Threshold Kleptographic Attacks on Discrete Logarithm Based Signatures

George Teşeleanu[1,2][0000−0003−3953−2744]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
tgeorge@dcti.ro
[2] Department of Computer Science
"Al.I.Cuza" University of Iaşi 700506 Iaşi, Romania,
george.teseleanu@info.uaic.ro

Abstract. In an ℓ out of n *threshold scheme*, ℓ out of n members must cooperate to recover a secret. A *kleptographic attack* is a backdoor which can be implemented in an algorithm and further used to retrieve a user's secret key. We combine the notions of *threshold scheme* and *kleptographic attack* to construct the first ℓ out of n *threshold kleptographic attack* on discrete logarithm based digital signatures and prove its security in the standard and random oracle models.

1 Introduction

Simmons [63, 64] was the first to study the use of digital signatures as a channel to convey information (subliminal channels). Later on, Young and Yung [67–71] combined subliminal channels and public key cryptography to leak a user's private key or a message (SETUP attacks). Young and Yung assumed a black-box environment[3], while mentioning the existence of other scenarios. These attacks need a malicious device manufacturer[4] to work. The input and output distributions of a device with SETUP should not be distinguishable from the regular distribution. However, if the device is reverse engineered, the deployed mechanism may be detectable.

Although SETUP attacks were considered far-fetched by some cryptographers, recent events [6,56] suggest otherwise. As a consequence, this research area seems to have been revived [5, 8, 22, 44, 58]. In [10], SETUP attacks implemented in symmetric encryption schemes are referred to as *algorithmic substitution attacks* (ASA). The authors of [10] point out that the sheer complexity of open-source software (*e.g.* OpenSSL) and the small number of experts who review them make ASAs plausible not only in the black-box model. ASAs in the symmetric setting are further studied in [8, 20] and, in the case of hash functions, in [3].

A practical example of leaking user keys is the Dual-EC generator. As pointed out in [13], using the Dual-EC generator facilitates a third party to recover a user's private key. Such an attack is a natural application of Young and Yung's work. Some real world SETUP attack examples may be found in [15, 16]. Building on the earlier work of [66] and influenced by the Dual-EC incident, [21, 22] provide the readers with a formal treatment of backdoored pseudorandom generators (PRNG).

A more general model entitled *subversion attacks* is considered in [5]. This model includes SETUP attacks and ASAs, but generic malware and virus attacks are also included. The authors provide subversion resilient signature schemes in the proposed model. Their work is further extended in [58, 59], where subversion resistant solutions for one-way functions, signature schemes and PRNGs are provided. In [58], the authors point out that the model from [5] assumes the system parameters are honestly generated (but this is not always the case). In the discrete logarithm case, examples of algorithms for generating trapdoored prime numbers may be found in [29, 35].

[3] A black-box is a device, process or system, whose inputs and outputs are known, but its internal structure or working is not known or accessible to the user (*e.g.* tamper proof devices).
[4] that implements the mechanisms to recover the secrets

A different method for protecting users from subversion attacks are *cryptographic reverse firewalls* (RF). RFs represent external trusted devices that sanitize the outputs of infected machines. The concept was introduced in [24, 49]. A reverse firewall for signature schemes is provided in [5].

In this paper, we extend the SETUP attacks of Young and Yung on digital signatures. We introduce the first SETUP mechanism that leaks a user's secret key, only if ℓ out of n malicious parties decide to do this. We assume that the signature schemes are implemented in a black-box equipped with a volatile memory, erased whenever someone tampers with it.

In the following we give a few examples where a threshold kleptographic signature may be useful.

Since digitally signed documents are just as binding as signatures on paper, if a recipient receives a document signed by A he will act according to A's instructions. Finding A's private key, can aid a law enforcement agency into collecting additional informations about A and his entourage. In order to protect citizens from abuse, a warrant must be issued by a legal commission before starting surveillance. To aid the commission and to prevent abuse, the manufacturer of A's device can implement a ℓ out of n threshold SETUP mechanism. Thus, A's key can be recovered only if there is a quorum in favor of issuing the warrant.

Digital currencies (*e.g.* Bitcoin) have become a popular alternative to physical currencies. Transactions between users are based on digital signatures. When a transaction is conducted, the recipient's public key is linked to the transfered money. Only the owner of the secret key can now spend the money. To protect his secret keys, a user can choose to store them in a tamper proof device, called a hardware wallet. Let's assume that a group of malicious entities manages to infect some hardware wallets and they implement an ℓ out of n threshold SETUP mechanism. When ℓ members decide, they can transfer the money from the infected wallets without the owner's knowledge. If $\ell - 1$ parties are arrested, the mechanism remains undetectable as long as the devices are not reverse engineered.

In accordance with the original works, we prove that the threshold SETUP mechanisms are polynomially indistinguishable from regular signatures. Depending on the infected signature, we obtain security in the standard or random oracle model (ROM). To do so, we make use of a public key encryption scheme (introduced in Section 3) and Shamir's secret sharing scheme [61]. ROM security proofs are easily deduced from the standard model security proofs provided in the paper. Thus, are omitted.

Structure of the paper. We introduce notations and definitions used throughout the paper in Section 2. In order to mount the SETUP attacks, we use a variant of the Generalized ElGamal encryption scheme [46] that is described in Section 3. In Section 4 we describe a SETUP attack on the Generalized ElGamal signature [46], extended in Section 5. Section 6 contains a series of applications of the described attacks. Countermeasures are provided in Section 7. We conclude in Section 8. Additional definitions are given in Appendix A. In **??** we prove the security margins claimed in Section 3.2. A two-party malicious signing protocol is presented in Appendix B. We provide a supplementary SETUP mechanism in Appendix C.

2 Preliminaries

Notations. Throughout the paper, λ will denote a security parameter. The action of selecting a uniformly random element x from a sample space X is denoted by $x \xleftarrow{\$} X$. We also denote by $x \leftarrow y$ the assignment of value y to variable x. The probability that event E happens is denoted by $Pr[E]$. The action of choosing a random element from an entropy smoothing[5] (ES) family $\mathcal{H}$ is further referred to as "H is ES". Encryption of message m with key k using the AES algorithm[6] is denoted by $AES_k(m)$.

2.1 Diffie-Hellman Assumptions

Definition 1 (Computational Diffie-Hellman - CDH). *Let $\mathbb{G}$ be a cyclic group of order q, g a generator of $\mathbb{G}$ and let A be a probabilistic polynomial-time algorithm (PPT algorithm) that returns an element from $\mathbb{G}$.*

[5] We refer the reader to Definition 10.

[6] We refer the reader to [19] for a description of AES.

We define the advantage

$$ADV^{\text{CDH}}_{\mathbb{G},g}(A) = Pr[A(g^x, g^y) = g^{xy} | x, y \xleftarrow{\$} \mathbb{Z}_q^*].$$

If $ADV^{\text{CDH}}_{\mathbb{G},g}(A)$ is negligible for any PPT algorithm A, we say that the Computational Diffie-Hellman problem is hard in $\mathbb{G}$.

Definition 2 (Decisional Diffie-Hellman - DDH). *Let $\mathbb{G}$ be a cyclic group of order q, g a generator of $\mathbb{G}$. Let A be a PPT algorithm which returns 1 on input (g^x, g^y, g^z) if $g^{xy} = g^z$. We define the advantage*

$$ADV^{\text{DDH}}_{\mathbb{G},g}(A) = |Pr[A(g^x, g^y, g^z) = 1 | x, y \xleftarrow{\$} \mathbb{Z}_q^*, z \leftarrow xy] - Pr[A(g^x, g^y, g^z) = 1 | x, y, z \xleftarrow{\$} \mathbb{Z}_q^*]|.$$

If $ADV^{\text{DDH}}_{\mathbb{G},g}(A)$ is negligible for any PPT algorithm A, we say that the Decisional Diffie-Hellman problem is hard in $\mathbb{G}$.

Definition 3 (Hash Diffie-Hellman - HDH). *Let $\mathbb{G}$ be a cyclic group of order q, g a generator of $\mathbb{G}$ and $H : \mathbb{G} \to \mathbb{Z}_q^*$ a hash function. Let A be a PPT algorithm which returns 1 on input (g^x, g^y, z) if $H(g^{xy}) = z$. We define the advantage*

$$ADV^{\text{HDH}}_{\mathbb{G},g,H}(A) = |Pr[A(g^x, g^y, H(g^{xy})) = 1 | x, y \xleftarrow{\$} \mathbb{Z}_q^*] - Pr[A(g^x, g^y, z) = 1 | x, y, z \xleftarrow{\$} \mathbb{Z}_q^*]|.$$

If $ADV^{\text{HDH}}_{\mathbb{G},g,H}(A)$ is negligible for any PPT algorithm A, we say that the Hash Diffie-Hellman problem is hard in $\mathbb{G}$.

Remark 1. The two first assumptions (CDH and DDH) are standard and are included for completeness. The HDH assumption was formally introduced in [1, 2], although it was informally introduced as a composite assumption in [11, 74]. According to [11], the HDH assumption is equivalent with the CDH assumption in ROM. If the DDH assumption is hard in $\mathbb{G}$ and H is ES, then the HDH assumption is hard in $\mathbb{G}$ [1, 51, 62]. In [32], the authors show that the HDH assumption holds, even if the DDH assumption is relaxed to the following assumption: $\mathbb{G}$ contains a large enough group in which DDH holds. One particular interesting group is $\mathbb{Z}_p^*$, where p is a "large"[7] prime. According to [32], it is conjectured that if $\mathbb{G}$ is generated by an element $g \in \mathbb{Z}_p^*$ of order q, where q is a "large"[8] prime that divides $p - 1$, then the DDH assumption holds. The analysis conducted in [32] provides the reader with solid arguments to support the hypothesis that HDH holds in the subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$.

2.2 Definitions and Security Models

Definition 4 (Signature Scheme). *A Signature Scheme consists of three PPT algorithms: KeyGen, Sign and Verification. The first one takes as input a security parameter and outputs the system parameters, the public key and the matching secret key. The secret key together with the Sign algorithm is used to generate a signature σ for a message m. Using the public key, the last algorithm verifies if a signature σ for a message m is generated using the matching secret key.*

Definition 5 (Public Key Encryption - PKE). *A Public Key Encryption (PKE) scheme consists of three PPT algorithms: KeyGen, Encrypt and Decrypt. The first one takes as input a security parameter and outputs the system parameters, the public key and the matching secret key. The public key together with the Encrypt algorithm is used to encrypt a message m. Using the secret key, the last algorithm decrypts any ciphertext encrypted using the matching public key.*

Remark 2. For simplicity, public parameters will further be implicit when describing an algorithm.

[7] at least 2048 bits, better 3072 bits
[8] at least 192 bits, better 256 bits

Definition 6 **(Indistinguishability from Random Bits - IND\$).** *The security model of indistinguishability from random bits for a PKE scheme $\mathcal{AE}$ is captured in the following game:*

KeyGen(λ): The challenger C generates the public key, sends it to adversary A and keeps the matching secret key to himself.

Query: Adversary A sends C a message m. The challenger encrypts m and obtains the ciphertext c_0. Let c_1 be a randomly chosen element from the same set as c_0. The challenger flips a coin $b \in \{0,1\}$ and returns c_b to the adversary.

Guess: In this phase, the adversary outputs a guess $b' \in \{0,1\}$. He wins the game, if $b' = b$.

