Practical algorithm substitution attack on extractable signatures

Zhao, Yi; Liang, Kaitai; Zhao, Yanqi; Yang, Bo; Ming, Yang; Panaousis, Emmanouil

DOI
10.1007/s10623-022-01019-1

Publication date
2022

Document Version
Final published version

Published in
Designs, Codes, and Cryptography

Designs, Codes and Cryptography (2022) 90:921–937
https://doi.org/10.1007/s10623-022-01019-1

Practical algorithm substitution attack on extractable signatures

Yi Zhao[1] · Kaitai Liang[2] · Yanqi Zhao[3] · Bo Yang[4] · Yang Ming[1] · Emmanouil Panaousis[5]

Received: 5 January 2021 / Revised: 30 January 2022 / Accepted: 2 February 2022 /
Published online: 5 March 2022

Abstract
An algorithm substitution attack (ASA) can undermine the security of cryptographic primitives by subverting the original implementation. An ASA succeeds when it extracts secrets without being detected. To launch an ASA on signature schemes, existing studies often needed to collect signatures with successive indices to extract the signing key. However, collection with successive indices requires uninterrupted surveillance of the communication channel and a low transmission loss rate in practice. This hinders the practical implementation of current ASAs, thus causing users to misbelieve that the threat incurred by ASA is only theoretical and far from reality. In this study, we first classify a group of schemes called extractable signatures that achieve traditional security (unforgeability) by reductions ending with key extraction, thus demonstrating that there is a generic and practical approach for ASA with this class of signatures. Further, we present the implementation of ASAs in which only two signatures and no further requirements are needed for the extraction of widely used discrete log-based signatures such as DSA, Schnorr, and modified ElGamal signature schemes. Our attack presents a realistic threat to current signature applications, which can also be implemented in open and unstable environments such as vehicular ad hoc networks. Finally, we prove that the proposed ASA is undetectable against polynomial time detectors and physical timing analysis.

Keywords Algorithm substitution attack · Extractable signatures · Discrete log · Arbitrary collection

Mathematics Subject Classification 11T71

Communicated by C. Padro.

Supported by the National Key R&D Program of China (Grant No. 2017YFB0802000), the National Natural Science Foundation of China (Grant Nos. 62072054, 61772326, 61802242, 61802241), the Fundamental Research Funds for the Central Universities, CHD (Grant No. 300102240102), European Union's Horizon 2020 research and innovation programme under grant agreement No. 952697 (ASSURED) and No. 101021727 (IRIS), the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2018JQ6088), the National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20180217).

Extended author information available on the last page of the article

1 Introduction

Cryptographic primitives have been widely used in information communication protocols to provide certain types of security properties, such as privacy-preserving authentication and data confidentiality. Some well-studied schemes were standardised by industry associations, such as the digital signature algorithm (DSA) or the elliptic curve digital signature algorithm (ECDSA) in IEEE, NIST, and ANSI. The security of these schemes should be trustworthy because of the theoretical and practical analysis of traditional cryptography. However, Snowden's revelations have indicated that novel attacking techniques evolve daily and might go beyond the scope of traditional cryptography. Emerging attacks can subvert cryptographic schemes to leak critical information. One main type of subversion we address in this study is the algorithm substitution attack (ASA). It allows an attacker to replace implemented algorithms with malicious code without being detected. The subverted algorithm has the same functionality as the original one, and the attacker with the backdoor can obtain access to secret information such as the client's private keys. In practice, cryptographic functions are usually invoked in a black-box manner. If the crypto library was replaced by a malicious update pack in a functionality-preserving manner, users would not notice any difference.

A signature is a fundamental primitive that provides integrity, public verifiability, and non-repudiation. It can be used as a building block for more complicated protocols such as authentication protocols. Some classic signature schemes have been adopted and developed for many industrial standards. Therefore, the effects of ASA on these schemes should be carefully examined. Young and Yung [24, 25] first demonstrated the harm caused by subverting algorithms in a signature via a kleptographic attack. Their attack aimed at establishing a secret channel in the signature to transfer messages rather than attacking the signature itself. The ASA was formalised by Bellare et al. [7], and they launched this attack on symmetric encryption schemes. Ateniese et al. [1] carefully investigated ASAs on signatures and obtained meaningful results. They presented generic stateful ASAs on randomised signatures and indicated that using a deterministic unique signature or cryptographic reverse firewall can resist ASAs. Aiming at the weakness of randomness, Liu et al. [11] discussed asymmetric ASAs on signature schemes in which the attacker's public keys are injected. Beak et al. [2] recently paid close attention to specific schemes such as DSA and proposed an efficient ASA method.

1.1 Motivation

Although existing studies have indicated that randomised signatures are prone to ASAs, and that theoretical attacks are not practical enough to incur actual harm, it is difficult to find devices in reality that are equipped with reverse firewalls [18], which can resist ASAs. Ateniese et al. [1] aimed to provide a generic algorithm to attack any randomised signature scheme; however, the resulting attack does not scale well. They designed a subverted algorithm that reveals one bit of the secret key sk from each signature. To recover the entire secret key, an attacker needs to collect all $||sk||$ signatures with different indices, that is, 1 to $||sk||$, which may not be practical in real-world applications. The asymmetric ASA proposed by Liu et al. [11] requires only two signatures with successive indices in an attempt to improve the efficiency of collection, but it incurs heavy computation costs for public key operations. In contrast, a symmetric ASA by Beak et al. [2] needs to collect three signatures with successive indices, which still holds "collection with successive indices" as a requirement.

In summary, one reason why ASAs are not practical is the signature collection with successive indices. To accomplish this, an attacker must continuously tap the communication of the signer. An ASA is often triggered by a virus or trojan in practice, and the attacker cannot estimate when the attack will begin. In mobile applications such as VANETs, users join and leave frequently, thus making it difficult for attackers to tap all the channels. In other words, if the attacker can tap all the communication channels of a device, the device would be physically under control. The other reason for the poor performance of ASAs is that they are stateful, and the literature does not explore what else can be provided by maintaining the *state* rather than the *label*.

