, \cdot)$ (and $h(\cdot)$ if in the random oracle model) polynomial times. The i-th state information St_i of $\mathsf{Sign}^{\mathsf{on}}$, which is kept secret from the adversary, is passed from the i-th execution of $\mathsf{Sign}^{\mathsf{off}}(sk)$.
3. $\mathcal{A}$ outputs a pair (m, Σ).

The adversary wins the game if the message m has never been queried to the on-line signing oracle $\mathsf{Sign}^{\mathsf{on}}(sk, St, \cdot)$ and $\mathsf{Ver}(pk, m, \Sigma) = 1$ holds. Let $\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}}$ be the *advantage* of the adversary $\mathcal{A}$ in breaking the signature scheme, i.e.,

$$\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}} = \Pr\left[\begin{matrix}\mathsf{Ver}(pk, m, \Sigma) = 1 : (pk, sk) \leftarrow \mathsf{KeyGen}(1^k); \\ (m, \Sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}^{\mathsf{off}}(sk), \mathsf{Sign}^{\mathsf{on}}(sk, St, \cdot), \eta(\cdot)}\end{matrix}\right]$$

where

$$\eta(\cdot) = \begin{cases} null & \text{(in the standard model)} \\ h(\cdot) & \text{(in the random oracle model)} \end{cases}$$

and $\mathcal{A}$ has never requested the signature of m from the on-line signing oracle. The probability is taken over the internal coin tosses of the algorithms KeyGen, $\mathsf{Sign}^{\mathsf{off}}$, $\mathsf{Sign}^{\mathsf{on}}$ and $\mathcal{A}$.

In detail, if $\mathcal{A}$ makes q_{off} off-line signing queries and q_{on} on-line singing queries, $\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}}$ is defined as:

$$\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}} = \Pr\left[\begin{array}{l} \mathsf{Ver}(pk, m, \Sigma) = 1 \text{ and } m \neq m_i \text{ for all } i \in \{1, \ldots, q_{\mathrm{on}}\} : \\ (pk, sk) \leftarrow \mathsf{KeyGen}(1^k); \\ (\Sigma_1^{\mathrm{off}}, St_1) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk); \\ \cdots\cdots\cdots \\ (\Sigma_{q_{\mathrm{off}}}^{\mathrm{off}}, St_{q_{\mathrm{off}}}) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk); \\ m_1 \leftarrow \mathcal{A}^{\eta}(pk, \Sigma_1^{\mathrm{off}}, \ldots, \Sigma_{q_{\mathrm{off}}}^{\mathrm{off}}); \\ \Sigma_1^{\mathrm{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk, St_1, m_1); \\ \cdots\cdots\cdots \\ m_{q_{\mathrm{on}}} \leftarrow \mathcal{A}^{\eta}(pk, \Sigma_1^{\mathrm{off}}, \Sigma_1^{\mathrm{on}}, m_1, \ldots, \Sigma_{q_{\mathrm{on}}}^{\mathrm{off}}, \Sigma_{q_{\mathrm{on}}+1}^{\mathrm{off}}, \ldots, \Sigma_{q_{\mathrm{off}}}^{\mathrm{off}}); \\ \Sigma_{q_{\mathrm{on}}}^{\mathrm{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk, St_{q_{\mathrm{on}}}, m_{q_{\mathrm{on}}}); \\ (m, \Sigma) \leftarrow \mathcal{A}^{\eta}(pk, \Sigma_1^{\mathrm{off}}, \Sigma_1^{\mathrm{on}}, m_1, \ldots, \Sigma_{q_{\mathrm{on}}}^{\mathrm{off}}, \Sigma_{q_{\mathrm{on}}}^{\mathrm{on}}, m_{q_{\mathrm{on}}}, \ldots, \Sigma_{q_{\mathrm{off}}}^{\mathrm{off}}) \end{array}\right]$$

Definition 6. *An adversary $\mathcal{A}$ $(t, q_{\mathrm{off}}, q_{\mathrm{on}}, \epsilon)$-breaks a divisible online/offline signature scheme $\mathcal{DOS}$ (in the standard model) if $\mathcal{A}$ runs in time at most t, makes at most q_{off} queries to the off-line signing oracle, at most q_{on} queries to the on-line signing oracle, and $\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}}$ is at least ϵ.*

An adversary $\mathcal{A}$ $(t, q_{\mathrm{off}}, q_{\mathrm{on}}, q_{\mathrm{h}}, \epsilon)$-breaks a divisible online/offline signature scheme $\mathcal{DOS}$ in the random oracle model if $\mathcal{A}$ runs in time at most t, makes at most q_{off} queries to the off-line signing oracle, at most q_{on} queries to the on-line signing oracle, at most q_{h} queries to the hash oracle, and $\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}}$ is at least ϵ.

A divisible on-line/off-line signature scheme $\mathcal{DOS}$ is EU-CMA secure if for every PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A},\mathcal{DOS}}$ is negligible.

Difference to the standard definition. The security definition of an ordinary on-line/off-line signature scheme is in the framework of the standard EU-CMA definition [16], where the adversary is only allowed to query the oracle $\mathsf{Sign}(sk, \cdot)$ (and a hash oracle if in the random oracle). In other word, in the attack game of an ordinary scheme, the off-line signature token is returned to the adversary only *after* the message to be signed is submitted, whereas in the game for a divisible scheme, the adversary obtains the off-line signature token of a message *before* he submits this message.

Thus, the unforgeability defined above is stronger than the unforgeability defined as usual for ordinary on-line/off-line signatures. Note, however, that the unforgeability defined as usual is enough for the applications where off-line signature tokens are not exposed in the off-line signing phase.

4 A Divisible On-line/Off-line Signature Scheme Based on The SDH Assumption

In this section, we propose an efficient divisible on-line/off-line signature scheme whose security is proven in the standard model. This scheme is based on Boneh and Boyen(BB)'s signature scheme [3].

4.1 Construction

Let $\mathbb{G}$ be a bilinear group of prime order p, where p's bit-length depends on the security parameter. Assume the message space is $\mathbb{Z}_p$. Note that using a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, one can extend the message domain to $\{0,1\}^*$. The new divisible on-line/off-line signature scheme is defined as $\mathsf{SDH\text{-}OS} = (\mathsf{KeyGen}, \mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}, \mathsf{Ver})$, where

- KeyGen. Pick a random generator $g \in \mathbb{G}$. Choose random $x, y, z \in_R \mathbb{Z}_p^*$, and compute $X = g^x \in \mathbb{G}\backslash\{1\}$, $Y = g^y \in \mathbb{G}\backslash\{1\}$ and $Z = g^z \in \mathbb{G}\backslash\{1\}$. Also compute $v = e(g, g) \in \mathbb{G}_T\backslash\{1\}$. The public key is (g, X, Y, Z, v). The private key is (x, y, z).
- $\mathsf{Sign}^{\mathsf{off}}$. (The i-th run). Choose a random $\theta \in \mathbb{Z}_p\backslash\{-x\}$. Compute $\sigma = g^{\frac{1}{(x+\theta)}}$ where $\frac{1}{(x+\theta)}$ is the inverse of $(x + \theta)$ in $\mathbb{Z}_p^*$. Store the state information θ. Output the off-line signature token σ.
- $\mathsf{Sign}^{\mathsf{on}}$. (The i-th run, on a message m). Retrieve from the memory the i-th state information θ. Compute $r, w \in \mathbb{Z}_p$ such that:

  $$m + yr + zw = \theta.$$

  (This can be done by first selecting a random $r \in \mathbb{Z}_p$, and computing $w = (\theta - m - yr)z^{-1} \mod p$.) Output the on-line signature token (r, w).
- Ver. Given a message $m \in \mathbb{Z}_p$ and a signature (σ, r, w), verify that whether $e(\sigma, Xg^mY^rZ^w) = v$.

