Hindawi
Security and Communication Networks
Volume 2021, Article ID 7897527, 8 pages
https://doi.org/10.1155/2021/7897527

Research Article

Practical SM2-Based Multisignature Scheme with Applications to Vehicular Networks

Lin Hou,[1] **Wei Liu**,[1] **Lisha Yao**,[1] **Xiaojian Liang**,[1] **and Guo-Qiang Zeng**[2]

[1]*College of Information Science and Technology, Jinan University, Guangzhou 510632, China*
[2]*College of Cyber Security and the National Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University, Guangzhou 510632, China*

Correspondence should be addressed to Wei Liu; weiliuscholar@gmail.com

Received 11 June 2021; Accepted 13 September 2021; Published 27 October 2021

Academic Editor: Rongmao Chen

In vehicular networks, the increasing value of transportation data and scale of connectivity also brings many security and privacy concerns. Peer authentication and message integrity are two vital security requirements to ensure safe transportation system. Because of the constrained resources of the units performing the cryptographic components, the proposed security-enhancing schemes should be lightweight and scalable. In this paper, we present a multisignature scheme derived from the SM2 signature which enables a group of parties to collaboratively sign a message and generate a compact joint signature at the end. Our scheme requires no preprocessing or interactions among the parties before signing, and its performance matches or surpasses known ones in terms of signing time, verification time, and signature size. Therefore, our scheme is also suitable for vehicular networks, with the goal to enhance security with small computation and storage cost.

1. Introduction

With the development of advanced information and communication-based technologies, intelligent transportation system (ITS) can provide a seamless transportation infrastructure and more functionalities for vehicles than a decade ago. Specifically, the Vehicle-to-Everything (V2X) communication technology in vehicular networks nowadays is able to support information sharing between vehicles and any other element involved in ITS [1, 2], including nearby vehicles (V2V), the infrastructure (V2I), mobile devices carried by pedestrians (V2P), and remote application servers or cloud platforms (V2N). The increasing scale of ITS ecosystem and the growing trend to integrate vehicular network deployment with other networks also bring concerns about cybersecurity for ITS since any message interception or modification by malicious units could result in fatal consequences [3, 4].

Digital signature is commonly used in vehicular networks to ensure integrity of messages exchanged among devices. However, the effectiveness of information propagation and routing, which are associated to delays and hence also have impacts on road safety, naturally depends on the computational overhead imposed by the applied security mechanisms [5]. Beyond traditional signature schemes, multisignature (MS) and aggregate signature (AS) are extended primitives considering multiuser setting to support cosigning and to reduce verification cost. The two primitives in common allow a group of signers to combine their individual signatures into a single short one. Specifically, an MS scheme [6, 7] enables a group of signers, each having a public key and a corresponding private key, to collaboratively produce a joint signature on a common message which can be publicly verified given the set of public keys of all signers. As a more general primitive, an AS scheme [8, 9] allows each of the signers to sign a different message, and all these individual signatures can still be aggregated into a single short one. As in the traditional signature scheme, the short combined signature should convince the verifier that all signers signed their designated messages.

Both MS and AS schemes have many potential uses in vehicular networks, such as in the distributed certificate

authority (CA) or in V2I/V2V communications. Unfortunately, the commonly used technologies including dedicated short-range communications (DSRC) and cellular-V2X (C-V2X) mainly exploit elliptic curve-based signature schemes, e.g., ECDSA and SM2, which to the best of our knowledge has very few MS or AS extensions due to their nonlinear construction.

In this paper, we propose a candidate multisignature scheme $MS-SM2$ based on the SM2 signature algorithm and specify the applications of $MS-SM2$ for vehicular networks. SM2 is a signature algorithm standard based on the elliptic curve published by the Chinese government and has been extensively used in cryptographic devices in finance and industry. Our proposed $MS-SM2$ scheme allows dynamic joining of signers (with certified public keys) and has no burdensome assumptions on the public-key infrastructure (PKI), which makes it plausible in vehicular networks.

1.1. Our Contributions. The original contribution of this work is mainly twofold:

(i) We first present a multisignature scheme $MS-SM2$ based on the SM2 signature by designing a cosigning protocol and prove its security in plain public-key and semihonest model. No preprocessing or any proof-of-knowledge step on the signer side is required in our scheme. The experimental results also show that our protocol is relatively practical for many applications.

(ii) We then illustrate some possible applications of $MS-SM2$ in vehicular networks, especially the usage in the multiple CAs architecture to reduce the certification storage for vehicles and RSUs and in V2I communication to reduce the computational overhead for RSUs.