The advantage of an adversary A attacking a PKE scheme is defined as

$$ADV_{\mathcal{AE}}^{\text{IND\$}}(A) = |2Pr[b = b'] - 1|,$$

where the probability is computed over the random bits used by C and A. A PKE scheme is IND\$ *secure, if for any PPT adversary A the advantage $ADV_{\mathcal{AE}}^{\text{IND\$}}(A)$ is negligible.*

Definition 7 **(Anonymity under Chosen Plaintext Attacks - ANO-CPA).** *The security model against anonymity under chosen plaintext attacks for a PKE scheme $\mathcal{AE}$ is captured in the following game:*

KeyGen(λ): The challenger C generates two public keys pk_0 and pk_1, sends them to adversary A and keeps the matching secret keys to himself.

Query: Adversary A sends C a message m. The challenger flips a coin $b \in \{0,1\}$ and encrypts m using pk_b. The resulting ciphertext c is sent to the adversary.

Guess: In this phase, the adversary outputs a guess $b' \in \{0,1\}$. He wins the game, if $b' = b$.

The advantage of an adversary A attacking a PKE scheme is defined as

$$ADV_{\mathcal{AE}}^{\text{ANO-CPA}}(A) = |2Pr[b = b'] - 1|,$$

where the probability is computed over the random bits used by C and A. A PKE scheme is ANO-CPA *secure, if for any PPT adversary A the advantage $ADV_{\mathcal{AE}}^{\text{ANO-CPA}}(A)$ is negligible.*

Definition 8 **(Secretly Embedded Trapdoor with Universal Protection -** SETUP**).** *A Secretly Embedded Trapdoor with Universal Protection (SETUP) is an algorithm that can be inserted in a system such that it leaks encrypted private key information to an attacker through the system's outputs. Encryption of the private key is performed using an asymmetric encryption scheme. It is assumed that the decryption function is accessible only to the attacker.*

Definition 9 **(**SETUP **indistinguishability - IND-SETUP).** *Let C_0 be a black-box system that uses a secret key sk. Let $\mathcal{AE}$ be the PKE scheme used by a SETUP mechanism as defined above, in Definition 8. We consider C_1 an altered version of C_0 that contains a SETUP mechanism based on $\mathcal{AE}$. Let A be a PPT algorithm which returns 1 if it detects that C_0 is altered. We define the advantage*

$$ADV_{\mathcal{AE},C_0,C_1}^{\text{IND-SETUP}}(A) = |Pr[A^{C_1(sk,\cdot)}(\lambda) = 1] - Pr[A^{C_0(sk,\cdot)}(\lambda) = 1]|.$$

If $ADV_{\mathcal{AE},C_0,C_1}^{\text{IND-SETUP}}(A)$ is negligible for any PPT algorithm A, we say that C_0 and C_1 are polynomially indistinguishable.

Remark 3. Definition 9 is a formalization of the indistinguishability property for a regular SETUP mechanism described in [68]. More general concepts may be found in [5] (*publicly undetectability*) and [58] (*cliptographic game*). A consequence of IND-SETUP is that C_0 and C_1 have the same security.

Remark 4. We consider that the attacks presented from now on are implemented in a device D that digitally signs messages. The owner of the device is denoted by V and his public key by pk_V. We assume that his secret key sk_V is stored only in D's volatile memory[9]. The victim V thinks that D signs messages using the signature scheme described in Section 2.3. We stress that *KeyGen* and *Verification* algorithms are identical to the ones from Section 2.3. Thus, *KeyGen* and *Verification* are omitted when presenting the attacks.

Throughout the paper, when presenting the SETUP mechanisms, we make use of the following two additional algorithms

– *Malicious Party(s) KeyGen* – used by the attacker(s) to generate his (their) parameters;
– *Recovering* – used by the attacker(s) to recover V's secret key.

The algorithms above are not implemented in D.

2.3 Generalized ElGamal Signature

Originally described in [26], the ElGamal digital signature scheme can easily be generalized to any finite cyclic group $\mathbb{G}$. We shortly describe the algorithms of the generalized ElGamal signature scheme, as presented in [46].

KeyGen(λ)*:* Generate a large prime number q, such that $q \geq 2^\lambda$. Choose a cyclic group $\mathbb{G}$ of order q and let g be a generator of the group. Let $h : \mathbb{G} \to \mathbb{Z}_q$ be a hash function. Choose $a \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y \leftarrow g^a$. Output the system parameters $pp = (q, g, \mathbb{G}, h)$ and the public key $pk_V = y$. The secret key is $sk_V = a$.

Sign(m, sk_V)*:* To sign a message $m \in \mathbb{G}$, first generate a random number $k \xleftarrow{\$} \mathbb{Z}_q^*$. Then compute the values $r \leftarrow g^k$ and $s \leftarrow k^{-1}[h(m) - a \cdot h(r)] \bmod q$. Output the signature (r, s).

Verification(m, r, s, pk_V)*:* To verify the signature (r, s) of message m, compute $v_1 \leftarrow y_V^{h(r)} \cdot r^s$ and $v_2 \leftarrow g^{h(m)}$. Output `true` if and only if $v_1 = v_2$. Else output `false`.

2.4 Young-Yung SETUP Attack on the Generalized ElGamal Signature

In [67–70], the authors propose a kleptographic version of ElGamal signatures and prove it secure in the standard model under the HDH assumption. The Young-Yung SETUP mechanism can be easily adapted to the generalized ElGamal signature, while maintaining its security. The algorithms of the generalized version are shortly described below. We assume that user V is the victim of a malicious user M. After D signs at least two messages, M can recover V's secret key and thus impersonate V.

Malicious Party KeyGen(pp)*:* Let $H : \mathbb{G} \to \mathbb{Z}_q^*$ be a hash function. Choose $x_M \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y_M \leftarrow g^{x_M}$. Output the public key $pk_M = y_M$. The public key pk_M and H will be stored in D's volatile memory. The secret key is $sk_M = x_M$; it will only be known by M and will not be stored in the black-box.

Signing Sessions: The possible signing sessions performed by D are described below. Let $i \geq 1$.

Session$_0(m_0, sk_V)$: To sign message $m_0 \in \mathbb{G}$, D does the following

$$k_0 \xleftarrow{\$} \mathbb{Z}_q^*, r_0 \leftarrow g^{k_0}, s_0 \leftarrow k_0^{-1}[h(m_0) - a \cdot h(r_0)] \bmod q.$$

The value k_0 is stored in D's volatile memory until the end of *Session*$_1$. Output the signature (r_0, s_0).

Session$_i(m_i, sk_V, pk_M)$: To sign message $m_i \in \mathbb{G}$, D does the following

$$z_i \leftarrow y_M^{k_{i-1}}, k_i \leftarrow H(z_i), r_i \leftarrow g^{k_i}, s_i \leftarrow k_i^{-1}[h(m_i) - a \cdot h(r_i)].$$

The value k_i is stored in D's volatile memory until the end of *Session*$_{i+1}$. Output the signature (r_i, s_i).

[9] If V knows his secret key, he is able to detect a SETUP mechanism using its description and parameters (found by means of reverse engineering a black-box, for example).

Recovering$(m_i, r_{i-1}, r_i, s_i, sk_M)$*:* Compute $\alpha \leftarrow r_{i-1}^{x_M}$ and $k_i \leftarrow H(\alpha)$. Recover a by computing

$$a \leftarrow h(r_i)^{-1}[h(m_i) - k_i \cdot s_i)].$$

Remark 5. Let S be an honest generator for the values r used by the Generalized ElGamal signature scheme and let σ_i denote the i-th internal state and $\rho_i = g^{\sigma_i}$ the i-th output of S. The mechanism described above can be seen as a malicious PRNG $\tilde{S}$ based on the honest PRNG S. We define the internal states and outputs of $\tilde{S}$ by

- $\tilde{\sigma}_0 = \sigma_0$, $\tilde{\rho}_0 = \rho_0$;
- $\tilde{\sigma}_i = H(y_M^{\tilde{\sigma}_{i-1}})$, $\tilde{\rho}_i = g^{\tilde{\sigma}_i}$, where $i \geq 1$.

In [22], the authors state that the Dual-EC generator does not output bits that are provably indistinguishable from random bits. To improve Dual-EC, they introduce $\tilde{S}$ and prove it secure under the HDH assumption.

3 Multiplicative ElGamal Encryption

The ElGamal encryption scheme was first described in [26]. The underlying group of the scheme is $\mathbb{Z}_p$, where p is a prime number. The scheme can easily be generalized to any finite cyclic group $\mathbb{G}$. The description of the generalized ElGamal can be found in [46]. Based on this description, we propose a new version of the ElGamal encryption scheme, which will later be used to deploy our SETUP mechanisms. We prove that the scheme is secure and that it preserves anonymity.

3.1 Scheme Description

KeyGen(λ)*:* Generate a large prime number q, such that $q \geq 2^\lambda$. Choose a cyclic group $\mathbb{G}$ of order q and let g be a generator of the group. Let $H : \mathbb{G} \to \mathbb{Z}_q^*$ be a hash function. Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y \leftarrow g^x$. Output the system parameters $pp = (q, g, \mathbb{G}, H)$ and the public key $pk = y$. The secret key is $sk = x$.

Encryption(m, pk)*:* To encrypt a message $m \in \mathbb{Z}_q^*$, first generate a random number $k \xleftarrow{\$} \mathbb{Z}_q^*$. Then compute the values $\alpha \leftarrow g^k$, $\beta \leftarrow y^k$, $\gamma \leftarrow H(\beta)$ and $\delta \leftarrow m \cdot \gamma$. Output the pair (α, δ).

Decryption(α, δ, sk)*:* To decrypt ciphertext (α, δ), compute $\epsilon \leftarrow \alpha^x$, $\zeta \leftarrow H(\epsilon)$. Recover the original message by computing $m \leftarrow \delta \cdot \zeta^{-1}$.

We need to prove that the scheme is sound. If the pair (α, δ) is generated according to the scheme, it is easy to see that $\delta \cdot \zeta^{-1} \equiv m \cdot H(y^k) \cdot [H(\alpha^x)]^{-1} \equiv m \cdot H((g^x)^k) \cdot [H((g^k)^x)]^{-1} \equiv m$.

Remark 6. In the original ElGamal encryption we have $m \in \mathbb{G}$ and $\delta \leftarrow m \cdot \beta$, but in modern use of Diffie-Hellman we have $\delta \leftarrow AES_\gamma(m)$.

3.2 Security Analysis

In this section we prove that the Multiplicative ElGamal is a secure encryption scheme and it preserves anonimity. We denote by $\mathcal{MEG}$ the Multiplicative ElGamal scheme.

Theorem 1. *If* HDH *is hard in* $\mathbb{G}$ *then* $\mathcal{MEG}$ *is* IND\$ *secure in the standard model. Formally, let* A *be an efficient PPT* IND\$ *adversary. There exists an efficient algorithm* B *such that*

$$ADV_{\mathcal{MEG}}^{\text{IND\$}}(A) \leq 2ADV_{\mathbb{G},g,H}^{\text{HDH}}(B).$$

Proof. Let A be an IND\$ adversary for $\mathcal{MEG}$ with access to "random coins" sampled uniformly from a set R. We construct an adversary B for the HDH assumption and then we provide an upper bound for the advantage of A.