Remark 4. To reduce the on-line signing cost, we can move the selection of r and computing $y \cdot r$ to the off-line phase. Thus, the on-line signing requires only 1 modular multiplication in $\mathbb{Z}_p$.

Completeness: Note that

$$\begin{aligned} e(\sigma, Xg^mY^rZ^w) &= e(g^{1/(x+\theta)}, g^{x+m+yr+zw}) \\ &= e(g^{1/(x+\theta)}, g^{x+\theta}) \\ &= e(g, g) = v \end{aligned}$$

Thus the proposed scheme satisfies the property of completeness.

4.2 Security

Theorem 1. *The divisible on-line/off-line signature scheme* SDH-OS *is EU-CMA secure, provided that the SDH assumption holds in Group* $\mathbb{G}$.

Proof. We prove this theorem by contradiction. Assume there exists an algorithm $\mathcal{A}$ which $(t, q_{\text{off}}, q-1, \epsilon)$ breaks the unforgeability of SDH-OS in the game defined in Section 3. Then we construct an algorithm $\mathcal{B}$ which breaks the q-SDH problem in polynomial time with a non-negligible probability $\epsilon' \geq \frac{\epsilon}{3} - \frac{q-1}{p}$.

Without loss of generality, we assume that $\mathcal{A}$ makes q_{off} off-line signing queries, and makes $q-1$ on-line signing queries on messages $\{m_i\}_{i\in\{1,\ldots,q-1\}}$ where $q-1 \leq q_{\text{off}}$. Let $\{(\sigma_i, r_i, w_i)\}_{i\in\{1,\ldots,q-1\}}$ be the $q-1$ full signatures returned by the signing oracle. At the end of $\mathcal{A}$'s attack game, $\mathcal{A}$ outputs a valid forgery (σ_*, r_*, w_*) on a new message m_* with probability at least ϵ. We can see one of the following cases, which cover all types of successful attacks of $\mathcal{A}$, must hold with probability at least $\epsilon/3$:

Case 1: $h^{m_*}Y^{r_*}Z^{w_*} \neq h^{m_i}Y^{r_i}Z^{w_i}$ for all $i \in \{1, \ldots, q-1\}$.
Case 2: $h^{m_*}Y^{r_*}Z^{w_*} = h^{m_i}Y^{r_i}Z^{w_i}$ for some $i \in \{1, \ldots, q-1\}$, and $r_* \neq r_i$.
Case 3: $h^{m_*}Y^{r_*}Z^{w_*} = h^{m_i}Y^{r_i}Z^{w_i}$ for some $i \in \{1, \ldots, q-1\}$, and $r_* = r_i$, but $w_* \neq w_i$.

Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$. Algorithm $\mathcal{B}$ is given a q-SDH problem instance $(g, g^\tau, g^{(\tau^2)}, \ldots, g^{(\tau^q)})$. To solve this problem instance, $\mathcal{B}$ selects a list of elements $c_1, \ldots, c_{q-1} \in_R \mathbb{Z}_p$. We may assume $c_i + \tau \neq 0$ for all $i \in \{1, \ldots, q-1\}$, or else $\mathcal{B}$ has already obtained τ and thus the q-SDH problem is solved. Next, $\mathcal{B}$ feeds Algorithm Γ in Lemma 1 with inputs $\mathsf{descr}(\mathbb{G})$, $(g, g^\tau, g^{(\tau^2)}, \ldots, g^{(\tau^q)})$, $(c_1, \ldots, c_{q-1})$ to get an algorithm Δ and $(h, u, s_1, \ldots, s_{q-1}) \in (\mathbb{G}\backslash\{1\})^{q+1}$. Note that as described in Lemma 1, $u = h^\tau$ and $s_i = h^{1/(\tau+c_i)}$. Algorithm $\mathcal{B}$ computes $v = e(h, h)$ and proceeds for each above case respectively as follows.

[CASE 1.]

Setup: Algorithm $\mathcal{B}$ selects $y, z \in_R \mathbb{Z}_p^*$ and sends to $\mathcal{A}$ a public key (h, X, Y, Z, v) where X is set to u, Y is set to h^y, and Z is set to h^z.

Simulating the Signing Oracle (Off-line): Upon the i-th query, if $1 \leq i \leq q-1$, $\mathcal{B}$ returns $\sigma_i = s_i$ as the i-th off-line signature token; else if $q \leq i \leq q_{\text{off}}$, $\mathcal{B}$ just returns a random element in $\mathbb{G}\backslash\{1\}$.

Simulating the Signing Oracle (On-line): Upon the i-th$(1 \leq i \leq q-1)$ query input m_i, $\mathcal{B}$ selects $r_i \in_R \mathbb{Z}_p$, sets $w_i = (c_i - m_i - yr_i)z^{-1} \mod p$, and outputs (r_i, w_i) as the i-th online signature token. It can be verified that (σ_i, r_i, w_i) is a valid signature on the message m_i.

Output: The simulated off-line/on-line singing oracles are identical to the real ones for $\mathcal{A}$. If Algorithm $\mathcal{A}$ outputs a valid forgery $(m_*, \sigma_*, r_*, w_*)$ satisfying the condition in Case 1, then we get a new solution (c_*, σ_*) for the 1-SDH problem instance (h, h^τ) where $c_* \stackrel{\text{def}}{=} m_* + yr_* + zw_*$. This is because from the verification equation we can get $\sigma_* = h^{1/(\tau+c_*)}$ and from $h^{m_*}Y^{r_*}Z^{w_*} \neq$

$h^{m_i}Y^{r_i}Z^{w_i}$ for all i we can get $c_* \neq c_i$ for all i. From Lemma 1, the original q-SDH problem instance can be solved in polynomial time by Algorithm Δ. Therefore if Case 1 occurs with probability at least $\epsilon/3$, $\mathcal{B}$ can successfully solve the original q-SDH problem instance with the same probability.

[CASE 2.]