1.2. Related Work. A trivial way to build a multisignature from standard signatures is to concatenate all stand-alone signatures signed individually. However, the resulted multisignature is of large size and particularly of size proportional to the number of signers, which does not scale well in practice [6, 7, 10]. Therefore, a multisignature should be short, meaning its length should be (ideally) independent from the number of signers and about the same as that of an ordinary stand-alone signature. Informally, the possibility of extending standard signature schemes to multisignatures comes from the homomorphism of the involved arithmetic operations of the underlying assumptions. However, the homomorphism also brings a serious vulnerability and allows adversaries to mount *rogue key attacks*, in which the attackers without valid key pairs can set its public key as a function of those from other honest signers and finally forge multisignatures. Micali et al. [6] described the formal model for the attack and showed a way to prevent such attacks known as *knowledge of secret key* (KOSK) assumption, in which users are required to prove knowledge of their secret keys during public key registration. Bellare and Neven [7] proposed a new practical multisignature scheme based on the Schnorr signature without KOSK assumption and proved that it can avoid rogue attack in the so-called *plain public key model*. There are several following-up work on constructing 2-round Schnorr-based multisignatures, i.e., all singers only need 2 rounds of communications to produce a multisignature [11–15]. Recently, public key aggregation is introduced to a multisignature scheme by which the verifier can check the validity of a multisignature only using a short aggregate key rather than a public key list [16, 17].

2. Preliminaries

For prime number p, $\mathbb{Z}_p$ denotes the additive group of integer modulo p. We consider elliptic curve E: $y^2 = x^3 + ax + b \pmod p$ in $\mathbb{Z}_p$, where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \neq 0 \pmod p$. The set of points on E along with the infinity point $\mathcal{O}$ constitutes an additive elliptic-curve group $E(\mathbb{Z}_p)$ under points addition, denoted by $\oplus$, with $\mathcal{O}$ being the identity. Let $G(x_G, y_G) \in E(\mathbb{Z}_p)(G \neq \mathcal{O})$ be the base point in $E(\mathbb{Z}_p)$ with order n. For $k \in \mathbb{Z}$, $Q(x_Q, y_Q) = [k]G$ denotes the scalar multiplication in $E(\mathbb{Z}_p)$.

Range $[x, y]$ denotes the set of integers i, $x \leq i \leq y$. Given a nonempty set S, $s \xleftarrow{\$} S$ denotes the operation of sampling an element of S uniformly at random and assigning it to s. For a randomized algorithm $\mathcal{A}$, $y \longleftarrow \mathcal{A}((x_1, \ldots, x_n); \rho)$ denotes the operation of running $\mathcal{A}$ on inputs $(x_1, \ldots, x_n)$ and random coins ρ then assigning its output to y.

2.1. Multisignature Scheme

2.1.1. Syntax. We follow the description of Bellare and Neven [7] and define a multisignature scheme as a tuple $MS = (\text{Setup}, \text{KeyGen}, \text{MSign}, \text{Vrfy})$. Note that the scheme is defined in the plain public key model, where the key generation is as same as that in any public-key cryptography and no more preprocessing protocol or key verification is required.

$\text{Setup}(1^\kappa) \longrightarrow pp$: the setup algorithm takes as input the security parameter κ and generates system parameters pp.

$\text{KeyGen}(pp) \longrightarrow (sk, pk)$: the key generation algorithm is a randomized algorithm executed by every signer on input pp to generate a key pair (sk, pk).

$\text{MSign}(pp, L, sk_i, m) \longrightarrow \sigma$: the MSign algorithm represents the signing protocol run by a group of signers who intend to collaboratively sign the same message m. Each signer i executes the protocol on input pp, a set of public keys of signers $L = \{pk_1, \ldots, pk_N\}$, private key sk_i and message m. The protocol outputs a multisignature σ.

$\text{Vrfy}(pp, L, m, \sigma) \longrightarrow 0/1$: the verification algorithm checks the validity of a multisignature σ on message m on behalf of the group of signers whose public keys are in set L and output 1 or 0 indicating the multisignature is valid or not.

2.1.2. Completeness. A multisignature scheme should satisfy the following *completeness* property, meaning that for any number n and message m, if $(pk_i, sk_i) \leftarrow \text{KeyGen}(pp)$ for $i \in \{1, \ldots, N\}$ and all signers run $\text{MSign}(pp, L, m, sk_i)$, then every signer will output the same signature σ such that $\text{Vrfy}(pp, L, m, \sigma) = 1$.

2.1.3. Security. The security of multisignature requires that it is infeasible to forge a signature involving at least one honest signer. We assume an adversary (forger) $\mathcal{F}$ that corrupts all other signers except the honest one and can choose their public keys in arbitrary ways as it likes, e.g., the rogue key attack. The unforgeability of multisignature in plain public key model is defined by the following three-phase game $\text{Exp}_{MS}^{UF-CMA}(\mathcal{F})$ between the forger $\mathcal{F}$ and a challenger.

Setup. The challenger generates system parameter $pp \leftarrow \text{Setup}(1^\kappa)$ and a challenge key pair $(pk^*, sk^*) \leftarrow \text{KeyGen}(pp)$ for the target honest signer. It returns (pp, pk^*) to $\mathcal{F}$.

Query. The forger $\mathcal{F}$ is allowed to make signature queries on any message m for any set L of signers with $pk^* \in L$. This signing oracle $\mathcal{O}(pp, \cdot, sk^*, \cdot)$ simulates the honest signer with key sk^* interacting in a signing protocol with other signers in list L. $\mathcal{F}$ can make any number of such queries concurrently.