$x \xleftarrow{\$} \mathbb{Z}_q^*, y \leftarrow g^x,$
$\rho \xleftarrow{\$} R, m \leftarrow A(\rho, y),$
$k \xleftarrow{\$} \mathbb{Z}_q^*, \alpha_0 \leftarrow g^k, \beta \leftarrow y^k, \gamma \leftarrow H(\beta), \delta_0 \leftarrow m \cdot \gamma,$
$\alpha_1 \xleftarrow{\$} \mathbb{G}, \delta_1 \xleftarrow{\$} \mathbb{Z}_q^*, b \xleftarrow{\$} \{0,1\},$
$b' \leftarrow A(\rho, y, \alpha_b, \delta_b).$

Fig. 1. The IND\$ game.

Algorithm $B(U, V, W)$:
$y \leftarrow U,$
$\rho \xleftarrow{\$} R, m \leftarrow A(\rho, y),$
$\alpha_0 \leftarrow V, \delta_0 \leftarrow m \cdot W,$
$\alpha_1 \xleftarrow{\$} \mathbb{G}, \delta_1 \xleftarrow{\$} \mathbb{Z}_q^*,$
$b \xleftarrow{\$} \{0,1\}, b' \leftarrow A(\rho, y, \alpha_b, \delta_b),$
If $b = b'$ then return 1 else return 0.

Fig. 2. Algorithm B for attacking HDH.

Figure 1 describes the IND\$ game. The first row sets up the public key. In the second row, A chooses the message m it wants to be challenged on. The challenger then picks a random k and computes the encryption (α_0, δ_0) of m. It also picks random choices (α_1, δ_1) from the same sampling sets, flips a bit b and reveals (α_b, δ_b). A then computes its guess b' for b. A wins if $b = b'$. Formally, the probability of A winning the IND\$ game is

$$|2Pr[b' = b] - 1| = ADV_{\mathcal{MEG}}^{\text{IND\$}}(A). \tag{1}$$

Figure 2 depicts the behavior of an adversary B who runs the IND\$ distinguisher A as a subroutine. B is given as input U, V, W, where $U \leftarrow g^u$ and $V \leftarrow g^v$, for random u, v, and W is either $H(g^{uv})$ or random. Algorithm B outputs a bit indicating its guess for which of these cases occurs, where 1 means B guesses $W = H(g^{uv})$. Formally, the HDH advantage of B is

$$ADV_{\mathbb{G},g,H}^{\text{HDH}}(B) = |Pr[B(U,V,W) = 1 | W = H(g^{uv})] - Pr[B(U,V,W) = 1 | W \xleftarrow{\$} \mathbb{Z}_q^*]|. \tag{2}$$

Lets us consider the case $W = H(g^{uv})$ and compute the probability of B outputting 1. We note that B is running A as the latter would run an attack on the IND\$ security of $\mathcal{MEG}$. Thus, we have

$$\begin{aligned} Pr[B(U,V,W) = 1 | W = H(g^{uv})] &= Pr[b = b' | W = H(g^{uv})] \\ &= \frac{1}{2}(2Pr[b = b' | W = H(g^{uv})] - 1 + 1) \\ &= \frac{1}{2} ADV_{\mathcal{MEG}}^{\text{IND\$}}(A) + \frac{1}{2}. \end{aligned} \tag{3}$$

We will now compute the probability of B outputting 1 when W random. Then if we multiply an element m from $\mathbb{Z}_q^*$ with an uniformly random element W of the same set, we obtain an uniformly random element. Raising g to a random value v, yields a random element of $\mathbb{G}$ because g generates $\mathbb{G}$. Thus, $\alpha_0, \delta_0, \alpha_1, \delta_1$ are random. Since A has to choose between random elements, we have that

$$Pr[B(U,V,W) = 1 | W \xleftarrow{\$} \mathbb{Z}_q^*] = Pr[b = b' | W \xleftarrow{\$} \mathbb{Z}_q^*] = \frac{1}{2}. \tag{4}$$

Finally, the statement is proven by combining the equalities (1) − (4). □

Theorem 2. *If* HDH *is hard in* $\mathbb{G}$ *then* $\mathcal{MEG}$ *is* ANO-CPA *secure in the standard model. Formally, let* A *be an efficient PPT* ANO-CPA *adversary. There exists an efficient algorithm* B *such that*

$$ADV_{\mathcal{MEG}}^{\text{ANO-CPA}}(A) \leq 4ADV_{\mathbb{G},g,H}^{\text{HDH}}(B).$$

Proof. Let A be an ANO-CPA adversary for $\mathcal{MEG}$ with access to "random coins" sampled uniformly from a set R. We construct two adversaries B_1, B_2 for the HDH assumption and then we provide an upper bound for the advantage of A.

$$x_0 \xleftarrow{\$} \mathbb{Z}_q^*, y_0 \leftarrow g^{x_0}, x_1 \xleftarrow{\$} \mathbb{Z}_q^*, y_1 \leftarrow g^{x_1},$$
$$\rho \xleftarrow{\$} R, m \leftarrow A(\rho, y_0, y_1),$$
$$b \xleftarrow{\$} \{0,1\}, k \xleftarrow{\$} \mathbb{Z}_q^*, \alpha \leftarrow g^k, \beta \leftarrow y_b^k, \gamma \leftarrow H(\beta), \delta \leftarrow m \cdot \gamma,$$
$$b' \leftarrow A(\rho, y_0, y_1, \alpha, \delta).$$

Fig. 3. The ANO-CPA game.

Algorithm $B(U, V, W)$:
$$y_0 \leftarrow U, z \xleftarrow{\$} \mathbb{Z}_q^*, y_1 \leftarrow g^z, \mu_0 \leftarrow H(V^z), \mu_1 \xleftarrow{\$} \mathbb{Z}_q^*,$$
$$\rho \xleftarrow{\$} R, m \leftarrow A(\rho, y_0, y_1),$$
$$b \xleftarrow{\$} \{0,1\}, \alpha \leftarrow V, \omega_0 \leftarrow W, \omega_1 \leftarrow \mu_b,$$
$$b' \xleftarrow{\$} \{0,1\}, \delta \leftarrow m \cdot \omega_{b'},$$
$$b'' \leftarrow A(\rho, y_0, y_1, \alpha, \delta),$$
If $b' = b''$ then return 1 else return 0.

Fig. 4. Algorithm B for attacking HDH.

Figure 3 describes the ANO-IND game. The first row sets up the public keys y_0 and y_1. In the second row, the adversary selects the message m it wants to be challenged on. The challenger then flips a bit b, chooses a random k and it reveals the encryption of m under y_b. A then computes its guess b' for b. A wins if $b = b'$. Formally, the probability of A winning the ANO-IND game is

$$|2Pr[b' = b] - 1| = ADV_{\mathcal{MEG}}^{\text{ANO-CPA}}(A). \tag{5}$$

Figure 4 depict the behavior of algorithm B who runs the ANO-IND distinguisher A as a subroutine. B is given as input U, V, W, where $U \leftarrow g^u$ and $V \leftarrow g^v$, for random u, v, and W is either $H(g^{uv})$ or random. B outputs a bit indicating its guess for which of these cases occurs, where 1 means B guesses $W = H(g^{uv})$. Formally, the HDH advantage of B is

$$ADV_{\mathbb{G},g,H}^{\text{HDH}}(B) = |Pr[B(U, V, W) = 1|W = H(g^{uv})] - Pr[B(U, V, W) = 1|W \xleftarrow{\$} \mathbb{Z}_q^*]|. \tag{6}$$

Let us consider the case $W = H(g^{uv})$ and compute the probability of B outputting 1. There are two sub-cases when $b = 0$ and when $b = 1$. In the former sub-case, we note that B is running A as the latter would run an attack on the ANO-IND security of $\mathcal{MEG}$. Thus, we have

$$\begin{aligned} Pr[B(U, V, W) = 1|W = H(g^{uv})] &= Pr[b' = b''|W = H(g^{uv})] \\ &= Pr[b' = b''|W = H(g^{uv}), b = 0]Pr[b = 0] \\ &+ Pr[b' = b''|W = H(g^{uv}), b = 1]Pr[b = 1] \\ &= \frac{1}{4}(2Pr[b = b'|W = H(g^{uv}), b = 0] - 1 + 1) \\ &+ \frac{1}{2}Pr[b' = b''|W = H(g^{uv}), b = 1] \\ &= \frac{1}{4}ADV_{\mathcal{MEG}}^{\text{ANO-CPA}}(A) + \frac{1}{4} \\ &+ \frac{1}{2}Pr[b' = b''|W = H(g^{uv}), b = 1]. \end{aligned} \tag{7}$$

The probability of B outputting 1 when W is random is

$$\begin{aligned} Pr[B(U, V, W) = 1|W \xleftarrow{\$} \mathbb{Z}_q^*] &= Pr[b' = b''|W = H(g^{uv})] \\ &= Pr[b' = b''|W \xleftarrow{\$} \mathbb{Z}_q^*, b = 0]Pr[b = 0] \\ &+ Pr[b' = b''|W \xleftarrow{\$} \mathbb{Z}_q^*, b = 1]Pr[b = 1]. \end{aligned} \tag{8}$$

Lets consider the sub-case $b = 1$. If we multiply an element m from $\mathbb{Z}_q^*$ with an uniformly random element ω_0 or ω_1 of the same set, we obtain an uniformly random element. Then A has two decide between two pairs that have the same distribution. Thus, we have

$$Pr[b' = b'' | W \xleftarrow{\$} \mathbb{Z}_q^*, b = 1] = \frac{1}{2}. \tag{9}$$

In the sub-case $b = 0$, we have that

$$Pr[b' = b'' | W \xleftarrow{\$} \mathbb{Z}_q^*, b = 0] = Pr[b' = b'' | W = H(g^{uv}), b = 1], \tag{10}$$

since in both case A receives one random element and one of the form $H(V^e)$, where e is random. Thus, equality (8) becomes

$$Pr[B(U, V, W) = 1 | W \xleftarrow{\$} \mathbb{Z}_q^*] = \frac{1}{2} Pr[b' = b'' | W = H(g^{uv}), b = 1] + \frac{1}{4} \tag{11}$$

Finally, the statement is proven by combining the equalities (5) – (11). □

4 A SETUP Attack on the Generalized ElGamal Signature

We further introduce a new SETUP mechanism. Compared to Young-Yung's attack, it is very easy to modify our mechanism to allow ℓ out of n malicious parties to recover V's secret key[10]. The best we were able to do, using Young-Yung's mechanism, was to devise an ℓ out of ℓ threshold scheme[11]. We point out, that like Young-Yung's mechanism, our proposed mechanism leaks data continuously to the attacker.

4.1 Scheme Description

To implement the attack, M works in almost the same environment as in Section 2.4. Thus, we only mention the differences between the two environments.

Signing Sessions: The possible signing sessions performed by D are described below. Let $i \geq 1$.

$Session_0(m_0, sk_V)$: To sign message $m_0 \in \mathbb{G}$, D does the following

$$k_0 \xleftarrow{\$} \mathbb{Z}_q^*, r_0 \leftarrow g^{k_0}, s_0 \leftarrow k_0^{-1}[h(m_0) - a \cdot h(r_0)] \bmod q.$$

The value k_0 is stored in D's volatile memory until the end of $Session_1$. Output the signature (r_0, s_0).

