Practical algorithm substitution attack on extractable signatures*

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

[1] School of Information Engineering, Chang'an University, Xi'an, 710064, China
[2] Faculty of Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, 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, UK

Abstract. Algorithm Substitution Attack (ASA) can undermine the security of cryptographic primitives by subverting the original implementation. ASA succeeds when it extracts secrets without being detected. To launch an ASA on signature schemes, existing research works often have to collect signatures with successive indices to extract the signing key. However, collection with successive indices requires uninterrupted surveillance of the communication channel and low loss rate of transmission in practice. This hinders the current ASAs from being practically implemented, and making users misbelieve the threat incurred by ASA is only at the theoretical level and much far from reality. In this paper, we first classify a group of schemes called extractable signatures that achieve traditional security (unforgeability) by reductions ending with key extraction, showing that there is a generic and practical approach for ASA to this class of signatures. Then, we present the implementations of ASAs that only two signatures and no further requirements are needed for extraction of widely used discrete log based signatures like DSA, Schnorr and Modified ElGamal signature schemes. Our attack presents a realistic threat to current signature applications, which can even be implemented in open and unstable environment like a vehicular ad hoc network. Finally, we prove the proposed ASA is undetectable against polynomial time detectors and physical timing analysis.

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

* Supported by the National Key R&D Program of China (2017YFB0802000), the National Natural Science Foundation of China (62072054,61772326,61802242, 61802241), the Fundamental Research Funds for the Central Universities, CHD(300102240102), the Natural Science Basic Research Plan in Shaanxi Province of China (2018JQ6088), the National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20180217).

1 Introduction

Cryptographic primitives have been widely used in information communication protocols to provide certain type of security properties, like privacy preserving authentication, and data confidentiality. Some well-studied schemes were standardized by industry associations such as DSA (Digital Signature Algorithm) or ECDSA (Elliptic Curve Digital Signature Algorithm) in IEEE, NIST and ANSI. The security of these schemes should be trustworthy due to the analysis of traditional cryptography theoretically and practically. However, Snowden's revelations have indicated that novel attacking techniques evolve day by day and might go beyond the scope of traditional cryptography. The emerging attacks can subvert the cryptographic schemes to leak critical information. One main type of subversions we deal with in this paper is called Algorithm Substitution Attack (ASA), which allows attacker to inject malicious code to replace the implemented algorithms without being detected. The subverted algorithm has the same functionality as the original one and the attacker with the backdoor can get access to secret information, e.g., client's private keys. In practice, cryptographic functions are usually invoked in black-box manner. If the crypto library was replaced by a malicious update pack in a functionality preserving way, users couldn't notice the differences.

Signature is a fundamental primitive which provides integrity, public verifiability and non-repudiation. It can be used as building block for more complicated protocols such as authentication protocols. Some classic signature schemes are adopted and developed in many industrial standards. Therefore the ASA effects on these schemes should be carefully examined. The first harm by subverting algorithms in signature was taken by Young and Yung [24, 25] via the kleptographic attack. Their attack aims to establish a secret channel in the signature to transfer messages rather than attacking signature itself. ASA was formalized by Bellare et al. [5] and they launched this attack on symmetric encryption schemes. Ateniese et al. [1] carefully investigated ASA on signatures and obtained some meaningful results. They presented generic stateful ASAs on randomized signatures and pointed out that using deterministic unique signature or cryptographic reverse firewall can resist ASAs. Aiming at the weakness of the 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 have paid close attention to some specific schemes like DSA and proposed an efficient ASA method.

1.1 Motivation

Although the existing works indicated that randomized signatures were prone to ASAs, and the theoretical attacks are not practical enough to incur actual harm, so it is difficult to find any device in reality equipped with reverse firewall [18] which can resist ASAs. Ateniese et al. [1] aimed to give a generic algorithm to attack any randomized signature schemes, but the resulting attack still does not scale well. They designed a subverted algorithm that reveals one bit of secret key sk from each signature. In order to recover the whole secret key, an attacker

needs to collect all $||sk||$ signatures with different indices 1 to $||sk||$, which may not be practical in real-world application. The asymmetric ASA proposed by Liu et al. [11] requires only 2 signatures with consecutive indices, trying to improve the efficiency of collection, but it yields heavy computation cost from public key operations. On the other hand, symmetric ASA by Beak et al. [2] needs to collect 3 signatures with successive indices which still put "collection with successive indices" as a requirement.