Forge. $\mathcal{F}$ outputs a set L^* of public keys, a message m^*, and a multisignature σ^*. The forger is said to win the game if $\text{Vrfy}(pp, L^*, m^*, \sigma^*) = 1$ with $pk^* \in L^*$ and the message m^* never appeared in Query phase.

The advantage of forger $\mathcal{F}$ in breaking the multisignature scheme is defined as the probability that $\mathcal{F}$ wins the above game (over the random coins of the challenger), denoted as $\text{Adv}_{MS}^{UF-\text{CMA}}(\mathcal{F})$.

Definition 1 (UF-CMA security). A multisignature scheme is (t, q_s, N, ε)-unforgeable if it holds that $\text{Adv}_{MS}^{UF-\text{CMA}}(\mathcal{F}) \leq \varepsilon$ for every forger $\mathcal{F}$ that runs in time at most t, makes at most q_s signing queries, produces forgeries on behalf of N parties, and wins the $\text{Exp}_{MS}^{UF-CMA}(\mathcal{F})$ game with negligible probability ε. In random oracle model, we define it as $(t, q_s, q_h, N, \varepsilon)$-unforgeable where q_h denotes the maximum number of hash queries.

2.2. SM2 Signature Algorithm. The SM2 signature algorithm is initialized by taking as input a security parameter κ and outputs $pp(E(\mathbb{Z}_p), \mathcal{O}, G, n, H(\cdot))$ as public parameters, in which H: $\{0,1\}^* \longrightarrow \mathbb{Z}_n$ is a cryptography hash function. The SM2 signature scheme is briefly reviewed in Table 1.

2.3. General Forking Lemma. We will use the general forking lemma [7] to prove the security of our scheme, which is a useful tool by extending the forking lemma of Pointcheval and Stern [18] without mentioning concrete signatures or random oracles.

Lemma 1 (general forking lemma). *Let H be a set of size $h(\geq 2)$, and $(h_1, \ldots, h_q) \xleftarrow{\$}$. Let $\mathcal{A}$ be a randomized algorithm that on input $\{x, (h_1, \ldots, h_q)\}$ returns a pair (i, σ), where $i \in \{0, \ldots, q\}$ and σ is a side output. For some randomized input generator IG, the accepting probability of algorithm $\mathcal{A}$, denoted by acc, is defined as $\Pr[i \geq 1 || x \xleftarrow{\$}]$. Consider randomized algorithm $\text{Fork}^{\mathcal{A}}$ associated with $\mathcal{A}$, taking as input x, proceeds as described in Algorithm 1. Let frk be the probability that $\Pr[b = 1 | x \xleftarrow{\$}]$. Then,*

$$\text{frk} \geq \text{acc}\left(\frac{\text{acc}}{q} - \frac{1}{h}\right). \tag{1}$$

2.4. Secure Multiparty Computation. Secure multiparty computation (MPC) enables a group to jointly perform a computation without disclosing any participant's private inputs. The participants agree on a function to compute and then can use an MPC protocol to jointly compute the output of that function on their secret inputs without revealing them [19]. There are several well-studied MPC protocols such as the GMW protocol [20] and the BGW protocol [21]. Both of the two schemes are based on the secret-sharing technique and can support both Boolean circuit and arithmetic circuit.

Here, we only present the general idea of a simple addition function to show how the protocols work. The basic idea is to allow each party holding the secret shares of the inputs; therefore, each party can locally sum up their shares and get a valid sharing of the final result. We describe it in a bit more detail in Figure 1.

3. SM2-Based Multisignature Scheme: $MS - SM2$

In this section, we present a multisignature scheme based on the SM2 signature in the plain public key model. Intuitively, the original signing algorithm of SM2 involves a nonlinear combination of secret key and randomness; therefore, it is nontrivial to extend it directly to a multisignature. To cope with the problem, in the protocol, we first exploit the linear part in SM2 to produce a semiaggregated signature and then employ a simple MPC protocol for addition to finally achieve the goal. Note that we slightly modify the output of original SM2 signing algorithm in protocol where we take the inverse of s instead to be the part of signature by each party. Therefore, the multisignature in our scheme is *almost* of the same structure as the original SM2 signature and remains practical. The unforgeability of the multisignature under chosen message attack can be proved in the random oracle model using general forking lemma [7, 16].

3.1. Construction. The initialization Setup algorithm and KeyGen algorithm of the multisignature are almost the same as that in the SM2 scheme, except that there are two hash functions used in multisignature scheme, denoted as H_0: $E(\mathbb{Z}_p) \longrightarrow \mathbb{Z}_n, H_1$: $\{0,1\}^* \longrightarrow \mathbb{Z}_n$. We now proceed to describe the signing protocol and verification algorithm of

Table 1: SM2 signature algorithm.

