Address-based Signature

Handong Cui
Department of Computer Science
The University of Hong Kong
Hong Kong
hdcui@cs.hku.hk

Tsz Hon Yuen
Department of Computer Science
The University of Hong Kong
Hong Kong
thyuen@cs.hku.hk

***Abstract*—In most applications using digital signatures, public keys are used to verify the signatures. The public keys are authenticated by a trusted certificate authority. However, this chain of trust is fragile in the real world. Systems like blockchain and eXpressive Internet Architecture (XIA) proposed to use the hash of a public key as an *address*. Then signatures are verified against *addresses*.**

In this paper, we first formalize the notion of *address-based signatures* to analyze the security of these address-based systems. We give a strong model which considers the security of multiple addresses, with respect to attackers who even know the randomness used by the system developers. The formalization of security model is important to understand the security of real world system.

We propose an efficient and secure construction of address-based signature, which can overcome the existing problems of address-based ECDSA and Schnorr signatures. Then we give two generic constructions of address-based signatures. We find out that these solutions are either not efficient enough (in terms of signature size and address size) or not secure enough.

***Index Terms*—Signature, Address, Blockchain**

I. Introduction

In public key infrastructure, a public key is used to verify a signature. The authenticity of the public key itself is validated by a digital certificate issued from a trusted Certificate Authority (CA). The public key infrastructure is quite cumbersome in practice. Current web browsers carry a lot of pre-installed intermediary certificates issued and signed by CAs. It increases the risk of a key compromise and it is difficult to manage which CAs or intermediate CAs to trust. In September 2018, Google Chrome distrusted millions of certificates issued by Symantec before December 2017, since Google believed that Symantec mis-issued SSL certificates. The chain of trust is difficult to maintain in the real world.

Identity-based cryptography [14] was proposed to deal with this problem, by using a human recognizable string (such as email address) as the user identity. For encrypting a message or verifying a signature, the user identity string is used as the public key. The shortcoming of identity-based cryptography is that a trusted authority is needed to generate the user's identity-based secret key. It is interesting to see how public key cryptography can be adapted to a distributed system without relying on trusted authority.

Expressive Internet Architecture. In 2006, the U.S. National Science Foundation started a project on future Internet architecture. The eXpressive Internet Architecture (XIA) [1], proposed in 2011, is one of the three projects that enters the Future Internet Architecture-Next Phase (FIA-NP) in 2014. XIA pointed out that one of the major problems of today's Internet is the lack of built-in security. They extended the self-certifying identifiers [2] in order to provide intrinsic security of the future Internet.

XIA proposed the use of the hash of public key for host and service as an *address* to provide accountability. To the best of the authors' knowledge, they do not specify any signature scheme. In the XIA prototype released on Github[1], they used 1024-bit RSA keys and the address is 160-bit hash output from the RSA key.

Blockchain Address and Signatures. Blockchain is a distributed ledger system proposed in 2009 [11], in which there is no trusted authority. People can generate as many public keys as they want and can use these keys for transactions. In most blockchain systems, the information about the public key is not stored in the blockchain directly. Instead, only the hash value of the public key (which is usually a 160-bit string) is used as the *address*. Transaction is accepted as long as the signature can be verified with the address.

The idea of *address* is used because the storage in blockchain is considered to be expensive. The transaction fee for each transaction is proportional to the length of the transaction. Therefore, the address is used to represent the recipient of a transaction in the blockchain, instead of using the longer public key.

As a result, an ideal digital signature for blockchain system requires that the sum of address size and the sum of signature size is minimal. It is different from classical public key signatures that only the signature size is considered. In Bitcoin and Ripple, they explicitly reveal the ECDSA public key along with the signature. In Ethereum, the public key can be derived from the ECDSA signature. Therefore, the overall transaction data size of Ethereum is more compact.

A. Address-based Signatures

The address system used in blockchain and XIA actually deviates from classical digital signatures. In this paper, we propose the concept of *address-based signature*. Address is defined as an arbitrary string describing the signer, and the address should be collision resistant between any two signers.

