Strong Known Related-Key Attacks and the Security of ECDSA

Tsz Hon Yuen(✉) and Siu-Ming Yiu

The University of Hong Kong, Pok Fu Lam, Hong Kong
{thyuen,smyiu}@cs.hku.hk

Abstract. The classical related-key attack (RKA) model fails to capture some real world systems that introduce related secret keys by design. In some blockchain applications, public keys are generated in a way that the corresponding secret keys are additively related. The difference between two secret keys are known to some third parties. In this paper, we propose the *Strong Known Related-Key Attack* (Strong KRKA) model to capture this scenario.

ECDSA has long been considered to be inferior to Schnorr signature in terms of security, in sprite of its popularity in the standardization and real world usage. In this paper we show that Schnorr signature is not secure in the Strong KRKA model. In contrast, the security of ECDSA in the Strong KRKA model can be reduced to the unforgeability of ECDSA under chosen message attack. This theoretical result gives a different view of the relative security level of ECDSA and Schnorr, since ECDSA was developed in 1992.

Keywords: Related-key attack · ECDSA · Schnorr signature · Blockchain

1 Introduction

The related-key attack (RKA) model captures real world attacks like tampering or fault injection attack. For the case of public key cryptosystem [1], it considers the security of encryption or signature with respect to a single public key. The encryption or verification algorithms is run by taking the original public key as the input. The RKA model mainly considers the attacks happened during the *run time* of the decryption or signing algorithms.

In this paper, we consider the case that related secret key is deliberately introduced to public key cryptosystem during the *design phase*. In this case, a pair of related secret keys correspond to a pair of related public keys. The relationship between secret keys is known to the adversary. As a result, the security of encryption or signature has to be considered with respect to multiple public keys. Looking ahead, we will introduce a new security model to capture this kind of public key cryptosystem. This security model is inspired by the non-hardened key derivation in Bitcoin Improvement Protocol (BIP) 32 and Bitcoin's stealth address.

J. K. Liu and X. Huang (Eds.): NSS 2019, LNCS 11928, pp. 130–145, 2019.
https://doi.org/10.1007/978-3-030-36938-5_8

BIP 32 Non-hardened Key Derivation. BIP 32 describes hierarchical deterministic wallet (HD wallet). A child key can be derived from a parent key. Suppose the parent secret key is x_0 and the parent public key is $X_0 = g^{x_0}$. There is also a *chain code* for the parent key c_0, which is a 32 bytes extended information about the key. Then to derive a non-hardened child key with index i, it computes

$$S = \mathsf{HMAC} - \mathsf{SHA512}_{c_0}(X_0||i),$$

where c_0 is used as the key of the hash function. Denote s_L as the first 32 bytes of S and s_R as the last 32 bytes of S. The child secret key $x_1 = x_0 + s_L$, the child public key $X_1 = X_0 \cdot g^{s_L}$ and the child chain code is $c_1 = s_R$.

The problem of using the standard Schnorr signature with BIP 32 non-hardened child key is that a signature from X_0 can be computed from a signature of X_1 (or vice versa). Note that the computation of s_L does not include secret key. If the chain code c_0 and the child index i is known, the adversary can compute s_L. If (R, z) is a valid Schnorr signature for X_0 (such that $g^z = RX_0^c$), then $(R, z + cs_L)$ is also a valid signature for X_1:

$$g^{z+cs_L} = RX_0^c g^{cs_L} = Rg^{c(x_0+s_L)} = RX_1^c,$$

where $c = H(R, m)$ for some message m. On the other hand, there is no known attack for ECDSA in this setting.

Bitcoin's stealth address also has a similar structure of related secret key. However, the related secret key is generated by two parties in a transaction. Details of stealth address will be discussed in Sect. 6.2. We will show the potential problem of using Schnorr signature with BIP 118 in Sect. 6.3.

1.1 Modelling Related-Key by Design

In our previous example, related-key in BIP 32 is used for digital signatures. For the ease of presentation, we mainly focus on the related-key of digital signature in this paper. In the classical RKA security model for signature [1], the adversary wins the security game if he can output a valid signature with respect to a challenge public key pk. The adversary can query the signing oracle on a message m and a function ϕ of the secret key sk (e.g., ask for a signature signed by $\phi(sk) = sk + \Delta$, where Δ is a constant chosen by the adversary).

Result 1: Strong Known RKA Model Captures Real Attack in Blockchain. In this paper, we require the following changes to the RKA security model:

1. Assume that the public key is computed from the secret key by a one-way function $\mathcal{T}$, i.e., $pk = \mathcal{T}(1^\lambda, sk)$, λ is the security parameter. Then the signing oracle on a message m_i and a function ϕ_i returns a valid signature σ_i signed by $\phi(sk_i)$; and a related public key $pk_i = \mathcal{T}(1^\lambda, \phi(sk_i))$. The adversary wins by outputting a valid signature with respect to any pk_i. This variant is called the Strong RKA security in [1].

Table 1. Difference between security models in this paper.

	(Chosen) Related Key Attack	Known Related Key Attack
Single public key	RKA [1]	KRKA
Multiple public keys	Strong RKA [1]	Strong KRKA

2. The related-key function ϕ is only known by the adversary but not chosen by the adversary. It is similar to the difference between the chosen message attack and the known message attack in digital signature. We call this variant as Known RKA (KRKA) security.

By combining the Strong RKA with Known RKA model, the Strong KRKA captures the security of signatures signed by secret keys derived by BIP 32. It is because (1) the adversary can ask for valid signatures signed by different parent and child keys and will try to forge any one of the corresponding public keys (Strong RKA), and (2) the adversary only knows that the parent and child keys are differed by s_L which is the output of a hash function, but the adversary cannot set s_L to arbitrary value (KRKA).