Key Generation	Signing	Verification
For user j, it generates	To sign message M, j computes	To verify $(M', \sigma\prime)$ with P_j,
sk: $d_j \xleftarrow{\$}$	1. $e = H(Z_j M)$	1. If $r', s' \notin [1, n-1]$,
pk: $P_j = [d_j]G$	2. $k \xleftarrow{\$}$	Return REJECT
Z_j: public hash bits of user	3. $(x_1, y_1) = [k]G$	2. $e' = H(Z_j M')$
	4. $r = (e + x_1) \pmod{n}$;	3. $t = r' + s' \pmod{n}$
	If $r = 0$ or $r + k = n$, go to Step 2	If $t = 0$, return REJECT
	5. $s = (1 + d_j)^{-1}(k - r \cdot d_j) \pmod{n}$;	4. $(x_1', y_1') = [s_1']G + [t]P_j$
j's information including pk	If $s = 0$, go to Step 2	5. If $r_1' = (e_1' + x_1') \pmod{n}$,
	6. Return signature $\sigma = (r, s)$	Return REJECT
		Else
		Return ACCEPT

(1) Select random coins ρ for $\mathscr{A}$
(2) $(h_1, \ldots, h_q) \xleftarrow{\$}$
(3) $(i, \sigma) \leftarrow \mathscr{A}(x, (h_1, \ldots, h_q); \rho)$;
(4) if $i = 0$ then
(5) return $(0, \varepsilon, \varepsilon)$;
(6) end
(7) $(h_{i\prime}, \ldots, h_{q\prime}) \xleftarrow{\$}$
(8) $(i\prime, \sigma\prime) \leftarrow \mathscr{A}(x, h_1, \ldots, h_{i-1}, h_i', \ldots, h_q'; \rho)$;
(9) if $(i = i\prime$ and $h_i \neq h_i')$ then
(10) return $(1, \sigma, \sigma\prime)$
(11) else
(12) return $(0, \varepsilon, \varepsilon\prime)$
(13) end

Algorithm 1: The forking algorithm Fork$^{\mathscr{A}}$.

PARAMETERS:
N: the number of parties;
x_i: the input of party P_i;
F: the function to compute (it is addition function here);

PROTOCOL:

1. Each party P_i creates N shares of input x_i using a (N, N)-secret sharing scheme, denote each share by $p_i(j)$.
2. P_i sends each share $p_i(j)$ $(j \in [1, \mathrm{N}], j \neq i)$ toP_j.
3. P_i computes $v_i = \sum_{j=1}^{\mathrm{N}} p_j(\mathrm{i})$. That is each party adds up all shares they received.
4. P_i broadcasts v_i to all other parties.
5. Each party computes $\mathrm{v} = \sum_{\mathrm{j}=1}^{\mathrm{N}} v_j$ to get the desired output.

Figure 1: The MPC protocol for addition $\mathscr{F}_{\mathrm{add}}$.

the $MS - SM2$ scheme. Note that we take L to be size of N for simplicity, where N is the maximum number of cosigners and $N \ll n$.

MSign(pp, L, m, sk_i): each signer i with secret key $sk_i = d_i$ and public key $pk_i = P_i$ in set L runs an interactive protocol to collaboratively sign a message m. The communication proceeds in a number of rounds, where in each round, every signer sends and receives messages to and from other signers and also performs some local computation.

(1) Choose $k_i \xleftarrow{\$}$, compute $K_i(x_{i,1}, y_{i,1}) = [k_i]G$ and $t_i = H_0(x_{i,1}, y_{i,1})$, and broadcast t_i.
(2) Upon receiving t_j from all other signers, broadcast $K_i(x_{i,1}, y_{i,1})$.
(3) Upon receiving $(x_{j,1}, y_{j,1})$ from all other signers, check the hash values and abort the protocol if for any j that $t_j \neq H_0(x_{j,1}, y_{j,1})$. Otherwise, set $e_i = H_1(Z_i \| L \| i_{i=1}^{N} K_i \| m)$, $r_i = e_i + x_{i,1} \pmod{n}$, and $\widetilde{s_i} = (k_i - r_i \cdot d_i) \pmod{n}$. Then, broadcast $\widetilde{s_i}$.
(4) Upon receiving $\widetilde{s_j}$ from all other signers, compute $\widetilde{s} = \sum_{i=1}^{N} \widetilde{s} \pmod{n}$ and run the protocol for $\mathscr{F}_{\mathrm{add}}$ with input $s_i = (1 + d_i) \cdot \widetilde{s}^{-1} \pmod{n}$ to get the addition $s = \sum_{i=1}^{N} s_i \pmod{n}$.

At the end the interactive protocol, the algorithm outputs a multisignature $\sigma = (K, s)$, where K is the set of all points $K_i(x_{i,1}, y_{i,1})$.

Vrfy(pp, L, m, σ): given a multiset of public keys L, message m, and multisignature σ, the verifier computes $e_i = H_1(Z_i \| L \| i_{i=1}^N K_i \| m)$ and $r_i = e_i + x_{i,1} \pmod n$, accepts the signature if $[s]\oplus_{i=1}^N K_i - [s]\oplus_{i=1}^N([r_i]P_i) = [N]G + \oplus_{i=1}^N P_i$, and outputs 1. Otherwise, it outputs 0.