To address this problem, various techniques may be explored to recover secret keys. Our study is inspired by theoretical work, namely, the forking lemma [19, 20], which can extract the secret key of the signer in discrete log-based signature schemes to reduce the security proof. In the forking lemma, the challenger rewinds the attacker and changes the random oracle answer to form a fork. If forgeries occur twice at the fork point, the challenger can extract the secret key by solving two linear equations. In our case, the attacker cannot perform any rewinding and the hash function is public rather than a random oracle so as to avoid obtaining two signatures with the same randomness and different hash values for the same message. However, we can still set parameters to form solvable equations to extract secret keys, which is helpful to realise the collection of signatures without index requirements for extraction in our proposed scheme.

1.2 Our contribution

In this section, we describe our results by comparing traditional security notions of signatures and ASA approaches with our concrete implementation of ASAs on discrete log-based signatures.

1.2.1 Proof technique vs. ASAs

In the past decade, provable security has been accepted as a security guarantee, rather than cryptanalysis. To prove the security of a cryptographic scheme, we often make a reduction that if an adversary can break the scheme, we can construct an algorithm that invokes the adversary as an inner process to solve a well-established hard problem such as a discrete log or RSA. With respect to signatures, the random oracle methodology [4] is widely employed because schemes proven to be secure in the standard model are often too complicated and inefficient. To make proofs in random oracle models more convincing, many studies [6, 14, 15, 26] have attempted to find proper instantiations of random oracles to ensure that the scheme is secure in the random oracle model (ROM), thus also being secure in the standard model. According to these results, signatures may be the most suitable primitive for ROMs. However, not all well-known signature schemes can be proven to be secure directly in ROMs. Bellare et al. [5] proposed that RSA and Rabin signature schemes are provably secure under ROM. However, those based on the discrete log problem do not have a direct solution. To solve this problem, the forking lemma [19, 20] was introduced to complete the reduction from the extraction of the signing key by rewinding the adversary with different random oracles.

Although the above techniques have performed well in proving traditional security in ROMs for a long time, the situation is different when the adversary is allowed to subvert algorithms. Extraction techniques such as forking lemma can work in an ideal environment

and have no impact on the real world when only traditional security is considered. However, in the ASA cases, the existence of an extraction algorithm implies that an adversary can subvert the signing algorithm to output signatures in extractable form. However, the indistinguishability of simulation in the proof of traditional security guarantees that the subversion is undetectable. Therefore, we can observe a contradiction in that the technique leading to traditional provable security contributes to ASAs.

1.2.2 Extractable signatures

In contrast to previous work on starting ASAs only through randomness, we follow another approach; that is, based on the above observations, we attack the weaknesses inherited by the primitive in the provable security structure. This is a key technique to avoid collection with successive indices. To achieve the ultimate goal, it first needs to capture the notion of signature schemes that are prone to ASAs in this manner. We present the definition of extractable signatures to include those schemes with an extraction algorithm in the security proof. We amplified the definition to highlight the different performances in the extraction process.

The notion of extractability (as well as a similar notion of simulation extractability) comes from the proof of knowledge protocol [3, 13, 21], where there exists an extractor such that if the prover can give proof of a statement, the secret witness can be extracted by the extractor with some internal state of the prover. In an ideal environment, the extracted secret can usually be used to simulate indistinguishable games with the adversary to complete the security reduction. To date, many extractable primitives have been proposed to solve different problems, including extractable functions [10], extractable one-way functions [8], extractable hash functions [16], extractable hash proof systems [23], and extractable commitment schemes [12]. However, our definition differs because extractors can only work in a simulation, and we need to find a subversion algorithm in practice. Therefore, we extend our definition to incorporate those schemes with modified extraction algorithms that can still keep the output signature indistinguishable from normal ones. The first challenge is how to accurately describe the scope of schemes. Not all provably secure signature schemes are extractable. Our definition excludes schemes such as the RSA signature, which reduces the security to find an inversion of a random element in the range of a one-way function keyed by a secret, but does not extract the secret that generates the one-way function. We also need to provide a precise description to show the "degree" of extractability of different schemes. The second challenge is to allow the conversion from extractors in simulation to an extractable algorithm in practice. For example, the forking lemma can be regarded as an extraction algorithm for discrete log-based signature schemes in simulation. However, the technique of rewinding the adversary and changing the answer to a random oracle is not applicable in realistic scenarios. Therefore, we need to carefully design the subverted algorithm to achieve the same result as the forking lemma without handling the random oracle and rewinding. This implies that the definition needs to contain more elements to play the role of extraction approaches.

1.2.3 Concrete ASA

In this work, we focus on ASA approaches that are based on extraction algorithms, which may help to find more efficient and practical ASA methods compared to generic methods. The concrete implementation of ASAs depends on different schemes, and the highlight of our ASA is that our attack does not need to collect signatures with successive indices; eavesdropping is also not necessary. The results are as follows:

Table 1 Performance comparison

Scheme	Collection mode on indices	Number of signatures	Scope
[1]	Successive indices	q	All randomised signatures
[2]	Successive indices	3	DSA type
[11]	Successive indices	2	Splittable signatures
Ours	Arbitrary indices	2	Extractable signatures

- Our subverted schemes only require two signatures regardless of their indices to recover the signing key. This implies that the attacker can collect signatures at any time from a public source without having to continue tapping the signer all the time. This is much more practical than existing works. We present our subverted schemes for DSA [17], modified ELGamal [19, 20], and Schnorr [22] signature schemes to demonstrate the generality of our method.
- The basic version of the subverted scheme only maintains a constant state and has a restriction that the message cannot be repeated. To incorporate duplicate messages, our scheme must maintain an internally changing state. We present a dynamic mechanism to maintain a state that can leverage the extraction computational efficiency and hardness for signature collection. This state not only represents the internal order or position, but also plays a more functional role in the scheme. The performance comparison between existing schemes and ours is shown in Table 1.