Setup: Algorithm $\mathcal{B}$ selects $x, z \in_R \mathbb{Z}_p^*$ and sends to $\mathcal{A}$ a public key (h, X, Y, Z, v) where X is set to h^x, Y is set to u, and Z is set to h^z.

Simulating the Signing Oracle (Off-line): Upon the i-th query, if $1 \leq i \leq q-1$, $\mathcal{B}$ selects $r_i \in_R \mathbb{Z}_p^*$, and returns $\sigma_i = (s_i)^{\frac{1}{r_i}}$ as the i-th off-line signature token; else if $q \leq i \leq q_{\text{off}}$, $\mathcal{B}$ just returns a random element in $\mathbb{G}\backslash\{1\}$.

Simulating the Signing Oracle (On-line): Upon the i-th$(1 \leq i \leq q-1)$ query input m_i, $\mathcal{B}$ sets $w_i = (c_i r_i - x - m_i)z^{-1} \mod p$, and outputs (r_i, w_i) as the i-th online signature token. It can be verified that (σ_i, r_i, w_i) is a valid signature on the message m_i.

Output: From $\mathcal{A}$'s view, the simulated oracles are indistinguishable to the real ones for $\mathcal{A}$. In particular, the only difference is that in the simulation r_i is uniformly distributed in $\mathbb{Z}_p^*$ whereas in the real world r_i is uniformly distributed in $\mathbb{Z}_p$. Thus for one signature the statistical difference is $1/p$ and for the whole game the difference is at most $(q-1)/p$. If Algorithm $\mathcal{A}$ outputs a valid forgery $(m_*, \sigma_*, r_*, w_*)$ satisfying the condition in Case 2, then for some i, $h^{m_*}Y^{r_*}Z^{w_*} = h^{m_i}Y^{r_i}Z^{w_i}$ and $r_* \neq r_i$ hold. Algorithm $\mathcal{B}$ can check to find this i and get $\tau = \text{dl}_h u = \text{dl}_h Y = [(w_i - w_*)z + (m_i - m_*)] \cdot (r_* - r_i)^{-1} \mod p$. Therefore if Case 2 occurs with probability at least $\epsilon/3$ in the real world, $\mathcal{B}$ can successfully solve the original q-SDH problem instance with probability at least $\epsilon/3 - (q-1)/p$.

[CASE 3.]

Setup: Algorithm $\mathcal{B}$ selects $x, y \in_R \mathbb{Z}_p^*$ and sends to $\mathcal{A}$ a public key (h, X, Y, Z, v) where X is set to h^x, Y is set to h^y, and Z is set to u.

Simulating the Signing Oracle (Off-line): Upon the i-th query, if $1 \leq i \leq q-1$, $\mathcal{B}$ selects $w_i \in_R \mathbb{Z}_p^*$, and returns $\sigma_i = (s_i)^{\frac{1}{w_i}}$ as the i-th off-line signature token; else if $q \leq i \leq q_{\text{off}}$, $\mathcal{B}$ just returns a random element in $\mathbb{G}\backslash\{1\}$.

Simulating the Signing Oracle (On-line): Upon the i-th$(1 \leq i \leq q-1)$ query input m_i, $\mathcal{B}$ sets $r_i = (c_i w_i - x - m_i)y^{-1} \mod p$, and outputs (r_i, w_i) as the i-th online signature token. It can be verified that (σ_i, r_i, w_i) is a valid signature on the message m_i.

Output: The argument is similar to that of Case 2. Finally Algorithm $\mathcal{B}$ can get $\tau = \text{dl}_h u = \text{dl}_h Z = [(r_i - r_*)y + (m_i - m_*)] \cdot (w_* - w_i)^{-1} \mod p$ for some i with probability at least $\epsilon/3 - (q-1)/p$, where $(m_*, \sigma_*, r_*, w_*)$ is a valid forgery output by $\mathcal{A}$ satisfying $h^{m_*}Y^{r_*}Z^{w_*} = h^{m_i}Y^{r_i}Z^{w_i}$ and $w_* \neq w_i$.

To sum up, there exists an algorithm $\mathcal{B}$, which can break the original q-SDH problem instance with probability at least $\epsilon/3 - (q-1)/p$, in polynomial time. This contradicts the q-SDH assumption and thus the theorem is proved. ■

The purpose of introducing $'z'$. Introducing an additional trapdoor z to the BB's original scheme enables the off-line signing oracle to generate the off-line signature token without knowing the message. Here $f(m,r,w) \stackrel{\text{def}}{=} h^m Y^r Z^w$ plays a role of "double-trapdoor hash function" in the scheme. Boneh and Boyen mentioned in [3] that the exposure of the off-line tokens (and the unused state informations) causes no harm if these tokens will not subsequently used to create signatures. However we note that for a divisible on-line/off-line signature scheme, an exposed (and unused) token also should can be used.

4.3 Comparison and Discussion

We compare our new scheme SDH-OS with some known divisible on-line/off-line signature schemes in Table 2. To achieve the same security level, we assume the parameter p in our new scheme and Schemes CMTW-OS, BCG-OS and Schnorr-OS are all k-bit long. When using an elliptic curves with $k = 160$, our scheme has the same security level with a 1024-bit key RSA signature [3]. In this case, our scheme has a 160-bit off-line signature length and a 320-bit on-line signature length. In comparison, we omit additions in the signing algorithm.

To our knowledge, the most efficient divisible on-line/off-line signature scheme is Scheme Schnorr-OS. However, its security proof is based on the random oracle model(ROM). Our scheme preserves all advantages of BB's original scheme: its security is proven in the standard model; its overall computational cost of signing is only one scalar exponentiation in the group $\mathbb{G}$ (i.e., roughly k squarings and $k/2$ multiplications[6] in $\mathbb{G}$), which is comparable to Scheme Schnorr-OS and is superior to other schemes whose security is proved in the standard model. Our new scheme's on-line signing requires only 1 modular multiplication in $\mathbb{Z}_p$. This is very efficient and comparable to other three schemes.

5 An Application to On-line/Off-line Threshold Signatures

Gennaro et al. [15] has proved that if a threshold signature scheme is simulatable, then its unforgeability can be reduced to the unforgeability of its underlying signature scheme. This provides a way to simplify the security proof of a threshold signature scheme. However, this result cannot be applied to on-line/off-line threshold signature schemes. Here we provide an extended result in Theorem 2 for on-line/off-line threshold signature schemes. This theorem essentially states that a sufficient condition for the security reduction of an on-line/off-line threshold signature scheme is that the simulatability is "divisible".

5.1 On-line/Off-line Threshold Signature Schemes

A threshold scheme [9, 10] is a distributed protocol which can tolerate a certain number of faults. These faults, which may be communication failures or malicious

[6] Suppose g_i are in some group $\mathbb{G}$, e_i are all k-bit random values and t is small compared to k. By using a variant of the "square-and-multiply" method for exponentiation(Algorithm 14.88, [18]), computing $g_1^{e_1} g_2^{e_2} \dots g_t^{e_t}$ requires roughly k squarings and $(1 - \frac{1}{2^t})k$ multiplications in $\mathbb{G}$.