Correctness: if $\sigma = (K, s)$ is a valid output of protocol, Vrfy algorithm always accepts and outputs 1. The equation only holds when all signers follow the protocol and use valid key pairs. Note that the integer computations are all modulo n, and we omit the notation for simplicity.

$$\begin{aligned}
[s]\mathop{\oplus}_{i=1}^{N} K_i - [s]\mathop{\oplus}_{i=1}^{N}([r_i]P_i) &= \left[s\sum_{i=1}^{N} k_i\right]G - \left[s\sum_{i=1}^{N}(r_i \cdot d_i)\right]G \\
&= \left[s \cdot \left(\sum_{i=1}^{N} k_i - \sum_{i=1}^{N}(r_i \cdot d_i)\right)\right]G \\
&= \left[\sum_{i=1}^{N} s_i \cdot \sum_{i=1}^{N} \widetilde{s_i}\right]G = \left[\sum_{i=1}^{N}(1 + d_i) \cdot \widetilde{s}^{-1} \cdot \widetilde{s}\right]G \\
&= \left[\sum_{i=1}^{N}(1 + d_i)\right]G = [N]G + \mathop{\oplus}_{i=1}^{N} P_i.
\end{aligned} \tag{2}$$

3.2. Security Proof. In general, we can treat the multisignature scheme as a multiparty computation protocol and prove its security in simulation-based framework for a clearer security guarantee. Unfortunately, the security of multisignature is traditionally defined in game-based framework, and on the other hand, simulation-based proof is complex in the random oracle model. Here, we follow the game-based definition of Bellare and Neven [7] and only show a proof sketch for the scheme.

The basic idea of game-based proof is to obtain from $\mathcal{F}$ two different forgeries σ and $\sigma\prime$ with the same randomness by employing the general forking lemma. As a result, we can extract the secret key from the target public key pk^*, which is usually a solution of the discrete-logarithm problem in the elliptic-curve group $E(\mathbb{Z}_p)$. For simplification, we take an equivalent verification equation into consideration, and if $\sigma = (K, s)$ and $\sigma\prime = (K, s\prime)$ satisfy

$$\begin{aligned}
\mathop{\oplus}_{i=1}^{N} K_i - \mathop{\oplus}_{i=1}^{N}([r_i]P_i) &= \left[s^{-1}N\right]G + \left[s^{-1}\right]\mathop{\oplus}_{i=1}^{N} P_i, \\
\mathop{\oplus}_{i=1}^{N} K_i - \mathop{\oplus}_{i=1}^{N}([r_i]P_i) &= \left[s^{-1}N\right]G + \left[s^{-1}\right]\mathop{\oplus}_{i=1}^{N} P_i,
\end{aligned} \tag{3}$$

then the secret key d^* corresponding to pk^* can be computed from the equation

$$\sum_{i=1}^{N}[(r_i' - r_i)d_i] = \left(s^{-1} - s'^{-1}\right)\left(N - \sum_{i=1}^{N} d_i\right). \tag{4}$$

However, in the process of MSign, each signer can check the value $\widetilde{s}$ before continuing to execute the protocol, which allows signers to quit cosigning immediately if there is any rogue key attack. Specifically, they can compute $[x_1', y_1'] = [\widetilde{s}]G + \sum_{i=1}^N([r_i]P_i)$ and check if x_1' is equal to the corresponding part in the result.

$$\begin{aligned}
[\widetilde{s}]G + \sum_{i=1}^{N}([r_i]P_i) &= \left[\sum_{i=1}^{N}\widetilde{s_i}\right]G + \left[\sum_{i=1}^{N}(r_i \cdot d_i)\right]G \\
&= \left[\sum_{i=1}^{N}(k_i - r_i \cdot d_i)\right]G + \left[\sum_{i=1}^{N}(r_i \cdot d_i)\right]G \\
&= \left[\sum_{i=1}^{N} k_i\right]G = \oplus_{i=1}^{N}\left(x_{i,1}, y_{i,1}\right).
\end{aligned} \tag{5}$$

Therefore, we can let the simulator halt if the forger successfully forged $\widetilde{s}$.

Lemma 2. *If there exists a $(t, q_s, q_h, N, \varepsilon)$-forger $\mathcal{F}'$ that can output a forgery $\widetilde{s}$, then there exists a PPT algorithm $\mathcal{A}$ which (t', ε')-solves the DL problem in $E(\mathbb{Z}_p)$.*

Proof. Note that $\widetilde{s} = \sum_{i=1}^N \widetilde{s_i} \pmod n$ and each $\widetilde{s_i}$ has similar structure with Schnorr signature. Therefore, the proof of Lemma 2 is similar to that of the $MS - BN$ scheme. Generally, given a $(t, q_s, q_h, N, \varepsilon)$-forger $\mathcal{F}'$, we first wrap it into an algorithm $\mathcal{B}$ that can be used in the general forking lemma. We then describe an algorithm $\mathcal{A}$ that on input $pk^* = P^*$ and runs $\text{Fork}^{\mathcal{B}}(pk^*)$ to output the corresponding discrete logarithm. □

