2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00038

Security on SM2 and GOST Signatures against Related Key Attacks

Handong Cui, Xianrui Qin, Cailing Cai, Tsz Hon Yuen
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong SAR
{hdcui,xrqin,clcai,thyuen}@cs.hku.hk

***Abstract*—The US Standard (EC)DSA is currently almost the most popular digital signature scheme. Chinese and Russian governments also proposed their counterparts: SM2 and GOST R 34.10 (GOST). Nowadays, there are already many industrial applications supporting SM2 and GOST digital signatures. Unfortunately, the existing analyses for SM2 and GOST are rather limited when compared to ECDSA.**

This paper focuses on the security of SM2 and GOST from the viewpoints of RKA security (related-key attack) and sKRKA security (strong known related key attack). RKA captures the real attacks of tampering and fault injection in hardware-stored secret keys. sKRKA, a recently proposed security model modified from RKA, captures the real attacks in the *BIP-32 HD wallet* and the *stealth address* used in Monero. It was proved that ECDSA is insecure in the RKA model (ICISC 2015) and but secure in the sKRKA model (NSS 2019).

In this work, we proved that GOST is insecure in both RKA and skRKA models, but SM2 is secure in both RKA and sKRKA models. This result well differentiates the security of ECDSA, SM2 and GOST, and demonstrates that Chinese SM2 is capable to construct secure cryptocurrency systems using *BIP-32 HD wallet* or *stealth address*, as secure as ECDSA, but outperforms ECDSA in resisting tampering or fault injection attacks.

***Index Terms*—SM2, GOST, related-key attack, strong known related key attack**

I. Introduction

The birth of Bitcoin [43], the first blockchain adopting ECDSA as the underlying digital signature scheme, leads to the prosperity of ECDSA nowadays. In the past decade, ECDSA drew a broad interest from researchers, especially on its security proofs [27], [28], [42], [52] and constructing threshold signature [22], [23], [25], [29], [31], [32], [39]. ECDSA has almost become the most widely used digital signature in industry.

In 2010, the Chinese state cryptography administration issued an SM2 digital signature based on EC (Elliptic Curve) for commercial use, as an alternative for ECDSA, which was later adopted as an international standard in ISO/IEC 11889-1 [10]. Up to now, SM2 has been applied in influential blockchain platforms like Huawei blockchain [40] and FISCO-BCOS blockchain [5], and will be supported by Hyperledger Fabric [6], [7]. There are also other industrial or financial applications using SM2 [1], [3], [12], [24]. In March 2021, ShangMi (SM) cipher suites for TLS 1.3 was formally issued in IETF RFC-8998 standard [9], including SM2 signature and other SM series of schemes, which will further nourish the development of applications supporting SM algorithms.

Russian Federal also issued an ECDSA counterpart GOST R 34.10 adopted in IETF RFC-7091 standard [8] (denoted by GOST for simplicity in the remaining part) which overcomes the *malleability* and *duplicate signature* problems of ECDSA [44]. GOST is applied in softwares like CryptoArm [2] and is possible to be supported by Mozilla Firefox [11]. There is also a separate implementation of Ethereum using GOST [4].

A. Similarities and Differences

(EC)DSA, SM2 and GOST are all rooted in the ElGamal type signature (1984) [26] and thus share many similarities from each other. All of them work in a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, contain a *conversion function* f mapping signed message into $\mathbb{Z}_q$ and a *hash function* H mapping group element into $\mathbb{Z}_q$, use the secret/public key pair of $\langle x, g^x \rangle$. $\mathbb{G}$ can be either a prime-order q subgroup of the multiplicative group in a field $\mathrm{GF}(p)$ or a prime-order q subgroup of an elliptic curve (EC) over some field $\mathrm{GF}(p^n)$. For the former setting, f is defined as $A \mapsto (A \mod p) \mod q$ where A is a group element in $\mathbb{G}$. For the latter EC setting, f is defined as the encoding of the x-coordinate of an EC point A, usually denoted by $A.x$. Highly abstractive schemes called *GenericElGamal* in the full version of [27] and *GenElGamal* in [41, Sec. 11] [28, Def. 4], both of which cover ECDSA, SM2 and GOST by setting different parameters. For differences, we note that SM2 and GOST have their own recommended hash functions respectively: SM3 and GOST R 34.11. Besides, for SM2, when computing the hash of the signed message, the message should be concatenated with the user's identity information, the public key and EC parameter, which is different from ECDSA and GOST.

B. Prior Security Analyses

ECDSA is somehow well-studied in some very idealized security models like the existential unforgeability proved in bijective random oracle model (BRO) [27] and in generic group model (GGM) [19], [21], [48], and the security against (strong known) related key attacks (sKRKA/RKA) analyzed in [42], [52], although there is no security proof in the standard model or the broadly accepted random oracle model (ROM) like Schnorr signature [47]. Up to now, however,

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00038

TABLE I: Comparison with different signatures. ROM/BRO/GGM stands for random oracle model / bijective random oracle model / generic group model.

	EUF-CMA	EUF-CM-RKA	EUF-CM-sKRKA
Schnorr	$\surd$ (ROM [45])	$\times$ [42] $\surd$ in its variant. [42]	$\times$ [52]
ECDSA	$\surd$ (BRO [27]/ GGM [19], [21], [48])	$\times$ [42] $\surd$ in its variant. [42]	$\surd$ (reduce to EUF-CMA in ROM, [52])
GOST	$\surd$ BRO, full version of [27]	$\times$ **(Our Result)**	$\times$ **(Our Result)**
SM2	$\surd$ BRO, full version of [27] $\surd$ GGM [54]	$\surd$ (reduce to EUF-CMA in GGM, **Our Result**)	$\surd$ (reduce to EUF-CMA in GGM, **Our Result**)

rigorous security analyses for SM2 and GOST are rather limited compared to ECDSA. To the best of our knowledge, the first security evaluation of SM2 is in [54] which proved the *existential unforgeability under adaptvively chosen message attack* (EUF-CMA) in GGM; and the security against *generalized key substitution attacks* (KSA) assuming H and f are non-programmable random oracles (NPROs). The existential unforgeability of *GenericElGamal* framework, which covers both GOST and SM2, is proved in the full version of [27] using BRO. The one-per-message unforgeability of both SM2 and GOST, which is a weaker security model than existential unforgeability, is proved in the ROM by a *GenElGamal* framework. To narrow the research gap, in this work, we consider the security against related-key attack (RKA) and strong known related key attack (sKRKA) for both SM2 and GOST. We summarize the major existing security analyses and our results in Table I.