Schemes	$\mathsf{Sign}^{\mathsf{off}}$	$\mathsf{Sign}^{\mathsf{on}}$	Ver	Signature Size Off-line/ On-line	Assumptions
New Scheme	k sq. in $\mathbb{G}$ $\frac{k}{2}$ mult. in $\mathbb{G}$	1 mult. in $\mathbb{Z}_p$	k sq. in $\mathbb{G}$ $\frac{7}{8}k$ mult. in $\mathbb{G}$ 1 pairing	k bits / $2k$ bits	q-SDH
CMTW-OS (Appendix A)	k sq. in $\mathbb{G}$ $\frac{3}{4}k$ mult. in $\mathbb{G}$ 1 stand. sig	1 mult. in $\mathbb{Z}_p$	k sq. in $\mathbb{G}$ $\frac{3}{4}k$ mult. in $\mathbb{G}$ 1 stand. ver	1 stand.sig / k bits	Sig, one-more-discrete-log
BCG-OS (Appendix B)	k sq. in $\mathbb{G}$ $\frac{7}{8}k$ mult. in $\mathbb{G}$ 1 stand. sig	1 mult. in $\mathbb{Z}_p$	k sq. in $\mathbb{G}$ $\frac{7}{8}k$ mult. in $\mathbb{G}$ 1 stand. ver	1 stand.sig / $2k$ bits	Sig, discrete log
Schnorr-OS (Appendix C)	k sq. in $\mathbb{G}$ $\frac{k}{2}$ mult. in $\mathbb{G}$	1 mult. in $\mathbb{Z}_p$	k sq. in $\mathbb{G}$ $\frac{3}{4}k$ mult. in $\mathbb{G}$	k bits/ k bits	ROM, one-more-discrete-log

Table 2. Comparisons amongst divisible on-line/off-line signature schemes. The word "stand." refers to operations or signature length of the underlying standard signature scheme. "Sig" in the assumption column means the security also depends on the security of the underlying standard signature scheme. Abbreviations used are: "sq." for squaring, and "mult." for multiplication.

faults, are modeled as an adversary which controls players to halt or divert from the protocol. In a formal definition[15], the ability of fault tolerance is called robustness of a protocol.

Definition 7 (On-line/Off-line Threshold Signatures). *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of n players. An on-line/off-line threshold signature scheme ($\mathcal{OTS}$) is a tuple of algorithms* $(\mathsf{T\text{-}KeyGen}, \mathsf{T\text{-}Sign}^{\mathsf{off}}, \mathsf{T\text{-}Sign}^{\mathsf{on}}, \mathsf{Ver})$.

- $(pk, sk, sk_1, \ldots, sk_n) \leftarrow \mathsf{T\text{-}KeyGen}(1^k)$. *The threshold key generation algorithm. It is a distributed PPT algorithm which on input a security parameter $k \in \mathbb{N}$, outputs a public/secret key pair (pk, sk) and key shares $sk_j (j = 1, .., n)$ of sk where sk is known to nobody and sk_j is only known to $P_j \in \mathcal{P}$.*
- $(\Sigma_i^{\mathrm{off}}, st_{i,1}, \ldots, st_{i,n}) \leftarrow \mathsf{T\text{-}Sign}^{\mathsf{off}}(sk_1, \ldots, sk_n)$. *The i-th execution of the off-line signing algorithm. It is a distributed PPT algorithm which on input the secret key shares $sk_j (j = 1, .., n)$, outputs a secret state information $st_{i,j}$ for each player P_j and an off-line signature token Σ_i^{off}, which is known to all players.*
- $\Sigma_i^{\mathrm{on}} \leftarrow \mathsf{T\text{-}Sign}^{\mathsf{on}}(sk_1, \ldots, sk_n, st_{i,1}, \ldots, st_{i,n}, m_i)$. *The i-th execution of the on-line signing algorithm. It is a distributed PPT algorithm which on input $sk_j, st_{i,j}$ of P_j and a message m_i, outputs an on-line signature token Σ_i^{on}. Here the $st_{i,j}$ of P_j is passed from the i-th execution of Algorithm* $\mathsf{T\text{-}Sign}^{\mathsf{off}}$. *The signature for m_i is defined as $\Sigma_i = (\Sigma_i^{\mathrm{off}}, \Sigma_i^{\mathrm{on}})$.*
- $0/1 \leftarrow \mathsf{Ver}(pk, m, \Sigma)$. *The verification algorithm. It is a PPT algorithm which on input the public key pk, a message m and a signature Σ, outputs 0 or 1 for* reject *or* accept *respectively.*

Remark 5. $\mathsf{T\text{-}Sign}^{\mathsf{off}}$ and $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ can be viewed as sub-algorithms of a complete signing algorithm. For simplicity, we use the notation $(\Sigma^{\mathrm{off}}, \Sigma^{\mathrm{on}}) \leftarrow (\mathsf{T\text{-}Sign}^{\mathsf{off}},$

$\mathsf{T\text{-}Sign}^{\mathsf{on}})(sk_1, \ldots, sk_n, m)$ to denote such a complete signing process: $(\Sigma^{\text{off}}, st_1, \ldots, st_n) \leftarrow \mathsf{Sign}^{\mathsf{off}}(sk_1, \ldots, sk_n)$ and $\Sigma^{\text{on}} \leftarrow \mathsf{Sign}^{\mathsf{on}}(sk_1, \ldots, sk_n, st_1, \ldots, st_n, m)$.

The Adversary. We assume that an adversary can corrupt up to t of the n players in the group. After a player is corrupted, all his incoming and outgoing messages, together with his private state information, are monitored by the adversary. All the information that the adversary sees in a protocol is called the *view* of the adversary in that protocol. In addition to *eavesdropping* adversaries, an adversary can also be *halting* or *malicious* [14], which means that the adversary may cause a corrupted player to halt or divert from the protocol. Adversaries can also be categorized as *static* or *adaptive* [5], based on whether the adversary chooses his victims before the attack begins or during it.

Robustness. Robustness of Scheme $\mathcal{OTS}$ means that the scheme will compute a correct output even in the presence of halting or malicious faults. Namely, even the inputs of corrupted players to the algorithm $\mathsf{T\text{-}KeyGen}$, $\mathsf{T\text{-}Sign}^{\mathsf{off}}$ and $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ are absent or wrong, a robust scheme $\mathcal{OTS}$ can still be successfully executed. Robustness is important for a threshold scheme, however we only focus on another important security notion called unforgeability, which is described as follows.