Let $q = q_h + q_s$, $T_0[\cdot], T_1[\cdot]$ be the programmed hash tables for oracles H_0 and H_1, respectively, and $h_1\{h_{1,1}, \ldots, h_{1,q}\}$ be the answers of queries to H_1. Two counters ctr_1 and ctr_2 are initialized to zero. An additional array $T_2[\cdot]$ records a unique index $1 \le i \le q_h + Nq_s$ to each public key P_i occurring either as a cosigner's public key in signature queries or H_1 queries, where $T_2[P^*] = 0$. On input $pp, h_1, P^* \in E(\mathbb{Z}_p)$, $\mathcal{B}$ plays the $\text{Exp}_{\text{MS}}^{\text{UF-CMA}}(\mathcal{F})$ game with $\mathcal{F}'$ with the target public key $pk^* = P^*$. $\mathcal{B}$ answers queries from $\mathcal{F}'$ by programming the oracles as follows:

(i) $H_0(K_i)$: if $H_0(K_i)$ is undefined, then $\mathcal{B}$ randomly assigns $T_0[K_i] \xleftarrow{\$}$ and then returns $t_i = T_0[K_i]$.

(ii) $H_1(Z_i \| L \| i_{i=1}^N K_i \| m)$: if $T_2[P_i]$ is undefined, then $\mathcal{B}$ increments ctr_2 and sets $T_2[P_i] = ctr_2$. Let $k = T_2[P_i]$; if $T_1[k, L \| i_{i=1}^N K_i \| m]$ has not yet been defined, then $\mathcal{B}$ assigns random values to all $T_1[j, L \| i_{i=1}^N K_i \| m]$ for $1 \le j \le q_h + Nq_s$, increases ctr_1, and assigns $T_1[0, L \| \oplus_{i=1}^N K_i \| m] = h_{1,\text{ctr}_1}$.

(iii) $\mathcal{O}^{\text{sign}}(L, m)$: if $P^* \notin L$, then $\mathcal{B}$ returns $\perp$ to the forger. Otherwise, it parses L as $\{P^*, P_2, \ldots, P_N\}$. $\mathcal{B}$ first checks whether $T_2[P_i]\,(2 \le i \le N)$ has already been defined, if not it increases ctr_2 and sets $T_2[P_i] = \text{ctr}_2$. Then, it increases counter ctr_1 and sets $e_1 = h_{1,ctr_1}$. It chooses $\widetilde{s_1} \xleftarrow{\$}$ and computes a elliptic curve point K_1 such that $K_1(x_{1,1}, y_{1,1}) = [\widetilde{s_1}]G + [r_1]P^*$, where $r_1 = h_{1,ctr_1} + x_{1,1}$. It finally sends $t_1 = H_0(K_1)$ to all cosigners. After receiving all t_j from $\mathcal{F}'$ (all other cosigners),

$\mathscr{B}$ looks up the corresponding K_j in table T_0 such that $t_j = T_0[K_j]$. If not all such values can be found, $\mathscr{B}$ randomly chooses $K_1' \xleftarrow{\$}$ and broadcasts K_1'. If there exists $K_{j\prime} \neq K_j$ such that $T_0[K_{j'}] = T_0[K_j]$, then $\mathscr{B}$ sets bad_1 = true and aborts the execution of $\mathscr{F}\prime$ by outputting $(0, \perp)$. Otherwise, $\mathscr{B}$ computes $K^* = i_{i=1}^N K_i$ and checks whether $T_1[0, L\|K^*\|m]$ has already been defined. If the entry was taken, $\mathscr{B}$ sets bad_2 = true and aborts the execution by outputting $(0, \perp)$. If not, $\mathscr{B}$ sets $T_1[0, L\|K^*\|m] = e_1$ and broadcasts K_1. Upon receiving all K_j, $\mathscr{B}$ stops the process if for any $2 \le j \le N$ such that $H_0(K_j) \neq t_j$. $\mathscr{B}$ then broadcasts $\tilde{s}_1$.

Finally, if $\mathscr{F}'$ outputs a valid forgery $(K, \tilde{s})$ on message m under the signer list L, then $\mathscr{B}$ checks $T_1[0, L\|i_{i=1}^N K_i\|m]$. Let J be the index that $h_{1,J} = T_1[0, L\|i_{i=1}^N K_i\|m]$. $\mathscr{B}$ returns $(J, (K, h_{1,J}, \tilde{s}, L))$. The accepting probability of $\mathscr{B}$ is as follows:

$$\left.\begin{aligned} acc_{\mathscr{B}} &= \Pr\left[\mathscr{F}'\text{succeeds} \wedge \overline{\text{bad}_1} \wedge \overline{\text{bad}_2}\right] \\ &\ge \Pr\left[\mathscr{F}'\text{succeeds}\right] - \Pr\left[\overline{\text{bad}_1}\right] - \Pr\left[\overline{\text{bad}_2}\right] \\ &\ge \varepsilon - \frac{(q_h + Nq_s + 1)^2}{2n} - \frac{2q_s(q_h + Nq_s)}{n} \end{aligned}\right. \quad (6)$$