Since there are two modifications to the classical RKA security model, we have 4 possible security models, as shown in Table 1. Interestingly, it is mention in [1] that there is no known application-relevant attack by the Strong RKA model. In this paper, we give a concrete example that a combination of Strong RKA model and KRKA model is useful to capture the security of BIP 32 non-hardened key derivation (and stealth address). BIP 32 is widely used as HD wallet for Bitcoin as well as other cryptocurrencies, such as Ethereum.

1.2 ECDSA and Schnorr Signature in Strong KRKA Model

Our new Strong KRKA model gives us a rather suprising result on the security of ECDSA and Schnorr signature. In short, we show that Schnorr signature is not secure in the Strong KRKA model. On the other hand, we can reduce the Strong KRKA security of ECDSA to the existential unforgeability against chosen message attack (EUF-CMA) of ECDSA.

There are many discussion about whether Schnorr signature or ECDSA is a better digital signature scheme in practice. The EUF-CMA of Schnorr signature is well-understood for years [10]. On the other hand, ECDSA is known to be malleable: if (s, t) is a ECDSA signature on a message m, then $(-s, t)$, is also a valid signature on m. Therefore ECDSA is not strongly unforgeable against chosen message attack (SUF-CMA). The ECDSA malleability is one of the causes of transaction malleability in the Bitcoin system, and a number of related attacks are found [4]. Comparatively, Schnorr signature is SUF-CMA secure [8].

Many people considers that Schnorr signature is more secure than (or at least as secure as) ECDSA. The only issue hindering the use of Schnorr signature is the patent problem. Since the expiry of the patent in 2008, there are calls to change ECDSA to Schnorr signature in various systems. For example, some developers and researchers suggest to use Schnorr signature to replace ECDSA in Bitcoin. The Schnorr signature also allows batch verification and can be easily converted to multi-signatures or threshold signatures. The ETSI specification [5] mentioned a number of advantages of using (EC-)Schnorr over ECDSA, including simpler signing algorithm, easier implementation of hash function, and Schnorr's security in the random oracle model. The Schnorr signature has no identified technical drawback compared to ECDSA in [5].

Result 2: Schnorr Signature is Not as Secure as ECDSA in the Strong Known RKA Model. It is commonly believed that Schnorr signature is more secure than ECDSA. For example, Schnorr signature is strongly unforgeable [8] but ECDSA is not. For Schnorr signature, the EUF-CMA is reduced to the discrete logarithm (DL) problem in the random oracle model [10]. The EUF-CMA security of ECDSA is reduced to the DL problem in the generic group model [2,3,12], or in the bijective random oracle model [6].

It is known that both Schnorr signature and ECDSA are not EUF-CMA secure in the RKA model (known as the EUF-CM-RKA security) [9]. In this paper, we will show that the Strong Known RKA model is just enough to differentiate between Schnorr signature and ECDSA. We will demonstrate that Schnorr signature is not EUF-CM-sKRKA secure. Other the other hand, ECDSA does not have the same weakness. In fact, we are able to show that (EC)DSA is EUF-CM-sKRKA secure in the random oracle model if (EC)DSA is EUF-CMA secure. To the best of the authors' knowledge, it is the first proof that (EC)DSA is potentially more secure than Schnorr signature in a model which is weaker than some well-established security model.

2 Backgrounds

Schnorr signature, DSA and ECDSA are the most well-known discrete logarithm (DL)-based digital signature schemes. DSA and ECDSA are commonly used in various standards even though no rigorous security proofs were given when these standards are set. On the other hand, the security of the Schnorr signature is well-known under the random oracle model [10]. However, the Schnorr signature remained patented until 2008 and hence its usage is relatively limited in the industry [5].

Schnorr Signature [11]**.** In a group $\mathbb{G}$ of prime order q with generator g, a signing key x coincides with exponent, a verification key $X = g^x$, a signature on a message m is (c, s), where:

$$c = H(g^r, m), \quad s = r + cx \mod q,$$

r is a random element randomly chosen from the exponent space and H is a collision resistant hash function that maps into the exponent space. Verification works by firstly recovering $g^r = g^s / X^c$ and then checking if $c = H(g^r, m)$.

(EC)DSA. DSA was firstly specified by NIST. ECDSA was proposed in 1992 in response to the NIST request. Both DSA and ECDSA use an extra *conversion*

Table 2. Comparing ECDSA and Schnorr signature. ROM stands for random oracle model and BRO stands for bijective random oracle model.

	EUF-CMA	MU-EUF-CMA	SUF-CMA	EUF-CM-RKA	EUF-CM-sKRKA
Schnorr signature	$\surd$ (ROM [10])	$\surd$ (ROM [8])	$\surd$ (ROM [8])	$\times$ [9]	$\times$ (this paper)
ECDSA	$\surd$ (BRO [6]/ generic group model [2,3,12])	$\surd$ (generic group model [7])	$\times$	$\times$ [9]	$\surd$ (reduce to EUF-CMA in ROM, this paper)

function f to map group elements into the exponent space $\mathbb{Z}_q$. An (EC)DSA signature on a message m is (s, t), where:

$$t = f(g^r), \quad s = (H(m) + xt)/r \mod q.$$

Verification works by firstly recovering $g^r = (g^{H(m)}X^t)^{1/s}$ and then checking if $t = f(g^r)$. For DSA defined in a prime-order subgroup of the multiplicative group of some prime field $GF(p)$, the conversion function f is define as $A \mapsto (A \mod p) \mod q$. For ECDSA, it is defined on elliptic curves over some finite field $\mathbb{F} = GF(p^n)$ and its group elements are points $(x, y) \in \mathbb{F} \times \mathbb{F}$. The conversion function f for ECDSA is the mapping $A \mapsto A.x \mod q$, where $A.x$ denotes the encoding of the x-coordinate of A as an integer.