EU-CMA. For an on-line/off-line threshold scheme $\mathcal{OTS}$, existential unforgeability against adaptive chosen message attacks(EU-CMA) is defined in the following game. The adversary $\mathcal{A}$, who corrupts up to t players, is allowed to make queries to an off-line signing oracle $\mathsf{T\text{-}Sign}^{\mathsf{off}}(sk_1, \ldots, sk_n)$ and an on-line signing oracle $\mathsf{T\text{-}Sign}^{\mathsf{on}}(sk_1, \ldots, sk_n, st_1, \ldots, st_n, \cdot)$ defined above. We assume that if $\mathcal{A}$ makes the i-th on-line signature query then it has already made the i-th off-line signature query. Let $\mathcal{B}$ be the set of currently corrupted players. The attack game is as follows:

1. All players run $\mathsf{T\text{-}KeyGen}$ on input 1^k to get $(pk, sk, sk_1, \ldots, sk_n)$. Let $\mathsf{View}_{\mathcal{A}}^{\mathsf{T\text{-}KeyGen}}$ be $\mathcal{A}$'s view in this phase, which includes pk, $\{sk_j | P_j \in \mathcal{B}\}$ and other information such as all internal coins of players in $\mathcal{B}$.
2. $\mathcal{A}$ is allowed to arouse $\mathsf{T\text{-}Sign}^{\mathsf{off}}(sk_1, \ldots, sk_n)$ and $\mathsf{T\text{-}Sign}^{\mathsf{on}}(sk_1, \ldots, sk_n, st_1, \ldots, st_n, \cdot)$ polynomial times. In an execution of Algorithm $\mathsf{T\text{-}Sign}^{\mathsf{on}}$, P_j's secret state information st_j is automatically passed from the same round execution of $\mathsf{T\text{-}Sign}^{\mathsf{off}}$, and the query message may depend on $\mathcal{A}$'s previous obtained views, i.e., the s-th message selected to query $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ can depend on all i-th off-line/on-line signing view for $i = 1, \ldots, s-1$, the s-th off-line signing view and the key generation view $\mathsf{View}_{\mathcal{A}}^{\mathsf{T\text{-}KeyGen}}$. At the end of this phase, $\mathcal{A}$ gets different messages signed.
3. $\mathcal{A}$ outputs a forgery (m, Σ).

The adversary wins the game if the message m has never been queried to the oracle $\mathsf{T\text{-}Sign}^{\mathsf{on}}(sk_1, \ldots, sk_n, st_1, \ldots, st_n, \cdot)$ and $\mathsf{Ver}(pk, m, \Sigma) = 1$ holds. Let $\mathrm{Adv}_{\mathcal{A},\mathcal{OTS}}$ be the *advantage* of the adversary $\mathcal{A}$ in breaking the signature scheme $\mathcal{OTS}$, i.e.,

$$\mathrm{Adv}_{\mathcal{A},\mathcal{OTS}} = \Pr\left[\begin{array}{l}\mathsf{Ver}(pk,m,\Sigma)=1 : (pk,sk,sk_1,\ldots,sk_n) \leftarrow \mathsf{T\text{-}KeyGen}(1^k);\\ (m,\Sigma) \leftarrow \mathcal{A}^{\mathsf{T\text{-}Sign^{off}}(sk_1,\ldots,sk_n),\mathsf{T\text{-}Sign^{on}}(sk_1,\ldots,sk_n,st_1,\ldots,st_n,\cdot)}\end{array}\right]$$

where $\mathcal{A}$ has never requested m to the on-line signing oracle and the probability is taken over the internal coin tosses of the algorithm $\mathsf{T\text{-}KeyGen}$, $\mathsf{T\text{-}Sign^{off}}$, $\mathsf{T\text{-}Sign^{on}}$ and $\mathcal{A}$.

Definition 8 (EU-CMA). *A scheme $\mathcal{OTS}$ is EU-CMA secure if for every PPT adversary $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A},\mathcal{OTS}}$ is negligible.*

5.2 The Simulation Theorem

Intuitively, the simulatability of a threshold scheme means that an adversary gains nothing useful except the public output while executing the algorithms in the scheme. This property enables the security reduction from the unforgeability of the threshold signature scheme to the unforgeability of its underlying signature scheme[14, 15]. However for an on-line/off-line threshold signature scheme, the definition of simulatability should be adapted to the situation where partial signature exposure problem exists.

Definition 9 (Simulatability). *An on-line/off-line threshold signature scheme $\mathcal{OTS} = (\mathsf{T\text{-}KeyGen}, \mathsf{T\text{-}Sign^{off}}, \mathsf{T\text{-}Sign^{on}}, \mathsf{Ver})$ is simulatable from a divisible on-line/off-line signature scheme $\mathcal{DOS} = (\mathsf{KeyGen}, \mathsf{Sign^{off}}, \mathsf{Sign^{on}}, \mathsf{Ver}_{\mathcal{DOS}})$ if the following properties hold for any PPT adversary $\mathcal{A}$ against $\mathcal{OTS}$:*

1. *Consistent distribution.*
   - *The key pair (pk, sk) generated by $\mathsf{T\text{-}KeyGen}(1^k)$ has the same distribution as the pair generated by $\mathsf{KeyGen}(1^k)$.*
   - *The signature $(\Sigma^{\mathrm{off}}, \Sigma^{\mathrm{on}})$ generated by $(\mathsf{T\text{-}Sign^{off}}, \mathsf{T\text{-}Sign^{on}})(sk_1, \ldots, sk_n, m)$ has the same distribution as the signature generated by $(\mathsf{Sign^{off}}, \mathsf{Sign^{on}})(sk, m)$ where $sk_j (j = 1, \ldots, n)$ are secret shares of sk.*
   - *Ver is the same as $\mathsf{Ver}_{\mathcal{DOS}}$.*
2. *The algorithm $\mathsf{T\text{-}KeyGen}$ is simulatable. That is, there exists a PPT simulator $\mathsf{SIM\text{-}T\text{-}KeyGen}$ that, on input the public key pk generated by an execution of KeyGen, simulates $\mathcal{A}$'s view $\mathsf{View}_{\mathcal{A}}^{\mathsf{T\text{-}KeyGen}}$ in executing Algorithm $\mathsf{T\text{-}KeyGen}$ which generates pk as public key.*
3. *The threshold signing algorithm is simulatable.*
   Let $(\Sigma^{\mathrm{off}}, \Sigma^{\mathrm{on}}) \leftarrow (\mathsf{T\text{-}Sign^{off}}, \mathsf{T\text{-}Sign^{on}})(sk_1, \ldots, sk_n, m)$ be an execution of $\mathcal{DOS}$'s signing algorithm.
   - *$\mathsf{T\text{-}Sign^{off}}$ is simulatable. That is, there exists a PPT simulator $\mathsf{SIM\text{-}T\text{-}Sign^{off}}$ that, on input the public key pk, the off-line signature Σ^{off} and the transcript of the execution of $\mathsf{SIM\text{-}T\text{-}KeyGen}$ (in particular the private information that is given to $\mathcal{A}$), simulates $\mathcal{A}$'s off-line singing view $\mathsf{View}_{\mathcal{A}}^{\mathsf{T\text{-}Sign^{off}}}$ in executing Algorithm $\mathsf{T\text{-}Sign^{off}}$ which generates Σ^{off} as the off-line signature token.*