$Session_i(m_i, sk_V, pk_M)$: To sign message $m_i \in \mathbb{G}$, D does the following

$$k_i \leftarrow k_{i-1} \cdot H(y_M^{k_{i-1}}), r_i \leftarrow g^{k_i}, s_i \leftarrow k_i^{-1}[h(m_i) - a \cdot h(r_i)] \bmod q.$$

The value k_i is stored in D's volatile memory until the end of $Session_{i+1}$. We remark that s_i is used as a data carrier for M. Output the signature (r_i, s_i).

[10] We refer the reader to Section 5.
[11] We refer the reader to Appendix C.

Recovering$(m_{i-1}, m_i, r_{i-1}, r_i, s_{i-1}, s_i, sk_M)$*:* Compute $\alpha \leftarrow [s_i \cdot H(r_{i-1}^{x_M})]^{-1}$. Recover a by computing

$$a \leftarrow \left(\alpha \cdot h(m_i) - s_{i-1}^{-1} \cdot h(m_{i-1})\right) \cdot \left(\alpha \cdot h(r_i) - s_{i-1}^{-1} \cdot h(r_{i-1})\right)^{-1} \bmod q.$$

The correctness of the *Recovering* algorithm can be obtained as follows. From *Session*$_{i-1}$ and *Session*$_i$, we obtain the value of k_{i-1}

$$k_{i-1} \equiv s_{i-1}^{-1}[h(m_{i-1}) - a \cdot h(r_{i-1})] \mod q \tag{12}$$

$$k_{i-1} \equiv [s_i \cdot H(y_M^{k_{i-1}})]^{-1} \cdot [h(m_i) - a \cdot h(r_i)] \mod q. \tag{13}$$

From equalities (12) and (13) we obtain

$$a \cdot \left(\alpha \cdot h(r_i) - s_{i-1}^{-1} \cdot h(r_{i-1})\right) \equiv \alpha \cdot h(m_i) - s_{i-1}^{-1} \cdot h(m_{i-1}) \bmod q.$$

Using the above equality and the fact that $y_M^{k_{i-1}} = r_{i-1}^{x_M}$, we obtain the correctness of the *Recovering* algorithm.

Remark 7. Let T be an honest generator for the values r used by the Generalized ElGamal signature scheme and let σ_i denote the i-th internal state and $\rho_i = g^{\sigma_i}$ the i-th output of T. The mechanism described above can be seen as a malicious PRNG $\tilde{T}$ based on the honest PRNG T. We define the internal states and outputs of $\tilde{T}$ by

– $\tilde{\sigma}_0 = \sigma_0$, $\tilde{\rho}_0 = \rho_0$;
– $\tilde{\sigma}_i = \tilde{\sigma}_{i-1} \cdot H(y_M^{\tilde{\sigma}_{i-1}})$, $\tilde{\rho}_i = g^{\tilde{\sigma}_i}$, where $i \geq 1$.

In the case of Dual-EC, if an attacker M knows output $\tilde{\rho}_{i-1}$ then he can compute the internal state $\tilde{\sigma}_i$. In the case of $\tilde{T}$, computing $\tilde{\sigma}_i$ also requires knowledge of the previous internal state $\tilde{\sigma}_{i-1}$. Since $\tilde{\sigma}_{i-1}$ is secret, the generator is not harmful on its own. But, if used to generate ephemeral keys g^k for ElGamal based signatures[12], it leads to a backdoor that enables M to break the security of the system.

4.2 Security Analysis

In this section we state the security margin for our variant of the ElGamal signature SETUP. We will defer the security proof of this scheme until the next section, since the scheme is a special case of the scheme described in Section 5.1. We denote by $\mathcal{GEGS}$ the Generalized ElGamal Signature and by $\mathcal{N}-\mathcal{GEGS}$ the scheme described in the previous subsection.

Theorem 3. *If the number of signatures is polynomial and* HDH *is hard in* $\mathbb{G}$ *then* $\mathcal{GEGS}$ *and* $\mathcal{N}-\mathcal{GEGS}$ *are* IND-SETUP *in the standard model. Formally, let A be an efficient PPT* IND-SETUP *adversary. There exists an efficient algorithm B such that*

$$ADV^{\text{IND-SETUP}}_{\mathcal{MEG},\mathcal{GEGS},\mathcal{N}-\mathcal{GEGS}}(A) \leq 4\Gamma ADV^{\text{HDH}}_{\mathbb{G},g,H}(B),$$

where Γ is the number of infected signatures.

Remark 8. Similarly to Theorem 3, we obtain that if T is a secure PRNG[13], then $\tilde{T}$ is a secure PRNG in the standard model.

Remark 9. As in the case of Dual-EC, it is easy to see that if in the $\mathcal{N}-\mathcal{GEGS}$ scheme, we replace y_M with $y'_M \stackrel{\$}{\leftarrow} \mathbb{G}$, the SETUP mechanism becomes benign. The security margin of the SETUP-free system remains the same as the one stated in Theorem 3.

[12] A well known vulnerability of ElGamal based signatures is that using the same k value twice, leads to secret key recovery [14].

[13] The outputs of T are computationally indistinguishable from the uniform distribution.

5 A Threshold SETUP Attack on the Generalized ElGamal Signature

In this section we introduce an ℓ out of n threshold SETUP attack, based on $\mathcal{N}-\mathcal{GEGS}$. In this secret sharing scenario, user V is the victim of n malicious parties (denoted by $\{M_i\}_{1\leq i\leq n}$) that somehow convince the manufacturer of D to implement the described SETUP mechanism. After D signs $n+1$ messages, any coalition of ℓ participants M_i can recover V's secret key. Once the key is obtained, V can be impersonated. We remark that starting from signature $\ell-1$ some coalitions of M_i can impersonate V.

5.1 Scheme Description

To ease description, we assume without loss of generality, that the first ℓ participants M_i decide to recover V's secret key and denote by $M=\{m_i\}_{0\leq i\leq \ell}$, $R=\{r_i\}_{0\leq i\leq \ell}$, $S=\{s_i\}_{0\leq i\leq \ell}$, $SK_M=\{sk_i\}_{1\leq i\leq \ell}$. We present our proposed threshold SETUP scheme below.

Malicious Parties KeyGen(pp): Let $H:\mathbb{G}\to\mathbb{Z}_q^*$ be a hash function. For each M_i, $1\leq i\leq n$, choose $x_i\xleftarrow{\$}\mathbb{Z}_q^*$ and compute $y_i\leftarrow g^{x_i}$. Output the public keys $pk_i=y_i$. The public keys pk_i and H will be stored in D's volatile memory. The secret keys are $sk_i=x_i$; they will only be known by the respective M_i and will not be stored in the black-box.

Signing Sessions: The possible signing sessions performed by D are described below. Let $1\leq i\leq n$ and $j>n$.

Session$_0(m_0, sk_V)$: To sign message $m_0\in\mathbb{G}$, D does the following

$$k_0\xleftarrow{\$}\mathbb{Z}_q^*, r_0\leftarrow g^{k_0}, s_0\leftarrow k_0^{-1}[h(m_0)-a\cdot h(r_0)] \bmod q.$$

The device also chooses $\{f_j\}_{1\leq j<\ell}$ at random from $\mathbb{Z}_q^*$ and forms the polynomial $f(z)=k_0+f_1\cdot z+\ldots+f_{\ell-1}\cdot z^{\ell-1}$. The polynomial $f(z)$ is stored in D's volatile memory until the end of *Session*$_n$. Output the signature (r_0,s_0).

Session$_i(m_i, sk_V, pk_i)$: To sign message $m_i\in\mathbb{G}$, D does the following

$$\begin{aligned}&k_i\leftarrow f(i)\cdot H(y_i^{k_0}),\ if\ f(i)\not\equiv 0 \bmod q;\\ &k_i\xleftarrow{\$}\mathbb{Z}_q^*,\ otherwise;\\ &r_i\leftarrow g^{k_i}, s_i\leftarrow k_i^{-1}[h(m_i)-a\cdot h(r_i)] \bmod q.\end{aligned}$$

We remark that s_i is used as a data carrier for M_i. Output the signature (r_i,s_i).

Session$_j(m_j, sk_V)$: To sign message $m_j\in\mathbb{G}$, D does the following

$$k_j\xleftarrow{\$}\mathbb{Z}_q^*, r_j\leftarrow g^{k_j}, s_j\leftarrow k_j^{-1}[h(m_j)-a\cdot h(r_j)] \bmod q.$$

Output the signature (r_j,s_j).

Recovering(M,R,S,SK_M): Compute $\alpha_i\leftarrow[s_i\cdot H(r_0^{x_i})]^{-1}$ and $\Delta_i\leftarrow\prod_{j\neq i}\frac{j}{j-i}$, $i,j\leq\ell$. Recover a by computing

$$a\leftarrow\left(\sum_{i=1}^{\ell}\alpha_i\cdot h(m_i)\cdot\Delta_i-s_0^{-1}\cdot h(m_0)\right)\cdot\left(\sum_{i=1}^{\ell}\alpha_i\cdot h(r_i)\cdot\Delta_i-s_0^{-1}\cdot h(r_0)\right)^{-1} \bmod q. \tag{14}$$

The correctness of the *Recovering* algorithm can be obtained as follows. From *Session*$_0$, we obtain the value of k_0

$$k_0\equiv s_0^{-1}[h(m_0)-a\cdot h(r_0)] \mod q. \tag{15}$$

From $Sessions_i$, we obtain M_i's share

$$f(i) \equiv [s_i \cdot H(y_i^{k_0})]^{-1} \cdot [h(m_i) - a \cdot h(r_i)] \mod q.$$

Using Lagrange interpolation we use the shares $f(i)$, $1 \le i \le \ell$ to recover k_0

$$k_0 \equiv \sum_{i=1}^{\ell} [s_i \cdot H(y_i^{k_0})]^{-1} \cdot [h(m_i) - a \cdot h(r_i)] \cdot \Delta_i \mod q. \tag{16}$$

From equalities (15) and (16) we obtain

$$a \cdot \left(\sum_{i=1}^{\ell} \alpha_i \cdot h(r_i) \cdot \Delta_i - s_0^{-1} \cdot h(r_0) \right) \equiv \sum_{i=1}^{\ell} \alpha_i \cdot h(m_i) \cdot \Delta_i - s_0^{-1} \cdot h(m_0) \mod q.$$

Using the above equality and the fact that $y_i^{k_0} = r_0^{x_i}$, we obtain the correctness of the *Recovering* algorithm.

Remark 10. The probability that key recovery is not possible due to failure is $\epsilon = 1 - \left(1 - \frac{1}{q}\right)^{n-\ell+1}$. Since q is a large prime number, we have that $\epsilon \simeq 0$.

Remark 11. When all n participants are required to recover V's secret key, the scheme described in Appendix C requires two infected signatures, while the above scheme requires n infected signatures. Thus, the scheme described in this section is less efficient in this case. Unfortunately, we could not devise a method to extend the scheme described in Appendix C to an ℓ out of n threshold scheme.

Remark 12. The mechanism described in this section requires the malicious parties to directly compute V's secret key. In some cases this raises security concerns. For example, if the mechanism is used for surveillance purposes and a warrant is issued, if V's secret key is directly computed, when the warrant expires V can still be impersonated. In Appendix B we present a two party protocol extension of our scheme in order to mitigate this issue. We could not find an extension for the scheme described in Appendix C.