We then construct the algorithm $\mathscr{A}$ that on input $pk^* = P^*$ and runs $\text{Fork}^{\mathscr{B}}(pk^*)$. According to the general forking lemma, it returns $(1, (K, h_{1,J}, \tilde{s}, L), (K, h_{1,J'}, \tilde{s}', L))$ with probability $frk_{\mathscr{A}}$. Note that the discrete logarithm with regard to P^* can be computed through $(K, h_{1,J}, \tilde{s}, L)$, $(K, h_{1,J'}, \tilde{s}', L)$. Therefore, the probability $\varepsilon\prime$ is as follows:

$$\left.\begin{aligned} \varepsilon' &\ge \text{frk}_{\mathscr{A}} \\ &\ge \text{acc}_{\mathscr{B}}\left(\frac{\text{acc}_{\mathscr{B}}}{q_h + q_s} - \frac{1}{n}\right) \\ &\ge \frac{\varepsilon^2}{q_h + q_s} - \frac{4q_s(q_h + Nq_s + 1)^2}{n(q_h + q_s)} \end{aligned}\right. \quad (7)$$

3.3. Experimental Results. We now present the concrete experimental results based on our implementation. We implemented the $MS - SM2$ scheme in Java and ran it on an EC2 instance of type CPU 2.50 GHz with 1 GB RAM. We use the standard SM2 curve and the SM3 hash algorithm. We ran experiments from 2 to 20 parties and compare our results in two-party setting with a related protocol from Zhang et al. [22] in Table 2. Note that [22] is an SM2-based two-party distributed signing protocol, which is slightly different from multisignature in the way that parties should also cooperate in key generation. Moreover, they omit the zero-knowledge proof component in their implementation, and our demo (https://github.com/lhoou/ms-sm2) as a simulation only includes local computation and omits the communication cost in real world. As for multiuser setting, the performances of our scheme are presented in Table 3.

4. Applications to Vehicular Networks

In this section, we describe two potential applications of $MS - SM2$ to vehicular networks. We first show that it can be employed in the architecture of multiple certificate authorities to reduce the number of certificates that are required for devices in the system including on-board units (OBU) and road-side units (RSU). In addition, we also specify its possible usage in the process of V2I communications. The goal is to reduce computation and storage overhead for the units while maintaining security properties.

4.1. Multi-CA Architecture. In vehicular networks, taking C-V2X, for example, certificate authorities usually include organizations for registration, communication authorization, and pseudonym authorization. Specifically, any device that is involved in the network should first require for registration certificate from registration CA and then require for other certificates from different CAs that are needed to send and receive messages in the network.

For instance, a vehicle is required to get a certificate from the registration CA using its unique identity before joining the network. It can then require a pseudonym certificate for the anonymous V2V communication and a secure V2I communication certificate from secure communication CA using its registration certificate. The vehicle can also apply multiple registration certificates from different registration CAs. To simplify the authentication process, the distributed CAs can employ $MS - SM2$ in order to jointly generate only one certificate or one registration certificate for the vehicle at the same time, instead of generating certificates one by one.

4.2. Cooperative V2I Communication. Cooperative communication in vehicular networks has been leveraged to offer various improvements on spectral efficiency, transmission reliability, and reduced transmission delay. Vehicles can cooperate with each other either directly or through an RSU, and the vehicular node which helps the source node to transmit its data is called a helper node or relay node [23].

(i) Cooperative traffic reports: vehicles in the same traffic area, such as in an accident or in a neighborhood, can cooperatively issue a traffic report including awareness messages (CAMs), safety importance, and vehicle heading and transmit a packet to the RSU attached with a $MS - SM2$ signature. The $MS - SM2$ signature can help the RSU to check validity of the packet and also reduce the computation cost of RSU.

(ii) RSU-assisted communication: when a source RSU fails to successfully transmit a packet to the targeted destination, it forwards the packet to the next RSU along the path using the backhaul wired connection. The new RSU relays the received packet to the targeted destination. In this scenario, both the source RSU and relayed RSU can jointly sign the packet

Table 2: Comparison of performances (in milliseconds) between [22] and our scheme in two-party setting.

Scheme	Key Generation	Signing	Verification
Ref. [22]	123.44 ms	152.34 ms	4.17 ms
Ours	51.16 ms	18.04 ms	10.20 ms

Table 3: The performances of our protocol in multiparty setting.

Number of parties	Signature Length (compressed)	Signing (local computation)	Verification
$N = 5$	192 B	13.648 ms	20.916 ms
$N = 10$	352 B	10.021 ms	44.191 ms
$N = 20$	674 B	9.646 ms	49.399 ms

using $MS - SM2$ to convince the target vehicle of the message transmitted, which can also prevent any malicious RSU from sending out frauds without collusion.

5. Conclusions

In this paper, we present a candidate multisignature scheme from the SM2 signature algorithm in the plain public-key model. Compared to a list of individual signatures, the storage volume of $MS - SM2$ signature reduces nearly 50% and the computation cost is relatively low. In addition, we specify in detail some potential applications of the $MS - SM2$ scheme to vehicular networks, especially in the scenario of cooperatively secure communication, with the goal of maximizing performance and compatibility. Because of the high-speed mobility, designing more efficient protocols with fewer communication rounds for vehicular networks is still a challenging research problem.