- $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ *is simulatable. That is, there exists a PPT simulator* $\mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{on}}$ *that, on input* m, Σ^{on} *and the transcript of the executions of* $\mathsf{SIM\text{-}Key}$ *and* $\mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{off}}$*, simulates* $\mathcal{A}$*'s on-line singing view* $\mathsf{View}_{\mathcal{A}}^{\mathsf{T\text{-}Sign}^{\mathsf{on}}}$ *in executing Algorithm* $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ *which generates* Σ^{on} *as the on-line signature token on the message* m*.*

Theorem 2. *An on-line/off-line threshold signature scheme* $\mathcal{OTS}$ *is EU-CMA secure, provided that it is simulatable from a divisible on-line/off-line signature scheme* $\mathcal{DOS}$ *which is EU-CMA secure.*

Proof. (sketch). We prove this theorem by contradiction. Assume under the conditions of the theorem there exists an algorithm $\mathcal{A}$ which breaks the EU-CMA of $\mathcal{OTS}$ in polynomial time with non-negligible probability, then we construct an algorithm $\mathcal{F}$ which breaks the EU-CMA of $\mathcal{DOS}$.

Let $\mathcal{DOS} = (\mathsf{KeyGen}, \mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}, \mathsf{Ver})$ and $\mathcal{OTS} = (\mathsf{T\text{-}KeyGen}, \mathsf{T\text{-}Sign}^{\mathsf{off}}, \mathsf{T\text{-}Sign}^{\mathsf{on}}, \mathsf{Ver})$. To break the EU-CMA of $\mathcal{DOS}$, $\mathcal{F}$ works as follows.

Setup: Suppose $\mathcal{F}$'s challenger runs KeyGen to get a public/private key pair (pk, sk) of $\mathcal{DOS}$ and pk is sent to $\mathcal{F}$. By the precondition, $\mathcal{OTS}$ is simulatable, so there exists a simulator $\mathsf{SIM} = (\mathsf{SIM\text{-}T\text{-}KeyGen}, \mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{off}}, \mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{on}})$ which can simulate $\mathcal{OTS}$'s key generation process and singing process for any adversaries. $\mathcal{F}$ starts $\mathsf{SIM\text{-}T\text{-}KeyGen}$ with input pk to simulate $\mathcal{A}$'s view in Algorithm $\mathsf{T\text{-}KeyGen}$.

Simulating the signing oracles for $\mathcal{A}$:

- When $\mathcal{A}$ arouses $\mathsf{T\text{-}Sign}^{\mathsf{off}}$, $\mathcal{F}$ first queries the oracle $\mathsf{Sign}^{\mathsf{off}}$, to get an off-line signature token Σ^{off}. Next, $\mathcal{F}$ runs $\mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{off}}$ with input Σ^{off} and the transcript of the execution of $\mathsf{SIM\text{-}T\text{-}KeyGen}$ to simulate the off-line singing view of $\mathcal{A}$.
- When $\mathcal{A}$ arouses $\mathsf{T\text{-}Sign}^{\mathsf{on}}$ to sign a message m, $\mathcal{F}$ submits the query m to the oracle $\mathsf{Sign}^{\mathsf{on}}$, to get an on-line signature token Σ^{on}. Then, $\mathcal{F}$ runs $\mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{on}}$ with input m,Σ^{on} and previous execution transcript to simulate $\mathcal{A}$'s off-line signing view.

Output: The simulated view of $\mathcal{A}$ is indistinguishable from the real one. With non-negligible probability, $\mathcal{A}$ outputs a valid forgery (m, Σ) for $\mathcal{OTS}$. $\mathcal{F}$ returns (m, Σ), which is also a valid forgery for $\mathcal{DOS}$, and thus the theorem is proved. ∎

Difference to the previous simulation theorem. The signing simulator for the on-line/off-line threshold signature scheme is divided into two sub-algorithms $(\mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{off}}, \mathsf{SIM\text{-}T\text{-}Sign}^{\mathsf{on}})$. This enables the simulator to generate an off-line signing view for $\mathcal{A}$ *before* the message m is known. However, the simulator for an ordinary threshold signing protocol cannot do this until the message and its signature $\Sigma = (\Sigma^{\mathrm{off}}, \Sigma^{\mathrm{on}})$ are both presented. Thus, in the revised simulation theorem, the "divisibility" of the signing simulation enables the security reduction from the unforgeability of $\mathcal{OTS}$ to the unforgeability of its underlying signature scheme $\mathcal{DOS}$.

Theorem 2 also provides an approach to construct on-line/off-line threshold signatures: given a divisible on-line/off-line signature scheme $\mathcal{DOS}$, we construct a scheme $\mathcal{OTS}$, which is a threshold realization of $\mathcal{DOS}$. If $\mathcal{OTS}$ is simulatable in the sense of Definition 9, then $\mathcal{OTS}$ is an EU-CMA secure on-line/off-line threshold signature scheme.

6 Conclusion and Discussions

We propose a new notion called divisible on-line/off-line signatures, in which off-line signature tokens can be sent to others before the messages to be signed are seen. We also propose an efficient construction, and prove its security under the new definition without resorting to the random oracle model. Furthermore, an application of the new notion to the on-line/off-line threshold signatures is presented. It is shown that a sufficient condition for the security reduction of an on-line/off-line threshold signature scheme is that it is simulatable from a divisible on-line/off-line signature scheme. Below we end with some further discussions.

1. *The gap between the security models of an ordinary on-line/off-line signature scheme and a divisible one.* It seems unlikely that Shamir-Tauman's general paradigm or BB's original scheme can be proven divisible, whilst these schemes are secure in a common sense. So Intuition tells us there exists a gap between the ordinary security model and the new one. However we cannot present a substantial attack against these schemes under the new model to illustrate this gap. This leaves us an open problem to find this potential gap.
2. *More shorter on-line signature length.* The main drawback of our divisible scheme is that the on-line signature length is $2 \log_2 p$, which is twice the length of Scheme CMTW-OS or Schnorr-OS. Thus, it remains an unsolved problem to find a divisible scheme whose security is proven in the standard model and whose performance is comparable to Scheme Schnorr-OS.

Acknowledgement

We thank the anonymous reviewers of the conference CT-RSA 2009 for their valuable comments and suggestions.