ECDSA vs Schnorr signature is shown in Table 2. For Schnorr signature, the existential unforgeability against chosen message attack (EUF-CMA) is reduced to the DL problem in the random oracle model [10]. The formal security of ECDSA is less studied than that of Schnorr signature. The EUF-CMA security of ECDSA is reduced to the DL problem in the generic group model [2,3,12], or in the bijective random oracle model [6].

The security of Schnorr signature in the multi-user setting is shown in [8]. The multi-user EUF-CMA security (MU-EUF-CMA) of ECDSA is shown in [7] using the generic group model.

3 Preliminaries

3.1 Notations

For a finite set $\mathbb{A}$, we use the symbol $a \leftarrow_s \mathbb{A}$ as the random sampling according to the uniform distribution. We also use $\leftarrow_s$ for assignments from randomized algorithms and $\leftarrow$ for deterministic algorithms. For any function $F : \mathbb{A} \rightarrow \mathbb{B}$, we write $\mathrm{Dom}(F)$ as the domain of F and $\mathrm{Rng}(F)$ as the range of F.

Let $\mathbb{G}$ be a cyclic group of prime order q, with generator g. Suppose $x \leftarrow_s \mathbb{Z}_q^*$. The discrete logarithm (DL) assumption is that given (g, g^x), no probabilistic polynomial time algorithm can output x.

3.2 Signature Schemes

A signature scheme consists of three algorithms:

- KeyGen: On input a security parameter 1^λ, it outputs a signing key sk and a verification key pk.
- Sign: On input a signing key sk and a message m, it outputs a signature σ or the failure indicator $\perp$.
- Verify: On input a verification key pk, a message m and a signature σ, it outputs 1 for acceptance or 0 for rejection.

A signature scheme is correct if for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$ and all m in the message space, $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$.

Unforgeability. The existential unforgeability under chosen message (EUF-CMA) game is defined in Algorithm 1. The game is executed with an adversary $\mathcal{A}$ by running INIT first and its output are the inputs to $\mathcal{A}$. Next, the Sign oracle queries of $\mathcal{A}$ are answered by the corresponding procedures. Finally, $\mathcal{A}$ calls FIN and terminates. Whenever the stop command is invoked, its argument is considered as the output of the game. We define the advantage of an adversary in the Game as the probability that the game outputs 1.

Algorithm 1. Game EUF-CMA.

```
1  Procedure INIT($1^\lambda$):
2      $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
3      $\mathbb{L} \leftarrow \emptyset$;
4      return $\mathsf{pk}$;
5  Procedure SIGN($m_i$):
6      $\sigma_i \leftarrow_s \mathsf{Sign}(\mathsf{sk}, m_i)$;
7      $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i)\}$;
8      return $\sigma_i$;
9  Procedure FIN($m^*, \sigma^*$):
10     if $m^* \in \mathbb{L}$ then
11         stop with 0;
12     if $\mathsf{Verify}(pk, m^*, \sigma^*) = 0$ then
13         stop with 0;
14     stop with 1;
```

Definition 1. *A signature scheme is (t, q_s, ϵ)-secure under the EUF-CMA if there is no adversary running in time t, with q_s queries to the signing oracle, has advantage larger than ϵ.*

4 RKA Security Model

The related-key attack (RKA) model is intended to capture real world attacks like tampering or fault injection attack. For example, an adversary manipulates a hardware-stored secret key by electromagnetic radiation and obtains the signature signed by the manipulated secret key.

4.1 RKA and Strong RKA Models

RKA is formalized as a security game that also allows an adversary to obtain signatures for modified keys. Denote the secret key space as $\mathcal{S}$. Thus, an adversary is allowed to query related-key deriving (RKD) functions $\phi_i : \mathcal{S} \rightarrow \mathcal{S}$ as well as messages to the signing oracle. We say that Φ is a class of RKD functions. For example, denote $\Phi^+ = \{\phi_i(x) = x + b_i : b_i \in \mathcal{S}\}$, $\Phi^* = \{\phi_i(x) = x * a_i : a_i \in \mathcal{S}\}$ and $\Phi^{\mathsf{aff}} = \{\phi_i(x) = a_i x + b_i : a_i, b_i \in \mathcal{S}\}$.

Φ-EUF-CM-RKA. [1] We recall existential unforgeability under chosen message and (chosen) RKA defined by RKD function class Φ. This security model of Φ-EUF-CM-RKA is formalized by Algorithm 2.

Algorithm 2. Game Φ-EUF-CM-RKA.

```
Procedure Init($1^\lambda$):
    Same as EUF-CMA;
Procedure Sign($m_i, \phi_i$):
    if $\phi_i \notin \{\Phi \cup$ identity map$\}$ then
        return $\perp$;
    $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_i(\mathsf{sk}), m_i)$;
    if $\phi_i$ is identity map then
        $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
    return $\sigma_i$;
Procedure Fin($m^*, \sigma^*$):
    Same as EUF-CMA;
```

Φ-EUF-CM-sRKA. [1] Bellare *et al.* extends the RKA security for *separable* signature. *Separable* signature means that for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, there exists a deterministic algorithm $\mathcal{T}$ such that $\mathsf{pk}' \leftarrow \mathcal{T}(1^\lambda, \mathsf{sk})$ and the distribution of pk is indistinguishable to pk'. This security model of Φ-EUF-CM-sRKA (Strong RKA) is formalized by Algorithm 3. The difference with the standard RKA model is highlighted.