C. Related Key Attack

As a special side-channel attack [33], related-key attack (RKA) means that an adversary can observe the outcome of a cryptosystem under an altered key (or called *related key*) and then breaks the system. The earliest discussion on RKA mainly focused on the security of block ciphers [17], [38] and pseudorandom functions (PRFs) [16], [34]. The RKA study was later expanded to broader views like cryptographic permutation [14]; semantic secure encryption [13]; public-key encryption, signatures and identity-based encryption [15]. RKA capture practical attacks since the adversary can alter the secret key of a cryptosystem by fault injection or tampering [18], [20] and the secret key can be a personal decryption key, signing key for certificate authority (CA) or SSL server, master key for IBE system or hardware-stored secret key for cryptocurrency wallet. In the formalization of RKA security of a signature scheme in [15], the adversary is allowed to query a *related-key deriving* (RKD) function ϕ and a message to the signing oracle, and obtain the signature of the queried message under the signing key $\phi(\mathsf{sk})$ where sk is the original signing key. The RKA security requires that the adversary outputs a valid signature of an unqueried message under sk with negligible probability. Yet, there are no analyses for the RKA security of SM2 and GOST.

D. Strong Known Related Key Attack

Strong known related-key attack (sKRKA) is proposed in [52] which is modified from the above mentioned RKA security model, which restricts that the adversary cannot choose freely but can know the RKD function ϕ used in the query phase and strengthen the winning condition by requiring that the adversary breaks the sKRKA security if and only if he outputs a valid signature of an unqueried message under any public key corresponding to $\phi(\mathsf{sk})$. This sKRKA model is meaningful since it captures the real attacks in *Bitcoin Improvement Protocol (BIP) 32 HD wallet* [50] using Schnorr signature and *stealth address* [49] implemented in Bitcoin and used in Monero. In [52], the authors provides efficient attack under sKRKA model to Schnorr and proved that ECDSA is sKRKA secure, which first demonstrates that ECDSA is somewhat more secure than Schnorr. Thus, ECDSA is a good alternative to Schnorr in the *BIP-32 HD wallet* and is secure enough to be leveraged in blockchain systems supporting *stealth address*. Yet, there are no sKRKA security analyses on SM2 and GOST, although SM2 is encouraged by the Chinese government to replace ECDSA in commercial systems and has been utilized in the Chinese financial blockchain FISCO-BCOS. To construct new blockchain systems supporting anonymous transaction like Monero or building cryptocurrency wallet like *BIP-32*, using SM2 or GOST as the underlying signature scheme, we cannot circumvent this sKRKA security issue.

E. Contributions

In this work, we differentiate the security of Schnorr, ECDSA, SM2 and GOST in the aspects of RKA and sKRKA security, both of which security models capture real attacks. Our results also provide valuable advice for the selection of signature scheme when constructing blockchain applications with different features. The major contributions are threefold:

- We provide two efficient attacks for GOST R 34.10 signature, respectively breaking its RKA and sKRKA security.
- We prove that SM2 is secure in both RKA and sKRKA models in ROM, which demonstrates that: SM2 is more secure than GOST, Schnorr, ECDSA (Schnorr and ECDSA are not RKA secure by [42]) against tampering or fault injection attacks and thus provides better protection for cryptosystems with hardware-stored keys; SM2

provides the sKRKA security as ECDSA (Schnorr and GOST are sKRKA insecure) and thus can be leveraged to construct cryptocurrency applications using *BIP-32 HD wallet*-like service or achieving anonymous transactions using the idea of *stealth address*.
- We specify the current open problems of SM2 and GOST. Hopefully, this can help the research on the two young but important digital signature schemes move forward in the near future.

II. Preliminaries

A. GOST R 34.10 Signature

We review the GOST R 34.10 [8] digital signature algorithm.

- $\mathsf{Setup}(1^\lambda)$: pick an elliptic curve with parameter (p, a, b, g, q)[1], pick a secure hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$. Output $\mathsf{params} = (p, a, b, g, q, H)$.
- $\mathsf{KeyGen}(\mathsf{params})$: choose randomly $x \xleftarrow{\$} \mathbb{Z}_q^*$, compute $X = g^x$, Output (x, X) as the private-public key pair.
- $\mathsf{Sign}(\mathsf{params}, x, m, \mathsf{ID})$: signer will
  1) Compute $R = g^r$ where r is randomly selected in $\mathbb{Z}_q^*$.
  2) $t = f(R)$, $h = H(m)$, $s = tx + rh \mod q$.

  The signature of m is $\sigma = (t, s)$.
- $\mathsf{Verify}(\mathsf{params}, Q, m, \sigma)$: verifier will
  1) Compute $h = H(m)$, $v = h^{-1} \mod q$.
  2) Parse σ as (t, s) and check $t, s \in \mathbb{Z}_q^*$.
  3) Compute $R' = g^{sv} X^{-tv}$.
  4) Check if $t = f(R') \mod q$.

  If all check equations are equivalent, the verification result is 1. Otherwise, it produces an output 0.

B. SM2 Signature

We review the SM2 [10] digital signature algorithm.

- $\mathsf{Setup}(1^\lambda)$: pick an elliptic curve with parameter (p, a, b, g, q), pick two secure hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$ and $H' : \{0,1\}^* \to \{0,1\}^{256}$. Output $\mathsf{params} = (p, a, b, g, q, H, H')$.
- $\mathsf{KeyGen}(\mathsf{params})$: choose randomly $x \xleftarrow{\$} \mathbb{Z}_q^*$, compute $X = g^x$, Output (x, X) as the private-public key pair.
- $\mathsf{Sign}(\mathsf{params}, x, m, \mathsf{ID})$: signer with ID will
  1) Compute $Z = H'(\mathsf{len}(\mathsf{ID})||\mathsf{ID}||a||b||g||X)$ where len produces the bit length of its input and $||$ denotes bit concatenation.
  2) Compute $h = H(Z||m)$, $R = g^r$ where r is randomly selected in $\mathbb{Z}_q^*$.
  3) $t = f(R)$, $c = t + h \mod q$, $s = (1 + x)^{-1}(r - cx) \mod q$.