[1]https://github.com/XIA-Project/xia-core

978-1-6654-0392-4/20/$31.00 ©2020 IEEE
DOI 10.1109/TrustCom50675.2020.00119

In this paper, we consider that the address is the hash of the public key. The definitions of the key generation and signing algorithms of address-based signatures are almost the same as classical digital signatures. During the verification of a signature over a message, only the address is given to the verifier. The verifier can still check the validity of the signature. This address-based signature is more suitable to capture the applications in blockchain system or XIA.

Advantages of Address-based Signatures. There are a number of advantages of using address-based signatures.

- Shorter Address. The most obvious advantage of using address-based signature is the shorter address as compared to the size of public key. This advantage will become more significant when we move to high security level (e.g., using 512-bit ECC) or post-quantum secure signature scheme (e.g., lattice-based signature usually has public key size $>$ 1000 bytes) in the future. Shorter address implies lower transaction fee for blockchain applications.
- Compatibility. The idea of verifying against address enables the whole system to use the same address data structure for different signature schemes. It is useful for XIA since different hosts may use different signature schemes. For example, users using ECDSA and Schnorr signature can use the same 160-bit address data structure. Signature schemes using 256-bit ECC and 512-bit ECC can use the same 160-bit address data structure. We only need a few bits to indicate the signature scheme and the bit-length used.

The ECDSA used in Ethereum can be viewed as an address-based signature. We review it in section IV.

B. Our Contributions

There are three main contributions in this paper.

Contribution 1: Formalization of address-based signature and its security model. In this paper, we propose the concept of address-based signatures to capture the real world scenario that address is used to verify a signature, instead of using the public key directly. The address-based ECDSA signature is already used in Ethereum but no formal security model is developed for address-based signature. We believe that it is beneficial to formalize this notion, since more than USD 250 billion of cryptocurrency (as of July 2020) is protected by the notion of address-based signature.

We give a formal notion of address-based signature, and define two security models for it: unforgeability and collision resistance. In both models, the attacker is allowed to obtain the randomness used to generate system parameters. It gives a strong security guarantee by considering the threat from system developers of a blockchain system, who may include a trapdoor in the system parameters.

Contribution 2: Constructing a new, compact address-based signature. In this paper, we propose a compact address-based signature scheme for cryptocurrency, summarized in Algorithm 1. It is as efficient as the ECDSA and Schnorr signature, but without their major drawbacks when used in blockchain. The efficiency comparison is shown in Fig 1.

Algorithm 1: Our Address-based Signature. The secret key is x, the public key is $X = g^x$ and the address is $A = H(X)$.

```
1  Function SIGN($x, m$)
2      $r \leftarrow_s \mathbb{Z}_p$;
3      $R = g^r$;
4      $c = H_{zp}(R, m)$;
5      $z = r^{-1}(c + x) \mod p$;
6      return $\sigma = (R, z)$;
7  Function VERIFY($A, m, \sigma = (R, z)$)
8      $c = H_{zp}(R, m)$;
9      $X = R^z g^{-c}$;
10     if $A = H(X)$ then
11         stop with 1;
12     stop with 0;
```

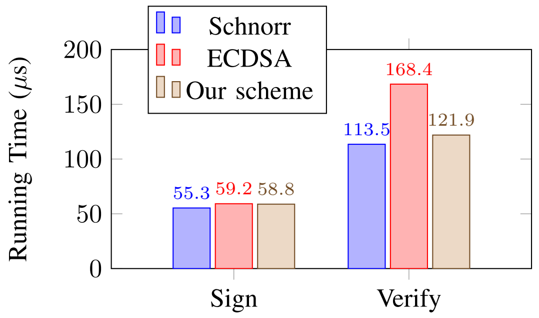

Fig. 1. Comparison of running time.

In terms of security proof, ECDSA is only proved secure in the generic group model [5] or the non-standard bijective random oracle model [9]. The security of our scheme is based on the discrete logarithm assumption in the random oracle model. In terms of blockchain application, Schnorr signature is not compatible with Bitcoin Improvement Protocol (BIP)-32 as shown in [17]. Our scheme is compatible with address using the BIP 32 non-hardened key derivation. Therefore, our proposed scheme is more suitable for address-based system such as blockchain and XIA.