1.3 Organisation

The remainder of this paper is organised as follows. In Sect. 2, we introduce the necessary preliminaries. Then, in Sect. 3, we introduce the notion of the simulation extractable signature, and show how existing schemes are incorporated in this frame. In Sect. 4, we present a concrete implementation of ASA on DSA with a restriction that does not allow repetitive messages and the proof. In Sect. 5, we show how to remove the restriction and leverage the extraction efficiency and the requirement of signature collection. In Sect. 6, the performance and timing analysis are evaluated. In Sect. 7, we show that our ASA method can be applied to many other schemes. The last section includes conclusions and future expectations.

2 Preliminaries

In this section, we introduce the definition of regular and subverted signature schemes with a refined formal description of the attack model.

2.1 Signature and its subversion

Definition 1 A regular signature scheme is a triple of algorithms $\mathcal{S} = (Gen, Sig, Ver)$. Gen is a key generation algorithm that takes a security parameter λ as input, and outputs a key pair (pk, sk). Sig is the signing algorithm that generates signature σ with signing key sk and message m. A randomised signing algorithm also takes a randomness r as input. Ver is a publicly computable algorithm that checks whether signature σ is valid or not.

Table 2 $Game^{PubDetect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$

$\mathcal{C}$		$\mathcal{A}$
$Gen(1^n) = (pk, sk)$		
$\overline{Gen}(1^n) = subk$		
	$pk \longrightarrow$	
	$\longleftarrow m_i$	
$b \leftarrow_R \{0, 1\}$		
$\sigma = Sig(m_i)$ if $b = 0$		
$\sigma = \overline{Sig}(m_i)$ if $b = 1$		
	$\sigma \longrightarrow$	
		Output b'

Definition 2 A subverted signature scheme for $\mathcal{S}$ is a tuple of algorithms $\overline{\mathcal{S}} = (\overline{Gen}, \overline{Sig}, \overline{Ver})$. $\overline{Gen}$ is a key generation algorithm that generates a subversion key $subk$. This key joins the signing process with a signing key. $\overline{Sig}(m, subk, sk) = (\sigma, state)$ is a subverted signing algorithm that generates a subverted signature $\overline{\sigma}$ with a subversion key $subk$, signing key sk, and message m. If $state = \emptyset$, then $\overline{Sig}$ is called stateless. $\overline{Ver}$ is the same as the original verification algorithm. The original verification algorithm is deterministic; therefore, the attacker cannot subvert it with undetectability. Thus, usually in ASAs, the subverted verification algorithms remain unchanged. This work does not consider the subverting verification algorithm.

2.2 Security notions of ASA

Let $\mathcal{A}$ be an ordinary user and $\mathcal{C}$ be a challenger in an ASA game. Given $\mathcal{S}$ and $\overline{\mathcal{S}}$, a basic security model of the ASA on a signature is the public undetectability, which is formally described by the game below (Table 2).

During the game, $\mathcal{A}$ can reboot the algorithm at any time. If $b' = b$, we say that $\mathcal{A}$ wins the game. We define $adv^{detect}_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$ as the advantage that $\mathcal{A}$ wins the game.

Definition 3 A subverted signature scheme $\overline{\mathcal{S}}$ is publicly undetectable if all probabilistic polynomial time (PPT) users $\mathcal{A}$ can win in $Game^{PubDetect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$ with only a negligible advantage.

Another stronger notion is secret undetectability. To define secret undetectability (Table 3), the adversary can access sk, which is not allowed in the public undetectability game. This means that even a user who generates valid signatures cannot distinguish subverted signatures from normal signatures.

Definition 4 A subverted signature scheme $\overline{\mathcal{S}}$ is secretly undetectable if all PPT users $\mathcal{A}$ can win in $Game^{SecDetect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$ with only a negligible advantage.

Table 3 $Game^{SecDetect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$

$\mathcal{C}$		$\mathcal{A}$
$Gen(1^n) = (pk, sk)$		
$\overline{Gen}(1^n) = subk$		
	(pk, sk) $\longrightarrow$	
	m_i $\longleftarrow$	
$b \leftarrow_R \{0, 1\}$		
$\sigma = Sig(m_i)$ if $b = 0$		
$\sigma = \overline{Sig}(m_i)$ if $b = 1$		
	σ $\longrightarrow$	
		Output b'

2.3 Pseudorandom functions

Definition 5 Let f be a random function and F be an efficient, length-preserving, and keyed function $\{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$. For any polynomial time distinguisher D, we define the advantage

$$adv_D = |\Pr[D^{F_k(\cdot)} = 1] - \Pr[D^{f(\cdot)} = 1]|,$$

where k is uniformly chosen. We say that F is a pseudorandom function if adv_D is negligible for any PPT D.

3 Extractable signature

In this section, we present our definition of extractable signatures, as well as measures to distinguish different levels of extractability. Similar to the notion of a splittable signature in [11] to capture the characteristics of schemes suitable for their attack, this class of signature schemes can cover the scope of schemes prone to our attack.

Definition 6 A signature scheme $\mathcal{S} = (Gen, Sig, Ver)$ is q-extractable if there exists an extractor that can compute the signing key from q signatures forged by the adversary in the simulation. Formally, for all polynomial time adversaries $\mathcal{A}$, there exists a polynomial time extractor E such that $(pk, sk) \leftarrow Gen(1^n)$, $\Pr[E^{\mathcal{A}}(pk, \{m_i\}_{i=1,\ldots,q}, \{\sigma_i\}_{i=1,\ldots,q}, a) = sk] = 1$, where a denotes some possible auxiliary inputs.