  The signature of m is $\sigma = (c, s)$.
- $\mathsf{Verify}(\mathsf{params}, \mathsf{ID}, Q, m, \sigma)$: verifier will
  1) Compute $Z = H'(\mathsf{len}(\mathsf{ID})||\mathsf{ID}||a||b||g||X)$.
  2) Parse σ as (c, s) and check $c, s \in \mathbb{Z}_q^*$.
  3) Compute $h = H(Z||m)$, $z = c + s$, and $R' = g^s X^z$.
  4) Check if $c = f(R') + h \mod q$.

  If all check equations are equivalent, the verification result is 1. Otherwise, it produces an output 0.

[1]elliptic curve (EC) parameter (p, a, b, g, q) refers to an EC $y^2 = x^3 + ax + b$ over some field GF(p^n) with g as a base EC point of prime order q.

C. RKA and Strong Known RKA Models

We recall Φ-EUF-CM-RKA and Φ-EUF-CM-sKRKA, which are respectively the existential unforgeability security models under chosen messages against RKA [15] and sKRKA [52], where $\Phi^+ = \{\phi_i(x) = x + b_i : b_i \in \mathcal{S}\}$, $\Phi^* = \{\phi_i(x) = x * a_i : a_i \in \mathcal{S}\}$ and $\Phi^{\mathsf{aff}} = \{\phi_i(x) = a_i x + b_i : a_i, b_i \in \mathcal{S}\}$.

Algorithm 1: Game Φ-EUF-CM-RKA.

Procedure $\textsc{Init}(1^\lambda)$**:**
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
  $\mathbb{L} \leftarrow \emptyset$;
  return pk;
Procedure $\textsc{Sign}(m_i, \phi_i)$**:**
  if $\phi_i \notin \{\Phi \cup$ *identity map*$\}$ **then**
    return $\perp$;
  $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_i(\mathsf{sk}), m_i)$;
  if ϕ_i *is identity map or* $\phi_i(\mathsf{sk}) = \mathsf{sk}$ **then**
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
  return σ_i;
Procedure $\textsc{Fin}(m^*, \sigma^*)$**:**
  if $m^* \in \mathbb{L}$ **then**
    stop with 0;
  if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 0$ **then**
    stop with 0;
  stop with 1;

Definition 1. *A signature scheme is (t, q_s, ϵ)-secure under the Φ-EUF-CM-RKA (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-RKA (resp. Φ-EUF-CM-sKRKA).*

Relations between RKA and sKRKA. The relationship between the RKA model and the Strong KRKA model which is an open problem and there is no straightforward implication from one to another. It is proved that ECDSA is not RKA secure [42] but it is sKRKA secure [52].

III. RKA Insecurity of GOST

A. RKA Attack under Addition Relation

We show that the GOST is not sRKA secure with respect to addition by providing a simple and efficient attack.

1) Choose $b \xleftarrow{\$} \mathbb{Z}_q^*$, query $(m^*, \phi(x) = x - b)$ to the RKA signing oracle and obtain the signature (t, s).
2) Output the message m^* and signature $(t^*, s^*) = (t, s + tb \mod q)$

Algorithm 2: Game Φ-EUF-CM-sKRKA.

Procedure $\textsc{Init}(1^\lambda)$**:**
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
- $\mathbb{L} \leftarrow \emptyset$;
- $\phi_0 \leftarrow$ 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 $\textsc{Sign}(m_i, j)$**:**
- **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 $\textsc{Fin}(i^*, m^*, \sigma^*)$**:**
- **if** $i^* \notin [0, q_s]$ **then**
  - stop with 0;
- **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;

The forged signature (t^*, s^*) on m^* is verified as follows. By siging oracle, we have $t = f(g^r)$ and $s = (x-b)t + rh$. Then we have $s + tb - tx = rh$. Let $h = H(m^*)$. We observe that

$$\begin{aligned} R' &= g^{\frac{s^*}{h}} X^{\frac{-t^*}{h}} = g^{\frac{s+tb}{h}} X^{\frac{-t}{h}} \\ &= g^{\frac{s+tb}{h}} g^{\frac{-tx}{h}} = g^{\frac{s+tb-tx}{h}} = g^r = R \end{aligned} \tag{1}$$

Hence, the forgery (t^*, s^*) is a valid signature on m^* under public key $X = g^x$. Thus, GOST is not Φ^+-EUF-CM-RKA secure.

B. RKA Attack under Multiplicative Relation

We show that the GOST is not RKA secure with respect to multiplication by providing a simple and efficient attack under the Φ^*-EUF-CM-RKA model.

1) Choose two distinct messages $m_0, m_1 \in \{0,1\}^*$ and let $h_0 \leftarrow H(m_0), h_1 \leftarrow H(m_1)$.
2) Let $a \leftarrow \frac{h_1}{h_0} \mod q$.
3) Query $(m_1, \phi(x) = ax)$ to the RKA signing oracle and obtain the signature (t, s).
4) Output the message $m^* = m_0$ and signature $(t^*, s^*) = (t, \frac{s}{a} \mod q)$

The forged signature (t^*, s^*) on m_0 is verified as follows. By signing oracle, we have $t = f(g^r)$ and $s = axt + rh_1$. Then we have $s - axt = rh_1$. We observe that

$$\begin{aligned} R' &= g^{\frac{s^*}{h_0}} X^{\frac{-t^*}{h_0}} = g^{\frac{s}{ah_0}} X^{\frac{-t}{h_0}} = g^{\frac{s}{h_1}} X^{\frac{-at}{h_1}} \\ &= g^{\frac{s}{h_1}} g^{\frac{-axt}{h_1}} = g^{\frac{s-axt}{h_1}} = g^r = R \end{aligned} \tag{2}$$

Hence, the forgery (t^*, s^*) is a valid signature on m_0 under public key $X = g^x$. Thus, GOST is not Φ^*-EUF-CM-RKA secure.

Algorithm 3: Game 0 is the Φ^{aff}-EUF-CM-RKA for SM2, in the random oracle model.