Remark 13. In the scheme described above, D plays the role of a trusted dealer, that leaks the shares using a subliminal channel to the n participants. This design choice was made in order to minimize communication between the malicious parties. The only moment when the participants communicate is when ℓ of them want to recover V's secret key.

Another possible scenario, was to use a secret sharing protocol with or without a trusted dealer between the n parties. After the participants agree on a shared public key $y_M = g^{x_M}$, the manufacturer implements, for example, the mechanism described in Section 2.4[14]. Note that this approach works without any modifications to the SETUP mechanism.

Remark 14. Let P be an honest generator for the values r used by the Generalized ElGamal signature scheme and let σ_i denote the i-th internal state and $\rho_i = g^{\sigma_i}$ the i-th output of P. The mechanism described above can be seen as a malicious PRNG $\tilde{P}$ based on the honest PRNG P. We define the internal states and outputs of $\tilde{P}$ by

- $\tilde{\sigma}_0 = \sigma_0$, $\tilde{\rho}_0 = \rho_0$;
- $\tilde{\sigma}_i = f(i) \cdot H(y_i^{\sigma_0})$, $\tilde{\rho}_i = g^{\tilde{\sigma}_i}$, where $f(z) = \sigma_0 + \sigma_1 \cdot z + \ldots + \sigma_{\ell-1} \cdot z^{\ell-1}$ and $1 \le i \le n$;
- $\tilde{\sigma}_j = \sigma_j$, $\tilde{\rho}_j = \rho_j$, where $j > n$.

Because σ_0 and σ_j, where $j > n$, are identical for P and $\tilde{P}$ generator $\tilde{P}$ remains unpredictable. In the case $1 \le i \le n$, a group of ℓ malicious parties can prove that their $\tilde{\rho}_i$ are not random, but they cannot compute $\tilde{P}$'s internal states. Thus, when used on its own $\tilde{P}$ is mostly harmless. Unfortunately, if it is used to generate r for ElGamal based signatures, then ℓ malicious parties can recover the V's secret key.

[14] that uses y_M

5.2 Security Analysis

In this subsection we prove that the threshold version described above, denoted $\mathcal{S}-\mathcal{GEGS}$, is indistinguishable from $\mathcal{GEGS}$ if the attacker corrupted at most $\ell-1$ out of n malicious parties M_i.

Theorem 4. *If* HDH *is hard in* $\mathbb{G}$ *then* $\mathcal{GEGS}$ *and* $\mathcal{S}-\mathcal{GEGS}$ *are* IND-SETUP *in the standard model as long as at most* $\ell-1$ *malicious parties are corrupted by* A*. Formally, let* A *be an efficient PPT* IND-SETUP *adversary. There exists an efficient algorithm* B *such that*

$$ADV^{\text{IND-SETUP}}_{\mathcal{MEG},\mathcal{GEGS},\mathcal{S}-\mathcal{GEGS}}(A) \leq 4(n-\ell+1)ADV^{\text{HDH}}_{\mathbb{G},g,H}(B).$$

Proof. Let A be an IND-SETUP adversary that is trying to distinguish between $\mathcal{GEGS}$ and $\mathcal{S}-\mathcal{GEGS}$. A has access to "random coins" sampled uniformly from a set R. Without loss of generality, we further assume that A has corrupted the first $\ell-1$ malicious participants.

$$\begin{aligned}
& a \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*, y \leftarrow g^a, x_1, \ldots, x_n \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*, y_1 \leftarrow g^{x_1}, \ldots, y_n \leftarrow g^{x_n},\\
& \mathcal{L}_1 \leftarrow \left(\cup_{i=1}^{\ell-1}\{x_i\}\right) \cup \left(\cup_{i=1}^{n}\{y_i\}\right), i \leftarrow 0;\\
C_0(a,m):\ & k \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*, r \leftarrow g^k, s \leftarrow k^{-1}[h(m) - a\cdot h(r)];\\
C_1(a,m):\ & k \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*, k_0 \leftarrow k, f_1, \ldots, f_{\ell-1} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*,\ if\ i = 0,\\
& k \leftarrow f(i)\cdot H(y_i^{k_0}),\ if\ 0 < i \leq n\ and\ f(i) \not\equiv 0 \bmod q,\\
& k \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*,\ otherwise,\\
& r \leftarrow g^k, s \leftarrow k^{-1}[h(m) - a\cdot h(r)], i \leftarrow i+1;\\
& b \stackrel{\$}{\leftarrow} \{0,1\}, \rho \stackrel{\$}{\leftarrow} R, b' \leftarrow A^{C_b(a,\cdot)}(\rho, y, \mathcal{L}_1).
\end{aligned}$$

Fig. 5. The IND-SETUP game.

Figure 5 describes the IND-SETUP game. The first and second rows set up the public keys. Then the $\mathcal{GEGS}$ and $\mathcal{S}-\mathcal{GEGS}$ oracles are described. The challenger then flips a bit b and reveals oracle C_b. A then computes its guess b' for b. A wins if $b = b'$.

We proceed by modifying oracle C_1 (described in Figure 5) into oracle C_2 (described in Figure 6). The only difference between the two oracles is that in C_2 the values k_i, $0 < i \leq \ell-1$, are chosen at random. Since Shamir's secret sharing scheme is information theoretically secure, an adversary cannot distinguish between C_1 and C_2.

$$\begin{aligned}
C_2(a,m):\ & k \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*, k_0 \leftarrow k, f_1, \ldots, f_{\ell-1} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*,\ if\ i = 0,\\
& k \leftarrow f(i)\cdot H(y_i^{k_0}),\ if\ \underline{\ell \leq i \leq n}\ and\ f(i) \not\equiv 0 \bmod q,\\
& k \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*,\ otherwise,\\
& r \leftarrow g^k, s \leftarrow k^{-1}[h(m) - a\cdot h(r)], i \leftarrow i+1.
\end{aligned}$$

Fig. 6. Oracle C_2.

Since $\mathcal{MEG}$ is IND\$ an adversary cannot distinguish between C_0 and C_2. Note that the number of k values that A has to distinguish is $n-\ell+1$. Thus, we obtain the security margin. ◻

Remark 15. Similarly to Theorem 4, we obtain that if P is a secure PRNG, then $\tilde{P}$ is a secure PRNG in the standard model.

Remark 16. As in the case of Dual-EC, it is easy to see that if in the $\mathcal{S}-\mathcal{GEGS}$ scheme, we replace y_i with $y'_i \stackrel{\$}{\leftarrow} \mathbb{G}$, $1 \leq i \leq n$, the SETUP mechanism becomes benign. The security margin of the SETUP-free system remains the same as the one stated in Theorem 4.

6 Other Applications

The schemes described in Section 5 and Appendix C can either directly be used on other signatures (*e.g.* variations of the Generalized ElGamal signature [46], Pointcheval-Stern signature [57]) or indirectly, *i.e.* some work must be done to recover g^k (*e.g.* Schnorr signature [60] - see Example 1, DSA [28]).

Example 1. To be more precise, we describe the method used in the case of Schnorr signatures. We place ourselves in the subgroup of order q generated by a $g \in \mathbb{Z}_p^*$, where p is prime. The signature generation algorithm is

$$k \xleftarrow{\$} \mathbb{Z}_q^*, r \leftarrow h(g^k||m), s \leftarrow a \cdot r + k \bmod q.$$

In order to recover g^k, one must compute

$$g^s \cdot y^{-r} \equiv g^{s-ar} \equiv g^k.$$

After finding a method to recover g^k, either directly or by computing it from the signature, it is fairly easy to use the methods described in Section 5 and Appendix C. All the signatures presented in this section either have g^k directly embedded in them or the recovering mechanism is similar to the one presented in Example 1.

Some signature schemes that can be tampered with and also have the same security as $\mathcal{S}-\mathcal{GEGS}$ are: variations of the Generalized ElGamal signature [46], ECDSA [4], ECDSA variants [45], Katz-Wang signature [38], KCDSA [42], Elliptic Curve GOST [25], EDL signature Goh-Jarecki variant [34], EDL signature Chevallier variant [17] and Elliptic Curve Nyberg-Rueppel [48].

If $\mathbb{G}$ is generated by an element $g \in \mathbb{Z}_p^*$ of order q, we can apply the same methods and obtain security in the standard model[15] for the following algorithms: DSA [28], GOST [47], Nyberg-Rueppel [52], Nyberg-Rueppel IEEE variant [48], Pointcheval-Stern signature [57], Schnorr signature [60] and Girault-Poupard-Stern (GPS) signature [33], if parameter A used by the GPS signature is prime.

Schnorr [60] and Girault-Poupard-Stern signatures [33] are derived from identification schemes. As a consequence, we can apply similar methods to infect these identification schemes and compromise V's secret key. Another identification scheme that offers the possibility of embedding a secret trapdoor is Okamoto's scheme [53].

Signcrypt algorithms [72, 73] use a variation of the ElGamal signature in order to authenticate messages and use a key derivation function based on the recipient's secret key in order to encrypt messages. So, if we embed the threshold SETUP mechanism in the signature and manage to recover the signer's secret key, then we can also decrypt all the messages that the signer receives.

Changing the setting to identity based signatures (IBS), we observe that Cha-Cheon IBS [18], Hess IBS [37] and Paterson IBS [55] can be infected and the resulting schemes are secure in ROM[15]. A signature that can be tampered with and obtain the same security as $\mathcal{S}-\mathcal{GEGS}$, is Bellare-Namprempre-Neven IBS [9]. This signature offers an extra feature, we can also modify the extraction algorithm, permitting ℓ out of n legitimate users to obtain the master key (used by the central authority to generate keys for any legitimate user).

When random numbers are not available or of questionable quality (*e.g.* malicious RNG), one may use deterministic signatures. One such example is the deterministic variant of the Schnorr signature scheme introduced in [50]. The authors suggest to choose $k \leftarrow h(\kappa, m, pp)$, where κ is a fixed secret. Unfortunately, this approach does not protect V. When implementing a SETUP attack for this scheme, we must ensure the same functionality as in the SETUP-free version (*i.e.* signing the same message multiple times yields the same signature). In the following we give two attacks for this deterministic signature. In the first attack, a malicious party replaces κ by $\kappa' \leftarrow H(y_M^a)$ and recovers V's secret key by computing $a \leftarrow h(r)^{-1}[h(m) - k \cdot s]$. This attack can be easily extended to an ℓ out of ℓ attack[16], but we were not able to extend it to an ℓ out of n attack. In the second attack, D stores a list $\mathcal{L}$ containing the messages received as input and the associated

[15] We refer the reader to Remark 1.

[16] We refer the reader to Appendix C.

signatures. When a message m is received, D will first search m in $\mathcal{L}$. If m is found, D will return the stored signature, else it will generate a new infected signature. If D runs out of memory, it reverts to $k \leftarrow h(\kappa, m, pp)$. To save memory an attacker could, for example, restrict D to maliciously signing only short messages.

7 Countermeasures

An intuitive protection against SETUP attacks is to use multiple devices manufactured by different companies. The underlying intuition is that it is less likely to corrupt all the vendors at the same time. Thus, by signing the same document with independent devices, the user has high confidence that at least one private key is not leaked to the attackers. Unfortunately, this intuitive protection comes with a performance cost (increased length of the signature and longer verification time).

