On the Security of the Schnorr Signature Scheme and DSA against Related-Key Attacks [⋆]

Hiraku Morita[1,2], Jacob C.N. Schuldt[2], Takahiro Matsuda[2], Goichiro Hanaoka[2], and Tetsu Iwata[1]

[1] Nagoya University, Nagoya, Japan
h_morita@echo.nuee.nagoya-u.ac.jp, iwata@cse.nagoya-u.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{jacob.schuldt,t-matsuda,hanaoka-goichiro}@aist.go.jp

Abstract. In the ordinary security model for signature schemes, we consider an adversary that may forge a signature on a new message using only his knowledge of other valid message and signature pairs. To take into account side channel attacks such as tampering or fault-injection attacks, Bellare and Kohno (Eurocrypt 2003) formalized related-key attacks (RKA), where stronger adversaries are considered. In RKA for signature schemes, the adversary can also manipulate the signing key and obtain signatures for the modified key. This paper considers RKA security of two established signature schemes: the Schnorr signature scheme and (a well-known variant of) DSA. First, we show that these signature schemes are secure against a weak notion of RKA. Second, we demonstrate that, on the other hand, neither the Schnorr signature scheme nor DSA achieves the standard notion of RKA security, by showing concrete attacks on these. Lastly, we show that a slight modification of both the Schnorr signature scheme and (the considered variant of) DSA yields fully RKA secure schemes.

Keywords: Related-key attacks, Schnorr signatures, DSA.

1 Introduction

1.1 Background

A signature scheme is a cryptographic public key primitive which guarantees validity of messages. Up until now, many schemes have been proposed such as the ElGamal signature scheme [18], the Schnorr signature scheme [31], and DSA [1]. The commonly accepted security notion for a signature scheme is existential unforgeability against chosen message attacks, which guarantees that even if an adversary can obtain signatures on arbitrary messages of its choice, the adversary cannot forge a valid signature on a new message. The Schnorr signature scheme, and two variants of DSA were proven to satisfy this notion in the random oracle model [27, 29], under the discrete logarithm (DL) assumption.

Related-key attacks (RKA), stronger attacks, were formalized by Bellare and Kohno [5]. RKA security captures security against practical attacks such as tampering or fault injection, which enable adversaries to alter a hardware-stored secret key and observe the output of the algorithm using the modified key. Thus, RKA security captures practical attacks which might cause security issues in practice. Therefore, it is an important question whether primitives are secure against RKA attacks even if they are already shown to be secure against ordinary attacks.

RKA for signature schemes allows an adversary to obtain not only valid message and signature pairs, but also signatures under a modified key. RKA security is defined with respect to the related-key deriving (RKD) functions with which an adversary is allowed to modify the secret key. For example, we consider linear functions, affine functions, and polynomial functions. Since RKA considers a broader class of attacks than ordinary attacks, security against RKA is much stronger than ordinary security.

However, only a few generic constructions for achieving RKA secure signatures have been proposed. Bellare, Cash, and Miller [4] studied relations between RKA secure primitives, and in particular showed that an RKA secure pseudorandom function (PRF) can be used to convert a signature scheme secure against ordinary attacks, into a scheme providing RKA security. The conversion is relatively simple:

[⋆] A preliminary version of this paper appears in the proceedings of the 18th Annual International Conference on Information Security and Cryptology, ICISC 2015 [26]. This is the full version.

before generating the verification and signing key, apply the PRF to the randomness used by the key generation algorithm, and then store the randomness instead of the generated signing key. Now, since the signing key of the original scheme is no longer stored, this has to be re-generated whenever a message is signed. This is done by applying the PRF to the stored randomness, and then re-running the key generation algorithm. Bellare, Cash, and Miller [4] showed that, via this conversion, it is possible to lift the RKA security of the PRF to the signature scheme. Used in combination with the recently proposed RKA secure PRF by Abdalla et al. [2], which is shown to be secure under the q-Diffie Hellman Inversion assumption, this allows the conversion of any (ordinary) signature scheme to a scheme which is RKA secure with respect to polynomial functions.

Goyal et al. [22] showed a similar conversion for achieving RKA secure signatures, but based on a correlated-input secure (CIS) hash function. Furthermore, Goyal et al. constructed a very efficient CIS hash function secure under the q-Diffie Hellman Inversion assumption, which would lead to signatures that are RKA secure with respect to polynomials. However, this construction only achieves selective security; a weak and non-adaptive security notion that requires the adversary to submit the RKD functions before seeing the verification key of the signature scheme.

Building upon the work on non-malleable key derivation functions (nm-KDFs) [17], Qin et al. [30] introduced the notion of continuous nm-KDFs, and used these in a similar conversion to the above to construct an RKA secure signature scheme with respect to polynomial functions under standard assumptions. The proposed construction of an nm-KDF can furthermore be extended to provide security with respect to any RKD function class that has the properties the authors denote "high output entropy" and "input-output collision resistant". Interestingly, the transformation into RKA-secure primitives shown in [13] can be understood as applying an nm-KDF [17, 30] to the secret key.

Since a signature scheme is an essential cryptographic primitive, clarifying the RKA security of various constructions is of interest from both a practical and a theoretical point of view. Specifically, studying the RKA security of well-known signatures such as the Schnorr signature scheme and DSA is important due to their widespread use, in particular in the case of DSA, which is employed in many practical implementations. However, besides the negative result by Bao et al. [3], who showed that the Schnorr signature scheme and DSA are not RKA secure against bit flipping attack, it is not known whether either scheme can provide any form of RKA security. Furthermore, simply applying the above transformations might not always be desirable due to the relatively high performance penalties these conversions imply.

1.2 Our Contributions

In this paper, we first show that both the Schnorr signature scheme and a DSA variant are secure against a weak notion of RKA (wRKA) that does not allow messages queried to the RKA signing oracle to be a part of a forgery. Second, we show that the Schnorr signature scheme and the original DSA are vulnerable to the standard notion of simple linear RKA. We then construct (standard) RKA secure signature schemes based on the Schnorr signature scheme and DSA. Specifically, as our main technical results, we show the following four results:

- The Schnorr signature scheme is secure against wRKA with respect to polynomial functions.
- A well-known variant of DSA by [29] is secure against wRKA with respect to polynomial functions.
- Slightly modifying the signing and verification algorithms of the Schnorr signature scheme yields an RKA secure scheme with respect to polynomial functions.
- Slightly modifying the signing and verification algorithms of DSA yields an RKA secure scheme with respect to polynomial functions.

In other words, the Schnorr signature scheme, which is secure against wRKA with respect to polynomial functions, but not RKA secure even for weak attacks with respect to linear functions, can achieve full RKA security with respect to polynomial functions by slightly modifying the scheme. While DSA is not RKA secure with respect to linear functions, the DSA variant from [29] is secure against wRKA, and by slightly modifying this scheme, full RKA security with respect to polynomial functions can be achieved. Both the improved Schnorr signature scheme and the improved DSA variant are proven to be RKA secure with respect to polynomial functions in the random oracle model, under the d-strong discrete logarithm (d-SDL) assumption. As a corollary, the improved signature schemes are RKA secure with respect to affine functions under the standard discrete logarithm (DL) assumption, since the 1-SDL assumption is equivalent to the DL assumption, and polynomials of degree 1 are affine functions.

Note that our modifications of the Schnorr signature scheme and DSA only increase the computational cost of signing with a single exponentiation, while the computational cost of verification, signature size, and key sizes remain unchanged. Hence, in contrast to using a conversion based on continuous nm-KDF [27, 29] or RKA secure PRFs [4, 2], our modifications maintain the efficiency of the Schnorr signature scheme and DSA. Furthermore, unlike all of the above mentioned conversions for achieving RKA security, our modifications of the Schnorr signature scheme and DSA do not require the verification and signing key to change. This is a virtue for schemes which are already deployed, such as DSA, since key management and verification key certificates remain unchanged. Lastly, we would like to emphasize that in our proofs of security for our improved Schnorr signature scheme and the improved DSA, we do not restrict the number of RKA signing oracle queries or rely on a "self-destruct" mechanism [16, 17] which prevents the adversary from making any further queries once it is detected that the signing key has been tampered with.