Procedure $\textsc{Init}$**:**
- pick $H : \{0,1\}^* \to \mathbb{Z}_q$;
- pick $H' : \{0,1\}^* \to \{0,1\}^{256}$;
- $x \leftarrow_s \mathbb{Z}_q^*$; $X \leftarrow g^x$;
- $\mathbb{L} \leftarrow \emptyset$;
- return X;

Procedure $\textsc{Sign}(m_i, \mathsf{ID}_i, a_i, b_i)$**:**
- **if** $a_i, b_i \notin \mathbb{Z}_q$ **then**
  - stop with 0;
- **if** $a_i x + b_i = x$ **then**
  - $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
- $r_i \leftarrow_s \mathbb{Z}_q$; $R_i \leftarrow g^{r_i}$;
- **if** $R_i = 1$ **then**
  - return $\perp$;
- $t_i \leftarrow f(R_i)$;
- **if** $t_i = 0$ **then**
  - return $\perp$;
- $Z_i \leftarrow H'(\mathsf{len}(\mathsf{ID}_i)||\mathsf{ID}_i||a||b||g||X)$;
- $h_i \leftarrow H(Z_i||m_i)$;
- $c_i \leftarrow t_i + h_i$;
- $\mathsf{sk}_i \leftarrow a_i x + b_i$;
- $s_i \leftarrow (1 + \mathsf{sk}_i)^{-1}[r_i - c_i \mathsf{sk}_i]$;
- $\sigma_i \leftarrow (c_i, s_i)$;
- return σ_i;

Procedure $\text{RO}(m)$**:**
- return $H(m)$;

Procedure $\textsc{Fin}(m^*, \mathsf{ID}^*, (c^*, s^*))$**:**
- **if** $m^* \in \mathbb{L}$ **then**
  - stop with 0;
- **if** $c^* = 0$ *or* $s^* = 0$ **then**
  - stop with 0;
- $Z^* \leftarrow H'(\mathsf{len}(\mathsf{ID}^*)||\mathsf{ID}^*||a||b||g||X)$;
- $h^* \leftarrow H(Z^*||m^*)$;
- $R^* \leftarrow g^{s^*} X^{c^*+s^*}$;
- **if** $c^* \neq f(R^*) + h^*$ **then**
  - stop with 0;
- stop with 1;

Algorithm 4: 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 $\textsc{Init}(1^\lambda)$**:**
- $H^O \leftarrow \emptyset$, $\mathbb{L} \leftarrow \emptyset$;
- pick $H' : \{0,1\}^* \to \{0,1\}^{256}$;
- $X \leftarrow \textsc{Init}_{\mathrm{CMA}}(1^\lambda)$;
- return X;

Procedure $\textsc{Sign}(m_i, \mathsf{ID}_i, a_i, b_i)$**:**
- **if** $X^{a_i} g^{b_i} = X$ **then**
  - $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
- $Z_i \leftarrow H'(\mathsf{len}(\mathsf{ID}_i)||\mathsf{ID}_i||a||b||g||X)$;
- isNewH $\leftarrow$ false;
- **while** isNewH = false **do**
  - $m'_i \leftarrow_s \mathcal{M}$;
  - $(c', s') \leftarrow \textsc{Sign}_{\mathrm{CMA}}(m'_i, \mathsf{ID}'_i)$; // $\textsc{Sign}_{\mathrm{CMA}}$ made H query on $Z'_i||m'_i$ to RO, obtained h'
  - $H(Z_i||m_i) \leftarrow h' - c' - s' + \frac{(s'+c')(1+b_i)}{a_i}$;
  - **if** $(\cdot, h_i) \notin H^O$ **then**
    - $H^O \leftarrow H^O \cup \{(Z_i||m_i, H(Z_i||m_i))\}$;
    - isNewH $\leftarrow$ true;
- $c_i = \frac{(s'+c')(1+b_i)}{a_i} - s'$;
- $s_i \leftarrow s' - \frac{b_i(s'+c')}{a_i}$;
- $\sigma_i \leftarrow (c_i, s_i)$;
- return σ_i;

Procedure $\mathrm{RO}(m)$**:**
- **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 $\textsc{Fin}(m^*, \mathsf{ID}^*, (c^*, s^*))$**:**
- **if** $m^* \in \mathbb{L}$ **then**
  - stop with 0;
- **if** $s^* = 0$ *or* $c^* = 0$ **then**
  - stop with 0;
- $Z^* \leftarrow H'(\mathsf{len}(\mathsf{ID}^*)||\mathsf{ID}^*||a||b||g||X)$;
- $h^* \leftarrow H(Z^*||m^*)$;
- $R^* \leftarrow g^{s^*} X^{c^*+s^*}$;
- **if** $c^* \neq f(R^*) + h^*$ **then**
  - stop with 0;
- run $\textsc{Fin}_{\mathrm{CMA}}(m^*, (c^*, s^*))$;

IV. sKRKA Insecurity of GOST

We show that the GOST is not sKRKA secure with respect to addition by providing a simple and efficient attack under the Φ^+-EUF-CM-sKRKA model.

Query. Adversary $\mathcal{A}$ is given δ_i such that $\phi_i(x) = x - \delta_i$ for $i \in [1, n]$. We additionally define $\delta_0 = 0$ and accordingly $\phi_0(x) = x$. $\mathcal{A}$ chooses $i^* \in [1, n]$ and queries $(m^*, \phi_{i^*}(x))$ to the RKA signing oracle and obtain the signature (t, s). We note that $t = f(g^r)$ and $s = (x - \delta_{i^*})t + rh$. Then we have $s + t\delta_{i^*} - tx = rh$.

Forgery under any related public key. $\mathcal{A}$ chooses an index $i' \in [0, n]$ and $i' \neq i^*$. Let $d \leftarrow \delta_{i'} - \delta_{i^*}$. $\mathcal{A}$ outputs the message m^* and signature $(t^*, s^*) = (t, s + td \mod q)$.

The forgery (t^*, s^*) is a valid signature on m^* under the i'-th related public key $X_{i'} = g^{x - \delta_{i'}}$ since

$$\begin{aligned} R' &= g^{\frac{s^*}{h}} X_{i'}^{\frac{-t^*}{h}} = g^{\frac{s+td}{h}} X_{i'}^{\frac{-t}{h}} = g^{\frac{s+td}{h}} g^{\frac{-t(x-\delta_{i'})}{h}} \\ &= g^{\frac{s+t(\delta_{i'}+d)-tx)}{h}} = g^{\frac{s+t\delta_{i^*}-tx}{h}} = g^r = R \end{aligned} \tag{3}$$

Therefore, we conclude that GOST is not Φ^+-EUF-CM-sKRKA secure. We note that if $i' = 0$, we obatin a valid forgery under the master public key X.

V. Security of SM2 in the RKA Model

We prove that SM2 is Φ-EUF-CM-RKA secure assuming H is a random oracle.