In summary, one reason that ASAs are not practical is mainly due to the signatures collection with successive indices. To do this, an attacker needs to tap the communication of the signer continuously. ASA is often triggered by virus or trojan in practice, and meanwhile, the attacker couldn't estimate when the attack will begin. In mobile applications like VANET, users come in and out frequently, thus making attackers hard to tap all the channels. In other words, if the attacker could 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 *state* rather than *label*.

To address the above problem, one may try to explore various techniques to recover secret keys. Our work is inspired by a theoretical work, the forking lemma [19, 20], which can extract the secret key of the signer in discrete log based signature schemes to finish reduction in the security proof. In forking lemma, the challenger rewinds the attacker and changes the random oracle answer to form a fork. If forgeries happened twice in the fork point, the challenger could extract the secret key by solving two linear equations. In our case, the attacker can't do any rewinding and hash function is public rather than random oracle so that we can't obtain two signatures with the same randomness and different hash values for the same message. But we can still set parameters to form solvable equations to extract secret keys, which is helpful for us to realize collection of signatures without index requirements for extraction in our subverted scheme.

1.2 Our Contribution

In this section, we describe our results with an observation on relationship between traditional security notions of signatures and ASA approaches to our concrete implementation of ASAs on discrete log based signatures.

Proof technique vs ASAs. In the past decade, provable security has been well 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 could break the scheme, we can construct an algorithm that invokes the adversary as inner process to solve a well established hard problem like discrete log or RSA. In the area of signature, random oracle methodology [6] is widely employed because schemes proven to be secure in standard model are often too complicated and inefficient. To make proofs in random oracle model more convincible, many works [4, 14, 15, 26] have tried to find proper instantiations of random oracle to keep the scheme secure in the random oracle model (ROM) still secure in standard model. According to these results, signature may be the most suitable

primitive for random oracle model so far. However, not all well known signature schemes can be proven secure directly in ROM. Bellare et al. [7] proposed that RSA and Rabin signature schemes provably secure under ROM. However, the ones based on discrete log problem do not have a direct solution. To solve this problem, forking lemma [19, 20] was introduced to complete the reduction from extraction of signing key by rewinding the adversary with different random oracles.

Although the techniques above have done extraordinarily to prove traditional security in ROM for a long time, the situation is different when the adversary is allowed to subvert algorithms. Extraction techniques like forking lemma can work in an ideal environments and have no impact on real world when only traditional security is considered. However, in the ASA cases, the existence of an extraction algorithm means an adversary can subvert the signing algorithm to output signatures with extractable form. Yet the indistinguishability of simulation in proof of traditional security guarantees that the subversion is undetectable. Therefore, we can see a contradiction that the technique leading to traditional provable security does contribute to ASAs.

Extractable signatures. Different from the previous work of starting ASA 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 the key technique to avoid collection with successive indices. In order to achieve the ultimate goal, it first needs to capture the notion of signature schemes which are prone to ASA in this way. We present the definition of extractable signatures to include those schemes with extraction algorithm in security proof. And we amplify 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, 21, 13], where exists an extractor that if the prover can give proof of a statement, the secret witness can be extracted by the extractor with some internal state of prover. In an ideal environment, usually the extracted secret can be used to simulate indistinguishable games with the adversary so as to complete the security reduction. So far, 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 does not follow exactly the same way because extractors can only work in simulation, but 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 here is how to describe the scope of schemes accurately. Not all provably secure signature scheme are extractable. Our definition excludes schemes like RSA signature, which reduces the security to find an inversion of random element in the range of one way function keyed by a secret, but does not extracting the secret generating the one way function. We also need to give precise description to show

the "degree" of extractability of different schemes. The second challenge is to allow conversion from extractors in simulation to extractable algorithm in practice. For example, 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 random oracle is not applicable in realistic scenarios. Therefore, we have to carefully design the subverted algorithm to achieve the same result with the forking lemma without handling the random oracle and rewinding. This means that the definition needs to contain more elements to play the role of extraction approaches.