Definition 2. *A signature scheme is (t, q_s, ϵ)-secure under the Φ-EUF-CM-RKA (resp. Φ-EUF-CM-sRKA) if there is no adversary running in time t, with q_s queries to the signing oracle, has advantage larger than ϵ in Game Φ-EUF-CM-RKA (resp. Φ-EUF-CM-sRKA).*

4.2 (Strong) Known-RKA Security

We give the new security model of existential unforgeability under chosen message and known RKA defined by RKD function class Φ. Recall the difference between the known message attack (KMA) and CMA for signature is that the adversary only knows the message-signature pairs in KMA, while the adversary is able to specify the message for the signing oracle in CMA. In the new known

Algorithm 3. Game Φ-EUF-CM-sRKA.

```
1  Procedure Init($1^\lambda$):
2      Same as EUF-CMA;
3  Procedure Sign($m_i, \phi_i$):
4      if $\phi_i \notin \{\Phi \cup$ identity map$\}$ then
5          return $\perp$;
6      $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_i(\mathsf{sk}), m_i)$;
7      $\mathsf{pk}_i \leftarrow \mathcal{T}(1^\lambda, \phi_i(\mathsf{sk}))$;
8      $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
9      return $(\mathsf{pk}_i, \sigma_i)$;
10 Procedure Fin($i^*, m^*, \sigma^*$):
11     if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ then
12         stop with 0;
13     if $\mathsf{Verify}(\mathsf{pk}_{i^*}, m^*, \sigma^*) = 0$ then
14         stop with 0;
15     stop with 1;
```

RKA model, the adversary only knows the RKD functions that he can query for the signing oracle (yet the adversary can still choose the message). It is weaker than the classical RKA model, in which the adversary can set the RKD functions to any function in Φ.

This security model of Φ-EUF-CM-KRKA is formalized by Algorithm 4. The difference with the standard RKA model is highlighted.

Algorithm 4. Game Φ-EUF-CM-KRKA.

```
1  Procedure Init($1^\lambda$):
2      $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
3      $\mathbb{L} \leftarrow \emptyset$;
4      $\phi_0 \leftarrow$ identity map;
5      $\mathbb{S} \leftarrow \{\phi_0\}$;
6      for $j \leftarrow 1$ to $q_s$ do
7          $\phi_j \leftarrow_s \Phi$;
8          $\mathbb{S} \leftarrow \mathbb{S} \cup \{\phi_j\}$;
9      return $\mathsf{pk}$, $\mathbb{S}$;
10 Procedure Sign($m_i$, $j$):
11     if $j \notin [0, q_s]$ then
12         return $\perp$;
13     $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_j(\mathsf{sk}), m_i)$;
14     if $j = 0$ then
15         $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
16     return $\sigma_i$;
17 Procedure Fin($m^*, \sigma^*$):
18     Same as EUF-CMA;
```

Finally, we give the combined security model of Strong Known RKA model formalized by Algorithm 5. The difference with the standard RKA model is highlighted.

Definition 3. *A signature scheme is (t, q_s, ϵ)-secure under the Φ-EUF-CM-KRKA (resp. Φ-EUF-CM-sKRKA) if there is no adversary running in time t, with q_s queries to the signing oracle, has advantage larger than ϵ in Game Φ-EUF-CM-KRKA (resp. Φ-EUF-CM-sKRKA).*

Algorithm 5. Game Φ-EUF-CM-sKRKA.

```
Procedure INIT($1^\lambda$):
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
    $\mathbb{L} \leftarrow \emptyset$;
    $\phi_0 \leftarrow$ identity map ;
    $\mathbb{S} \leftarrow \{\phi_0\}$ ;
    for $j \leftarrow 1$ to $q_s$ do
        $\phi_j \leftarrow_s \Phi$ ;
        $\mathbb{S} \leftarrow \mathbb{S} \cup \{\phi_j\}$ ;
    return pk, $\mathbb{S}$ ;
Procedure SIGN($m_i$, $j$ ):
    if $j \notin [0, q_s]$ then
        return $\perp$;
    $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_j(\mathsf{sk}), m_i)$;
    $\mathsf{pk}_i \leftarrow \mathcal{T}(1^\lambda, \phi_j(\mathsf{sk}))$ ;
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$ ;
    return $(\mathsf{pk}_i, \sigma_i)$ ;
Procedure FIN($i^*, m^*, \sigma^*$) :
    if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ then
        stop with 0;
    if $\mathsf{Verify}(\mathsf{pk}_{i^*}, m^*, \sigma^*) = 0$ then
        stop with 0;
    stop with 1;
```

4.3 Relationship Between Models

We summarize the relationship between different security models in Table 3 with EUF-CMA. We also include the strong unforgeability (SUF-CMF) and multi-user security (MU-EUF-CMA) for completeness.

According to the definition of the security models, it is obvious that the RKA model is stronger than KRKA model (chosen relation vs. known relation). Similarly, the Strong RKA model is stronger than Strong KRKA model.

On the other hand, the Strong RKA model is stronger than the RKA model (forgery on multiple public keys vs. forgery on a single public key). Similarly, the Strong KRKA model is stronger than KRKA model.

There is no straightforward relationship between the RKA model and the Strong KRKA model. It is known that ECDSA is not RKA secure [9]. In the next section, we will show that ECDSA is secure in the Strong KRKA model if ECDSA is EUF-CMA secure. We leave the relationship between the RKA model and the Strong KRKA model as an interesting open problem.