Theorem 1. *Let $\mathcal{A}$ be an adversary that (τ, q_s, ϵ)-breaks the Φ^{aff}-EUF-CM-RKA security of SM2, 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 SM2, where*

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

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

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

- **Game 0** in Algorithm 3 provides the EUF-CM-RKA experiment for SM2. The random oracle is provided by RO and accordingly $\epsilon = \mathrm{Adv}_{\mathcal{A}_0}$.
- **Game 1** has two modifications from **Game 0**: line 2 in **Game 0** is replaced by $H^O \leftarrow \emptyset$; the return value of $\mathrm{RO}(m)$ in **Game 0** is set h if $(m, h) \in H^O$, otherwise set a random value sampled from $\mathbb{Z}_q \backslash \mathrm{Rng}(H^O)$ and H^O is updated to $H^O \cup \{(m, h)\}$ as well. We have $\mathrm{Adv}_{\mathcal{A}_0} = \mathrm{Adv}_{\mathcal{A}_1}$ by the random oracle model.

- Algorithm 4 constructs an adversary $\mathcal{A}_{\mathrm{CMA}}$ to break the EUF-CMA security of SM2, by running as the challenger of **Game 1** and making use of the output from $\mathcal{A}_1$. $\mathcal{A}_{\mathrm{CMA}}$ uses the output of $\textsc{Init}_{\mathrm{CMA}}$ from its EUF-CMA challenger of standard SM2 to simulate the challenger of **Game 1** in line 4. $\mathcal{A}_1$ uses the public key X to check the relation of $a_i x + b_i = x$ inseatd of using the secret key x unkonwn to him. These changes are indistinguishable to $\mathcal{A}_1$. The $\textsc{Sign}$ procedure in Algorithm 4 is simulated by using the signing oracle output from the EUF-CMA challenger of SM2. In the end, the validation of the output from $\mathcal{A}_1$ is same as except line 38. We next show that $\mathrm{Adv}_{\mathcal{A}_1} \leq \epsilon_{\mathrm{CMA}} + q_s q_H / q$.

$$\begin{aligned} c_i + s_i &= \frac{(s'+c')(1+b_i)}{a_i} - s' + s' - \frac{b_i(s'+c')}{a_i} \\ &= \frac{(s'+c')}{a_i} \end{aligned} \tag{4}$$

$$\begin{aligned} g^{r_i} &= g^{s_i}(X^{a_i} g^{b_i})^{s_i+c_i} = g^{s_i+b_i(s_i+c_i)} X^{a_i(s_i+c_i)} \\ &= g^{s_i + \frac{b_i}{a_i}(s'+c')} X^{s'+c'} = g^{s' - \frac{b_i(s'+c')}{a_i} + \frac{b_i}{a_i}(s'+c')} X^{s'+c'} \\ &= g^{s'} X^{s'+c'} = g^{r'} \end{aligned} \tag{5}$$

Then we have $f(g^{r_i}) = f(g^{r'}) = t' = t_i$. We observe that:

$$\begin{aligned} c_i &= c' - h' + H(Z_i||m_i) = t' + H(Z_i||m_i) \\ &= t_i + H(Z_i||m_i) \end{aligned} \tag{6}$$

Therefore, the signing oracle output correctly runs the verification of (c_i, s_i) against the related key $\mathsf{pk}_i = X^{a_i} g^{b_i}$:

When $\mathcal{A}_1$ produces a valid forgery $(m^*, \mathsf{ID}^*, (c^*, s^*))$, line 38 of Algorithm 4 is always reached. Now we explain that m^* was not queried to $\textsc{Sign}_{\mathrm{CMA}}$ in line 13. Observe that in line 12, 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 $h' = H(Z_i'||m_i')$ by line 14. By the random oracle model, $\mathcal{A}_1$ cannot find some m_i' and use it as m^* with probability more than $\frac{q_s q_H}{q}$. Hence, we have $\mathrm{Adv}_{\mathcal{A}_1} \leq \epsilon_{\mathrm{CMA}} + q_s q_H / q$.

To sum up, we have $\epsilon \leq \epsilon_{\mathrm{CMA}} + q_s q_H / q$. The running time is dominated by $O(q_s)$ exponentiation in the signing oracle queries. □

VI. Security of SM2 in the sKRKA Model

We prove that SM2 is Φ-EUF-CM-sKRKA secure assuming H is a random oracle.

Theorem 2. *Let $\mathcal{A}$ be an adversary that (τ, q_s, ϵ)-breaks the Φ^{aff}-EUF-CM-sKRKA security of SM2, 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 SM2, where*

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

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

Proof. The proof is 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 which is omitted for simplicity.

- **Game 0*** in Algorithm 5 describes the EUF-CM-sKRKA experiment for SM2. The random oracle is provided by RO. Thus, $\epsilon = \mathrm{Adv}_{\mathcal{A}_{0^*}}$.
- **Game 1*** has two modifications from **Game 0***: line 2 in **Game 0*** is replaced by $H^O \leftarrow \emptyset$; the return value of $\mathrm{RO}(m)$ in **Game 0*** is set h if $(m,h) \in H^O$, otherwise set a random value sampled from $\mathbb{Z}_q \backslash \mathrm{Rng}(H^O)$ and H^O is updated to $H^O \cup \{(m,h)\}$. We have $\mathrm{Adv}_{\mathcal{A}_{0^*}} = \mathrm{Adv}_{\mathcal{A}_{1^*}}$ by the random oracle model.
- Algorithm 6 describes an adversary $\mathcal{A}_{\mathrm{CMA}}$ to break the EUF-CMA security of SM2, by running as the challenger of **Game 1*** and making use of the output from $\mathcal{A}_{1^*}$. $\mathcal{A}_{\mathrm{CMA}}$ uses the output of $\textsc{Init}_{\mathrm{CMA}}$ from its EUF-CMA challenger of SM2 to simulate the challenger of **Game 1*** in line 10. This change is indistinguishable to $\mathcal{A}_{1^*}$. The $\textsc{Sign}$ procedure in Algorithm 6 is simulated by using the signing oracle output from the challenger of the EUF-CMA security. In the end, the validation of the output from $\mathcal{A}_{1^*}$ is same as except line 43, 44 and 54. We next show that $\mathrm{Adv}_{\mathcal{A}_{1^*}} \leq (q_s+1)(\mathrm{Adv}_{\mathcal{A}_{\mathrm{CMA}}} + q_s q_H / q)$.