Concrete ASA. In this work, we focus on those ASA approaches based on extraction algorithms which may help to find more efficient and practical ASA methods than the generic one. The concrete implementation of ASAs depend on different schemes respectively.The highlight of our ASA is that our attack doesn't need to collect signatures with successive indices, and even eavesdropping is not necessary either. The results are as follows:

- Our subverted schemes only require 2 signatures regardless of their indices to recover the signing key. This means that the attacker can collect signatures at any time from public source without having to keep tapping the signer all the time. This is much more practical than existing works. We will give our subverted schemes for DSA [17], Modified ELGamal [19, 20] and Schnorr [22] signature schemes to show the generality of our method.
- The basic version of the subverted scheme only maintains a constant state and has a restriction that the message can not repeat. To incorporate duplicate messages, our scheme needs to maintain an internal changing state. We present a dynamic mechanism to maintain a state which can leverage the extraction computational efficiency and the hardness for signature collection. This state can not only represent the internal order or position but play a more functional role in the scheme.

Table 1. Performance Comparison

scheme	collection mode on indices	number of signatures	Scope
[1]	successive indices	q	all randomized signatures
[2]	successive indices	3	DSA type
[11]	successive indices	2	splittable signatures
Ours	arbitrary indices	2	extractable signatures

1.3 Organization

The rest of this paper is organized as follows: In section 2, we will introduce the preliminaries needed. Then we introduce the notion of simulation extractable signature in section 3, and show how existing schemes are incorporated in this

frame. In section 4 we present a concrete implementation of ASA on DSA with the restriction that does not allow repetitive messages and proof. In section 5 we show how to remove the restriction and leverage between extraction efficiency and requirement of signature collection. in section 6 The evaluation of performance and timing analysis are shown. In section 7, we show that our ASA method can be applied to many other schemes. The last section includes the conclusion and future expectation.

2 Preliminaries

In this section, we will introduce the definition of regular and subverted signature schemes with refined formal description of 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 randomized signing algorithm also takes a randomness* r *as input.* Ver *is a publicly computable algorithm that checks whether the signature* σ *is valid or not.*

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 will join 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 subversion key* $subk$, *signing key* sk *and message* m. *If* $state = \emptyset$, $\overline{Sig}$ *is called stateless.* $\overline{Ver}$ *is the same as the original verification algorithm. The original verification algorithm is deterministic so that the attacker is hard to subvert it with undectectability. Thus usually in ASAs subverted verification algorithms just remain unchanged. This work does not consider subverting verification algorithm either.*

2.2 Security Notions of ASA

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

During the game, $\mathcal{A}$ can reboot the algorithm at any time. If $b' = b$, we say $\mathcal{A}$ wins the game. We define $adv_{\mathcal{A}}^{detect} = |\Pr[b' = b] - 1/2|$ 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) user* $\mathcal{A}$ *can win in Game*$_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}^{PubDetect}$ *with only negligible advantage.*

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

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

Another stronger notion is secret undetectability. In order to define secret undetectability, the adversary can get access to sk which is not allowed in public undetectability game. That means even a user who generates valid signatures can not tell subverted signatures apart form normal ones.

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 $b = 1$		
	$\sigma \longrightarrow$	
		Output b'

Definition 4. *A subverted signature scheme $\overline{\mathcal{S}}$ is secretly undetectable if all probabilistic polynomial time (PPT) user $\mathcal{A}$ can win in $Game^{SecDetect}_{\mathcal{A},\mathcal{S},\overline{\mathcal{S}}}$ with only negligible advantage.*

2.3 Pseudorandom functions

Definition 5. *Let f be a random function and F be an efficient, length-preserving and keyed function $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. 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 for any PPT D that adv_D is negligible.

3 Extractable signature

In this section we present our definition of extractable signatures as well as measures to distinguish different level of extractability. Just like the notion of 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 simulation. Formally, for all polynomial time adversary $\mathcal{A}$, there exists a polynomial time extractor E 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 can check some signature schemes to see which kind of schemes are covered by our definition. First, schemes proven secure by 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 [7] is 1-extractable but the RSA scheme in the same literature is not extractable because the secret key cannot be extracted in the reduction. Due to the same reason, those identity based encryption induced signatures [9] are also not extractable. Our goal is to find more efficient ASA methods on these extractable signatures. So we need a measure to show the effect of ASAs.

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

Discussion. According to the definition above, the generic ASA on randomized signature schemes in [1] realizes (succ, q)-extraction. Liu et al.'s scheme in [11] realizes $(\text{succ}, 2)$-extraction while Beak et al.'s scheme [2] realizes $(\text{succ}, 3)$-extractable. 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 to subverted prover with trapdoored CRS to realize 1-extraction. However, for ordinary signature schemes, the random oracle methodology for extraction are infeasible directly. Thus, we need to find an alternative approach to achieve the same extraction result according to the structure of schemes respectively. We leave the question to find a generic transformation as an open problem. In the following sections, we will show our results for different schemes.