References

[1] Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The One-More-RSA-Inversion problems and the security of Chaum's blind signature scheme. Report 2001/002, Cryptology ePrint Archive (May 2002)

[2] Bellare, M., Palacio, A.: GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Yung, M., (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002)

[3] Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology 21(2), pp. 149–177 (2008)

[4] Bresson, E., Catalano, D., Gennaro, R.: Improved on-line/Off-line threshold signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 217–232. Springer, Heidelberg (2007)
[5] Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 98–115. Springer, Heidelberg (1999)
[6] Catalano, D., Di Raimondo, M., Fiore, D., Gennaro, R.: Off-line/On-line signatures: Theoretical aspects and experimental results. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 101–120. Springer, Heidelberg (2008)
[7] Chen, X., Zhang, F., Susilo, W., Mu, Y.: Efficient generic on-line/off-line signatures without key exposure. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 18–30. Springer, Heidelberg (2007)
[8] Crutchfield, C., Molnar, D., Turner, D., Wagner, D.: Generic on-line/Off-line threshold signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 58–74. Springer, Heidelberg (2006)
[9] Desmedt, Y.: Threshold cryptography. European Transactions on Telecommunications 5(4), 449–457 (1994)
[10] Desmedt, Y.: Some recent research aspects of threshold cryptography. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 158–173. Springer, Heidelberg (1998)
[11] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory IT-31(4), 469–472 (1985)
[12] Even, S., Goldreich, O., Micali, S.: On-line/Off-line digital signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 263–275. Springer, Heidelberg (1990)
[13] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
[14] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 354–371. Springer, Heidelberg (1996)
[15] Gennaro, R., Rabin, T., Jarecki, S., Krawczyk, H.: Robust and efficient sharing of RSA functions. Journal of Cryptology 13(2), pp. 273–300 (2000)
[16] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
[17] Kurosawa, K., Schmidt-Samoa, K.: New online/Offline signature schemes without random oracles. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 330–346. Springer, Heidelberg (2006)
[18] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
[19] National Institute of Standards and Technology (NIST). The Digital Signature Standard, 186. Federal Information Processing Standards Publication (FIPS PUB) (May 1994)
[20] Rabin, T.: A simplified approach to threshold and proactive RSA. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 88–104. Springer, Heidelberg (1998)
[21] Schmidt-Samoa, K., Takagi, T.: Paillier's cryptosystem modulo p^2q and its applications to trapdoor commitment schemes. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 296–313. Springer, Heidelberg (2005)

[22] Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), pp. 161–174 (1991)
[23] Shamir, A., Tauman, Y.: Improved online/Offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
[24] Xu, S., Mu, Y., Susilo, W.: Online/Offline signatures and multisignatures for AODV and DSR routing security. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 99–110. Springer, Heidelberg (2006)
[25] Yu, P., Tate, S.R.: An online/offline signature scheme based on the strong rsa assumption. In: AINA Workshops (1), pp. 601–606 (2007)

In appendices, we give elliptic curve analogues of existing divisible on-line/off-line signature schemes in order to fairly compare them with our proposed scheme. Suppose E is an elliptic curve over a finite field $\mathbb{F}$. Let g be a point in E with prime order p and let $\mathbb{G}$ be a group generated by g.

A. Crutchfield et al.'s Divisible On-line/Off-line Scheme CMTW-OS

The on-line/off-line signature scheme $\mathsf{CMTW\text{-}OS}$ is extracted from [8]. In [8], the authors construct an on-line/off-line threshold signature scheme which is a threshold realization of this underlying scheme. Let $\mathcal{S} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ be an ordinary signature scheme. The on-line/off-line signature scheme $\mathsf{CMTW\text{-}OS} = (\mathsf{KeyGen}, \mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}, \mathsf{Ver})$, where

- KeyGen. Choose $x \in_R \mathbb{Z}_p$, and let $h = g^x$. Run the key generation algorithm of $\mathcal{S}$ to obtain (pk, sk) which is public/private key pair for $\mathcal{S}$. The public key of $\mathcal{OS}$ is (E, p, g, h, pk), and the private key is (x, sk).
- $\mathsf{Sign}^{\mathsf{off}}$. (The i-th run). Choose $r, m \in_R \mathbb{Z}_p$, and compute $u = g^r h^m$. Use the signing algorithm of $\mathcal{S}$ to obtain $\sigma = \mathsf{S}_{sk}(u)$. Store the state information r, m. The off-line signature token is σ.
- $\mathsf{Sign}^{\mathsf{on}}$. (The i-th run, on a message $m' \in \mathbb{Z}_p$). Retrieve m, r from the memory. Compute $r' = r + (m - m')x \mod p$. The on-line signature token is r'.
- Ver. (On a message-signature pair (m', Σ) where $\Sigma = (\sigma, r')$). Verify that whether $\mathsf{V}_{pk}(g^{r'} h^{m'}, \sigma) = 1$.

Theorem 3. *The scheme* $\mathsf{CMTW\text{-}OS}$ *is a divisible on-line/off-line signature scheme which is EU-CMA secure, provided that the underlying signature scheme* $\mathcal{S}$ *is existentially unforgeable against generic chosen message attacks and the one-more-discrete-log assumption holds in* $\mathbb{G}$.

The proof of this theorem is omitted. Please refer to Theorem 2 of [8] for details. [7]

[7] The proof in [8] reduces the security of the on-line/off-line scheme $\mathsf{CMTW\text{-}OS}$ to the one-more-discrete-log assumption, or the collision resistance of a trapdoor hash function, or the unforgeability of $\mathcal{S}$. A small modification of this proof can simply reduce the security to the one-more-discrete-log assumption or the unforgeability of $\mathcal{S}$.

B. Bresson et al.'s Divisible On-line/Off-line Scheme BCG-OS

The on-line/off-line signature scheme BCG-OS is extracted from [4]. In [4], the authors construct an on-line/off-line threshold signature scheme which is a threshold realization of this underlying scheme. Let $\mathcal{S} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ be an ordinary signature scheme. The on-line/off-line signature scheme BCG-OS = ($\mathsf{KeyGen}, \mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}$, Ver), where

- KeyGen. Choose $x, y \in_R \mathbb{Z}_p^*$, and let $h_1 = g^x, h_2 = g^y$. Run the key generation algorithm of $\mathcal{S}$ to obtain (pk, sk) which is public/private key pair for $\mathcal{S}$. The public key is (E, p, g, h_1, h_2, pk), and the private key is (x, y, sk).
- $\mathsf{Sign}^{\mathsf{off}}$. (The i-th run). Choose $r, s, m \in_R \mathbb{Z}_p$, and compute $u = g^m h_1^r h_2^s$. Use the signing algorithm of $\mathcal{S}$ to obtain $\sigma = \mathsf{S}_{sk}(u)$. Store the state information r, s, m. The off-line signature token is σ.
- $\mathsf{Sign}^{\mathsf{on}}$. (The i-th run, on a message m'). Retrieve r, s, m from the memory. Choose $r' \in_R \mathbb{Z}_p$ and compute $s' = s + y^{-1}[(m - m') + (r - r')x] \mod p$. The on-line signature token is (r', s').
- Ver. (On a message-signature pair (m', Σ) where $\Sigma = (\sigma, r', s')$). Verify that whether $\mathsf{V}_{pk}(g^{m'} h_1^{r'} h_2^{s'}, \sigma) = 1$.