$$h' = \frac{1}{a_j}(s'+c')(a_j - a_{j^*} + a_j b_{j^*} - a_{j^*} b_j) + H(Z_i||m_i) \tag{7}$$

$$\begin{aligned} c' &= (s'+c')(1 - \frac{a_{j^*}}{a_j} + b_{j^*} - \frac{a_{j^*} b_j}{a_j}) + (H(Z_i||m_i) + t_i) \\ &= (s'+c')(1 - \frac{a_{j^*}}{a_j} + b_{j^*} - \frac{a_{j^*} b_j}{a_j}) + c_i \end{aligned} \tag{8}$$

Then, we obtain:

$$\begin{aligned} s' &= (s'+c') - c' \\ &= (s'+c') - (s'+c')(1 - \frac{a_{j^*}}{a_j} + b_{j^*} - \frac{a_{j^*} b_j}{a_j}) - c_i \\ &= (s'+c')\frac{a_{j^*} + a_{j^*} b_j - a_j b_{j^*}}{a_j} - c_i \end{aligned} \tag{9}$$

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

$$\begin{aligned} g^{r_i} &= g^{s_i}(X^{a_j} g^{b_j})^{s_i+c_i} \\ &= g^{\frac{a_{j^*}}{a_j}(s'+c') - t' - H(Z_i||m_i)} (X^{a_j} g^{b_j})^{\frac{a_{j^*}}{a_j}(s'+c')} \\ &= g^{\frac{a_{j^*}}{a_j}(s'+c') - c_i + \frac{a_{j^*} b_j}{a_j}(s'+c')} (X^{a_{j^*}})^{s'+c'} \\ &= g^{\frac{a_{j^*}}{a_j}(s'+c') - c_i + \frac{a_{j^*} b_j - a_j b_{j^*}}{a_j}(s'+c')} (X^{a_{j^*}} g^{b_{j^*}})^{s'+c'} \\ &= g^{(s'+c')\frac{a_{j^*} + a_{j^*} b_j - a_j b_{j^*}}{a_j} - c_i} (X^{a_{j^*}} g^{b_{j^*}})^{s'+c'} \\ &= g^{s'} (X^{a_{j^*}} g^{b_{j^*}})^{s'+t'+h'} \\ &= g^{r'} \end{aligned} \tag{10}$$

Algorithm 5: Game 0* is the Φ^{aff}-EUF-CM-sKRKA for SM2, in the random oracle model.

Procedure INIT:
pick $H : \{0,1\}^* \to \mathbb{Z}_q$;
pick $H' : \{0,1\}^* \to \{0,1\}^{256}$;
$x \leftarrow_s \mathbb{Z}_q^*$; $X \leftarrow g^x$;
$\mathbb{L} \leftarrow \emptyset$;
$a_0 \leftarrow 1, b_0 \leftarrow 0$;
$\mathbb{S} \leftarrow \{(a_0, b_0)\}$;
for $j \leftarrow 1$ **to** q_s **do**
  $\phi_j(x) := a_j x + b_j \leftarrow_s \Phi^{\text{aff}}$;
  $\mathbb{S} \leftarrow \mathbb{S} \cup \{(a_j, b_j)\}$;
return $X, \mathbb{S}$;
Procedure SIGN(m_i, ID_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 $\perp$;
$Z_i \leftarrow H'(\mathsf{len}(\mathsf{ID}_i)||\mathsf{ID}_i||a||b||g||X)$;
$h_i \leftarrow H(Z_i||m_i)$;
$c_i \leftarrow t_i + h_i$;
$\mathsf{sk}_i \leftarrow a_j x + b_j$; // $j \in [0, q_s]$
$\mathsf{pk}_i \leftarrow g^{\mathsf{sk}_i}$;
$s_i \leftarrow (1 + \mathsf{sk}_i)^{-1}[r_i - c_i \mathsf{sk}_i]$;
$\sigma_i \leftarrow (c_i, s_i)$;
$\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
return $(\mathsf{pk}_i, \sigma_i)$;
Procedure RO(m):
return $H(m)$;
Procedure FIN$(i^*, m^*, \mathsf{ID}^*(c^*, s^*))$:
if $i^* \notin [0, q_s]$ **then**
  stop with 0;
if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ **then**
  stop with 0;
if $c^* = 0$ *or* $s^* = 0$ **then**
  stop with 0;
$Z^* \leftarrow H'(\mathsf{len}(\mathsf{ID}^*)||\mathsf{ID}^*||a||b||g||X^{a_{i^*}} g^{b_{i^*}})$;
 // $i^* \in [0, q_s]$
$h^* \leftarrow H(Z^*||m^*)$;
$R^* \leftarrow g^{s^*}(X^{a_{i^*}} g^{b_{i^*}})^{c^*+s^*}$; // $i^* \in [0, q_s]$
if $c^* \neq f(R^*) + h^*$ **then**
  stop with 0;
stop with 1;

Then we have $f(g^{r_i}) = f(g^{r'}) = t' = t_i$. Observe that:

$$\begin{aligned} c_i &= c' - h' + H(Z_i||m_i) = t' + H(Z_i||m_i) \\ &= t_i + H(Z_i||m_i) \end{aligned} \tag{11}$$

Hence (c_i, s_i) is a valid signature with respect to pk_i. When $\mathcal{A}_1$ outputs a valid forgery $(i^*, m^*, (c^*, s^*))$, line 54 of Algorithm 6 is reached if $i^* = j^*$. It happens with probability $\frac{1}{q_s+1}$. By the checking of line 45, m^* was not queried to SIGN_{CMA} in line 16. Now we show that m^* was not queried to SIGN_{CMA} in line 22. Observe that in line 21, 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 $h' = H(Z_i'||m_i')$ by line 23. 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 $\text{Adv}_{\mathcal{A}_{1^*}} \leq (q_s + 1)(\epsilon_{\text{CMA}} + q_s q_H/q)$.

To sum up, we have $\epsilon \leq (q_s + 1)(\epsilon_{\text{CMA}} + q_s q_H/q)$. The running time is dominated by $O(q_s)$ exponentiation in the signing oracle queries. □

VII. OPEN PROBLEMS ON SM2 AND GOST

We specify the open problems for SM2 and GOST.