Data Availability

The data, including algorithms and proofs, used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported in part by the National Key Research and Development Plan of China (Grant no. 2020YFB1005600), Key Area Research and Development Program of Guangdong Province (Grant nos. 2020B0101360001 and 2020B0101090004), National Natural Science Foundation of China (Grant nos. 61825203, U1736203, 61732021, and 61902067), Major Program of Guangdong Basic and Applied Research Project (2019B030302008), and Foundation for Young Innovative Talents in Ordinary Universities of Guangdong (2018KQNCX255).

References

[1] N. Xia and C.-S. Yang, "Vehicular communications: standards and challenges," in *Lecture Notes in Computer Science*, S.-L. Peng, G. Lee, R. Klette, and C.-H. Hsu, Eds., in *Proceedings of the Internet of Vehicles. Technologies and Services for Smart Cities - 4th International Conference, IOV 2017*, vol. 10689, pp. 1–12, Springer, Kanazawa, Japan, November, 2017.

[2] L. Tuyisenge, M. Ayaida, S. Tohmé, and L.-E. Afilal, "Network architectures in internet of vehicles (iov): review, protocols analysis, challenges and issues," in *Proceedings of the Internet of Vehicles. Technologies and Services Towards Smart City - 5th International Conference, IOV 2018*, A. M. J. Skulimowski, Z. Sheng, S. Khemiri-Kallel, C. Cérin, and C.-H. Hsu, Eds., vol. 11253, pp. 3–13, Springer, Paris, France, November, 2018.

[3] A. Yang, J. Weng, N. Cheng, J. Ni, X. Lin, and X. Shen, "Deqos attack: degrading quality of service in vanets and its mitigation," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4834–4845, 2019.

[4] A. Yang, J. Weng, K. Yang, C. Huang, and X. Shen, "Delegating authentication to edge: a decentralized authentication architecture for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2020.

[5] C. Pedro, A. Zúquete, S. Sargento, and M. Luís, "The impact of ECDSA in a VANET routing service: insights from real data traces," *Ad Hoc Networks*, vol. 90, 2019.

[6] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures: extended abstract," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 245–254, Philadelphia, PA, USA, November, 2001.

[7] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pp. 390–399, Alexandria, VA, USA, November, 2006.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, E. Biham, Ed., vol. 2656, pp. 416–432, Springer, Warsaw, Poland, May, 2003.

[9] Y. Zhao, "Practical aggregate signature from general elliptic curves, and applications to blockchain," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (Asia CCS'19)*, pp. 529–538, New York, NY, USA, 2019.

[10] K. Itakura, "A public-key cryptosystem suitable for digital multisignature," *NEC Research and Development*, vol. 71, pp. 1–8, 1983.

[11] B. Ali, J.H. Cheon, and S. Jarecki, "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," in *Proceedings of the 2008 ACM Conference*

on Computer and Communications Security, CCS 2008, pp. 449–458, Alexandria, VA, USA, October, 2008.

[12] C. Ma, J. Weng, Y. Li, R. Deng, and Deng, "Efficient discrete logarithm based multi-signature scheme in the plain public key model," *Designs, Codes and Cryptography*, vol. 54, no. 2, pp. 121–133, 2010.

[13] E. Syta, I. Tamas, D. Visher et al., "Keeping authorities "honest or bust" with decentralized witness cosigning," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 526–545, SP, San Jose, CA, USA, May, 2016.

[14] E. Syta, P. Jovanovic, E. Kokoris-Kogias et al., "Scalable bias-resistant distributed randomness," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP 2017*, pp. 444–460, San Jose, CA, USA, May, 2017.

[15] M. Drijvers, K. Edalatnejad, B. Ford et al., "On the security of two-round multi-signatures," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy, SP 2019*, pp. 1084–1101, IEEE, San Francisco, CA, USA, May 19-23, 2019.

[16] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *IACR Cryptology ePrint Archive*, vol. 68, 2018.

[17] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," *Lecture Notes in Computer Science*, in *Proceedings of the Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 435–464, Brisbane, Australia, December, 2018.

[18] D. Pointcheval and J. Stern, "Security proofs for signature schemes," *Advances in Cryptology—EUROCRYPT '96*, vol. 1070, pp. 387–398, 1996.

[19] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations Trends Privacyand Security*, vol. 2, no. 2-3, pp. 70–246, 2018.

[20] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, New York, NY, USA, January 1987.

[21] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *STOC*ACM, New York, NY, USA, 1988.

[22] Y. Zhang, D. He, M. Zhang, and K.-K. R. Choo, "A provable-secure and practical two-party distributed signing protocol for SM2 signature algorithm," *Frontiers of Computer Science*, vol. 14, no. 3, p. 143803, 2020.

[23] E. Ahmed and H. Gharavi, "Cooperative vehicular networking: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 996–1014, 2018.