4 Our ASA scheme on DSA

We first present a basic version of $(\text{arb}, 2)$-extractable ASA scheme on DSA with an additional restriction that there are no duplicate messages. Because same messages lead to the same signature from the signing algorithm in this scheme, which contradicts the fact that DSA is randomized, so it is detectable. We will then show how to remove this restriction. But it is still worth saying that even the basic version still has practical value, 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 not likely to happen in practice. By exploiting this additional assumption, our ASA can realize signature collection with arbitrary indices.

4.1 Standard DSA Scheme

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

- Gen_{DSA}: Randomly choose an 1024 bit prime p and a 160 bit prime q that $q|p-1$. Choose a generator g by computing $g = h^{(p-1)/q} \bmod p$ from random h until $g \neq 1$. Randomly select an integer x that $1 \leq x \leq q-1$ as the signing key. Then compute $y = g^x \bmod p$ as the verification key. Let H denote SHA-1 function for future extension 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

Now we replace some parts of 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 pick κ uniformly as 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 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 $s_1 = (t_1 k)^{-1}(e + xr_1) \bmod q$ where $e = H(m)$. The signature for m_1 is (r_1, s_1). The state τ is set to be 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 is noticeable that our scheme maintains a constant state since it is chosen, which is the key for achieving collection without requirement on indices. Existing works with small number of signatures like [11] and [2] require collection with successive indices because the state to generate next signature depends on the previous one. Thus missing the former will make for the adversary hard to compute state for key extraction. The state in our algorithm is irrelevant to order, so that 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 in 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 in 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, 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 s to both sides of equations, and 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. Attacker can obtain

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

  by solving equations with overwhelming probability.

4.4 Undetectability Analysis

Theorem 1. *Let* $(Gen_{DSA}, Sig_{DSA}, Ver_{DSA})$ *be a standard DSA scheme, and* $(\overline{Gen}_{DSA}, \overline{Sig}_{DSA}, \overline{Ver}_{DSA})$ *a subverted DSA scheme as 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 the 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 $\mathcal{A}$ issues q signature queries, and after $j-1$th query, the signing algorithm is rebooted (Without the loss of generality, we assume there is just one time reboot. More reboots are handled just the same as the one time approach.). Then we start with $Game_0$.
$Game_0$: This game is just the same as the game description in definition 4 using Sig_{DSA} to answer signature queries.
$Game_1$: In this game, we modify the first answer to signature queries. Given m_1 as the first query, using $\overline{Sig}_{DSA}$ 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 with 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 with Sig_{DSA}.

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

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

Lemma 2. *The view of 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 to generate 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 reboot. So F determines whether the views of adversary in both games are indistinguishable. The formal reduction proof is given below.

Given an adversary $\mathcal{A}$ that can detect whether ASA happens with advantage ϵ, one can construct an algorithm $\mathcal{C}$ which invokes $\mathcal{A}$ as internal process can determine whether F is a PRF with related advantage as follows:
1. $\mathcal{C}$ invokes $\mathcal{A}$ by running Gen to obtain (pk, sk) and sends pk to $\mathcal{A}$. The PRF

challenger $\mathcal{O}(F, f)$ runs generate samples a PRF F with secret key κ. This implicitly sets $subk = (F, \kappa)$ in $\overline{Gen}_{DSA}$. Thus $\mathcal{C}$ doesn't need to run $\overline{Gen}_{DSA}$.
2. To answer the lth query that $(1 \leq l \leq i-1)$, $\mathcal{C}$ transfers the queries m_l to its PRF challenger as PRF queries to get $t_l = F(\kappa, m_l)$ to compute signatures by $\overline{Sig}_{DSA}$.
To answer lth query that $l = i$, $\mathcal{C}$ submits m_i to its PRF challenger as a challenge to get a response t_i. Then use this value to compute the signature $(r_i = g^{t_i k} \mathrm{mod} p \mathrm{mod} q, s_i = (t_i k)^{-1}(e + x r_i))$.
To answer lth query that $(i < l \leq q)$, $\mathcal{C}$ 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 with $Gamei$. So we can conclude that if an adversary $\mathcal{A}$ has advantage ϵ in detecting ASA, $\mathcal{C}$ has also advantage ϵ in judging whether F is a PRF. If t_i is uniformly distributed, $\mathcal{A}$ should have no advantage at all.