1) SM2, GOST, as well as ECDSA lack security proofs of existential unforgeability in the well-accepted ROM, instead are proved in some too idealized models like BRO and GGM which may lead to further debatable conclusions.
2) The security in a multi-user setting has not yet been concerned for SM2 and GOST although there is one analysis for ECDSA [30].
3) The non-linearity structure of SM2 makes it not easy to build multi-signature as concise as Schnorr. The proposed threshold schemes for SM2 [36] and GOST [37] only support the t-out-of-n threhosld setting the security of which is guaranteed when n is number of all participants, $t-1$ is the maximum number of corrupted parties and $n \geq 2t+1$. Until now there is no *full* threshold signature scheme for either SM2 or GOST, which means t in the t-out-of-n setting can be setting any value less than n.
4) There is one blind ECDSA signature scheme proposed in [51] constructed from homomorphic encryption and zero-knowledge proof, the idea of which can be used to easily construct blind SM2 or GOST. Another efficient construction of blind SM2 without using homomorphic encryption is proposed in [53]. A blind signature version of GOST

Algorithm 6: 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 $\textsc{Init}(1^\lambda)$**:**
- $H^O \leftarrow \emptyset$, $\mathbb{L} \leftarrow \emptyset$;
- pick $H' : \{0,1\}^* \rightarrow \{0,1\}^{256}$;
- $j^* \leftarrow_s [0, q_s]$;
- $a_0 \leftarrow 1, b_0 \leftarrow 0$;
- $\mathbb{S} \leftarrow \{(a_0, b_0)\}$;
- **for** $j \leftarrow 1$ **to** q_s **do**
  - $\phi_j(x) := a_j x + b_j \leftarrow_s \Phi^{\mathrm{aff}}$;
  - $\mathbb{S} \leftarrow \mathbb{S} \cup \{(a_j, b_j)\}$;
- $X' \leftarrow \textsc{Init}_{\mathrm{CMA}}(1^\lambda)$;
- $X \leftarrow (X' g^{-b_{j^*}})^{1/a_{j^*}}$;
- return $X, \mathbb{S}$;

Procedure $\textsc{Sign}(m_i, \mathsf{ID}_i, j)$**:**
- **if** $j = j^*$ **then**
  - $\mathsf{pk}_i \leftarrow X'$;
  - $\sigma_i \leftarrow \textsc{Sign}_{\mathrm{CMA}}(m_i)$;
- **else**
  - $Z_i \leftarrow H'(\mathsf{len}(\mathsf{ID}_i)||\mathsf{ID}_i||a||b||g||X)$;
  - isNewH $\leftarrow$ false;
  - **while** isNewH = false **do**
    - $m_i' \leftarrow_s \mathcal{M}$;
    - $(c', s') \leftarrow \textsc{Sign}_{\mathrm{CMA}}(\mathsf{ID}_i', m_i')$; // $\textsc{Sign}_{\mathrm{CMA}}$ made H query on $Z_i'||m_i'$ to RO, obtained h'
    - $H(Z_i||m_i) \leftarrow h' - \frac{1}{a_j}(s' + c'))(a_j - a_{j^*} + a_j b_{j^*} - a_{j^*} b_j)$;
    - **if** $(\cdot, h_i) \notin H^O$ **then**
      - $H^O \leftarrow H^O \cup \{(Z_i||m_i, H(Z_i||m_i))\}$;
      - isNewH $\leftarrow$ true;
  - $t_i \leftarrow c' - h'$ (Note $t_i = t'$);
  - $c_i = c' - h' + H(Z_i||m_i)$;
  - $s_i \leftarrow \frac{a_{j^*}}{a_j}(s' + c') - c_i$;
  - $\mathsf{pk}_i \leftarrow X^{a_j} g^{b_j}$;
  - $\sigma_i \leftarrow (c_i, s_i)$;
- $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
- return $(\mathsf{pk}_i, \sigma_i)$;

Procedure $\mathrm{RO}(m)$**:**
- **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 $\textsc{Fin}(i^*, m^*, \mathsf{ID}^*, (c^*, s^*))$**:**
- **if** $i^* \notin [0, q_s]$ **then**
  - stop with 0;
- **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* $c^* = 0$ **then**
  - stop with 0;
- $Z^* \leftarrow H'(\mathsf{len}(\mathsf{ID}^*)||\mathsf{ID}^*||a||b||g||X')$;
- $h^* \leftarrow H(Z^*||m^*)$;
- $R^* \leftarrow g^{s^*} {X'}^{c^* + s^*}$;
- **if** $c^* \neq f(R^*) + h^*$ **then**
  - stop with 0;
- run $\textsc{Fin}_{\mathrm{CMA}}(m^*, (c^*, s^*))$;

is proposed in [35]. Unfortunately, all of them lack the rigorous security proof for one-more-unforgeability [46].

VIII. Conclusion

In a nutshell, our results well distinguish the advantages/disadvantages over these existing signature schemes. We close the gap between the hopeful abundance of use of SM2 and GOST in the near future, and the lack of the security analyses currently. In this paper, we first analyze the (strong known) related key attack for SM2 and GOST. Firstly, we give efficient attacks breaking the RKA and sKRKA security of GOST. Secondly, through game-hopping proofs, we proved that SM2 is secure in both RKA and sKRKA models in ROM, leading the following two meaningful implications:

1) SM2 is more secure than Schnorr, ECDSA, GOST when resisting tampering or fault injection attacks.
2) SM2 is as secure as ECDSA (better than Schnorr) in terms of constructing blockchain applications supporting *BIP-32* or *stealth address*.

References

[1] "Chainsql," http://www.chainsql.net/secret_setup.html.
[2] "Cryptoarm," https://www.cryptopro.ru/news/2018/09/mobilnoe%20prilozhenie%20kriptoarm%20gost%20s%20podderzhkoi%20gost%20r%203410%202012%20dobavleno%20v%20appstore.

[3] "Dongjin," http://www.donjin.com/fangan/375.shtml.

[4] "Ethereum implementation using gost," https://www.ledgerinsights.com/blockchain-standards-russia/.

[5] "Financial services blockchain consortium (shenzhen) fisco," http://www.fisco-bcos.org/.

[6] "Hyperledger-twgc," https://training.linuxfoundation.cn/news/155.

[7] "Hyperledger-ursa," https://wiki.hyperledger.org/display/ursa/0000-OpenSSL-Interface.

[8] "Ietf rfc - gost r 34.10-2012: Digital signature algorithm," https://www.ietf.org/rfc/rfc7091.txt.

[9] "Ietf rfc - shangmi (sm) cipher suites for tls 1.3," https://tools.ietf.org/html/rfc8998.

[10] "Iso/iec 11889-1:2015: Information technology — trusted platform module library — support for smx family of algorithms — annex c.1.2: Sm2 digital signature algorithm," https://www.iso.org/standard/66510.html.