1.3 Related Work

Gennaro et al. [19] show how to recover the key of almost any cryptographic primitive assuming the adversary can tamper arbitrarily with the key of the primitive. This implies that RKA security cannot be achieved for every set of RKD functions. On the other hand, Damgård et al. [12, 13] showed that in a security model which restricts the number of RKA queries that an adversary is allowed to make, it is possible to achieve security for arbitrary RKD functions. In contrast to this model, which is denoted the bounded leakage and tampering model, we will in this paper consider unrestricted adversaries which are allowed to make an arbitrary number of RKA signing oracle queries. Since Dziembowski, Pietrzak, and Wichs introduced non-malleable codes [15], they have been studied and found to have a good application in the construction of RKA secure cryptosystems. While non-malleable codes in themselves are not sufficient to provide full RKA security, continuous non-malleable codes, which were initiated in [16], enables this. However, the security of the constructions presented in [16] relies on a self-destruct mechanism that will prevent an attacker from interacting with the system once it has been detected that the internal state of the systems is being tampered with. In contrast, the continuous nm-KDF proposed by Qin et al. [30] does not require a self-destruct mechanism, and can be used to construct RKA secure public key primitives for a large class of RKD functions. Jafargholi and Wichs [23] defined two factors which yield four levels of security of continuous nm-KDF depending on (I) whether tampering is applied to the original secret key persistently or applied to the changed secret key (classified by "persistent" and "non-persistent"), (II) whether tampering to an invalid codeword causes a "self-destruct" or not. Lastly, Bellare, Cash, and Miller [4] showed how any RKA secure identity-based encryption scheme leads to an RKA secure signature scheme, and Goyal et al. [22] showed that the Boneh-Boyen signature scheme [11] satisfied RKA security with respect to a class of certain polynomial RKD functions.

We note that the signature schemes EdDSA by Bernstein et al. [8] and ECDSA$^+$ by Koblitz and Menezes [25] resemble our schemes provided in Sect. 5.1 and 6.1, respectively, in the sense that one of the inputs to the hash function is the verification key. However, the schemes in [8] and [25] are proposed for a different context and RKA security is not considered.

2 Preliminaries

Here, we review basic notation and definitions of terminology.

2.1 Notation

Throughout the paper, we will use the following notation: For the set of natural numbers $\mathbb{N}$, let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathbb{G}$ be a group of prime order q, where q is a λ-bit prime. Let g be a generator of $\mathbb{G}$. Let $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$. A function $F : \mathbb{N} \to \mathbb{R}$ is negligible if it vanishes faster than the inverse of any polynomial. We write $\Pr[A : B]$ to denote a probability that the predicate A is true after the event B occurred. $\mathcal{O}(\cdot)$ denotes an order.

2.2 *d*-Strong Discrete Logarithm Assumption

We recall the d-strong discrete logarithm (d-SDL) assumption introduced by Goyal et al. [22]. Let d be a natural number. The d-SDL problem is to compute x given an input $(g, g^x, g^{x^2}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$, where $x \xleftarrow{\$} \mathbb{Z}_q$.

For an adversary $\mathcal{A}$ that solves the d-SDL problem over $\mathbb{G}$, we define the advantage as follows:

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{A},\mathbb{G}}(\lambda) = \Pr\left[x' = x : \begin{array}{c} x \xleftarrow{\$} \mathbb{Z}_q \\ x' \leftarrow \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^d}) \end{array}\right].$$

The d-SDL assumption over $\mathbb{G}$ says that the advantage $\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{A},\mathbb{G}}(\lambda)$ is negligible for any polynomial time algorithm $\mathcal{A}$.

It is clear that the 1-SDL assumption is equivalent to the standard DL assumption. Similar to the d-Strong Diffie-Hellman problem [11], the d-SDL problem is easier than the standard DL problem. In particular, more efficient solving algorithms, similar to Jao and Yoshida's algorithm [24] for the d-Strong Diffie-Hellman problem, can likely be constructed for the d-SDL problem.

2.3 General Forking Lemma

Bellare and Neven [6] generalized the forking lemma that was introduced by Pointcheval and Stern [27, 28] for security proofs of signature schemes.

Lemma 1 ([6]). *Fix an integer $Q \geq 1$ and a set Z of size $q \geq 2$. Let IG be a randomized algorithm called the input generator that outputs a string X. Suppose that a probabilistic algorithm $\mathcal{F}$ on input $X, h_1, \ldots, h_Q$ outputs a pair of an integer J and a side output V, where $h_1, \ldots, h_Q \in Z$ and the integer J is in the range $0, \ldots, Q$.*

Suppose that **acc** *denotes the following probability:*

$$\mathbf{acc} = \Pr\left[J \geq 1 : \begin{array}{c} X \xleftarrow{\$} \mathrm{IG} \\ h_1, \ldots, h_Q \xleftarrow{\$} Z \\ (J, V) \xleftarrow{\$} \mathcal{F}(X, h_1, \ldots, h_Q) \end{array}\right].$$

The forking algorithm $\mathcal{B}_{\mathcal{F}}$ associated to $\mathcal{F}$ is the randomized algorithm that takes X as input, and proceeds as follows:

```
1. Pick randomness $\rho_{\mathcal{F}}$ for $\mathcal{F}$ at random
2. $h_1, \ldots, h_Q \xleftarrow{\$} Z$
3. $(I, V) \leftarrow \mathcal{F}(X, h_1, \ldots, h_Q; \rho_{\mathcal{F}})$
4. If $I = 0$ then return $(0, \perp, \perp)$
5. $h'_1, \ldots, h'_Q \xleftarrow{\$} Z$
6. $(I', V') \leftarrow \mathcal{F}(X, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_Q; \rho_{\mathcal{F}})$
7. If ($I = I'$ and $h_I \neq h'_I$) then return $(1, V, V')$
8. Else return $(0, \perp, \perp)$
```

Let the probability that $\mathcal{B}_{\mathcal{F}}$ outputs $(1, V, V')$ be

$$\mathbf{frk} = \Pr[d = 1 : X \xleftarrow{\$} \mathrm{IG};\ (d, V, V') \xleftarrow{\$} \mathcal{B}_{\mathcal{F}}(X)].$$

Then

$$\mathbf{frk} \geq \mathbf{acc} \cdot \left(\frac{\mathbf{acc}}{Q} - \frac{1}{q}\right).$$

2.4 Signature

We recall the syntax of signature schemes, introduce functions with respect to which RKA security is considered, and lastly define RKA security for a signature scheme.

Signature Scheme. A signature scheme Σ consists of three algorithms: key generation algorithm, signing algorithm, and verification algorithm. We write

$$\Sigma = (\text{KeyGen}, \text{Sign}, \text{Verify}),$$

where these algorithms have the following interfaces:

$$\begin{aligned}(sk, vk) &\leftarrow \text{KeyGen}(1^\lambda),\\ \sigma &\leftarrow \text{Sign}(m, sk),\\ 1/0 &\leftarrow \text{Verify}(m, \sigma, vk),\end{aligned}$$

and sk, vk, and σ are a signing key, a verification key and a signature, respectively. For any message m and any key pair (sk, vk) generated by KeyGen, the following correctness should be satisfied:

$$\text{Verify}(m, \text{Sign}(m, sk), vk) = 1.$$

Related-Key Attack. In the ordinary attack model, an adversary is allowed to obtain signatures on arbitrary messages of its choice. In the RKA model, an adversary is also allowed to modify the signing key and obtain signatures on arbitrary messages of its choice under the modified signing key.

The RKA model, for instance, captures a realistic attack in which an adversary manipulates a hardware-stored secret key by electromagnetic radiation and obtains the outputs of the signing algorithm. This is called tampering or a fault injection attack. RKA is formalized as a security game that also allows an adversary to obtain signatures for modified keys. Thus, an adversary is allowed to query related-key deriving (RKD) functions [5] as well as messages to the signing oracle.

An RKD function is a function $\phi : K \to K$, where K is the signing key space. Let Φ be a class of RKD functions. The RKD function class Φ consists of operations by which an adversary is allowed to manipulate a signing key. Normally, Φ is assumed to contain the identity function id so that RKA security implies standard EUF-CMA [21]. We assume that it is easy to check whether a function is contained in a class Φ, and that RKD functions are efficiently computable.