Discussion We evaluate some signature schemes to determine which types of schemes are covered by our definition. First, schemes proven secure by the forking lemma belong to this category. The schemes in [19, 20] are 2-extractable. The proof of knowledge schemes, which can be regarded as an extension of signature [13], is 1-extractable. The Rabin scheme in [5] is 1-extractable, whereas the RSA scheme in the same literature is not extractable because the secret key cannot be extracted in the reduction. For the same reason, identity-based encryption-induced signatures [9] are also not extractable. Our goal is to find more efficient ASA methods for these extractable signatures. Therefore, we need a measure to indicate the effects of ASAs.

Definition 7 Let q be the minimum number of signatures required for the key extraction. Let succ/arb denote the mode in which the signatures are collected with a successive/arbitrary order of indices. A subverted signature scheme $\overline{\mathcal{S}}$ realises (succ/arb, q)-extraction if from q queried subverted signatures collected in the succ/arb mode, the signing key can be extracted.

Discussion According to the above definition, the generic ASA on randomised signature schemes in [1] realises (succ, q)-extraction. Liu et al.'s scheme in [11] realises (succ, 2)-extraction, whereas Beak et al.'s scheme [2] achieves (succ, 3)-extraction. We believe that a q-extractable signature scheme has at least one ASA approach to implement (arb, q)-extraction. For a non-interactive proof of knowledge protocol, the extractor can be transformed directly into a subverted prover with trapdoor CRS to realise 1-extraction. However, for ordinary signature schemes, the random oracle methodology for extraction is directly infeasible. Thus, we need to find an alternative approach to achieve the same extraction results according to the structure of the schemes. We leave the question of finding a generic transformation as an open problem. In the following sections, we present our results for different schemes.

4 Our ASA scheme on DSA

We first present a basic version of the (arb, 2)-extractable ASA scheme on DSA with an additional restriction that there are no duplicate messages. Because the same messages lead to the same signature from the signing algorithm in this scheme, which contradicts the fact that DSA is randomised, it is detectable. We then show how to remove this restriction. However, it is still worth noting that even the basic version has practical value. This is because in the actual implementation, the final message to be signed is usually formed by a message concatenated with a timestamp, which is usually used to stop replay attacks in many protocols. Therefore, resigning the same message is unlikely to occur in practice. By exploiting this additional assumption, the proposed ASA can realise signature collection with arbitrary indices.

4.1 Standard DSA scheme

Before proposing our subverted scheme, we first describe the original DSA scheme $(Gen_{DSA}, Sig_{DSA}, Ver_{DSA})$ for comparison. $H : \{0,1\}^* \rightarrow \mathcal{Z}_q$ is a collision-resistant hash function.

- Gen_{DSA}: Randomly choose a 1024-bit prime p, and a 160-bit prime q such that $q|p-1$. Choose a generator g by computing $g = h^{(p-1)/q} \bmod p$ from a random h until $g \neq 1$. Randomly select an integer x such that $1 \leq x \leq q-1$ is the signing key. Then, we compute $y = g^x \bmod p$ as the verification key. Let H denote the SHA-1 function for future extensions because other schemes may not use SHA-1.
- Sig_{DSA}: Given a message m, select a randomness $1 \leq k \leq q-1$ and compute $R = g^k \bmod p$ and $r = R \bmod q$. The signature is computed as $s = k^{-1}(e + xr) \bmod q$, where $e = H(m)$. The output is (r, s).
- Ver_{DSA}: Given a signature (r, s) of a message m, compute $e = H(m)$ and $X = g^{es^{-1}} y^{rs^{-1}} \bmod p \bmod q$. The signature is valid if $r = X$.

4.2 Subverted DSA scheme

We now replace some parts of the algorithms above to form a subverted scheme $(\overline{Gen}_{DSA}, \overline{Sig}_{DSA}, \overline{Ver}_{DSA})$.

- $\overline{Gen}_{DSA}$: Select a PRF $F : \{0,1\}^l \times \{0,1\}^\rho \to \{0,1\}^n$, and select κ uniformly as an evaluation key. Let $subk = (F, \kappa)$.
- $\overline{Sig}_{DSA}$: A state τ is set to be null initially. Given the first message m_1, signing key x, and subversion key $subk$, compute $t_1 = F(\kappa, m_1)$. Then, we select a randomness $1 \le k \le q-1$ and compute $R_1 = g^{t_1 k} \bmod p$ and $r_1 = R_1 \bmod q$. The signature is computed as $s_1 = (t_1 k)^{-1}(e + xr_1) \bmod q$, where $e = H(m)$. The signature for m_1 is (r_1, s_1). The state τ is set to k.
  To sign a latter message m_i, compute $t_i = F(\kappa, m_i)$ and $R_i = g^{t_i k} \bmod p$. Let $r_i = R_i \bmod q$. The signature is computed as $s_i = (t_i k)^{-1}(e + xr_i) \bmod q$. The signature for m_i is (r_i, s_i). The state is still k.
- $\overline{Ver}_{DSA}$: It is the same as Ver_{DSA}.

Remark It can be observed that our scheme maintains a constant state since it is chosen, which is the key for achieving collection without the requirement of indices. Existing works with a small number of signatures, such as [2, 11], require collection with successive indices because the state to generate the next signature depends on the previous one. Thus, missing the former will make it difficult for the adversary to compute the state for key extraction. The state in our algorithm is irrelevant to order; therefore, any two signatures can work.

4.3 Key extraction

The process of extracting the key from any two subverted signatures is analogous to the process of the forking lemma. The details are as follows.