Table 3. Relationship between different security models. Model A $\Rightarrow$ Model B means that Model A is weaker than Model B. Model A $\rightarrow$ Model B means that there exists a scheme secure in Model A but not secure in Model B. The grey box indicates the major work of this paper.

$$\begin{tikzcd}
\text{EUF-CM-sRKA} & \text{EUF-CM-sKRKA} \arrow[l, Leftarrow] \arrow[dl] & \text{MU-EUF-CMA} & \\
\text{EUF-CM-RKA} \arrow[u, Rightarrow] & \text{EUF-CM-KRKA} \arrow[l, Leftarrow] \arrow[u, Rightarrow] & \text{EUF-CMA} \arrow[l, Leftarrow] \arrow[u, Rightarrow] \arrow[r, Rightarrow] & \text{SUF-CMA}
\end{tikzcd}$$

5 Security of Schnorr Signature and ECDSA

It is known that both Schnorr signature and DSA are not secure in the Φ^{aff}-EUF-CM-RKA model [9]. It is straightforward to see that ECDSA is also not secure in the Φ^{aff}-EUF-CM-RKA model. In this section, we show that (EC)DSA is secure against our Strong Known RKA Model, but Schnorr signature is not. Therefore, it gives an important separation between the security between these two schemes.

5.1 Insecurity of Schnorr Signature in (Strong) KRKA Model

We show that the Schnorr signature scheme is not Known RKA secure with respect to additive functions by providing a simple and efficient attack. This additive relation between secret keys is realistic in the real world, such as BIP 32. In later section, we will also show that by using stealth address in Bitcoin, multiple secret keys of the same user are related additively.

According to the security model, an adversary $\mathcal{A}$ is given $\delta_1, \ldots, \delta_n$ such that $\phi_i(\mathsf{sk}) = \mathsf{sk} + \delta_i$ for $i \in [1, n]$. Then $\mathcal{A}$ queries the RKA signing oracle with input (m^*, ϕ_1) for some random message m^*. The oracle returns the signature (z, c) such that:

$$R = g^z g^{-c(\mathsf{sk}+\delta_1)}, \quad c = H(R, m^*).$$

Then $\mathcal{A}$ returns $(z - c\delta_1, c)$ as an forgery for the message m^*. Hence Schnorr signature scheme is not Known RKA secure. By similar argument, we can see that Schnorr signature is not Strong KRKA secure.

5.2 Security of (EC)DSA in Strong KRKA Model

We first show that the Φ^{aff}-EUF-CM-sKRKA security of (EC)DSA can be reduced to the EUF-CMA security, under the ROM model. The KRKA security of (EC)DSA follows from the Strong KRKA security.

Security of Φ^{aff}-EUF-CM-sKRKA. We prove that (EC)DSA is secure in the Φ^{aff}-EUF-CM-sKRKA model. The class of additive RKD functions Φ^{aff} captures the case that the secret keys are linearly related, e.g., $\phi(\mathsf{sk}) = a \cdot \mathsf{sk} + b$ for some $a, b \in \mathbb{Z}_q$. Therefore, the Φ^{aff}-EUF-CM-sKRKA model already captures the attacks in the Φ^{+}-EUF-CM-sKRKA model.

The proof differs from the proof of EUF-CMA in [6] in a few ways. Firstly, the simulation of the signing oracle is modified to capture the signature with respect to the class of RKD functions Φ^{aff}. Secondly, the extraction of secret key in the proof of EUF-CMA in [6] uses the simulation transcript of a past signing oracle query, and it requires the collision resistant property of the hash function H. However, the same argument no longer holds if the past signing oracle query includes RKD functions. The collision resistant property is not enough. We discover that the security can be shown alternatively if we use the random oracle model for H. As a result, we also have to add the relevant simulation of the random oracle model and make sure that it is consistent with the rest of the proof.

Algorithm 6. Game 0 is the Φ^{aff}-EUF-CM-sKRKA for (EC)DSA, in the random oracle model.

```
Procedure Init:
    pick $H : \{0,1\}^* \to \mathbb{Z}_q$;
    $x \leftarrow_s \mathbb{Z}_q^*$; $X \leftarrow g^x$;
    $\mathbb{L} \leftarrow \emptyset$;
    $a_0 \leftarrow 1, b_0 \leftarrow 0$;
    $\mathbb{S} \leftarrow \{(a_0, b_0)\}$;
    for $j \leftarrow 1$ to $q_s$ do
        $\phi_j(x) := a_j x + b_j \leftarrow_s \Phi^{\mathrm{aff}}$;
        $\mathbb{S} \leftarrow \mathbb{S} \cup \{(a_j, b_j)\}$;
    return $X, \mathbb{S}$;
Procedure Sign($m_i, j$):
    $r_i \leftarrow_s \mathbb{Z}_q$; $R_i \leftarrow g^{r_i}$;
    if $R_i = 1$ then
        return $\perp$;
    $t_i \leftarrow f(R_i)$;
    if $t_i = 0$ then
        return $\perp$;
    $h_i \leftarrow H(m_i)$;
    $\mathsf{sk}_i \leftarrow a_j x + b_j$ // $j \in [0, q_s]$
    $\mathsf{pk}_i \leftarrow g^{\mathsf{sk}_i}$;
    $u_i \leftarrow h_i + \mathsf{sk}_i t_i$;
    if $u_i = 0$ then
        return $\perp$;
    $s_i \leftarrow u_i / r_i$;
    $\sigma_i \leftarrow (s_i, t_i)$;
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
    return $(\mathsf{pk}_i, \sigma_i)$;
Procedure RO($m$):
    return $H(m)$;
Procedure Fin($i^*, m^*, (s^*, t^*)$):
    if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ then
        stop with 0;
    if $s^* = 0$ or $t^* = 0$ then
        stop with 0;
    $h^* \leftarrow H(m^*)$;
    $U^* \leftarrow g^{h^*} (X^{a_{i^*}} g^{b_{i^*}})^{t^*}$ // $i^* \in [0, q_s]$
    if $U^* = 1$ then
        stop with 0;
    $R^* \leftarrow (U^*)^{1/s^*}$;
    if $t^* \neq f(R^*)$ then
        stop with 0;
    stop with 1;
```