Following [7], we consider three types of RKD functions: linear functions, affine functions, and polynomial functions. In the following, K is assumed to have an appropriate algebraic structure (group or finite field). In this paper, we will consider signature schemes whose signing key space is $\mathbb{Z}_q$ with prime q, which constitutes a field, as required.

Linear functions. Assume that $(K, *)$ is a group. The class of linear functions is defined as follows: $\Phi^{\text{lin}} = \{\phi_\Delta \mid \Delta \in K\}$, where $\phi_\Delta(k) = k * \Delta$ for a key $k \in K$. Note that "$*$" represents addition or multiplication depending on the group that is considered.

Affine functions. Assume that K is a finite field. The class of affine functions is defined as follows: $\Phi^{\text{aff}} = \{\phi_{\alpha,\beta} \mid \alpha, \beta \in K\}$, where $\phi_{\alpha,\beta}(k) = \alpha \cdot k + \beta$ for a key $k \in K$.

Polynomial functions. Assume that K is a finite field. The class of polynomial functions is defined as follows: $\Phi^{\text{poly}(d)} = \{\phi_f \mid f \in K_d[x]\}$, where $K_d[x]$ is the set of polynomials over K with degree at most d, and $\phi_f(k) = f(k)$ for a key $k \in K$.

RKA security is getting stronger and harder to achieve, as it moves from linear functions to affine functions to polynomial functions. In this paper, we only consider such algebraic operations.

***Φ*-EUF-CM-RKA [4].** We recall existential unforgeability under chosen message and RKA defined by RKD function class Φ. This security of a signature scheme, which we will denote by Φ-EUF-CM-RKA, is formalized by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Initialization. The challenger $\mathcal{B}$ runs KeyGen(1^λ) to obtain a signing key sk and a verification key vk. $\mathcal{B}$ sets a list $M \leftarrow \emptyset$. Then, $\mathcal{B}$ gives vk to $\mathcal{A}$.

RKA signing oracle query. For adaptive queries (m_i, ϕ_i) by $\mathcal{A}$, $\mathcal{B}$ returns the signatures

$$\sigma_i \leftarrow \text{Sign}(m_i, \phi_i(sk)),$$

where $\phi_i \in \Phi$. If $\phi_i(sk) = sk$, $\mathcal{B}$ records m_i in the list M.

Output. Suppose that $\mathcal{A}$ outputs (m^*, σ^*). If $\text{Verify}(m^*, \sigma^*, vk) = 1$ and $m^* \notin M$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Let F be the event that $\mathcal{B}$'s output is 1 in the above game. We define the advantage of $\mathcal{A}$ against Φ-EUF-CM-RKA security as

$$\text{Adv}_{\mathcal{A},\Sigma}^{\Phi\text{-euf-cm-rka}}(\lambda) := \Pr[F].$$

If the advantage $\text{Adv}_{\mathcal{A},\Sigma}^{\Phi\text{-euf-cm-rka}}(\lambda)$ is negligible for any probabilistic polynomial time algorithm $\mathcal{A}$, a signature scheme Σ is said to be Φ-EUF-CM-RKA secure.

We note that the security definition is strong in the sense that the adversary can reuse the message m_i as the forgery even if (m_i, ϕ_i) has been queried to the RKA signing oracle as long as $\phi_i(sk) \neq sk$.

Φ-wEUF-CM-RKA. We also consider a weaker variant of the above notion following the traditional weak existential unforgeability against adaptive chosen-message attacks [21] and the weak existential unforgeability of message authentication codes against RKA [9]. By requiring that the adversary in the above security experiment, produces a forgery on a message m^* which has not previously been submitted to the RKA signing oracle, we obtain the weaker security notion Φ-wEUF-CM-RKA.

Although it can be argued that, in some scenarios, the weaker notion Φ-wEUF-CM-RKA is sufficient to guarantee security, we note that the standard notion used in the literature, corresponds to the stronger notion Φ-EUF-CM-RKA defined above. We will show that the Schnorr signature scheme is $\Phi^{\text{poly}(d)}$-wEUF-CM-RKA secure, but the scheme is vulnerable with respect to Φ^{lin}-EUF-CM-RKA as we demonstrate in Sect. 4.1. The improved Schnorr signature scheme presented in Sect. 5.1 will be proven to be $\Phi^{\text{poly}(d)}$-EUF-CM-RKA secure. We furthermore show that one of the DSA variants from [29] is $\Phi^{\text{poly}(d)}$-wEUF-CM-RKA secure, but the original DSA is vulnerable with respect to Φ^{lin}-EUF-CM-RKA as we demonstrate in Sect. 4.2. Note that it is not known whether the DSA variant is vulnerable to $\Phi^{\text{poly}(d)}$-EUF-CM-RKA, but the improved DSA presented in Sect. 6.1 will be proven to be $\Phi^{\text{poly}(d)}$-EUF-CM-RKA secure. For further details, see Sect. 4, 5, and 6.

We note that, stronger models of RKA security that is often called fault attacks have been considered for round-based symmetric encryption schemes [10, 14, 20].These models allow the adversary to introduce faults (i.e. modification of the input or the internal state) in the individual rounds of the encryption algorithm, which, for example, lead to recovering a secret key. A similar extension, in which the adversary can choose when in the execution of the signing algorithm it would like to modify the signing key, could be considered for the RKA security of signature schemes. However, in this paper, we focus on the standard RKA notion (and its weaker variant) introduced above.

2.5 Schnorr Signature Scheme

The Schnorr signature scheme was proposed by Schnorr in 1989 [31] and was proven to be secure in the random oracle model based on the discrete logarithm assumption [27]. Recall that $\mathbb{G}$ is a group of prime order q, and g is a generator. The three algorithms, key generation, signing, and verification algorithms, are defined as follows.

- KeyGen: This algorithm takes 1^λ as input, and generates a signing key sk and a verification key vk as follows.
  1. Choose $x \xleftarrow{\$} \mathbb{Z}_q$ and let $y \leftarrow g^x$.
  2. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
  3. Output $sk = x, vk = (y, H)$.
- Sign: This algorithm takes a message $m \in \{0,1\}^*$ and the signing key sk as input, and generates a signature σ as follows.
  1. Choose $t \xleftarrow{\$} \mathbb{Z}_q$ and let $r \leftarrow g^t$.
  2. Let $h \leftarrow H(m \,\|\, r)$.
  3. Let $s \leftarrow x \cdot h + t \bmod q$.
  4. Output $\sigma \leftarrow (h, s)$.
- Verify: This algorithm takes a message m, a signature σ, and the verification key vk as input, and verifies the signature as follows.
  1. Let $r' \leftarrow g^s y^{-h}$.
  2. Let $h' \leftarrow H(m \,\|\, r')$.
  3. If $h' = h$, return 1, otherwise return 0.

2.6 DSA

DSA was proposed as the US Digital Signature Standard [1] in 1994. First, we recall the original DSA scheme.

Let p and q be primes, where q is a prime factor of $p-1$. Let $g \in \mathbb{Z}_p^*$ be a generator of prime order q. DSA is defined by the following three algorithms:

- KeyGen: This algorithm takes 1^λ as input, and generates a signing key sk and a verification key vk as follows.
  1. Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and let $y \leftarrow g^x \bmod p$.
  2. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
  3. Output $sk = x, vk = (y, H)$.
- Sign: This algorithm takes a message $m \in \{0,1\}^*$ and the signing key sk as input, and generates a signature σ as follows.
  1. Choose $t \xleftarrow{\$} \mathbb{Z}_q^*$ and let $r \leftarrow (g^t \bmod p) \bmod q$.
  2. Let $s \leftarrow t^{-1}(H(m) + x \cdot r) \bmod q$.
  3. Output $\sigma \leftarrow (r, s)$.
- Verify: This algorithm takes a message m, a signature $\sigma = (r, s)$, and the verification key $vk = (y, H)$ as input, and verifies the signature as follows.
  1. Let $r' \leftarrow (g^{H(m)/s} y^{r/s} \bmod p) \bmod q$.
  2. If $r' = r$, output 1, otherwise output 0.