There is also theoretical interest in the security proof of our construction. In the proof, we show that the unforgeability of the signature requires the address hash function H to be *always second-preimage resistant*. This is a non-trivial result, since *always second-preimage resistance* is not implied by *collision resistance* as shown in [12]. This result also demonstrates the importance of formalizing the notion of address-based signature.[2]

Contribution 3: Constructing address-based signature from existing signature schemes. We show that address-based signature can also be constructed from other classical digital signatures using generic constructions GC1 in §VII-A

[2]Practical hash function like SHA256 is likely to be both collision resistant and always second-preimage resistant. Nevertheless, the security proof shows that we need these two distinct properties from the hash function. From the theoretic point of view, it is interesting to see how these two theoretic properties of hash function can affect two different properties (collision resistant and unforgeability) of a practical signature scheme.

Address-based Signature	Signature	Address	Total cost	Major drawbacks
ECDSA (SEC standard)	(r, s)	$H(X)$	672 bits	Malleable, non-standard security proof, repeated computation in verification.
ECDSA (Used in Ethereum)	(r, s, v)	$H(X)$	680 bits	Non-standard security proof.
ECDSA + GC 2 (Used in Bitcoin/Ripple)	(r, s, X)	$H(X)$	936 bits	Malleable, non-standard security proof, longer signature
Schnorr	(R, z)	$H(X)$	680 bits	Not secure in strong known related-key attack [17] (e.g., Bitcoin BIP32)
Key-prefix Schnorr + GC 1	(c, z)	X	776 bits	Longer address
Key-prefix Schnorr + GC 2 (Proposed in Bitcoin BIP)	$(c, z), X$	$H(X)$	936 bits	Longer signature
BLS/BB + GC 1	σ	X	776 bits	Slow verification using pairing, longer address
BLS/BB + GC 2	σ, X	$H(X)$	936 bits	Slow verification using pairing, longer signature
Our construction (§V-A)	(R, z)	$H(X)$	680 bits	-

TABLE I

COMPARISON OF ADDRESS-BASED SIGNATURES. THE TOTAL COMMUNICATION COST IS THE SUM OF ADDRESS AND THE SIGNATURE SIZE. GC STANDS FOR THE GENERIC CONSTRUCTION IN SECTION VII. X IS THE PUBLIC KEY. DETAILS OF THE SYMBOLS ARE EXPLAINED IN SECTION VII.

and GC2 in §VII-B. We compare the efficiency of these schemes in Table I. When implemented in secp256k1, the secret key can be represented by 32 bytes. The compressed public key can be represented by 33 bytes (32 bytes for the x coordinate and 1 byte (0x02 or 0x03) for the sign of the y coordinate). In some cryptocurrencies (e.g., Bitcoin), the address is appended with a checksum value, but some cryptocurrencies do not have it (e.g., Ethereum). In Bitcoin, ECDSA signature is further encoded in DER format. For fair comparison, we omit the sizes of address checksum and signature encoding in Table I.

As summarized in Table I, the most efficient schemes are the address-based ECDSA, Schnorr signature and our construction. For Schnorr signature, it is known to the Bitcoin community that Schnorr signature is not compatible with some Bitcoin Improvement Protocol (BIP), such as BIP32's non-hardened derivation. As a result, key-prefixed Schnorr is preferred in the recent BIP proposal [16]. However, using key-prefixed Schnorr will result in a 50% increase in signature size if we keep the old address structure, or a 60% increase in address size if we keep the old signature structure. The pairing-based signatures like BLS [4] and BB [3] require a larger public key. Therefore, our construction is the optimal solution for address-based signature.

II. BACKGROUNDS

Intractability Assumption. Using the notation in [13], for a security parameter 1^λ, a cyclic group $\mathbb{G}$, its order p and the generator g are classified as the algebraic structure SI of the discrete logarithm (DL) problem instance. The DL assumption holds if there is no probabilistic polynomial time (PPT) adversary can output x when given $X = g^x$ and SI, where x is randomly chosen from $\mathbb{Z}_p$.

Hash Functions. We review some properties of hash functions from [12]. We treat a hash function H as a family of functions, $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$.