Lemma 3. *The view of the adversary in $Game_{j-1}$ and $Game_j$ have the perfectly 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. That is similar to the situation in Lemma1. And We can have the same conclusion.

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

5 Remove the Restriction

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

5.1 Self rebooting Subverted DSA Scheme

- $\overline{Gen}_{sDSA}$: 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 select a randomness $1 \leq k \leq q-1$ and compute $R_1 = g^{t_1 k} \mathrm{mod}\ p$ and $r_1 = R_1 \mathrm{mod}\ q$. The signature is computed as $s_1 = (t_1 k)^{-1}(e + x r_1) \mathrm{mod}\ q$ where $e = H(m)$. The signature for m_1 is (r_1, s_1).

The state $(\tau, count)$ is set to be $(k, count + 1)$.
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 $s_i = (t_i k)^{-1}(e + xr_i) \bmod q$. The signature for m_i is (r_i, s_i). The state is $(k, i + 1)$.
When $count = u$, the signing algorithm reboots. The state is reset to be $(\perp, 1)$.

5.2 A Trade-off

Note that in our key extraction algorithm, t_i is asked to be recovered by the attacker without any interactions. When there is just the message which is public, recovering t_i is not challenging. In above subverted signing algorithm, we add a counter inside without any information sent outside. Thus, the attacker needs to guess the value of *counter*. Because 2 signatures are needed for extraction, the attacker has to guess the right pair (t_i, t_j). We assume the signatures are collected during one interval between two reboots. There are $u(u-1)/2$ guesses which requires $u(u-1)/4$ times guessing on average to extract the right signing key.

One may think that smaller u will reduce the time cost for extraction. But smaller u will make signature collection harder because two signatures have to be in the same interval between reboots. When u is getting smaller, reboot happens more frequently such that the signatures to be chosen for extraction are fewer. Thus, we need to balance the collection hardness and extraction efficiency.

6 Efficiency and Timing analysis

In this section, we implement the DSA, subverted DSA and self rebooting subverted DSA schemes by the python language with python 3.6.5 version [6]. The experiment platform is based on Intel(R) Core(TM) i5-2450M CPU 2.50 GHz 6.00GB RAM Ubuntu 16.0.4 LTS OS. We test the signature algorithm of DSA, subverted DSA and self rebooting subverted DSA schemes with the key size of 2048 and select the values $(p, q) = (2048, 224)$ and the parameters $p =$
27272474141258804355794771020398686769045427768844080557514210634254 9324115317389377588869721785543268554040536654333591334167098822719495433762844087189353577839213597421942324010193430652206865991451797613097817972044436049649264944395444393861781267469581946012406298930938387235152547666893929226811977701024606115749458782928541816966732495803017356118819256252165835401044029978248973648374570500927264752101005937942639149810661469917144588400076218433079814575479386210269251958537294758142512203389086545772145157253536385138165862694987022540889202619217204558281421355067853244378649146179732898423507595 1, $q =$
1495479779689622116344929534191025936417837799294008966244474761437 9

We run the signature algorithm 10,000 times and capture an average running time. The time cost of signature for DSA, subverted DSA and self rebooting

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

subverted DSA schemes is shown in Figure 1, where our subverted DSA scheme is most effective.

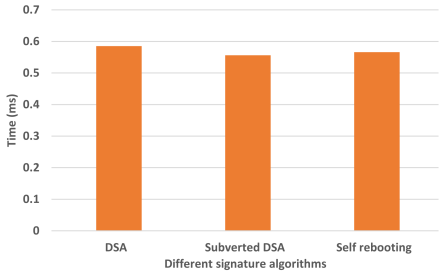

Fig. 1. The time cost of signature for DSA, subverted DSA and self rebooting subverted DSA schemes

We also evaluate the efficiency of key extraction. With the same massage, we also can extract the key by the self rebooting operations. We run the key extraction algorithm 10,000 rounds and capture an average running time. The time cost of key extraction, as shown in Figure 2. We choose different counter value from $u = 0$ to $u = 1000$ to test the time cost for key extraction. The continuous u values valid show the trend of key extraction time. From the Figure 2, we can find the experimental results are consistent with the theoretical analysis.