Variants of DSA. While the original scheme has not been proven to be secure, Pointcheval and Vaudenay [29] proved that two variants of DSA are secure in the sense of standard security in the random oracle model. The first DSA variant uses one additional random oracle H', and the first step of signing algorithm computes $r \leftarrow H'(g^t \bmod p)$. The second DSA variant's main difference is that a hash function takes as input not only a message but also the value r. Looking ahead, we will consider a slight modified version of this second variant of DSA in Sect. 6.

On the Collision Resistance of the DSA mapping from $\mathbb{Z}_p^*$ to $\mathbb{Z}_q$. Note that in Step 1 of the signing algorithm of DSA, we have to map an element $g^t \in \mathbb{Z}_p^*$ to an element $r \in \mathbb{Z}_q$. In [29], Pointcheval and Vaudenay considered this mapping an abstract function from $\mathbb{G}$ to $\mathbb{Z}_q$, where $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ of order q. To prove security of their second variant of DSA, Pointcheval and Vaudenay made the assumption that this function has a certain collision resistance property. In this paper, we take a similar approach as [29], and assume this function, which we will denote $F_{p,q}$, has the following property:

Let $F_{p,q} : \mathbb{G} \to \mathbb{Z}_q$ be the mapping defined by $\overline{g} \mapsto \overline{g} \bmod q$, where $\overline{g} \in \mathbb{G}$, and $\mathbb{G}, q, p$ are the parameters of the group over which DSA is constructed (i.e. $\mathbb{G}$ is a subgroup of $\mathbb{Z}_p^*$ of order q). We say that $F_{p,q}$ is ϵ-collision-resistant if no probabilistic polynomial time algorithm $\mathcal{A}$ can find two distinct elements $\overline{g}_1, \overline{g}_2 \in \mathbb{G}$ such that $F_{p,q}(\overline{g}_1) = F_{p,q}(\overline{g}_2)$ with probability more than ϵ. When ϵ is negligible in the security parameter, we simply say that $F_{p,q}$ is collision resistant.

3 wRKA Security of Signature Schemes

In this section, we show that the Schnorr signature scheme and the second variant of DSA from [29] are $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA secure. We remind the reader that $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA security requires that the message m^* in the forgery must be new and that it has not been submitted to the RKA signing oracle.

First, we show the following theorem regarding the Schnorr signature scheme.

Theorem 1. *Let d be a positive integer. Under the d-SDL assumption over $\mathbb{G}$, the Schnorr signature scheme is $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA secure in the random oracle model.*

More precisely, for any probabilistic polynomial time algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}$, making q_S RKA signing oracle queries, and q_H random oracle queries to H, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ with running time $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ that satisfies the following equation:

$$\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) \le \left((q_H + q_S) \left(\mathrm{Adv}_{\mathcal{B},\mathbb{G}}^{d\text{-sdl}}(\lambda) + \frac{2q_S + 1}{q} \right) \right)^{1/2}.$$

The proof is very similar to that of Theorem 3 in Sect. 5.2. We highlight the differences between the two proofs after presenting the proof of Theorem 3.

Next, we show the following theorem regarding the second DSA variant from [29].

Theorem 2. *Let d be a positive integer, and assume the mapping $F_{p,q}$ is collision resistant. Under the d-SDL assumption over $\mathbb{G}$, the second DSA variant is $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA secure in the random oracle model.*

More precisely, assume that $F_{p,q}$ is ϵ-collision-resistant. Then, for any probabilistic polynomial time algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}$, making q_S RKA signing oracle queries, and q_H random oracle queries to H, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ with running time $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ that satisfies the following equation:

$$\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) \leq \left((q_H + q_S)\left(\mathrm{Adv}_{\mathcal{B},\mathbb{G}}^{d\text{-sdl}}(\lambda) + \frac{1}{q} + \frac{2\epsilon}{q_H + q_S}\right)\right)^{1/2}.$$

Since the proof is quite similar to that of Theorem 4 in Sect. 6.2, the differences between the two proofs are highlighted after Theorem 4.

4 Related-Key Attacks against Signature Schemes

In this section, we show related-key attacks against the Schnorr signature scheme and DSA. As mentioned in Sect. 2.4, linear functions as RKD functions can be described as addition or multiplication depending on the group used as the signing key space.

4.1 Related-Key Attack against Schnorr Signature

We show that the Schnorr signature scheme is not RKA secure with respect to linear functions or addition by providing a simple and efficient attack. That is, we show that the Schnorr signature scheme is not Φ^{lin}-EUF-CM-RKA secure.

An adversary $\mathcal{A}$ forges a signature as follows.

1. Choose an arbitrary message $m' \in \{0,1\}^*$ and an arbitrary value $b \in \mathbb{Z}_q^*$.
2. Query $(m', \phi(x) = x - b)$ to the RKA signing oracle and obtain the signature (h', s') as a response.
3. Output a message m' and forgery $(h', s' + b \cdot h')$.

Now, let us confirm that the forgery is valid. First, the reply from the RKA signing oracle, (h', s'), must have been computed by the following procedure:

– Choose $t' \xleftarrow{\$} \mathbb{Z}_q$ and let $r' \leftarrow g^{t'}$.
– Let $h' \leftarrow H(m' \,\|\, r')$.
– Let $s' \leftarrow (x - b) \cdot h' + t' \bmod q$.

The forged signature $(h', s' + b \cdot h')$ on the message m' is verified as follows.

$$r'' = g^{s'+b\cdot h'} y^{-h'} = g^{(x-b)\cdot h'+t'+b\cdot h'} y^{-h'} = g^{(x-b)\cdot h'+t'+b\cdot h'-x\cdot h'} = g^{t'} = r'.$$

4.2 Related-Key Attack against DSA

We next show that DSA is not RKA secure with respect to linear functions or multiplication by providing a simple and efficient attack. That is, we show that DSA is not Φ^{lin}-EUF-CM-RKA secure.

An adversary $\mathcal{A}$ forges a signature as follows.

1. Choose two distinct messages $m_0, m_1 \in \{0,1\}^*$ and let $z_0 \leftarrow H(m_0), z_1 \leftarrow H(m_1)$.
2. Let $a \leftarrow \dfrac{z_1}{z_0} \bmod q$.
3. Query $(m_1, \phi(x) = ax)$ to the RKA signing oracle and obtain the signature $(r, s = t^{-1}(z_1 + axr))$.
4. Output a message $m^* = m_0$ and the signature $(r^*, s^*) = (r, \dfrac{s}{a} \bmod q)$.

Note that even if a is 1, the attack still works.

The forged signature $(r, \frac{s}{a} \bmod q)$ on the message m_0 will be verified as follows. First, we compute w^*:

$$w^* = (s^*)^{-1} = \frac{a}{s} = \frac{ta}{z_1 + axr} = \frac{ta}{a \cdot z_0 + axr} = \frac{t}{z_0 + xr}.$$

Then, we compute $u_1 = w^* z_0 \bmod q$ and $u_2 = rw^* \bmod q$. Now we can check

$$\begin{aligned} r' &= (g^{H(m_0)/s^*} y^{r^*/s^*} \bmod p) \bmod q = (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (g^{w^* z_0} y^{rw^*} \bmod p) \bmod q = (g^{w^* z_0 + xrw^*} \bmod p) \bmod q \\ &= (g^{w^*(z_0+xr)} \bmod p) \bmod q = (g^t \bmod p) \bmod q = r. \end{aligned}$$

Thus, the forgery output by $\mathcal{A}$ is valid.

5 Improved Schnorr Signature Scheme and Its RKA Security

As described in Sect. 4.1, the original Schnorr signature scheme is not RKA secure with respect to linear functions. In this section, we show that a slight modification yields an RKA-secure signature scheme with respect to polynomial functions. We refer to this scheme as the improved Schnorr signature scheme.