In [5], the authors show that unique signatures schemes[17] are secure against subversion algorithms that satisfy the verifiability condition[17]. The SETUP mechanisms described in Section 5 and in Appendix C meet the verifiability condition, thus unique signature schemes are secure against these mechanisms. If device D uses a re-randomizable signature scheme[17], then another method of protection is the usage of an external un-tamperable cryptographic reverse firewall[17]. The role of the external device is to prevent data exfiltration, while maintaining functionality and preserve security. The reverse firewall fulfills these requirements by re-randomizing the signature. By using such a device, V protects himself from our mechanisms.

A more general approach to subversion resistant signatures may be found in [58, 59]. The authors propose splitting every generation algorithm into two parts: a random string generation algorithm RG and a deterministic algorithm DG. The deterministic algorithm can be tested extensively, thus ensuring that it is *almost consistent* with the specifications. This forces the malicious parties to concentrate their efforts on the RG algorithm. To ensure that any backdoor implemented in the RG algorithm does not affect the deterministic part, the authors use two RG algorithms, concatenate the outputs and hash them, before passing the data to DG. Since we use a backdoor in the generator to leak information about the secret key, the immunization techniques proposed in [58, 59] protect the user against our proposed mechanisms.

Another method to counter these SETUP attacks is to use threshold signatures. We could not find any reference or devise a method to apply SETUP mechanisms to this setting. Some examples of threshold signatures are: threshold Schnorr signature scheme [65], threshold DSS signature [30] and threshold signature schemes for ElGamal variants [39].

A variation of the Schnorr identification scheme is presented in [40]. The author proves that the modified scheme is secure when the ephemeral key k is chosen by an attacker, if GDH[18] is hard in the underlying group $\mathbb{G}$. A variant of the Okamoto identification scheme is introduced in [40] and proven secure, under the same assumptions [41]. Since s is never sent in clear, only $\tilde{g}^s$ is sent, for some $\tilde{g}$, we cannot recover the secret key by forcing a collision for k. Thus, our proposed mechanisms do not work for these variants. Applying the Fiat-Shamir transform [27], we obtain signatures schemes that are also secure against our attacks.

In [36], the authors introduce a deterministic method for controlling a PRNG implemented in a black-box device. The idea is to let the user install a blinding factor $U = g^u$ in the device. The user keeps the value u secret. After a successful installation, the device will start generating pseudorandom numbers. Each time a number is generated, the device will also output some control data (r', i). Using (r', i) and u, the user can check if the device is attempting to cheat. Thus, no manufacturer will risk implementing our proposed mechanisms.

8 Conclusions

In this paper we introduced two threshold SETUP mechanisms that allow a group of malicious parties to recover a user's secret key. We adapted Shamir's secret sharing scheme [61] and the Hashed ElGamal encryption scheme [62] in order to infect the Generalized ElGamal signature scheme [46]. Depending on the

[17] See Appendix A for a definition of the concept.

[18] *i.e.* in $\mathbb{G}$ DDH can be solved in polynomial time, but CDH is hard

underlying group of the signature scheme, we prove that our proposed schemes are secure in the standard or random oracle model.

As an application of the devised threshold SETUP methods, we present other schemes that can be modified in order to recover a user's secret key. We, also provide countermeasures for the mechanisms described in Section 5 and in Appendix C.

Future Work. An interesting area of research would consist in finding a method to extend SETUP attacks applied to encryption schemes to threshold SETUP attacks. Also, it would be interesting to see if one can mount a successful SETUP attack or threshold SETUP attack if threshold signature schemes are used.

In Appendix C we describe an ℓ out of ℓ threshold SETUP mechanism that uses only two sessions in order to recover V's secret key. An extension to ℓ out of n would be more efficient than the approach from Section 5.1.

Acknowledgments

The author would like to thank Adrian Atanasiu, Alejandro Hevia , Tanja Lange, Diana Maimuţ and Ferucio Laurenţiu Ţiplea, and the anonymous reviewers for their helpful comments.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. IACR Cryptology ePrint Archive **1999/7** (1999)
2. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: CT-RSA 2001. Lecture Notes in Computer Science, vol. 2020, pp. 143–158. Springer (2001)
3. Albertini, A., Aumasson, J.P., Eichlseder, M., Mendel, F., Schläffer, M.: Malicious Hashing: Eve's Variant of SHA-1. In: SAC 2014. Lecture Notes in Computer Science, vol. 8781, pp. 1–19. Springer (2014)
4. Association, A.B., et al.: Working Draft: American National Standard X9. 62-1998 Public Key Cryptography for the Financial Services Industry (1998)
5. Ateniese, G., Magri, B., Venturi, D.: Subversion-Resilient Signature Schemes. In: ACM-CCS 2015. pp. 364–375. ACM (2015)
6. Ball, J., Borger, J., Greenwald, G.: Revealed: How US and UK Spy Agencies Defeat Internet Privacy and Security. The Guardian **6** (2013)
7. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in Public-Key Encryption. In: ASIACRYPT 2001, Lecture Notes in Computer Science, vol. 2248, pp. 566–582. Springer (2001)
8. Bellare, M., Jaeger, J., Kane, D.: Mass-Surveillance without the State: Strongly Undetectable Algorithm-Substitution Attacks. In: ACM-CCS 2015. pp. 1431–1440. ACM (2015)
9. Bellare, M., Namprempre, C., Neven, G.: Security Proofs for Identity-Based Identification and Signature Schemes. Journal of Cryptology **22**(1), 1–61 (2009)
10. Bellare, M., Paterson, K.G., Rogaway, P.: Security of Symmetric Encryption Against Mass Surveillance. In: CRYPTO 2014. Lecture Notes in Computer Science, vol. 8616, pp. 1–19. Springer (2014)
11. Bellare, M., Rogaway, P.: Minimizing the Use of Random Oracles in Authenticated Encryption Schemes. In: ICICS 1997. Lecture Notes in Computer Science, vol. 1334, pp. 1–16. Springer (1997)
12. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography. UCSD CSE **207**, 207 (2005)
13. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: A Standardized Back Door. In: The New Codebreakers, Lecture Notes in Computer Science, vol. 9100, pp. 256–281. Springer (2016)
14. Cantero, H.M., Peter, S., Bushing, S.: Console Hacking 2010–PS3 Epic Fail. In: 27th Chaos Communication Congress (2010)
15. Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.P., Rescorla, E., Shacham, H.: A Systematic Analysis of the Juniper Dual EC Incident. In: ACM-CCS 2016. pp. 468–479. ACM (2016)
16. Checkoway, S., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H., Fredrikson, M.: On the Practical Exploitability of Dual EC in TLS Implementations. In: USENIX Security Symposium. pp. 319–335. USENIX Association (2014)

17. Chevallier-Mames, B.: An Efficient CDH-Based Signature Scheme with a Tight Security Reduction. In: CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 511–526. Springer (2005)
18. Choon, J.C., Cheon, J.H.: An Identity-Based Signature from Gap Diffie-Hellman Groups. In: PKC 2003. Lecture Notes in Computer Science, vol. 2567, pp. 18–30. Springer (2003)
19. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Science & Business Media (2013)
20. Degabriele, J.P., Farshim, P., Poettering, B.: A More Cautious Approach to Security Against Mass Surveillance. In: FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 579–598. Springer (2015)
21. Degabriele, J.P., Paterson, K.G., Schuldt, J.C., Woodage, J.: Backdoors in Pseudorandom Number Generators: Possibility and Impossibility Results. In: CRYPTO 2016. Lecture Notes in Computer Science, vol. 9814, pp. 403–432. Springer (2016)
22. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A Formal Treatment of Backdoored Pseudorandom Generators. In: EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9056, pp. 101–126. Springer (2015)
23. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 494–510. Springer (2004)
24. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message Transmission with Reverse Firewalls—Secure Communication on Corrupted Machines. In: CRYPTO 2016. Lecture Notes in Computer Science, vol. 9814, pp. 341–372. Springer (2016)
25. Dolmatov, V., Degtyarev, A.: GOST R 34.10-2012: Digital Signature Algorithm (2013)
26. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
27. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
28. FIPS, P.: 186-4. Digital Signature Standard (DSS) (2013)
29. Fried, J., Gaudry, P., Heninger, N., Thomé, E.: A Kilobit Hidden SNFS Discrete Logarithm Computation. In: EUROCRYPT 2017. Lecture Notes in Computer Science, vol. 10210, pp. 202–231. Springer (2017)
30. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust Threshold DSS Signatures. In: EUROCRYPT 1996. Lecture Notes in Computer Science, vol. 1070, pp. 354–371. Springer (1996)
31. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: EUROCRYPT 1999. Lecture Notes in Computer Science, vol. 1592, pp. 295–310. Springer (1999)
32. Gennaro, R., Krawczyk, H., Rabin, T.: Secure Hashed Diffie-Hellman over Non-DDH Groups. In: EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 361–381. Springer (2004)
33. Girault, M., Poupard, G., Stern, J.: On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. Journal of Cryptology **19**(4), 463–487 (2006)
34. Goh, E.J., Jarecki, S.: A Signature Scheme as Secure as the Diffie-Hellman Problem. In: EUROCRYPT 2003, Lecture Notes in Computer Science, vol. 2656, pp. 401–415. Springer (2003)
35. Gordon, D.: Designing and Detecting Trapdoors for Discrete Log Cryptosystems. In: CRYPTO 1992. Lecture Notes in Computer Science, vol. 740, pp. 66–75. Springer (1993)
36. Hanzlik, L., Kluczniak, K., Kutyłowski, M.: Controlled Randomness - A Defense against Backdoors in Cryptographic Devices. In: MyCrypt 2016. Lecture Notes in Computer Science, vol. 10311, pp. 215–232. Springer (2016)
37. Hess, F.: Efficient Identity Based Signature Schemes Based On Pairings. In: SAC 2002. Lecture Notes in Computer Science, vol. 2595, pp. 310–324. Springer (2002)
38. Katz, J., Wang, N.: Efficiency Improvements for Signature Schemes With Tight Security Reductions. In: ACM-CCS 2003. pp. 155–164. ACM (2003)
39. Kim, S., Kim, J., Cheon, J.H., Ju, S.H.: Threshold Signature Schemes for ElGamal Variants. Computer Standards & Interfaces **33**(4), 432–437 (2011)
40. Krzywiecki, Ł.: Schnorr-Like Identification Scheme Resistant to Malicious Subliminal Setting of Ephemeral Secret. In: SECITC 2016. Lecture Notes in Computer Science, vol. 10006, pp. 137–148. Springer (2016)
41. Krzywiecki, Ł., Kutylowski, M.: Security of Okamoto Identification Scheme: a Defense against Ephemeral Key Leakage and Setup. In: ACM-SCC@ASIACCS 2017. pp. 43–50. ACM (2017)
42. Lim, C.H., Lee, P.J.: A Study on the Proposed Korean Digital Signature Algorithm. In: ASIACRYPT 1998. Lecture Notes in Computer Science, vol. 1514, pp. 175–186. Springer (1998)
43. Lindell, Y.: Fast Secure Two-Party ECDSA Signing. In: CRYPTO 2017. Lecture Notes in Computer Science, vol. 10402, pp. 613–644. Springer (2017)