Theorem 1. *Let $\mathcal{A}$ be an adversary that (τ, q_s, ϵ)-breaks the Φ^{aff}-EUF-CM-sKRKA security of (EC)DSA, with q_H random oracle queries. Then, there exists an adversary $\mathcal{A}_{\mathrm{CMA}}$ that $(\tau_{\mathrm{CMA}}, q_s, \epsilon_{\mathrm{CMA}})$-breaks the EUF-CMA security of (EC)DSA, where:*

$$\epsilon \leq (q_s + 1)(\epsilon_{\mathrm{CMA}} + \frac{q_s q_H}{q}), \quad \tau_{\mathrm{CMA}} = \tau + O(q_s)\tau_e,$$

where τ_e is the time of exponentiation in $\mathbb{G}$.

Proof. The security is shown by a game-hopping proof. We define $\mathrm{Adv}_{\mathcal{A}_i}(1^\lambda)$ as the advantage of the adversary $\mathcal{A}$ in Game i, with security parameter λ. We omit the security parameter for simplicity.

– **Game 0** in Algorithm 6 gives the complete EUF-CM-sKRKA for (EC)DSA. The random oracle is provided by RO. Therefore, $\epsilon = \mathrm{Adv}_{\mathcal{A}_0}$.
– **Game 1** in Algorithm 7 is modified from Game 0 that the hash function H is now replaced by sampling. By the random oracle model, we have $\mathrm{Adv}_{\mathcal{A}_0} = \mathrm{Adv}_{\mathcal{A}_1}$.

Algorithm 7. Game 1 is the same as Game 0 except the Procedure INIT and RO (highlighted in gray box).

```
1  Procedure INIT:
2      $H^O \leftarrow \emptyset$;
3      $x \leftarrow_s \mathbb{Z}_q^*$; $X \leftarrow g^x$;
4      $\mathbb{L} \leftarrow \emptyset$;
5      $a_0 \leftarrow 1, b_0 \leftarrow 0$;
6      $\mathbb{S} \leftarrow \{(a_0, b_0)\}$;
7      for $j \leftarrow 1$ to $q_s$ do
8          $\phi_j(x) := a_j x + b_j \leftarrow_s \Phi^{\text{aff}}$;
9          $\mathbb{S} \leftarrow \mathbb{S} \cup \{(a_j, b_j)\}$;
10     return $X, \mathbb{S}$;

11 Procedure RO($m$):
           if $(m, h) \in H^O$ then
               return $h$;
           $h \leftarrow_s \mathbb{Z}_q \backslash \text{Rng}(H^O)$;
           $H^O \leftarrow H^O \cup \{(m, h)\}$;
12
13     return $h$;
```

– Finally, Algorithm 8 shows how to build an adversary $\mathcal{A}_{\text{CMA}}$ to break the EUF-CMA security of (EC)DSA, by running as the challenger of Game 1 and making use of the output from $\mathcal{A}_1$. $\mathcal{A}_{\text{CMA}}$ uses the output of INIT_{CMA} from its challenger (of the EUF-CMA security) to simulate the challenger of Game 1 in line 9. This change is indistinguishable to $\mathcal{A}_1$. The SIGN procedure in Algorithm 8 is simulated by using the signing oracle output from the challenger of the EUF-CMA security. Finally, the validation of the output from $\mathcal{A}_1$ is same as except line 38, 39 and 51. We want to show that $\text{Adv}_{\mathcal{A}_1} \leq (q_s + 1)(\text{Adv}_{\mathcal{A}_{\text{CMA}}} + q_s q_H / q)$.

We can see that the signing oracle output is correct by running the verification of (s_i, t_i) against the related key pk_j:

$$\begin{aligned} g^{r_i} &= g^{\frac{H(m_i)}{s_i}} (X^{a_j} g^{b_j})^{\frac{t_i}{s_i}} = g^{\frac{a_{j*} H(m_i)}{s' a_j}} (X^{a_j} g^{b_j})^{\frac{a_{j*} t_i}{s' a_j}} \\ &= g^{\frac{(a_{j*} b_j) t_i + a_{j*} H(m_i)}{s' a_j}} (X^{a_{j*}})^{\frac{t_i}{s'}} = g^{\frac{(a_{j*} b_j - a_j b_{j*}) t_i + a_{j*} H(m_i)}{s' a_j}} (X^{a_{j*}} g^{b_{j*}})^{\frac{t_i}{s'}} \\ &= g^{\frac{H(m_i')}{s'}} (X^{a_{j*}} g^{b_{j*}})^{\frac{t'}{s'}} = g^{\frac{H(m_i')}{s'}} X'^{\frac{t'}{s'}} = g^{r'}. \end{aligned}$$

Then we have $f(g^{r_i}) = f(g^{r'}) = t' = t_i$. Hence (s_i, t_i) is a valid signature with respect to pk_i.

When $\mathcal{A}_1$ outputs a valid forgery $(i^*, m^*, (s^*, t^*))$, line 51 of Algorithm 8 is reached if $i^* = j^*$. It happens with probability $\frac{1}{q_s+1}$. By the checking of line 40, m^* was not queried to SIGN_{CMA} in line 15. If m^* was also not queried to SIGN_{CMA} in line 20, then $\mathcal{A}_{\text{CMA}}$ wins by line 51.