- Given any two message-signature pairs $(m_i, (r_i, s_i))$ and $(m_j, (r_j, s_j))$ with $m_i \neq m_j$, the attacker can extract the signing key by establishing equations from the verification process.

$$r_i = g^{e_i s_i^{-1}} g^{r_i s_i^{-1}} \bmod p \bmod q$$
$$r_j = g^{e_j s_j^{-1}} g^{r_j s_j^{-1}} \bmod p \bmod q$$

- Substitute r_i, r_j on the left side, and y with $g^{t_i k} \bmod p \bmod q$ and $g^{t_j k} \bmod p \bmod q$ and g^x, to obtain equations in the exponentiation:

$$t_i k = e_i s_i^{-1} + x r_i s_i^{-1} \bmod q$$
$$t_j k = e_j s_j^{-1} + x r_j s_j^{-1} \bmod q$$

- Multiply both sides of the equations by s to obtain

$$t_i k s_i = e_i + x r_i \bmod q$$
$$t_j k s_j = e_j + x r_j \bmod q$$

  This is a linear equation with two unknowns, x and k. The attacker can obtain

$$x = \left(t_i s_i e_j - t_j s_j e_i\right) / \left(t_j s_j r_i - t_i s_i r_j\right)$$

  by solving equations with a significant probability.

4.4 Undetectability analysis

Theorem 1 *Let* $(Gen_{DSA}, Sig_{DSA}, Ver_{DSA})$ *be a standard DSA scheme, and let* $(\overline{Gen}_{DSA}, \overline{Sig}_{DSA}, \overline{Ver}_{DSA})$ *be a subverted DSA scheme, as described above.* $\mathcal{A}$ *is a detector in the game defined in Definition 4.* F *is an efficient, length-preserving, and keyed function* $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. *Then,* $(\overline{Gen}_{DSA}, \overline{Sig}_{DSA}, \overline{Ver}_{DSA})$ *is undetectable, assuming the pseudorandomness of function* F *and no duplicate messages for signing.*

Proof The proof proceeds as a sequence of variants of $Game^{Detect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$ in Definition 4. Assume that $\mathcal{A}$ issues q signature queries, and after the $j-1$th query, the signing algorithm is rebooted (Without the loss of generality, we assume there is only a one-time reboot. More reboots are handled in the same manner as in the one-time approach.). Then, we start with $Game_0$.
$Game_0$: This game is the same as the game described in Definition 4 using Sig_{DSA} to answer signature queries.
$Game_1$: In this game, we modify the first answer to the signature queries. Given m_1 as the first query, we use $\overline{Sig}_{DSA}$ to generate a signature (r_1, s_1). Other queries are still answered with Sig_{DSA}.
$Game_i$ $(2 \le i \le j-1)$: In this game, the first i answers are generated using $\overline{Sig}_{DSA}$. Other queries are answered using Sig_{DSA}.
$Game_i$ $(j \le i \le q)$: In this game, the first i answers are generated using $\overline{Sig}_{DSA}$. Among them, the jth query is answered with $\overline{Sig}_{DSA}$ by resetting the state to be empty. Other queries are answered using Sig_{DSA}. □

Lemma 1 *The views of the adversary in* $Game_0$ *and* $Game_1$ *have the exact same distribution.*

Proof In $Game1$, the first answer is generated with $\overline{Sig}_{DSA}$. In $\overline{Sig}_{DSA}$, the first signature is generated in the same manner as Sig_{DSA} by selecting a new randomness with the only difference that $\overline{Sig}_{DSA}$ multiplies a constant t_1 before exponential computation. Therefore, the output of $\overline{Sig}_{DSA}$ has the same distribution as Sig_{DSA}. □

Lemma 2 *The views of the adversary in* $Game_{i-1}$ *and* $Game_i$, *where* $(2 \le i \le j-1)$ *and* $(j+1 \le i \le q)$, *are indistinguishable if* F *is a PRF.*

Proof The only difference between $Game_{i-1}$ and $Game_i$ is the algorithm used to generate the ith answer. In $Game_i$, the ith answer is computed via $\overline{Sig}_{DSA}$, in which F is used to compute t_i. Then, $t_i \cdot state$ is used as the role of randomness in Sig_{DSA}. Note that $state$ is constant before the reboot. Thus, F determines whether the views of the adversary in both games are indistinguishable. The formal reduction proof is given below.

Given an adversary $\mathcal{A}$ that can detect whether ASA occurs with advantage ϵ, we can construct an algorithm $\mathcal{C}$ that invokes $\mathcal{A}$ as an internal process, which can determine whether F is a PRF with the following advantages:

1. $\mathcal{C}$ invokes $\mathcal{A}$ by running Gen to obtain (pk, sk), and sends pk to $\mathcal{A}$. The PRF challenger $\mathcal{O}(F, f)$ generates samples of PRF F with a secret key κ. This implicitly sets $subk = (F, \kappa)$ in $\overline{Gen}_{DSA}$. Thus, $\mathcal{C}$ does not need to run $\overline{Gen}_{DSA}$.
2. To answer the lth query, $(1 \le l \le i-1)$, $\mathcal{C}$ transfers the queries m_l to its PRF challenger as PRF queries to obtain $t_l = F(\kappa, m_l)$ to compute signatures by $\overline{Sig}_{DSA}$.

To answer the lth query, $l = i$, $\mathcal{C}$ submits m_i to its PRF challenger as a challenge to obtain a response t_i. This value is then used to compute the signature $(r_i = g^{t_i k} \bmod p \bmod q, s_i = (t_i k)^{-1}(e + x r_i))$.