5.1 Construction

Our slight modification of the Schnorr signature scheme is as follows. The hash function is modified to take an extra input, which will correspond to a recalculated value of the verification key. Suppose that $\mathbb{G}$ is a group of prime order q, and g is a generator. The improved Schnorr signature scheme is defined as follows:

- KeyGen: This algorithm takes 1^λ as input, and generates a signing key sk and a verification key vk as follows.
  1. Choose $x \xleftarrow{\$} \mathbb{Z}_q$ and let $y \leftarrow g^x$.
  2. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
  3. Output $sk = x$ and $vk = (y, H)$.
- Sign: This algorithm takes a message $m \in \{0,1\}^*$ and the signing key sk as input, and generates a signature σ as follows.
  1. Choose $t \xleftarrow{\$} \mathbb{Z}_q$ and let $r \leftarrow g^t$.
  2. Let $\psi \leftarrow g^x$.
  3. Obtain $h \leftarrow H(m \parallel r \parallel \psi)$.
  4. Let $s \leftarrow x \cdot h + t \bmod q$.
  5. Output $\sigma \leftarrow (h, s)$.
- Verify: This algorithm takes a message m, a signature σ, and the verification key vk as input, and verifies the signature as follows.
  1. Let $r' \leftarrow g^s y^{-h}$.
  2. Let $h' \leftarrow H(m \parallel r' \parallel y)$.
  3. If $h' = h$, output 1, otherwise output 0.

Note that the second step of the signing algorithm, computation of $\psi \leftarrow g^x$, should not be altered to simply use the verification key y as ψ. That is, the signing algorithm computes $\psi = g^x$ each time it computes a signature.

Given that the verification key is recomputed from the signing key, one might wonder whether RKA security can be achieved simply by comparing the recomputed verification key with the original (assuming that the original verification key is available to the signing algorithm). However, for this to work, the additional assumption that the original verification key is stored and remains unchanged, is required. In the RKA setting, this seems unlikely to hold since the adversary is assumed to be capable of modifying the signing key, which should be better protected than the verification key. Furthermore, if the adversary is capable of modifying the stored signing key, a similar attack to Sect. 4.1 and 4.2 will be possible: an attacker queries $(m', \phi(x) = x - b)$ under the modified verification key yg^{-b} in the second step of the attack. In contrast, our schemes provided in this section and in Sect. 6.1 can be shown RKA secure without any additional assumptions regarding stored values.

5.2 Proof of Security

We prove the following theorem about the improved Schnorr signature scheme.

Theorem 3. *Let d be a positive integer. Under the d-SDL assumption over $\mathbb{G}$, the signature scheme in Sect. 5.1 is $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA secure in the random oracle model.*

More precisely, for any probabilistic polynomial time algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}$, making q_S RKA signing oracle queries, and q_H random oracle queries to H, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ with running time $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ that satisfies the following equation:

$$\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) \leq \left((q_H + q_S) \left(\mathrm{Adv}_{\mathcal{B},\mathbb{G}}^{d\text{-sdl}}(\lambda) + \frac{2q_S + 1}{q} \right) \right)^{1/2}. \quad (1)$$

Proof. Let $\mathcal{A}$ be any probabilistic polynomial time adversary that attacks the $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA security of the improved Schnorr signature scheme making q_S RKA signing oracle queries and q_H random oracle queries to H. We will show how the algorithm $\mathcal{B}$ that solves the d-SDL problem is constructed, and then we will show the above inequality between the advantage of $\mathcal{A}$ and that of $\mathcal{B}$ (Eq. (1)).

First, we specify the following to use the general forking lemma. Suppose that IG is the input generator that determines $\mathcal{G} \leftarrow (\mathbb{G}, q, g)$, picks $x \xleftarrow{\$} \mathbb{Z}_q$, computes $y_1 \leftarrow g^x, y_2 \leftarrow g^{x^2}, \ldots, y_d \leftarrow g^{x^d}$, and outputs $X = (\mathcal{G}, y_1, y_2, \ldots, y_d)$. Let the set Z be $\mathbb{Z}_q$, and let the integer Q be $q_H + q_S$. Let $\mathcal{F}$ be an algorithm that uses randomness $\rho_{\mathcal{F}}$ consisting of randomness $\rho_{\mathcal{A}}$ for $\mathcal{A}$ and q_S values $s_1, \ldots, s_{q_S} \in \mathbb{Z}_q$, takes $X = (\mathcal{G}, y_1, y_2, \ldots, y_d)$ and $h_1, \ldots, h_{q_H+q_S} \in \mathbb{Z}_q$ as input, and internally runs $\mathcal{A}$ as follows:

Procedure of $\mathcal{F}$. $\mathcal{F}$ sets $pp \leftarrow \mathcal{G}$ and sets $vk \leftarrow y_1$. Note that values $h_1, \ldots, h_{q_H}$ will be used for the response to $\mathcal{A}$'s hash oracle queries, and values $s_1, \ldots, s_{q_S}, h_{q_H+1}, \ldots, h_{q_H+q_S}$ will be used for the response to $\mathcal{A}$'s RKA signing oracle queries.

Initialization. $\mathcal{F}$ proceeds as follows.
- Let $L_H \leftarrow \emptyset, M \leftarrow \emptyset$.
- Run $\mathcal{A}$ with public parameter pp, input $vk \leftarrow y$, and randomness $\rho_{\mathcal{A}}$.

Hash oracle query. For the i-th ($1 \leq i \leq q_H$) hash oracle query (m_i, r_i, y_i) from $\mathcal{A}$, $\mathcal{F}$ responds as follows:
- If $(m_i, r_i, y_i, h, \cdot)$ for some $h \in \mathbb{Z}_q$ is recorded in L_H ($\cdot$ is an arbitrary value), $\mathcal{F}$ returns h.
- Otherwise, $\mathcal{F}$ records $(m_i, r_i, y_i, h_i, \perp)$ in the list L_H and returns h_i.

RKA signing oracle query. For the j-th ($1 \leq j \leq q_S$) RKA signing oracle query (m_j, ϕ_j) from $\mathcal{A}$, $\mathcal{F}$ computes the following value.

$$r_j \leftarrow g^{s_j} (g^{\phi_j(x)})^{-h_{q_H+j}} = g^{s_j - a_0 \cdot h_{q_H+j}} \{ (g^{x^d})^{a_d} \cdot (g^{x^{d-1}})^{a_{d-1}} \cdots (g^x)^{a_1} \}^{-h_{q_H+j}},$$

where $\phi_j(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \cdots + a_1 \cdot x + a_0$. Then,
- If there is already an entry of the form $(m_j, r_j, g^{\phi_j(x)}, \cdot, \cdot)$ in the list L_H, then $\mathcal{F}$ gives up and terminates with output $(0, \perp)$.
- Otherwise, $\mathcal{F}$ records $(m_j, r_j, g^{\phi_i(x)}, h_{q_H+j}, s_j)$ in L_H, where h_{q_H+j} is a part of $\mathcal{F}$'s input, and s_j is a part of $\mathcal{F}$'s randomness $\rho_{\mathcal{F}}$, then returns a signature $\sigma_j = (h_{q_H+j}, s_j)$ to $\mathcal{A}$.

Output. When $\mathcal{A}$ outputs a message m^* and a forged signature $\sigma^* = (h^*, s^*)$, $\mathcal{F}$ verifies it by using the verification key and the list M. For the forged signature $\sigma^* = (h^*, s^*)$ on the message m^*, we assume that $\mathcal{A}$ has made a hash query (m^*, r^*, y), where

$$r^* = g^{s^*} y^{-h^*}.$$

We assume this without loss of generality because we are always able to construct an adversary $\mathcal{A}'$ that queries (m^*, r^*, y) to the hash oracle by using $\mathcal{A}$ that might not query (m^*, r^*, y) to the hash oracle. Note that such an adversary $\mathcal{A}'$ increases hash oracle query only once. Thus, $\mathcal{F}$ always finds $(m^*, r^*, y, h, \cdot)$ for some $h \in \mathbb{Z}_q$ in the list L_H. Let $J \in \{1, \ldots, q_H\}$ be the index such that $h = h_J$ found in this process. Then,
- If $h^* = h_J$ and the last element of $(m^*, r^*, y, h^*, \cdot)$ is $\perp$ (i.e. h^* has been computed by hash oracle, but has not been computed by RKA signing oracle), then $\mathcal{F}$ outputs $(J, V = (h^*, s^*))$. Here, the case means that h^* satisfies the equation $h^* = H(m^*, r^*, y)$, and thus $\sigma^* = (h^*, s^*)$ is a valid signature for m^* in the experiment simulated by $\mathcal{F}$.