44. Maimuţ, D., Teşeleanu, G.: Secretly Embedding Trapdoors into Contract Signing Protocols. In: SECITC 2017. Lecture Notes in Computer Science, vol. 10543. Springer (2017)
45. Malone-Lee, J., Smart, N.P.: Modifications of ECDSA. In: SAC 2002. Lecture Notes in Computer Science, vol. 2595, pp. 1–12. Springer (2002)
46. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC press (1996)
47. Michels, M., Naccache, D., Petersen, H.: GOST 34.10-A Brief Overview of Russia's DSA. Computers & Security **15**(8), 725–732 (1996)
48. Microprocessor, Committee, M., et al.: IEEE Standard Specifications for Public-Key Cryptography. IEEE Computer Society (2000)
49. Mironov, I., Stephens-Davidowitz, N.: Cryptographic Reverse Firewalls. In: ASIACRYPT 2015. Lecture Notes in Computer Science, vol. 9057, pp. 657–686. Springer (2015)
50. M'Raïhi, D., Naccache, D., Pointcheval, D., Vaudenay, S.: Computational Alternatives to Random Number Generators. In: SAC 1998. Lecture Notes in Computer Science, vol. 1556, pp. 72–80. Springer (1998)
51. Naor, M., Reingold, O.: Number-Theoretic Constructions of Efficient Pseudo-Random Functions. Journal of the ACM (JACM) **51**(2), 231–262 (2004)
52. Nyberg, K., Rueppel, R.A.: A New Signature Scheme Based on the DSA Giving Message Recovery. In: ACM-CCS 1993. pp. 58–61. ACM (1993)
53. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: CRYPTO 1992. Lecture Notes in Computer Science, vol. 740, pp. 31–53. Springer (1992)
54. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Eurocrypt 1999. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999)
55. Paterson, K.G.: ID-Based Signatures from Pairings on Elliptic Curves. Electronics Letters **38**(18), 1025–1026 (2002)
56. Perlroth, N., Larson, J., Shane, S.: NSA Able to Foil Basic Safeguards of Privacy on Web. The New York Times **5** (2013)
57. Pointcheval, D., Stern, J.: Security Proofs For Signature Schemes. In: EUROCRYPT 1996. Lecture Notes in Computer Science, vol. 1070, pp. 387–398. Springer (1996)
58. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10032, pp. 34–64. Springer (2016)
59. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Destroying Steganography via Amalgamation: Kleptographically CPA Secure Public Key Encryption. IACR Cryptology ePrint Archive **2016/530** (2016)
60. Schnorr, C.P.: Efficient Identification and Signatures For Smart Cards. In: CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
61. Shamir, A.: How to Share a Secret. Communications of the ACM **22**(11), 612–613 (1979)
62. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs. IACR Cryptology ePrint Archive **2004/332** (2004)
63. Simmons, G.J.: The Subliminal Channel and Digital Signatures. In: EUROCRYPT 1984. Lecture Notes in Computer Science, vol. 209, pp. 364–378. Springer (1984)
64. Simmons, G.J.: Subliminal Communication is Easy Using the DSA. In: EUROCRYPT 1993. Lecture Notes in Computer Science, vol. 765, pp. 218–232. Springer (1993)
65. Stinson, D.R., Strobl, R.: Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates. In: ACISP 2001. Lecture Notes in Computer Science, vol. 2119, pp. 417–434. Springer (2001)
66. Vazirani, U.V., Vazirani, V.V.: Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design. In: FOCS 1983. pp. 23–30. IEEE (1983)
67. Young, A., Yung, M.: The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone? In: CRYPTO 1996. Lecture Notes in Computer Science, vol. 1109, pp. 89–103. Springer (1996)
68. Young, A., Yung, M.: Kleptography: Using Cryptography Against Cryptography. In: EUROCRYPT 1997. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer (1997)
69. Young, A., Yung, M.: The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems. In: CRYPTO 1997. Lecture Notes in Computer Science, vol. 1294, pp. 264–276. Springer (1997)
70. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. John Wiley & Sons (2004)
71. Young, A., Yung, M.: Malicious Cryptography: Kleptographic Aspects. In: CT-RSA 2005, Lecture Notes in Computer Science, vol. 3376, pp. 7–18. Springer (2005)
72. Zheng, Y.: Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost (Signature)+ Cost (Encryption). In: CRYPTO 1997, Lecture Notes in Computer Science, vol. 1294, pp. 165–179. Springer (1997)
73. Zheng, Y., Imai, H.: How to Construct Efficient Signcryption Schemes on Elliptic Curves. Information Processing Letters **68**(5), 227–233 (1998)
74. Zheng, Y., Seberry, J.: Immunizing Public Key Cryptosystems Against Chosen Ciphertext Attacks. IEEE Journal on Selected Areas in Communications **11**(5), 715–724 (1993)

A Additional Preliminaries

Definition 10 (Entropy Smoothing - ES). *Let $\mathbb{G}$ be a cyclic group of order q, $\mathcal{K}$ the key space and $\mathcal{H} = \{h_i\}_{i\in\mathcal{K}}$ a family of keyed hash functions, where each h_i maps $\mathbb{G}$ to $\mathbb{Z}_q^*$. Let A be a PPT algorithm which returns 1 on input (i, y) if $y = h_i(z)$, where z is chosen at random from $\mathbb{G}$. Also, let We define the advantage*

$$ADV^{\text{ES}}_{\mathcal{H}}(A) = |Pr[A(i, h_i(z)) = 1 | i \xleftarrow{\$} \mathcal{K}, z \xleftarrow{\$} \mathbb{G}] - Pr[A(i, h) = 1 | i \xleftarrow{\$} \mathcal{K}, h \xleftarrow{\$} \mathbb{Z}_q^*]|.$$

If $ADV^{\text{ES}}_{\mathcal{H}}(A)$ is negligible for any PPT algorithm A, we say that $\mathcal{H}$ is Entropy Smoothing.

Remark 17. In [23], the authors prove that CBC-MAC, HMAC and Merkle-Damgård constructions satisfy the above definition, as long as the underlying primitives satisfy some security properties.

Definition 11 (Unique Signature Scheme). *Let S be a signature scheme and pk be a public key generated by the KeyGen algorithm of S. We say that S is a Unique Signature Scheme if for any message m and any signatures of m, $\sigma_1 \neq \sigma_2$*

$$Pr[\mathit{Verification}(m, \sigma_1, pk) = \mathit{Verification}(m, \sigma_2, pk) = \texttt{true}]$$

is negligible.

Definition 12 (Re-Randomizable Signature Scheme). *Let S be a signature scheme and (pk, sk) be a public/secret key pair generated by the KeyGen algorithm of S. We say that S is a Re-Randomizable Signature Scheme if there exists a PPT algorithm ReRand such that for all messages m the output of $\mathit{ReRand}(m, \sigma, pk)$ is statistically indistinguishable from $\mathit{Sign}(m, sk)$.*

Definition 13 (Verifiability Condition). *Let A be a subversion algorithm for a signature scheme S. Let pk be a public key generated by the KeyGen algorithm of S. We say A satisfies the verifiability condition if for all messages m and all signatures generated by A for message m*

$$Pr[it\ \mathit{Verification}(m, \sigma, pk) = 1] \tag{17}$$

is non-negligible.

Definition 14 (Reverse Firewall). *Let S be a signature scheme and pk be a public key generated by the KeyGen algorithm of S. A Reverse Firewall for S consists of two algorithms: KeyGen and Patch. The first algorithm takes as input a security parameter and pk and outputs some initial state. The last algorithm takes as input the current state and a message/signature pair and outputs a modified signature or a special symbol $\perp$ and an updated state.*

B Two-Party Malicious Signing

In [43], the author introduces a two-party protocol for signing with ECDSA. Based on this idea, we sketch a protocol that extends $\mathcal{S}-\mathcal{GEGS}$. Using this extension, two malicious parties M_1 and M_2 can impersonate V without explicitly computing sk_V.

As the Paillier cryptosystem [54] is later used in the protocol, we shortly describe its homomorphic properties. We denote the public and private key pair of M_1 by (pk_p, sk_p), Paillier encryption by $Enc(pk_p, \cdot)$ and Paillier decryption by $Dec(sk_p, \cdot)$. Let n be a large composite number in the Paillier scheme sense, $c_1 \leftarrow Enc(pk_p, m_1)$ and $c_2 \leftarrow Enc(pk_p, m_2)$, where messages $m_1, m_2 \in \mathbb{Z}_n$. The upcoming properties hold

- the *addition* of m_1 and m_2 modulo n (represented by $c_1 \oplus c_2$ in the current section): $Dec(sk_p, c_1c_2 \bmod n^2) = m_1 + m_2 \bmod n$;
- the *multiplication* of m_1 by a constant t modulo n (represented by $t \odot c_1$ in the current section): $Dec(sk_p, c_1^t \bmod n^2) = tm_1 \bmod n$.

Before the malicious signing protocol can start, the two parties must agree on the protocol's parameters. In Figure 7 we present the parameters agreement protocol. The protocol uses an ideal commitment scheme and an ideal non-interactive zero-knowledge proof. For concrete instantiation of the two, we refer the reader to [43].

M_1		M_2
α_1		α_2
$Q \leftarrow g^{\alpha_1}$ Compute DL proof π_1. Generate a Paillier key-pair (pk_p, sk_p). Generate a proof π_2 that pk_p is generated correctly. $c_{key} \leftarrow Enc(pk_p, \alpha_1)$ Generate a proof π_3 of a DL in c_{key}.		
	$\xrightarrow{Q,\pi_1,\pi_2,\pi_3}$	
		Verify proofs π_1, π_2 and π_3. If `false` abort, else store c_{key}.

Fig. 7. Parameters Generation.

Let m_3 be the message that M_1 wants to sign. By combining equation (14) with the $\mathcal{GEGS}$ signing operation we obtain the following equation for malicious signing m_3

$$s_3 \leftarrow k_3^{-1}\xi_1^{-1}[\xi_1 \cdot h(m_3) - \xi_2 \cdot h(r_3)] \bmod q, \tag{18}$$

where

$$\xi_1 \leftarrow \left(\sum_{i=1}^{2} \alpha_i \cdot h(r_i) \cdot \Delta_i - s_0^{-1} \cdot h(r_0)\right) \qquad \text{and} \qquad \xi_2 \leftarrow \left(\sum_{i=1}^{2} \alpha_i \cdot h(m_i) \cdot \Delta_i - s_0^{-1} \cdot h(m_0)\right).$$

In Figure 8 we describe in detail the two-party protocol for signing m_3. To simplify the protocol, instead of $h(m)$ and $h(r)$ we simply write m and r. As in Figure 7, we use a commitment scheme and a zero knowledge protocol.

We can observe that, by using c_{key} and the homomorphic properties of the Paillier cryptosystem, M_2 can encrypt $u_1 \leftarrow k_{33}\xi_1$ and $u_2 \leftarrow k_{34}^{-1}[\xi_1 \cdot h(m_3) - \xi_2 \cdot h(r_3)]$. After M_1 receives the ciphertexts, it decrypts them and computes $k_{31}u_1$ and $k_{32}u_3$. Now, M_1 can compute m_3's signature (r_3, s_3), where $k_3 \leftarrow k_{31}k_{32}k_{33}k_{34}$.

C An ℓ out of ℓ Threshold Attack on the Generalized ElGamal Signature

In this section, we introduce an ℓ out of ℓ threshold version of the Young-Yung SETUP mechanism. In this particular case, the proposed scheme is more efficient than the one proposed in Section 5.