To answer the lth query, $(i < l \leq q)$, $\mathcal{C}$ computes signatures via Sig_{DSA}.
We can see that if $t_i = F(\kappa, m_i)$, the game is $Game_{i-1}$. If t_i is a random number, then the signature has exactly the same distribution as $Gamei$. Thus, we can conclude that if an adversary $\mathcal{A}$ has an advantage ϵ in detecting ASA, $\mathcal{C}$ also has an advantage ϵ in judging whether F is a PRF. If t_i is uniformly distributed, then $\mathcal{A}$ should have no advantage. □

Lemma 3 *The views of the adversary in $Game_{j-1}$ and $Game_j$ have the exact same distribution.*

Proof The reason these two games are distributed statistically close is exactly the same as Lemma1. In $Game_j$, the signing algorithm is rebooted, and the randomness is fresh. This is similar to the situation in Lemma1. We can thus draw the same conclusion.

We can see that $Game_q$ is the game in which all the queries are answered with PRF, which is simply the situation of the subverted algorithm. To summarise, if there exists an adversary who can distinguish between a PRF and a truly random function with advantage ϵ, one can then construct an algorithm that can distinguish $Game_0$ and $Game_q$ with advantage $(q-2)\epsilon$. This completes the reduction. □

5 Removing the restriction

The ASA above works only if there are no duplicate messages. Although this is the case in most applications, the restriction is still not satisfied theoretically. We present a self-rebooting mechanism to remove the restriction and show that there is a trade-off between the signature collection hardness and extraction efficiency. This trade-off does not affect the undetectability, which is inherited from the basic scheme.

5.1 Self-rebooting subverted DSA scheme

- $\overline{Gen}_{sDSA}$: It is the same as $\overline{Gen}_{DSA}$.
- $\overline{Sig}_{sDSA}$: A state $(\tau, count)$ is set to be $(\perp, 1)$ initially. Let u be the upper bound of $count$. Given the first message m_1, signing key x, and subversion key $subk$, compute $t_1 = F(\kappa, m_1, count)$. Then, we select a randomness $1 \leq k \leq q-1$ and compute $R_1 = g^{t_1 k} \bmod p$ and $r_1 = R_1 \bmod q$. The signature is computed as $s_1 = (t_1 k)^{-1}(e + x r_1) \bmod q$, where $e = H(m)$. The signature for m_1 is (r_1, s_1). The state $(\tau, count)$ is set to $(k, count + 1)$.
  To sign a latter message m_i, compute $t_i = F(\kappa, m_i, i)$ and $R_i = g^{t_i k} \bmod p$. Let $r_i = R_i \bmod q$. The signature is computed as $s_i = (t_i k)^{-1}(e + x r_i) \bmod q$. The signature for m_i is (r_i, s_i). The state is $(k, i + 1)$.
  When $count = u$, the signing algorithm is rebooted. The state is reset to $(\perp, 1)$.

5.2 Trade-off

Note that in our key extraction algorithm, t_i must be recovered by the attacker without any interactions. When there is a message that is public, recovering t_i is not challenging. In the above subverted signing algorithm, we add a counter inside without any information sent outside. Thus, the attacker must guess the value of $counter$. Since two signatures are needed for extraction, the attacker has to guess the right pair (t_i, t_j). We assume that the signatures

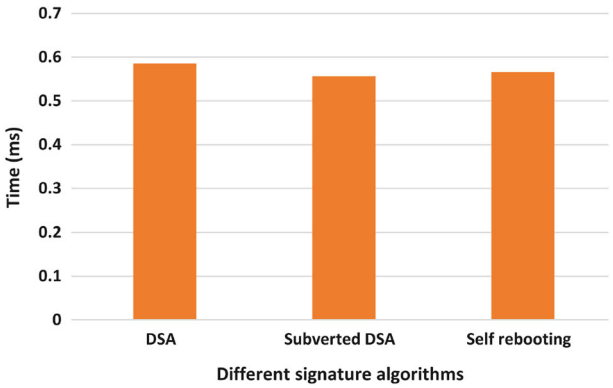

Fig. 1 Time costs of signatures for the DSA, subverted DSA, and self-rebooting subverted DSA schemes

are collected during one interval between two reboots. There are $u(u-1)/2$ guesses, which require $u(u-1)/4$ guesses, on average, to extract the correct signing key.

One may consider that a smaller u will reduce the time required for extraction. However, a smaller u will make signature collection more difficult because two signatures must be in the same interval between reboots. When u becomes smaller, reboots occur more frequently such that fewer signatures are chosen for extraction. Thus, it is necessary to balance the collection hardness and extraction efficiency.

6 Efficiency and timing analysis

We implemented the DSA, subverted DSA, and self-rebooting subverted DSA schemes using the Python language, version 3.6.5.[1] The experimental platform was based on an Intel(R) Core(TM) i5-2450M, with a 2.50 GHz CPU and 6.00 GB RAM, running the Ubuntu 16.0.4 LTS OS. We tested the signature algorithm of the DSA, subverted DSA, and self-rebooting subverted DSA schemes with a key size of 2048. The values $(p, q) = (2048, 224)$ were selected with parameters $p =$ 27272474141258804355794771020398686760904542776884408055751421063425499324115317389377588869721785543268554040536654333591334167098822719495433762844087189353577839213597421942324010193430652206865991451797613097817972044436049649264944395444393861781267469581946012406298930938387235152547666893929226811977701024606115749458782928541816966732495803017356118819256252165835401044029978248973648374570500927264752101005937942639149810661469917144588400076218433079814575479386210269251958537294758142512203389086545772145157253536385138165862694987022540889202619217204558281421355067853244378649146179732898423507595 1,
$q =$ 1495479779689622116344929534191025936417837799294008966244474761437 9

We ran the signature algorithm 10,000 times and captured the average running time. The time costs of signatures for the DSA, subverted DSA, and self-rebooting subverted DSA schemes are shown in Fig. 1, where our subverted DSA scheme is shown to be the most effective.

We also evaluated the efficiency of the key extraction. With the same message, we can also extract the key by self-rebooting operations. We ran the key extraction algorithm for

[1] https://www.python.org/downloads/release/python-365/.