[11] "Mozilla firefox 28.0," https://club.cnews.ru/blogs/entry/podderzhka_rossijskih_kriptoalgoritmov_gost_r_34_102012_i_gost_r_34_112012_v_brauzere_mozilla_firefox_28_0.

[12] "Patent-cn106130738a," https://patents.google.com/patent/CN106130738A/zh.

[13] B. Applebaum, D. Harnik, and Y. Ishai, "Semantic security under related-key attacks and applications." in *ICS 2011*, vol. 2011. Citeseer, 2011, pp. 45–60.

[14] M. Bellare and D. Cash, "Pseudorandom functions and permutations provably secure against related-key attacks," in *CRYPTO 2010*. Springer, 2010, pp. 666–684.

[15] M. Bellare, D. Cash, and R. Miller, "Cryptography secure against related-key attacks and tampering," in *ASIACRYPT 2011*. Springer, 2011, pp. 486–503.

[16] M. Bellare and T. Kohno, "A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications," in *EUROCRYPT 2003*. Springer, 2003, pp. 491–506.

[17] E. Biham, "New types of cryptanalytic attacks using related keys (extended abstract)," in *EUROCRYPT 1993*, ser. Lecture Notes in Computer Science, vol. 765. Springer, 1993, pp. 398–409.

[18] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *CRYPTO 1997*. Springer, 1997, pp. 513–525.

[19] I. F. Blake, G. Seroussi, and N. P. Smart, *Advances in elliptic curve cryptography*. Cambridge University Press, 2005, vol. 317.

[20] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *EUROCRYPT 1997*. Springer, 1997, pp. 37–51.

[21] D. R. Brown, "Generic groups, collision resistance, and ecdsa," *Designs, Codes and Cryptography*, vol. 35, no. 1, pp. 119–152, 2005.

[22] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled, "Uc non-interactive, proactive, threshold ecdsa with identifiable aborts," in *ACM CCS 2020*, 2020, pp. 1769–1787.

[23] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Bandwidth-efficient threshold ec-dsa," in *IACR International Conference on Public-Key Cryptography - PKC 2020*. Springer, 2020, pp. 266–296.

[24] Dingxuan, "Julongchain," http://www.julongchain.com/index.html.

[25] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Threshold ecdsa from ecdsa assumptions: the multiparty case," in *IEEE S&P 2019*. IEEE, 2019, pp. 1051–1066.

[26] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[27] M. Fersch, E. Kiltz, and B. Poettering, "On the provable security of (ec) dsa signatures," in *ACM CCS 2016*, 2016, pp. 1651–1662.

[28] M. Fersch, E. Kiltz, and B. Poettering, "On the one-per-message unforgeability of (ec) dsa and its variants," in *TCC 2017*. Springer, 2017, pp. 519–534.

[29] A. Gagol and D. Straszak, "Threshold ecdsa for decentralized asset custody," Cryptology ePrint Archive, Report 2020/498, 2020. https://eprint.iacr. org ..., Tech. Rep., 2020.

[30] S. Galbraith, J. Malone-Lee, and N. P. Smart, "Public key signatures in the multi-user setting," *Information Processing Letters*, vol. 83, no. 5, pp. 263–266, 2002.

[31] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ecdsa with fast trustless setup," in *ACM CCS 2018*, 2018.

[32] R. Gennaro and S. Goldfeder, "One round threshold ecdsa with identifiable abort." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 540, 2020.

[33] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering," in *TCC 2004*. Springer, 2004, pp. 258–277.

[34] D. Goldenberg and M. Liskov, "On related-secret pseudorandomness," in *Theory of Cryptography Conference*. Springer, 2010, pp. 255–272.

[35] H. Hosseini, B. Bahrak, and F. Hessar, "A gost-like blind signature scheme based on elliptic curve discrete logarithm problem," *arXiv preprint arXiv:1304.2094*, 2013.

[36] Y. Jie, L. Yu, C. Li-yun, and N. Wei, "A sm2 elliptic curve threshold signature scheme without a trusted center," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 10, no. 2, pp. 897–913, 2016.

[37] S. Kim, J. Kim, J. H. Cheon, and S.-h. Ju, "Threshold signature schemes for elgamal variants," *Computer Standards & Interfaces*, vol. 33, no. 4, pp. 432–437, 2011.

[38] L. R. Knudsen, "Cryptanalysis of loki 91," in *AUSCRYPT 1992*. Springer, 1992, pp. 196–208.

[39] Y. Lindell and A. Nof, "Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody," in *ACM CCS 2018*, 2018.

[40] H. T. C. Ltd, "Huawei blockchain whitepaper," https://www.huaweicloud.com/content/dam/cloudbu-site/archive/hk/en-us/about/analyst-reports/images/4-201804-Huawei%20Blockchain%20Whitepaper-en.pdf.

[41] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[42] H. Morita, J. C. Schuldt, T. Matsuda, G. Hanaoka, and T. Iwata, "On the security of the schnorr signature scheme and dsa against related-key attacks," in *ICISC 2015*. Springer, 2015, pp. 20–35.

[43] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[44] T. Q. Phong and N. Q. Toan, "Some security comparisons of gost r 34.10-2012 and ecdsa signature schemes," 2017.

[45] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *EUROCRYPT 1996*. Springer, 1996, pp. 387–398.

[46] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[47] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *EUROCRYPT 1989*. Springer, 1989, pp. 239–252.

[48] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart, "Flaws in applying proof methodologies to signature schemes," in *CRYPTO 2002*. Springer, 2002, pp. 93–110.

[49] N. van Saberhagen, "Cryptonote v 2.0 whitepaper," https://bytecoin.org/old/whitepaper.pdf.

[50] P. Wuille, "Bip 0032," https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.

[51] X. Yi and K.-Y. Lam, "A new blind ecdsa scheme for bitcoin transaction anonymity," in *ACM AsiaCCS 2019*, 2019, pp. 613–620.

[52] T. H. Yuen and S.-M. Yiu, "Strong known related-key attacks and the security of ecdsa," in *NSS 2019*. Springer, 2019, pp. 130–145.

[53] Y. Zhang, D. He, F. Zhang, X. Huang, and D. Li, "An efficient blind signature scheme based on sm2 signature algorithm," in *INSCRYPT 2020*. Springer International Publishing, 2020, pp. 368–384.

[54] Z. Zhang, K. Yang, J. Zhang, and C. Chen, "Security of the sm2 signature scheme against generalized key substitution attacks," in *SSR 2015*. Springer, 2015, pp. 140–153.