– Otherwise, $\mathcal{F}$ outputs $(0, \perp)$.

The above completes the description of $\mathcal{F}$. We observe that the probability that $\mathcal{F}$ fails to answer $\mathcal{A}$'s RKA signing oracle queries is at most $q_S(q_H + q_S)/q$ because of the following reasons. First, $\mathcal{A}$ is not able to compute the hidden value $r_i = g^{s_i} \cdot (g^{\phi(x)})^{-h_i}$ computed by $\mathcal{F}$ because s_i and h_i are hidden from $\mathcal{A}$. Thus the probability that the list L_H already contains an entry $(m_i, r_i, g^{\phi(x)}, \cdot, \cdot)$ when $\mathcal{A}$ makes one RKA signing oracle query is at most $(q_H + q_S)/q$, where $q_H + q_S$ represents the size of L_H. Then, since $\mathcal{A}$ makes at most q_S RKA signing oracle queries, the probability that $\mathcal{F}$ fails to answer $\mathcal{A}$'s RKA signing queries is at most $q_S(q_H + q_S)/q$. Here, note that $\mathcal{F}$ perfectly simulates the EUF-CM-RKA experiment for $\mathcal{A}$ in the random oracle model unless $\mathcal{F}$ fails to answer $\mathcal{A}$'s RKA signing oracle queries. If $\mathcal{A}$ succeeds in outputting a valid forgery pair $(m^*, \sigma^* = (h^*, s^*))$, and $\mathcal{F}$ succeeds in answering RKA signing oracle queries, then $\mathcal{F}$ always outputs $(J, V = (h^*, s^*))$ such that $J \geq 1$ and $h^* = h_J$. Since the probability that $\mathcal{F}$ outputs such (J, V) is **acc**, we obtain

$$\mathbf{acc} \geq \mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) - \frac{q_S(q_H + q_S)}{q}.$$

By the general forking lemma (Lemma 1), we have the following inequality about the probability **frk** that $\mathcal{B}_{\mathcal{F}}$ succeeds in forking (i.e., terminates at Step 7 in the lemma),

$$\mathbf{frk} \geq \mathbf{acc} \cdot \left(\frac{\mathbf{acc}}{q_H + q_S} - \frac{1}{q} \right).$$

Thus we have

$$\mathbf{frk} \geq \frac{(\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda))^2}{q_H + q_S} - \frac{2q_S + 1}{q}.$$

Next, we relate **frk** with the advantage of another algorithm $\mathcal{B}$ that solves the d-SDL problem.

Procedure of $\mathcal{B}$. $\mathcal{B}$ takes $\mathcal{G}$ and $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$ as input, and it sets $X = (\mathcal{G}, (g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}))$. The algorithm $\mathcal{B}$ performs the forking procedure $\mathcal{B}_{\mathcal{F}}$ corresponding to $\mathcal{F}$ as described above. Let (d, V, V') be the output of $\mathcal{B}_{\mathcal{F}}$. If $d = 0$, $\mathcal{B}$ gives up and aborts. Otherwise, let $V = (h^*_{(1)}, s^*_{(1)})$ and $V' = (h^*_{(2)}, s^*_{(2)})$. At this point, it is guaranteed that $h^*_{(2)} \neq h^*_{(1)}$. By using V and V', we have $r^* = g^{s^*_{(1)}} y^{-h^*_{(1)}} = g^{s^*_{(2)}} y^{-h^*_{(2)}}$, which implies $y = g^{(s^*_{(1)} - s^*_{(2)})/(h^*_{(1)} - h^*_{(2)})}$. Thus $\mathcal{B}$ computes $x = (s^*_{(1)} - s^*_{(2)})/(h^*_{(1)} - h^*_{(2)}) \bmod q$, and terminates with output x.

The above completes the description of $\mathcal{B}$, and this x is the required discrete logarithm. In other words, $\mathcal{B}$ solves the given instance of the d-SDL problem. The running time of $\mathcal{B}$, $t_{\mathcal{B}}$, is the sum of the running time of $\mathcal{B}_{\mathcal{F}}$, and the time to compute x. Since $\mathcal{B}_{\mathcal{F}}$ runs $\mathcal{F}$ twice, $\mathcal{F}$ runs $\mathcal{A}$ once, and $\mathcal{F}$ answers $\mathcal{A}$'s RKA signing and hash oracle queries, the running time of $\mathcal{B}$ satisfies $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$. $\mathcal{B}$ succeeds in computing the discrete logarithm x such that $g^x = y$ whenever $\mathcal{B}_{\mathcal{F}}$ terminates and outputs (d, V, V') such that $d = 1$. Therefore, we obtain

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) = \mathbf{frk}$$

and then we obtain

$$\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) \geq \frac{\left(\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda)\right)^2}{q_H + q_S} - \frac{2q_S + 1}{q}.$$

Finally, we obtain

$$\mathrm{Adv}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}_{\mathcal{A},\Sigma}(\lambda) \leq \left((q_H + q_S) \left(\mathrm{Adv}^{d\text{-sdl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \frac{2q_S + 1}{q} \right) \right)^{1/2},$$

which completes the proof. $\square$

We note that the proof of Theorem 1 is obtained by changing the hash oracle query and the components of the list L_H to (m_i, r_i) and $(m_j, r_j, h_{q_H+j}, s_j)$, respectively. Moreover, when the forgery is produced, the last element of $(m^*, r^*, h^*, \cdot)$ is $\perp$ since the message must not have been submitted to the RKA signing oracle as required by the security notion $\Phi^{\mathrm{poly}(d)}$-wEUF-CM-RKA. In the proof of Theorem 3, on the other hand, the last element of the forgery $(m^*, r^*, y, h^*, \cdot)$ is $\perp$ since the message m^* has not been submitted to the RKA signing oracle with ϕ such that $\phi(sk) = sk$, thus $y \neq g^{\phi(x)}$.

The 1-SDL assumption is equivalent to the ordinary DL assumption, which leads to the following result.

Corollary 1. *The improved Schnorr signature scheme is RKA secure with respect to affine functions in the random oracle model under the DL assumption over $\mathbb{G}$.*

6 Improved DSA and Its RKA Security

As described in Sect. 4.2, the original DSA is not RKA secure with respect to linear functions. In this section, we show that a slight modification yields an RKA-secure signature scheme with respect to polynomial functions. We refer to this scheme as the improved DSA.

6.1 Construction

Based on one of DSA variants (introduced as "second variant" in [29]), we construct an RKA secure variant of DSA with respect to polynomial functions. The slight modification of DSA variant is as follows. The hash function is modified to take an extra input, which will correspond to a recalculated value of the verification key. Suppose that q is a prime, p is a prime such that $p-1 \bmod q = 0$, and $\mathbb{G} \subseteq \mathbb{Z}_p^*$ is a group of prime order q. Let $g \in \mathbb{G}$ be a generator. Let $F_{p,q} : \mathbb{G} \to \mathbb{Z}_q$ be the mapping defined by $\overline{g} \mapsto \overline{g} \bmod q$, where $\overline{g} \in \mathbb{G}$, and $\mathbb{G}, q, p$ are the parameters of the group.

The improved DSA is defined as follows:

- KeyGen: This algorithm takes 1^λ as input, and generates the signing key sk and the verification key vk as follows.
  1. Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and let $y \leftarrow g^x \bmod p$.
  2. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
  3. Output $sk = x$ and $vk = (y, H)$.
- Sign: This algorithm takes a message $m \in \{0,1\}^*$, the verification key vk, and the signing key sk as input, and generates a signature σ as follows.
  1. Choose $t \xleftarrow{\$} \mathbb{Z}_q^*$ and let $r \leftarrow F_{p,q}(g^t \bmod p)$.
  2. Let $\psi \leftarrow g^x \bmod p$.
  3. Let $s \leftarrow t^{-1}(H(m \,\|\, r \,\|\, \psi) + x \cdot r) \bmod q$.
  4. Output $\sigma \leftarrow (r, s)$.
- Verify: This algorithm takes a message m, a signature σ, and the verification key vk as input, and verifies the signature as follows.
  1. Let $r' \leftarrow F_{p,q}(g^{H(m \,\|\, r \,\|\, y)/s} y^{r/s} \bmod p)$.
  2. If $r' = r$, output 1, otherwise output 0.