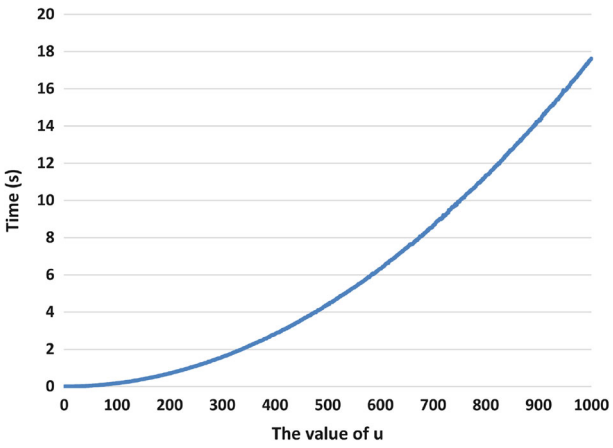

Fig. 2 Time cost for key extraction

10,000 rounds and captured the average running time. The time cost of the key extraction is shown in Fig. 2. We chose different counter values from $u = 0$ to $u = 1000$ to test the time cost for key extraction. The continuous valid u values show the trend of the key extraction time. From Fig. 2, we can see that the experimental results are consistent with the theoretical analysis.

6.1 Immune to timing analysis

It should be noted that in Fig. 1, our subverted scheme has a noticeable difference from the normal one, which may lead to detection through timing analysis. The time difference arises from the absence of randomness generation in the most subverted signing process. However, we can add some dummy computations to eliminate the difference because the subversion algorithm is the lower one. For the dummy computation, we call the PRF function instead of idle time. We tested the PRF function, which costs $0.003172421455383301ms$ for an average running time of 10,000 rounds. Therefore, the subverted DSA scheme runs approximately 10 times more than the dummy computation to replace the free time, while the self-rebooting subverted DSA scheme runs approximately 6 times more than the dummy computation. Thus, our scheme becomes undetectable against timing analysis.

7 Our ASA on more schemes

Our ASA can be implemented on most discrete log-based signature schemes, such as the forking lemma. We chose two more classic signature schemes to demonstrate the generality of our technique.

7.1 Subverted Schnorr signature scheme

7.1.1 Review of the original Schnorr scheme

- Gen_{SCH}: Generate a cyclic group $\mathcal{G}$ with a large prime order q. Let g be the generator and $H : \{0,1\}^* \to \mathcal{Z}_q$ be a collision-resistant hash function. Randomly choose an integer $x \in \mathcal{Z}_q$ and compute $y = g^x$. The verification key is y, and the signing key is x.
- Sig_{SCH}: Given a message m, select an integer $k \in \mathcal{Z}_q$ uniformly and compute $r = g^k$. Let $e = H(m, r)$. Compute $s = k + xe \bmod q$, and the signature is (r, s).
- Ver_{SCH}: Given a signature (r, s) of a message m, compute $e = H(m, r)$ and check whether $e = H(g^s y^{-e})$.

7.1.2 Our subverted Schnorr scheme

- $\overline{Gen}_{SCH}$: Select a PRF $F : \{0,1\}^l \times \{0,1\}^\rho \to \{0,1\}^n$, and select κ uniformly as an evaluation key. Let $subk = (F, \kappa)$.
- $\overline{Sig}_{SCH}$: A state τ is set to be empty initially. Given the first message m_1, signing key x, and subversion key $subk$, compute $t_1 = F(\kappa, m_1)$. Then, we select a randomness $1 \le k \le q-1$ and compute $r_1 = g^{t_1 k}$ and $r_1 = R_1 \bmod q$. The signature is computed as $s_1 = k + xe \bmod q$, where $e = H(m_1, r_1)$. The signature for m_1 is (r_1, s_1). The state τ is set to k.
  To sign a latter message m_i, compute $t_i = F(\kappa, m_i)$ and $r_i = g^{t_i k}$. The signature is computed as $s_i = (t_i k)^{-1}(e + xr_i) \bmod q$, where $e = H(m_i, r_i)$. The signature for m_i is (r_i, s_i). The state is still k.
- $\overline{Ver}_{SCH}$: It is the same as Ver_{SCH}.

7.2 Subverted modified ElGamal signature scheme

7.2.1 Review of the original modified ElGamal scheme

- Gen_{ElG}: Generate a cyclic group $\mathcal{G}$ with a large prime order q. Let g be the generator and $H : \{0,1\}^* \to \mathcal{Z}_q$ be a collision-resistant hash function. Randomly choose an integer $x \in \mathcal{Z}_q$ and compute $y = g^x$. The verification key is y, and the signing key is x.
- Sig_{ElG}: Given a message m, select an integer $k \in \mathcal{Z}_q$ uniformly and compute $r = g^k$. Let $e = H(m, r)$. Solve the linear equation $e = xr + ks \bmod q - 1$ to obtain s. The signature is (r, s).
- Ver_{ElG}: Given a signature (r, s) of a message m, compute $e = H(m, r)$ and check whether $g^e = y^r r^s$.

7.2.2 Our subverted ElGamal scheme

- $\overline{Gen}_{ElG}$: Select a PRF $F : \{0,1\}^l \times \{0,1\}^\rho \to \{0,1\}^n$, and select κ uniformly as an evaluation key. Let $subk = (F, \kappa)$.
- $\overline{Sig}_{ElG}$: A state τ is set to be empty initially. Given the first message m_1, signing key x, and subversion key $subk$, compute $t_1 = F(\kappa, m_1)$. Then, select a randomness $1 \le k \le q-1$ and compute $r_1 = g^{t_1 k}$. s_1 is computed by solving $e_1 = xr_1 + ks_1 \bmod q - 1$, where $e = H(m_1, r_1)$. The signature for m_1 is (r_1, s_1). The state τ is set to k.