We now show that m^* was not queried to SIGN_{CMA} in line 20. Observe that in line 19, m_i' is randomly chosen from the message space and it is not given to the $\mathcal{A}_1$. $\mathcal{A}_1$ can only calculate $H(m_i')$ as in line 21. By the random oracle model, $\mathcal{A}_1$ cannot find some m_i' and use it as m^* with probability more than $\frac{q_s q_H}{q}$. Therefore, we have $\text{Adv}_{\mathcal{A}_1} \leq (q_s + 1)(\epsilon_{\text{CMA}} + q_s q_H / q)$.

Algorithm 8. The construction of adversary $\mathcal{A}_{\mathrm{CMA}}$ against EUF-CMA, using the adversary $\mathcal{A}_1$ for Game 1. (Interaction with the challenger of EUF-CMA is highlighted in the gray box).

```
Procedure Init($1^\lambda$):
    $H^O \leftarrow \emptyset$, $\mathbb{L} \leftarrow \emptyset$;
    $j^* \leftarrow_s [0, q_s]$;
    $a_0 \leftarrow 1, b_0 \leftarrow 0$;
    $\mathbb{S} \leftarrow \{(a_0, b_0)\}$;
    for $j \leftarrow 1$ to $q_s$ do
        $\phi_j(x) := a_j x + b_j \leftarrow_s \Phi^{\mathrm{aff}}$;
        $\mathbb{S} \leftarrow \mathbb{S} \cup \{(a_j, b_j)\}$;
    $X' \leftarrow \mathrm{Init}_{\mathrm{CMA}}(1^\lambda)$ ;
    $X \leftarrow (X' g^{-b_{j^*}})^{1/a_{j^*}}$;
    return $X, \mathbb{S}$;
Procedure Sign($m_i, j$):
    if $j = j^*$ then
        $\mathsf{pk}_i \leftarrow X'$;
        $\sigma_i \leftarrow \mathrm{Sign}_{\mathrm{CMA}}(m_i)$ ;
    else
        isNewH $\leftarrow$ false;
        while isNewH = false do
            $m'_i \leftarrow_s \mathcal{M}$;
            $(s', t') \leftarrow \mathrm{Sign}_{\mathrm{CMA}}(m'_i)$ ;
            $h_i \leftarrow \frac{(a_{j^*} b_j - a_j b_{j^*}) t_i + a_{j^*} H(m_i)}{a_j}$;
            if $(\cdot, h_i) \notin H^O$ then
                $H^O \leftarrow H^O \cup \{(m'_i, h_i)\}$;
                isNewH $\leftarrow$ true;
        $t_i \leftarrow t'$;
        $s_i \leftarrow \frac{s' a_j}{a_{j^*}}$;
        $\mathsf{pk}_i \leftarrow X^{a_j} g^{b_j}$;
        $\sigma_i \leftarrow (s_i, t_i)$;
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
    return $(\mathsf{pk}_i, \sigma_i)$;
Procedure RO($m$):
    if $(m, h) \in H^O$ then
        return $h$;
    $h \leftarrow_s \mathbb{Z}_q \backslash \mathrm{Rng}(H^O)$;
    $H^O \leftarrow H^O \cup \{(m, h)\}$;
    return $h$;
Procedure Fin($i^*, m^*, (s^*, t^*)$):
    if $i^* \neq j^*$ then
        stop with 0;
    // $\mathsf{pk}_{i^*} = \mathsf{pk}_{j^*} = X'$
    if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ then
        stop with 0;
    if $s^* = 0$ or $t^* = 0$ then
        stop with 0;
    $h^* \leftarrow H(m^*)$;
    $U^* \leftarrow g^{h^*} X'^{t^*}$;
    if $U^* = 1$ then
        stop with 0;
    $R^* \leftarrow (U^*)^{1/s^*}$;
    if $t^* \neq f(R^*)$ then
        stop with 0;
    run $\mathrm{Fin}_{\mathrm{CMA}}(m^*, (s^*, t^*))$;
```

To conclude, we have $\epsilon \le (q_s + 1)(\epsilon_{\mathrm{CMA}} + q_s q_H / q)$. Finally, the running time is dominated by $O(q_s)$ exponentiation in the signing oracle queries. $\square$

The security of (EC)DSA under the EUF-CMA attack can be reduced to the DL problem in the bijective random oracle model [6] or in the generic group model [2,3,12].

6 Strong KRKA Attack in the Bitcoin System

The Strong KRKA security model not only captures the tampering attack, it can also be used to capture the security of some variants in Bitcoin system, such as BIP 32 non-hardened key derivation and *stealth address*. Combining with the result of the previous section, ECDSA is secure with the use of these Bitcoin variants, while the standard Schnorr signature is not secure.

6.1 BIP 32 Non-hardened Key Derivation

BIP 32 describes how a hierarchical deterministic wallet (HD wallet) generate keys from a single seed. We have described how non-hardened secret keys are derived in Sect. 1 according to BIP 32. Every parent secret key is linearly related to its child secret key by design. Therefore, all non-hardened secret keys derived in BIP 32 are linearly related. Note that the adversary can only know the difference between secret keys, but he cannot set it to arbitrary value by the security of the HMAC-SHA512 function.

BIP 32 standardizes the key generation process in HD wallet and it does not consider what message to be signed with these keys. Strong KRKA attack is dangerous in the setting that the message to be signed is not related to the signer public key/address.