Note that the computation of a hash function at the third step of the signing algorithm takes as input not only a message and the value r, but also $\psi = g^x$. This computation is different from that of the second DSA variant [29].

6.2 Proof of Security

We prove the following theorem about the improved DSA.

Theorem 4. *Let d be a positive integer, and assume the mapping $F_{p,q}$ is collision resistant. Under the d-SDL assumption over $\mathbb{G}$, the signature scheme in Sect. 6.1 is $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA secure in the random oracle model.*

More precisely, assume that $F_{p,q}$ is ϵ-collision-resistant. Then, for any probabilistic polynomial time algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}$, making q_S RKA signing oracle queries, and q_H random oracle queries to H, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ with running time $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$ that satisfies the following equation:

$$\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) \leq \left((q_H + q_S) \left(\mathrm{Adv}_{\mathcal{B},\mathbb{G}}^{d\text{-sdl}}(\lambda) + \frac{1}{q} + \frac{2\epsilon}{q_H + q_S} \right) \right)^{1/2} . \quad (2)$$

Proof. An inequality between the advantage of $\mathcal{A}$ and that of $\mathcal{B}$ (Eq. (2)) is obtained in the same manner as the proof of Theorem 3. Now, we only show how the forking algorithm $\mathcal{F}$ and the algorithm $\mathcal{B}$ which solves the d-SDL problem are constructed.

As is done in Sect. 5.2, we first specify the following to use the general forking lemma. Suppose that IG is the input generator that determines $\mathcal{G} \leftarrow (\mathbb{G}, q, g)$, picks $x \xleftarrow{\$} \mathbb{Z}_q$, computes $y_1 \leftarrow g^x, y_2 \leftarrow g^{x^2}, \ldots, y_d \leftarrow g^{x^d}$, and outputs $X = (\mathcal{G}, y_1, y_2, \ldots, y_d)$. Let the set Z be $\mathbb{Z}_q$, and let the integer Q be $q_H + q_S$. Let $\mathcal{F}$ be an algorithm that uses randomness $\rho_{\mathcal{F}}$ consisting of randomness $\rho_{\mathcal{A}}$ for $\mathcal{A}$ and $\overline{r}_1, \ldots, \overline{r}_{q_H+q_S} \in \mathbb{Z}_q$, takes $X = (\mathcal{G}, y_1, y_2, \ldots, y_d)$ and $\overline{h}_1, \ldots, \overline{h}_{q_H+q_S} \in \mathbb{Z}_q$ as input (where we consider $\overline{r}_i$ are random values for r_i/s_i and $\overline{h}_i$ are random values for h_i/s_i.), and internally runs $\mathcal{A}$ as follows:

Procedure of $\mathcal{F}$. $\mathcal{F}$ sets $pp \leftarrow \mathcal{G}$ and sets $vk \leftarrow y_1$. Note that values indexed from 1 to q_H, $\overline{r}_1, \ldots, \overline{r}_{q_H}$, $\overline{h}_1, \ldots, \overline{h}_{q_H}$ will be used for the response to $\mathcal{A}$'s hash oracle queries, and the rest of the values $\overline{r}_{q_H+1}, \ldots, \overline{r}_{q_H+q_S}, \overline{h}_{q_H+1}, \ldots, \overline{h}_{q_H+q_S}$ will be used for the response to $\mathcal{A}$'s RKA signing oracle queries.

Initialization. $\mathcal{F}$ proceeds as follows.
- Let $L_H \leftarrow \emptyset, M \leftarrow \emptyset$.
- Run $\mathcal{A}$ with public parameter pp, input $vk \leftarrow y$, and randomness $\rho_{\mathcal{A}}$.

Hash oracle query. For the i-th $(1 \leq i \leq q_H)$ hash oracle query (m_i, r_i, y_i) from $\mathcal{A}$, $\mathcal{F}$ responds as follows:
- If $(m_i, r_i, y_i, h, \cdot)$ for some $h \in \mathbb{Z}_q$ is recorded in L_H ($\cdot$ is an arbitrary value), $\mathcal{F}$ returns h.
- Otherwise, $\mathcal{F}$ obtains h_i as follows: From $\overline{r}_i$ and r_i, $\mathcal{F}$ obtains s_i. From $\overline{h}_i$ and s_i, $\mathcal{F}$ obtains h_i. Then, $\mathcal{F}$ records $(m_i, r_i, y_i, h_i, \perp)$ in the list L_H and returns h_i.

RKA signing oracle query. For the j-th $(1 \leq j \leq q_S)$ RKA signing oracle query (m_j, ϕ_j) from $\mathcal{A}$, $\mathcal{F}$ computes the value $r_j \leftarrow F_{p,q}(g^{\overline{h}_j}(g^{\phi_j(x)})^{\overline{r}_j} \bmod p)$, where $\phi_j(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \cdots + a_1 \cdot x + a_0$. Note that $\mathcal{F}$ can compute $g^{\phi(x)}$ by using its input X. From $\overline{r}_{q_H+j}$ and r_{q_H+j}, $\mathcal{F}$ obtains s_{q_H+j}. From $\overline{h}_{q_H+j}$ and s_{q_H+j}, $\mathcal{F}$ obtains h_{q_H+j}.
- If there is already an entry of the form $(m_j, r_{q_H+j}, g^{\phi_j(x)}, \cdot, \cdot)$ in the list L_H, then $\mathcal{F}$ gives up and terminates with output $(0, \perp)$
- Otherwise, $\mathcal{F}$ records $(m_j, r_{q_H+j}, g^{\phi_j(x)}, h_{q_H+j}, s_{q_H+j})$ in L_H, and then $\mathcal{F}$ returns a signature $\sigma_j = (r_{q_H+j}, s_{q_H+j})$ to $\mathcal{A}$.

Output. When $\mathcal{A}$ outputs a message m^* and a forged signature $\sigma^* = (r^*, s^*)$, $\mathcal{F}$ verifies it by using the verification key and the list M. For the forged signature $\sigma^* = (r^*, s^*)$ on the message m^*, we assume that $\mathcal{A}$ has made a hash query (m^*, r^*, y) without loss of generality. Thus, $\mathcal{F}$ always finds $(m^*, r^*, y, h, \cdot)$ for some $h \in \mathbb{Z}_q$ in the list L_H. Let $J \in \{1, \ldots, q_H\}$ be the index such that $h = h_J$ found in this process. Then,
- If $h^* = h_J$ and the last element of $(m^*, r^*, y, h^*, \cdot)$ is $\perp$ (i.e. h^* has been computed by hash oracle, but has not been computed by RKA signing oracle), where $h^* = H(m^*, r^*, y)$, then $\mathcal{F}$ outputs $(J, V \leftarrow (r^*, s^*))$. Here, the case means $\sigma^* = (r^*, s^*)$ is a valid signature for m^* in the experiment simulated by $\mathcal{F}$.
- Otherwise, $\mathcal{F}$ outputs $(0, \perp)$.

The above completes the description of $\mathcal{F}$. Note that unless one of $\mathcal{A}$'s RKA signing oracle queries causes a collision in the list L_H (in which case $\mathcal{F}$ will terminate with output $(0, \perp)$), $\mathcal{F}$ will provide a perfect simulation for $\mathcal{A}$. Furthermore, since a collision in the list L_H will directly lead to the ϵ-collision-resistance of the mapping function $F_{p,q}$ to be broken, the probability that F terminates with output different from $(0, \perp)$ will be at least $\mathrm{Adv}_{\mathcal{A},\Sigma}^{\Phi^{\mathrm{poly}(d)}\text{-euf-cm-rka}}(\lambda) - \epsilon$.