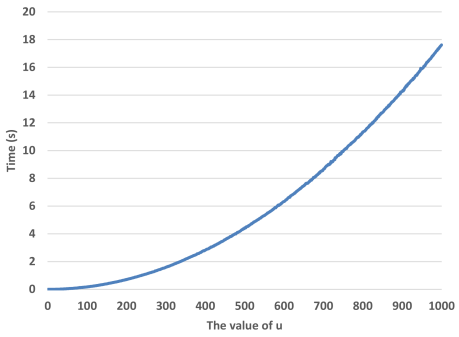

Fig. 2. The time cost for key extraction

6.1 Immune to Timing analysis

One may notice that in Figure 1 our subverted scheme have noticeable difference with normal one, which may lead to detection through timing analysis. The time

difference comes from no randomness generation in 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 the idle time. We test the PRF function which costs $0.003172421455383301ms$ for the average running time with 10,000 rounds. So, a subverted DSA scheme runs about 10 times of dummy computation to replace the free time while the self rebooting subverted DSA scheme runs about 6 times dummy computation. Thus our scheme will become undetectable against timing analysis.

7 Our ASA on more Schemes

Our ASA can be implemented on most discrete log based signature schemes like forking lemma. We choose two more classic signature schemes to show the generality of our technique.

7.1 Subverted Schnorr signature scheme

Review Original Schnorr Scheme.

- Gen_{SCH}: Generate a cyclic group $\mathcal{G}$ with big 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 $s = k + xe \bmod q$ The signature is (r, s).
- Ver_{SCH}: Given a signature (r, s) of a message m, compute $e = H(m, r)$ and check whether $e = H(g^s y^{-e})$.

Our subverted Schnorr Scheme.

- $\overline{Gen}_{SCH}$: Select a PRF $F : \{0,1\}^l \times \{0,1\}^\rho \to \{0,1\}^n$ and pick κ uniformly as 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 select a randomness $1 \leq k \leq q-1$ 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 be 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

Review Original Modified ElGamal Scheme.

- Gen_{ElG}: Generate a cyclic group $\mathcal{G}$ with big 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 $g^e = y^r r^s$.

Our subverted ElGamal Scheme.

- $\overline{Gen}_{ElG}$: Select a PRF $F : \{0,1\}^l \times \{0,1\}^\rho \to \{0,1\}^n$ and pick κ uniformly as 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 $1 \leq k \leq q - 1$ 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 be 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 the ASAs and arm their devices with reverse firewalls. We aim to find generic and more efficient ASA methods for those un-extractable schemes in the future. In addition, loosing the goal for ASAs as signing key extraction to arbitrary forgeries of signature for instance, might bring to more subversion approaches.

References

1. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signatures: Definitions, constructions and applications. *Theor. Comput. Sci.*, 820:91–122, 2020.
2. Joonsang Baek, Willy Susilo, Jongkil Kim, and Yang-Wai Chow. Subversion in practice: How to efficiently undermine signatures. *IEEE Access*, PP(99):1–1, 2019.

3. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.
4. Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via uces. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 398–415. Springer, 2013.
5. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In *International Cryptology Conference*, 2014.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.
7. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.
8. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 505–514. ACM, 2014.
9. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004.
10. Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 595–613. Springer, 2009.
11. Liu Chi, Rongmao Chen, Wang Yi, and Yongjun Wang. Asymmetric subversion attacks on signature schemes. 2018.
12. Giovanni Di Crescenzo. Equivocable and extractable commitment schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2002.
13. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *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*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.
14. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, 2011.
15. Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 201–220. Springer, 2014.

16. Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1317–1328. ACM, 2016.
17. Kravitz, David, and William. Digital signature algorithm.
18. Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015.
19. David Pointcheval and Jacques Stern. *Security Proofs for Signature Schemes.* Advances in Cryptology EUROCRYPT 96. Springer Berlin Heidelberg., 1996.
20. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
21. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.
22. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.
23. Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 314–332. Springer, 2010.
24. Adam Young and Moti Yung. The dark side of "black-box" cryptography or: Should we trust capstone? In *International Cryptology Conference on Advances in Cryptology*, 1996.
25. Adam L. Young and Moti Yung. The prevalence of kleptographic attacks on discrete-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. Mark Zhandry. The magic of elfs. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 479–508. Springer, 2016.