C.1 Scheme Description

To implement their attack, the ℓ malicious parties work in almost the same environment as in Section 5. Thus, we only mention the differences between the environments. We denote by $PK_M = \{pk_i\}_{1 \le i \le \ell}$ and present these changes below.

M_1		M_2
m_3, α_1		m_3, α_2, c_{key}
$k_{31}, k_{32} \xleftarrow{\$} \mathbb{Z}_q^*$ $R_1 \leftarrow g^{k_{31}k_{32}}$ Compute DL proof π_1. Compute commit τ to R_1, π_1.		
	$\xrightarrow{\tau}$	
		$k_{33}, k_{34} \xleftarrow{\$} \mathbb{Z}_q^*$ $R_2 \leftarrow g^{k_{33}k_{34}}$ Compute DL proof π_2.
	$\xleftarrow{R_2, \pi_2}$	
Verify proof π_2. If `false` abort.		
	$\xrightarrow{R_1, \pi_1}$	
		Verify commit τ and proof π_1. If `false` abort. $r_3 \leftarrow R_1^{k_{33}k_{34}}$ $\rho_1, \rho_2 \xleftarrow{\$} \mathbb{Z}_{q^2}^*$ $v_1 \leftarrow \alpha_2 r_2 \Delta_2 - s_0^{-1} r_0 \bmod q$ $c_1 \leftarrow Enc(pk_p, (k_{33}v_1 \bmod q) + \rho_1 q)$ $v_2 \leftarrow k_{33} r_1 \Delta_1 \bmod q$ $c_2 \leftarrow v_2 \odot c_{key}$ and $c_3 \leftarrow c_1 \oplus c_2$ $v_3 \leftarrow r_3(\alpha_2 m_2 \Delta_2 - s_0^{-1} m_0) \bmod q$ $v_4 \leftarrow k_{34}^{-1}(m_3 v_1 - v_3) \bmod q$ $c_4 \leftarrow Enc(pk_p, v_4 + \rho_2 q)$ $v_5 \leftarrow k_{34}^{-1}(m_3 r_1 \Delta_1 - r_3 m_1 \Delta_1) \bmod q$ $c_5 \leftarrow v_5 \odot c_{key}$ and $c_6 \leftarrow c_4 \oplus c_5$
	$\xleftarrow{c_3, c_6}$	
$r_3 \leftarrow R_2^{k_{31}k_{32}}$ $u_1 \leftarrow Dec(sk_p, c_3)$ $u_2 \leftarrow k_{31} u_1 \bmod q$ $u_3 \leftarrow Dec(sk_p, c_6)$ $s_3 \leftarrow k_{32}^{-1} u_1^{-1} u_3 \bmod q$ Verify signature (r_3, s_3). If `false` abort, else output (r_3, s_3).		

Fig. 8. Two-Party Malicious Signing.

Signing Sessions: The possible signing sessions performed by D are described below. Let $i \geq 1$.

Session$_0(m_0, sk_V)$: To sign message $m_0 \in \mathbb{G}$, D does the following

$$k_0 \xleftarrow{\$} \mathbb{Z}_q^*, r_0 \leftarrow g^{k_0}, s_0 \leftarrow k_0^{-1}[h(m_0) - a \cdot h(r_0)].$$

The value k_0 is stored in D's volatile memory until the end of *Session*$_1$. Output the signature (r_0, s_0).

Session$_i(m_i, sk_V, PK_M)$: To sign message $m_i \in \mathbb{G}$, D does the following

$$z_i \leftarrow (y_1 \cdot \ldots \cdot y_\ell)^{k_{i-1}}, k_i \leftarrow H(z_i), r_i \leftarrow g^{k_i}, s_i \leftarrow k_i^{-1}[h(m_i) - a \cdot h(r_i)].$$

The value k_i is stored in D's volatile memory until the end of *Session*$_{i+1}$. Output the signature (r_i, s_i).

Recovering$(m_i, r_{i-1}, r_i, s_i, SK_M)$*:* Compute $\alpha_i \leftarrow r_{i-1}^{x_i}$ and $k_i \leftarrow H(\alpha_1 \cdot \ldots \cdot \alpha_\ell)$. Recover a by computing

$$a \leftarrow h(r_i)^{-1}[h(m_i) - k_i \cdot s_i)].$$

Remark 18. Let Q be an honest generator for the values r used by the Generalized ElGamal signature scheme and let σ_i denote the i-th internal state and $\rho_i = g^{\sigma_i}$ the i-th output of Q. The mechanism described above can be seen as a malicious PRNG $\tilde{Q}$ based on the honest PRNG Q. We define the internal states and outputs of $\tilde{Q}$ by

- $\tilde{\sigma}_0 = \sigma_0$, $\tilde{\rho}_0 = \rho_0$;
- $\tilde{\sigma}_i = H(z_i)$, $\tilde{\rho}_i = g^{\tilde{\sigma}_i}$, where $z_i \leftarrow (y_1 \cdot \ldots \cdot y_\ell)^{\tilde{\sigma}_{i-1}}$, $i \geq 1$.

Unlike $\tilde{P}$ from Remark 14, $\tilde{Q}$ can be harmful by itself[19]. A coalition of ℓ malicious parties that know an output $\tilde{\rho}_{i-1}$ can compute the next internal state $\tilde{\sigma}_i$. $\tilde{Q}$ is a threshold variant of the generator described in Remark 5.

C.2 Security Analysis

In this subsection we show that the scheme described above, denoted $\mathcal{F} - \mathcal{GEGS}$, cannot be distinguished from $\mathcal{GEGS}$ if adversary A corrupted at most $\ell - 1$ malicious parties M_i.

Theorem 5. *If the number of signatures is polynomial and* HDH *is hard in* $\mathbb{G}$ *then* $\mathcal{GEGS}$ *and* $\mathcal{F} - \mathcal{GEGS}$ *are* IND-SETUP *in the standard model as long as at most* $\ell - 1$ *malicious parties are corrupted by* A*. Formally, let* A *be an efficient PPT* IND-SETUP *adversary. There exists an efficient algorithm* B *such that*

$$ADV^{\text{IND-SETUP}}_{\mathcal{MEG},\mathcal{GEGS},\mathcal{F}-\mathcal{GEGS}}(A) \leq 4\Gamma ADV^{\text{HDH}}_{\mathbb{G},g,H}(B),$$

where Γ *is the number of infected signatures.*

Proof. Let A be an IND-SETUP adversary that is trying to distinguish between $\mathcal{GEGS}$ and $\mathcal{F} - \mathcal{GEGS}$. A has access to "random coins" sampled uniformly from a set R. Without loss of generality, we further assume that A has corrupted the first $\ell - 1$ malicious participants.

$$\begin{aligned}
& a \xleftarrow{\$} \mathbb{Z}_q^*, y \leftarrow g^a, x_1, \ldots, x_\ell \xleftarrow{\$} \mathbb{Z}_q^*, y_1 \leftarrow g^{x_1}, \ldots, y_\ell \leftarrow g^{x_\ell}, \\
& \mathcal{L}_1 \leftarrow \left(\cup_{i=1}^{\ell-1}\{x_i\}\right) \cup \left(\cup_{i=1}^{\ell}\{y_i\}\right), i \leftarrow 1; \\
C_0(a, m) :\ & k \xleftarrow{\$} \mathbb{Z}_q^*, r \leftarrow g^k, s \leftarrow k^{-1}[h(m) - a \cdot h(r)]; \\
C_1(a, m) :\ & k_0 \xleftarrow{\$} \mathbb{Z}_q^*,\ if\ i = 0, \\
& z_i \leftarrow (y_1 \cdot \ldots \cdot y_\ell)^{k_{i-1}}, k_i \leftarrow H(z_i),\ if\ 1 \leq i \leq \Gamma, \\
& r \leftarrow g^{k_i}, s \leftarrow k_i^{-1}[h(m) - a \cdot h(r)], i \leftarrow i + 1; \\
& b \xleftarrow{\$} \{0, 1\}, \rho \xleftarrow{\$} R, b' \leftarrow A^{C_b(a,\cdot)}(\rho, y, \mathcal{L}_1).
\end{aligned}$$

Fig. 9. The IND-SETUP game.

Figure 9 describes the IND-SETUP game. The first and second rows set up the public keys. Then the $\mathcal{GEGS}$ and $\mathcal{F} - \mathcal{GEGS}$ oracles are described. The challenger then flips a bit b and reveals oracle C_b. A then computes its guess b' for b. A wins if $b = b'$.

[19] *i.e* not only when used with ElGamal based signatures

$$
\begin{aligned}
& a \xleftarrow{\$} \mathbb{Z}_q^*, y \leftarrow g^a, x_1, \ldots, x_\ell \xleftarrow{\$} \mathbb{Z}_q^*, y_1 \leftarrow g^{x_1}, \ldots, y_\ell \leftarrow g^{x_\ell}, \\
& \underline{x_t \xleftarrow{\$} \mathbb{Z}_q^*, y_t \leftarrow g^{x_t}}, \mathcal{L}_1 \leftarrow \left(\cup_{i=1}^{\ell-1}\{x_i\}\right) \cup \left(\cup_{i=1}^{\ell}\{y_i\}\right), i \leftarrow 1; \\
C_0(a,m): & \ k \xleftarrow{\$} \mathbb{Z}_q^*, r \leftarrow g^k, s \leftarrow k^{-1}[h(m) - a \cdot h(r)]; \\
C_1(a,m): & \ k_0 \xleftarrow{\$} \mathbb{Z}_q^*, \ if \ i = 0, \\
& \underline{z_i \leftarrow y_t^{k_{i-1}}}, k_i \leftarrow H(z_i), \ if \ 1 \leq i \leq \Gamma, \\
& r \leftarrow g^{k_i}, s \leftarrow k_i^{-1}[h(m) - a \cdot h(r)], i \leftarrow i + 1; \\
& b \xleftarrow{\$} \{0,1\}, \rho \xleftarrow{\$} R, b' \leftarrow A^{C_b(a,\cdot)}(\rho, y, \mathcal{L}_1).
\end{aligned}
$$

Fig. 10. The new IND-SETUP game.

We proceed by changing the initial IND-SETUP game (described in Figure 9) into a new IND-SETUP game (described in Figure 10). In addition to the original set up, in the new version, we choose an extra secret internal state y_t. Another change is the way we compute the k_i values from oracle C_1. In the original game we multiply the element $y_1 \cdot \ldots \cdot y_{\ell-1}$ from $\mathbb{G}$ with an uniformly random element y_ℓ of the same set and we obtain an uniformly random element. In the new game we directly use a random value y_t for computing the k_i values, thus the change is statically indistinguishable. Since these are the only changes, an adversary will not notice any difference between the IND-SETUP games.

Since $\mathcal{MEG}$ is IND\$ an adversary cannot distinguish between C_0 and C_1. Note that the number of k values that A has to distinguish is n. Thus, we obtain the security margin. □

Remark 19. Similarly to Theorem 5, we obtain that if Q is a secure PRNG, then $\tilde{Q}$ is a secure PRNG in the standard model.

Remark 20. As in the case of Dual-EC, it is easy to see that if in the $\mathcal{F} - \mathcal{GEGS}$ scheme, we replace y_i with $y_i' \xleftarrow{\$} \mathbb{G}$, $1 \leq i \leq n$, the SETUP mechanism becomes benign. The security margin of the SETUP-free system remains the same as the one stated in Theorem 5.