Procedure of $\mathcal{B}$. $\mathcal{B}$ takes a d-SDL problem $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}) \in \mathbb{G}^{d+1}$ as input, and simulates the challenger for an adversary $\mathcal{A}$ in the $\Phi^{\mathrm{poly}(d)}$-EUF-CM-RKA security game. It sets $X = (\mathcal{G}, (g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^d}))$. The algorithm $\mathcal{B}$ performs the forking procedure $\mathcal{B}_{\mathcal{F}}$ corresponding to $\mathcal{F}$ as described above. Let (d, V, V') be the output of $\mathcal{B}_{\mathcal{F}}$. If $d = 0$, $\mathcal{B}$ gives up and aborts. Otherwise, let $V = (r^*, s^*_{(1)})$ and $V' = (r^*, s^*_{(2)})$. At this point, it is guaranteed that $h^*_{(2)} \neq h^*_{(1)}$, thus $s^*_{(2)} \neq s^*_{(1)}$. By using V and V', we have $r' = F_{p,q}(g^{h^*_{(1)}/s^*_{(1)}} y^{r^*/s^*_{(1)}} \bmod p) = F_{p,q}(g^{h^*_{(2)}/s^*_{(2)}} y^{r^*/s^*_{(2)}} \bmod p)$, which implies

$$y = g^{(s^*_{(1)} h^*_{(2)} - s^*_{(2)} h^*_{(1)})/r^*(s^*_{(2)} - s^*_{(1)})}$$

Thus, $\mathcal{B}$ obtains

$$x = \frac{s^*_{(1)} h^*_{(2)} - s^*_{(2)} h^*_{(1)}}{r^*(s^*_{(2)} - s^*_{(1)})} \bmod q.$$

This x is the required discrete logarithm. In other words, $\mathcal{B}$ solves the given instance of the d-SDL problem. The running time of $\mathcal{B}$, $t_{\mathcal{B}}$, is the sum of the time to run $\mathcal{A}$ twice, the time to compute x, and the time to answer the RKA signing and hash oracle queries. Therefore $t_{\mathcal{B}} = 2t_{\mathcal{A}} + \mathcal{O}(q_S + q_H)$. $\square$

Now, we highlight the differences between the proof of Theorem 2 and Theorem 4. The proof of Theorem 2 is obtained by changing the hash oracle query to (m_i, r_i) and the components of the list L_H to $(m_j, r_j, h_{q_H+j}, s_j)$. Furthermore, when the forgery is produced and the algorithm $\mathcal{F}$ checks the validity, the last element of $(m^*, r^*, h^*, \cdot)$ is $\perp$ since the security notion $\Phi^{\text{poly}(d)}$-wEUF-CM-RKA does not allow the message to have been submitted to the RKA signing oracle. In the proof of Theorem 4, on the other hand, the security notion $\Phi^{\text{poly}(d)}$-EUF-CM-RKA guarantees that the last element of the forgery $(m^*, r^*, y, h^*, \cdot)$ is $\perp$ since the message m^* has not been submitted to the RKA signing oracle with ϕ such that $\phi(sk) = sk$, thus $y \neq g^{\phi(x)}$. The 1-SDL assumption is equivalent to the ordinary DL assumption, which leads to the following result.

Corollary 2. *If the DL assumption over $\mathbb{G}$ holds and the function $F_{p,q}$ is collision-resistant, then the improved DSA is RKA secure with respect to affine functions in the random oracle model.*

7 Conclusions

We analyzed the RKA security of the Schnorr signature scheme and DSA. We showed that the Schnorr signature scheme and the second DSA variant from [29] are weak RKA secure with respect to polynomial functions ($\Phi^{\text{poly}(d)}$-wEUF-CM-RKA), but the Schnorr signature scheme and the original DSA are not fully secure against relatively weak attacks based on linear functions (Φ^{lin}-EUF-CM-RKA). It is not known whether the second DSA variant is vulnerable with respect to $\Phi^{\text{poly}(d)}$-EUF-CM-RKA. We leave this as an open problem. However, we proved that simple modifications yield schemes, the improved Schnorr signature scheme and the improved DSA scheme, which are RKA secure with respect to polynomial functions ($\Phi^{\text{poly}(d)}$-EUF-CM-RKA) in the random oracle model. The RKA security with respect to polynomial functions is proven under the d-SDL assumption. Interestingly, considering the case of $d = 1$, our results show that our improved Schnorr scheme and the improved DSA are RKA secure with respect to affine functions in the random oracle model under the ordinary DL assumption. Moreover, our simple modification of the original Schnorr scheme and the considered DSA variant does not require the public or private key from the original schemes to change, and only increases the computational cost of the signing algorithm with a single exponentiation while no other computational cost or the signature size will increase. However, the improved schemes do not address bit-flipping attacks, such as those highlighted by Bao et al. [3]. It remains future work to construct schemes which are provably secure against these attacks.

Acknowledgements. The second author is supported by JSPS KAKENHI Grant Number 15K16006.

References

1. National Institute of Standards and Technology (NIST), FIPS Publication 186: Digital Signature Standards (DSS) (1994)
2. Abdalla, M., Benhamouda, F., Passelègue, A., Paterson, K.G.: Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 77–94. Springer (2014)
3. Bao, F., Deng, R.H., Han, Y., Jeng, A.B., Narasimhalu, A.D., Ngair, T.: Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 115–124. Springer (1997)
4. Bellare, M., Cash, D., Miller, R.: Cryptography Secure against Related-Key Attacks and Tampering. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 486–503. Springer (2011)
5. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer (2003)

6. Bellare, M., Neven, G.: Multi-signatures in the plain public-Key model and a general forking lemma. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) CCS 2006. pp. 390–399. ACM (2006)
7. Bellare, M., Paterson, K.G., Thomson, S.: RKA Security beyond the Linear Barrier: IBE, Encryption and Signatures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 331–348. Springer (2012)
8. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.: High-Speed High-Security Signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer (2011)
9. Bhattacharyya, R., Roy, A.: Secure Message Authentication Against Related-Key Attack. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 305–324. Springer (2013)
10. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Jr., B.S.K. (ed.) CRYPTO '97. LNCS, vol. 1294, pp. 513–525. Springer (1997)
11. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer (2004)
12. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: Bounded Tamper Resilience: How to Go beyond the Algebraic Barrier. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 140–160. Springer (2013)
13. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: The Chaining Lemma and Its Application. In: Lehmann, A., Wolf, S. (eds.) ICITS 2015. LNCS, vol. 9063, pp. 181–196. Springer (2015)
14. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer (2003)
15. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-Malleable Codes. In: Yao, A.C. (ed.) ICS 2010. pp. 434–452. Tsinghua University Press (2010)
16. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous Non-malleable Codes. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 465–488. Springer (2014)
17. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 111–128. Springer (2014)
18. Gamal, T.E.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO '84. LNCS, vol. 196, pp. 10–18. Springer (1984)
19. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer (2004)
20. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2004. LNCS, vol. 3373, pp. 27–41. Springer (2004)
21. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2), 281–308 (1988)
22. Goyal, V., O'Neill, A., Rao, V.: Correlated-Input Secure Hash Functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer (2011)
23. Jafargholi, Z., Wichs, D.: Tamper Detection and Continuous Non-malleable Codes. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 451–480. Springer (2015)
24. Jao, D., Yoshida, K.: Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 1–16. Springer (2009)
25. Koblitz, N., Menezes, A.J.: The random oracle model: a twenty-year retrospective. Des. Codes Cryptography 77(2-3), 587–610 (2015)
26. Morita, H., Schuldt, J.C., Matsuda, T., Hanaoka, G., Iwata, T.: On the Security of the Schnorr Signature Scheme and DSA against Related-Key Attacks. In: ICISC 2015 (to appear)
27. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 387–398. Springer (1996)
28. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptology 13(3), 361–396 (2000)
29. Pointcheval, D., Vaudenay, S.: On Provable Security for Digital Signature Algorithms (1996)
30. Qin, B., Liu, S., Yuen, T.H., Deng, R.H., Chen, K.: Continuous Non-malleable Key Derivation and Its Application to Related-Key Security. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 557–578. Springer (2015)
31. Schnorr, C.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO '89. LNCS, vol. 435, pp. 239–252. Springer (1989)