Remark 6. To reduce the on-line signing cost, we can move the selection of r' and computing $(r - r') \cdot x$ to the off-line phase. Thus, the on-line signing requires only 1 modular multiplication in $\mathbb{Z}_p$.

Theorem 4. *The scheme* BCG-OS *is a divisible on-line/off-line signature scheme which is EU-CMA secure, provided that the underlying signature scheme $\mathcal{S}$ is existentially unforgeable against generic chosen message attacks and the discrete logarithm assumption holds in $\mathbb{G}$.*

The proof of this theorem is also omitted. Please refer to Theorem 1 of [4] for details.

C. Proving the Schnorr Signature Scheme [22] is Divisible

A variant of the Schnorr signature scheme can be naturally viewed as a divisible on-line/off-line signature scheme: Schnorr-OS = ($\mathsf{KeyGen}, \mathsf{Sign}^{\mathsf{off}}, \mathsf{Sign}^{\mathsf{on}}, \mathsf{Ver}$).

- KeyGen. Choose $x \in_R \mathbb{Z}_p$, and let $h = g^x$. Let H be a hash function: $H : \{0,1\}^* \to \mathbb{Z}_p$. The public key is (E, p, g, h, H), and the private key is x.
- $\mathsf{Sign}^{\mathsf{off}}$. (The i-th run). Choose $r \in_R \mathbb{Z}_p$, and compute $u = g^r$. Store the state information r. The off-line signature token is u.
- $\mathsf{Sign}^{\mathsf{on}}$. (The i-th run, on a message m). Retrieve r from the memory. Set $c = H(m \| u)$ and compute $s = r - cx \mod p$. The on-line signature token is s.
- Ver. (On a message-signature pair (Σ, m) where $\Sigma = (u, s)$. Verify that whether $g^s h^{H(m\|u)} = u$.

Remark 7. The signature token is defined as $(u, s) \in \mathbb{G} \times \mathbb{Z}_p$ instead of $(c, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$. This is to decrease the on-line signature length because the value of c can't be computed in the off-line phase.

Theorem 5. *The scheme* Schnorr-OS *is a divisible on-line/off-line signature scheme which is EU-CMA secure in the random oracle model, provided that the one-more-discrete-log assumption holds in* $\mathbb{G}$*.*

Proof. Assume there exists an Algorithm $\mathcal{A}$ which $(t, q_{\text{off}}, q_{\text{on}}, q_{\text{h}}, \epsilon)$ breaks the unforgeability of Schnorr-OS in the random-oracle based game defined in Section 3. Then we construct an Algorithm $\mathcal{B}$ which breaks the q_{on}-DL problem in polynomial time with probability $\epsilon' \geq (\frac{\epsilon}{q_{\text{h}}} - \frac{1}{p})^2$.

Let g be a point in E with prime order p. Let $\mathbb{G}$ be a group generated by g. Algorithm $\mathcal{B}$ is given a q_{on}-DL problem instance $(\mathsf{descr}(\mathbb{G}), g, v_1, \ldots, v_{q_{\text{on}}+1})$ where $\mathsf{descr}(\mathbb{G}) = (E, p)$. To find all the values of $\text{dl}_g v_i$, Algorithm $\mathcal{B}$ works as follows:

Setup: Algorithm $\mathcal{B}$ sets $h = v_{q_{\text{on}}+1}$ and gives to $\mathcal{A}$ the public key (E, p, g, h, H) where H is modeled as a random oracle defined in the following.

Simulating the hash oracle: Maintain a H-list of pairs, which is initialized to empty. Upon a query input $M \in \{0,1\}^*$, returns a random value $c \in \mathbb{Z}_p$ if M is not in the H-list. Then add (M, c) to the H-list.

Simulating the signing oracle (off-line): Upon the i-th query, return v_i as the off-line signature token.

Simulating the signing oracle (on-line): Upon the i-th query input m_i, Algorithm $\mathcal{B}$ makes query $m_i \| v_i$ to the hash oracle to obtain an answer, say c_i. Next, Algorithm $\mathcal{B}$ makes query $v_i h^{-c_i}$ to the oracle $\mathsf{DL}_g(\cdot)$ to obtain $w_i = \text{dl}_g(v_i h^{-c_i})$ and returns w_i as the answer for the query m_i. $\mathcal{B}$ makes at most q_{on} queries to the oracle $\mathsf{DL}_g(\cdot)$ since $\mathcal{A}$ makes at most q_{on} on-line signing queries.

Rewinding: At the beginning of the game, Algorithm $\mathcal{B}$ randomly selects a value $d \in q_{\text{h}}$. Upon the d-th hash query, the simulated hash oracle returns a random value $c \in \mathbb{Z}_p$ as usual. If $\mathcal{A}$ successfully outputs a valid signature forgery on a message, Algorithm $\mathcal{B}$ then resets $\mathcal{A}$ to the step where $\mathcal{A}$ has just sent the d-th hash query. This time, the simulated hash oracle again randomly selects a value, say c', and returns it as the answer. Next, $\mathcal{B}$ continues the game which runs the second instance of $\mathcal{A}$, which has the same inputs and internal coins with the first instance.

Output: Parse the d-th hash query as $m \| t$ where $t \in \mathbb{Z}_p$. Algorithm $\mathcal{B}$ successfully ends the game if the following events occur:

1) The first instance of $\mathcal{A}$ successfully outputs a valid signature forgery (t, s_1) for the message m where s_1 is some value in $\mathbb{Z}_p$, and
2) The second instance of $\mathcal{A}$ successfully outputs a valid signature forgery (t, s_2) for the message m where s_2 is some value in $\mathbb{Z}_p$, and
3) $c \neq c'$.

If the above events occur, we can conclude that $g^{s_1} h^c = g^{s_2} h^{c'} = t$ and $c \neq c'$. Thus, $\mathcal{B}$ can output the solution of the q_{on}-DL problem as follows:

$\mathrm{dl}_g(v_{q_{\mathrm{on}}+1}) = \mathrm{dl}_g(h) = (s_1 - s_2)(c' - c)^{-1} \mod p$.
$\mathrm{dl}_g(v_i) = w_i + c_i \mathrm{dl}_g(v_{q_{\mathrm{on}}+1}) \mod p$ for i from 1 to q_{on}.

Let's estimate the probability that $\mathcal{B}$ successfully ends the game. By a standard argument similar to the reset lemma of [2], we get that $\mathcal{B}$ succeeds with probability at least $(\frac{\epsilon}{q_{\mathrm{h}}} - \frac{1}{p})^2$. Thus, $\mathcal{B}$ successfully breaks q_{on}-DL problem in polynomial time with non-negligible probability. This contradicts the one-more-discrete-log assumption and thus the theorem is proved. ∎