6.2 Stealth Address

The idea of stealth address was firstly proposed in a Bitcoin forum[1]. It allows the recipient to remain anonymous, even after sharing his stealth address. The most common version of stealth address was proposed by CryptoNote in 2013[2]. Stealth address was implemented for Bitcoin and is widely used as a cornerstone to many anonymous cryptocurrencies, such as Monero.

The stealth address is described as follows. Suppose that the recipient Bob has a long term secret key $(a, b) \in \mathbb{Z}_p^2$ and public key $(A = g^a, B = g^b) \in \mathbb{G}^2$. The sender Alice picks a random number $r \leftarrow_s \mathbb{Z}_q$ and puts $R = g^r$ in the transaction. The one-time recipient address is (the hash of) $Y = A \cdot g^{H'(B^r)}$, where $H' : \mathbb{G} \rightarrow \mathbb{Z}_q$ is a collision resistant hash function. Bob can use b (which is known as the viewing key) to check if he is the intended recipient of the transaction with (R, Y) by checking if $Y = A \cdot g^{H'(R^b)}$. Bob's one-time secret key corresponding to Y is $a + H(R^b)$.

Related-Key Attack for Stealth Address. If Alice sends some Bitcoin to Bob in two different transactions, then Bob's one-time secret keys are $y_1 = a + H(R_1^b)$ and $y_2 = a + H(R_2^b)$ respectively. Therefore, y_1 and y_2 are linearly related: $\delta = y_1 - y_2 = H(R_1^b) - H(R_2^b)$. If the Bitcoin system uses the Schnorr signature, there is potential attack when Bob uses y_2 to output (z, c) for a

[1] https://bitcointalk.org/index.php?topic=5965.0.
[2] CryptoNote v 2.0 Whitepaper. https://cryptonote.org/whitepaper.pdf.

message m. In this case, Alice with the knowledge of $\delta = H(B^{r_1}) - H(B^{r_2})$, can output a signature $(z' = z + c\delta, c)$ for the same message m. We can see that it is a valid signature for Y_1:

$$g^{z'} = g^{z+c\delta} = (RY_2^c) \cdot g^{c\delta} = Rg^{y_2 c + c\delta} = RY_1^c, \quad c = H(R, m).$$

This attack can be launched simply by the knowledge of the difference δ of the two secret keys. This attack is captured in the Strong KRKA model, by signing oracle queries with the addition function.

Note that for the case of stealth address, the adversary can only know the difference δ, but he cannot set δ to arbitrary value since $\delta = H(B^{r_1}) - H(B^{r_2})$. He cannot find such r_1 and r_2 satisfying this relation, assuming the pseudo-randomness of the output of H. Therefore, the attack is precisely captured by the Strong KRKA model, but not the classical RKA model.

We have shown that ECDSA is secure in the Strong KRKA model for affine functions. Therefore, ECDSA is not affected by the use of stealth address. There is potential threat of using stealth address with standard Schnorr signature.

6.3 BIP 118 SIGHASH_NOINPUT

We note that in the normal use case of Bitcoin transaction with BIP 32 key/stealth address, the signer's address (=hash of his public key) is included in the message. The Strong KRKA attack on the standard Schnorr signature does not apply to this use case. However, we cannot guarantee what message will be signed in the future update of the Bitcoin protocol.

BIP 118 is useful for building Lightning Network channels to increase the scalability of Bitcoin system and to enable micropayment over Bitcoin. In particular, a new signing flag SIGHASH_NOINPUT is proposed, such that the signature does not commit to any of the inputs. All fields related to the input address/sequence/outpoint are replaced with string of 0s. Therefore, using standard Schnorr signature with BIP 32 non-hardened key/stealth address and BIP 118 SIGHASH_NOINPUT are insecure.

7 Conclusion

In this paper, we showed that, for the first time, ECDSA is potentially more secure than the standard Schnorr signature in the Strong Known RKA model. The Strong Known RKA model captures the attack on BIP 32 and stealth address in Bitcoin and other cryptocurrencies. Therefore if Schnorr signature or other DL-type signatures (including multi-signatures, aggregate signatures, threshold signatures, etc.) are proposed in the blockchain system, it is highly recommended to evaluate their Strong Known RKA security.

Acknowledgment. This project is partially supported by the grant of the University of Hong Kong (Project No. 201901159007), and the CRF grant (CityU: C1008-16G) of the Government of HKSAR, Hong Kong.

References

1. 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, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_26
2. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptography **35**(1), 119–152 (2005)
3. Brown, D.R.L.: On the provable security of ECDSA. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, pp. 21–40. Cambridge University Press, Cambridge (2005)
4. Decker, C., Wattenhofer, R.: Bitcoin transaction malleability and MtGox. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 313–326. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_18
5. ETSI: Electronic signatures and infrastructures (ESI); cryptographic suites. ETSI Technical Specification 119 312 (v1.2.1) (2017)
6. Fersch, M., Kiltz, E., Poettering, B.: On the provable security of (EC)DSA signatures. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) CCS 2016, pp. 1651–1662. ACM (2016)
7. Galbraith, S.D., Malone-Lee, J., Smart, N.P.: Public key signatures in the multiuser setting. Inf. Process. Lett. **83**(5), 263–266 (2002)
8. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 33–61. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_2
9. Morita, H., Schuldt, J.C.N., Matsuda, T., Hanaoka, G., Iwata, T.: On the security of the Schnorr signature scheme and DSA against related-key attacks. In: Kwon, S., Yun, A. (eds.) ICISC 2015. LNCS, vol. 9558, pp. 20–35. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30840-1_2
10. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_33
11. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
12. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in applying proof methodologies to signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 93–110. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_7