– $\overline{Ver}_{ElG}$: It is the same as Ver_{ElG}.
To sign a latter message m_i, compute $t_i = F(\kappa, m_i)$ and $r_i = g^{t_i k} \bmod q$. s_i is computed by solving $e_i = xr_i + ks_i \bmod q - 1$. The signature for m_i is (r_i, s_i). The state is still k.

8 Conclusion

Our work considers how to subvert extractable signature schemes more effectively than existing approaches. Efficient attacks on widely deployed signature schemes, such as DSA, may warn people not to ignore the security threat incurred by ASAs and to arm their devices with reverse firewalls. We aim to find generic and more efficient ASA methods for these unextractable schemes in the future. Further, using ASAs for arbitrary forgeries of signatures instead of for signing key extraction, for instance, might bring about more subversion approaches.

References

1. Ateniese G., Magri B., Venturi D.: Subversion-resilient signatures: Definitions, constructions and applications. Theor. Comput. Sci. **820**, 91–122 (2020).
2. Baek J., Susilo W., Kim J., Chow Y.W.: Subversion in practice: How to efficiently undermine signatures. IEEE Access. **7**, 68799–68811 (2019).
3. Bellare M., Goldreich O.: On defining proofs of knowledge. In: Brickell E.F. (ed.) Advances in Cryptology—CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16–20, 1992, Proceedings. Lecture Notes in Computer Science vol. 740, pp. 390–420. Springer (1992)
4. Bellare M., Rogaway P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning D.E., Pyle R., Ganesan R., Sandhu R.S., Ashby V, (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM (1993)
5. Bellare M., Rogaway P.: The exact security of digital signatures—How to sign with RSA and rabin. In Maurer UM, (ed.) Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12–16, 1996, Proceeding. Lecture Notes in Computer Science, vol. 1070, pp. 399–416. Springer (1996)
6. Bellare M., Hoang V.T., Keelveedhi S.: Instantiating random oracles via uces. In: Canetti R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2013. Proceedings, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 398–415. Springer (2013)
7. Bellare M., Paterson K. G., Rogaway P.: Security of symmetric encryption against mass surveillance. In: International Cryptology Conference (2014)
8. Bitansky N., Canetti R., Paneth O., Rosen A.: On the existence of extractable one-way functions. In: Shmoys D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31–June 03, 2014. pp. 505–514. ACM (2014)
9. Boneh D., Lynn B., Shacham H.: Short signatures from the weil pairing. J. Cryptol. **17**(4), 297–319 (2004).
10. Canetti R., Dakdouk R.R.: Towards a theory of extractable functions. In: Reingold O, (ed.) Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15–17, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5444, pp. 595–613. Springer (2009)
11. Chi L., Chen R., Yi W., Wan, Y.: Asymmetric subversion attacks on signature schemes. (2018)
12. Crescenzo G.D.: Equivocable and extractable commitment schemes. In: Cimato S, Galdi C, Persiano G, (eds.) Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11–13, 2002. Revised Papers. Lecture Notes in Computer Science, vol. 2576, pp. 74–87. Springer (2002)
13. Groth J.: On the size of pairing-based non-interactive arguments. In: Fischlin M., Coron J.-S. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory

and Applications of Cryptographic Techniques, Vienna, Austria, May 8–12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666 , pp. 305–326. Springer (2016)
14. Hofheinz D., Kiltz E.: Programmable hash functions and their applications. J. Cryptol. **25**, 484–527 (2011).
15. Hohenberger S., Sahai A., Waters B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 201–220. Springer (2014)
16. Kiayias A., Liu F.-H., Tselekounis Y.: Practical non-malleable codes from l-more extractable hash functions. In: Weippl E.R., Katzenbeisser S., Kruegel C., Myers A.C., Halevi S, (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24–28, 2016, pp. 1317–1328. ACM (2016)
17. Kravitz D.W.: Digital signature algorithm.
18. Mironov I., Stephens-Davidowitz N.: Cryptographic reverse firewalls. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques (2015)
19. Pointcheval D., Stern J.: Security proofs for signature schemes. Advances in Cryptology - EUROCRYPT '96. Springer, Berlin (1996)
20. Pointcheval D., Stern J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000).
21. Rackoff C., Simon D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum J, (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11–15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 433–444. Springer (1991)
22. Schnorr C.-P.: Efficient identification and signatures for smart cards. In: Brassard G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
23. Wee H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15–19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6223, pp. 314–332. Springer (2010)
24. Young A., Yung M.: The dark side of "black-box" cryptography or: Should we trust capstone? In: International Cryptology Conference on Advances in Cryptology (1996)
25. Young A.L., Yung M.: The prevalence of kleptographic attacks on discret-log based cryptosystems. In: Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 1997, Proceedings. (1997)
26. Zhandry M.: The magic of elfs. In: Robshaw M., Katz J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 479–508. Springer (2016)

Authors and Affiliations

Yi Zhao[1] · Kaitai Liang[2] · Yanqi Zhao[3] · Bo Yang[4] · Yang Ming[1] · Emmanouil Panaousis[5]

✉ Yanqi Zhao
zhaoyanqi2021@xupt.edu.cn

Yi Zhao
yizhao@chd.edu.cn

Kaitai Liang
Kaitai.Liang@tudelft.nl

Bo Yang
boyang@snnu.edu.cn

Yang Ming
yangming@chd.edu.cn

Emmanouil Panaousis
E.Panaousis@greenwich.ac.uk

1 School of Information Engineering, Chang'an University, Xi'an 710064, China

2 Faculty of Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, Delft, The Netherlands

3 School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

4 School of Computer Science, Shaanxi Normal University, Xi'an 710062, China

5 School of Computing and Mathematical Sciences, University of Greenwich, London, UK