We review the *always second-preimage resistance* property as follows. Let m be a number such that $\{0,1\}^m \subseteq M$. Let $\mathcal{A}$ be an adversary. Then a hash function family is *always second-preimage resistant* if the probability:

$$\max_{K \in \mathcal{K}} \left\{ \Pr \begin{bmatrix} M \leftarrow_R \{0,1\}^m; M' \leftarrow \mathcal{A}(M) : \\ (M \neq M') \wedge (H_K(M) = H_K(M')) \end{bmatrix} \right\}$$

is negligible.

We review the *collision resistance* property as follows. A hash function family is *collision resistant* if the probability:

$$\Pr \begin{bmatrix} K \leftarrow_R \mathcal{K}; (M, M') \leftarrow \mathcal{A}(K) : \\ (M \neq M') \wedge (H_K(M) = H_K(M')) \end{bmatrix}$$

is negligible.

III. SECURITY MODEL

A. Address-based Signature

Address-based signature is quite similar to the identity-based signature in the sense that a string is used as the "public key" to verify the signature. The advantage of identity-based signature is that the string can be something meaningful such as email address. However, there is a trusted third party to issue identity-based secret keys. On the other hand, there is no trusted third party to issue address-based keys to different users. Therefore, it makes address-based signature suitable for XIA and blockchain applications.

An address-based signature scheme consists of the following four algorithms:

1) $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, it outputs a master public key mpk.
2) $\mathsf{KeyGen}(\mathsf{mpk})$: It outputs an address addr, a public key pk and a secret key sk.
3) $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$: On input a master public key mpk, a secret key sk and a message m, it outputs a signature σ.
4) $\mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}, m, \sigma)$: On input a master public key mpk, an address addr, a message m and a signature σ, it outputs $(1, \mathsf{pk})$ for valid signature (where pk is the corresponding public key) and outputs $(0, \perp)$ otherwise.

Threat Model. In the system of address-based signature, there are three parties:

- Signer: the one who generates key pairs and signs a message.
- Verifier: the one who verifies the signature. In the case of cryptocurrency, it can be the one receiving the money, or the miners running the consensus algorithms.
- System developer: the one who generates the system parameters mpk. In real world, the system developers usually justify their choice of parameters by using parameters from the standard, or setting parameters as a hash of some meaningful string.

In this paper, we consider the security of each signer against other parties in the system. As in the standard unforgeability requirement, the attacker can obtain signatures for some messages from the target signer. We require that no PPT adversary can forge a signature on a new message. The second security requirement is collision resistance, which means that no PPT adversary can output two distinct secret keys with the same address. In these two security models, we additionally allow the attacker to obtain all randomness used by the system developer to generate the system parameters. It models the attack on unforgeability/collision resistance even against honest-but-curious system developer.

B. Unforgeability

We define our security model similar to the notion of *multi-user existential unforgeability against chosen message attack* [10]. The original security model in [10] does not consider whether the system parameters are generated by a trusted third party or not. In this paper, we take this into consideration and define *trapdoorless multi-user existentially unforgeable under chosen message attack* (TMU-EUF-CMA). In the first step of this security game, the challenger generates the system parameters by running $\mathsf{Setup}(1^\lambda; \rho)$ with randomness ρ. The adversary is additionally given ρ in the first step. This is motivated by the trustless setup assumption used in most public blockchain.

It is easy to see that Schnorr signature is TMU-EUF-CMA secure, since the system parameter par only contains a cyclic group $\mathbb{G}$ with prime order p, its generator g and a cryptographic hash function H_{zp}. The generation of system parameters does not involve a trapdoor. On the other hand, RSA-based signature with common modulus $N = pq$ is not secure in this model.

The notion of *existential unforgeability under chosen message attack* is defined as the following game between a challenger and an adversary.

1) The challenger runs $\mathsf{mpk} \leftarrow \mathsf{Setup}(1^\lambda; \rho)$. The challenger runs $(\mathsf{addr}_i, \mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{mpk})$ for $i \in [1, q_k]$. He stores $(\mathsf{addr}_i, \mathsf{pk}_i, \mathsf{sk}_i)$. The adversary is given mpk and $(\mathsf{addr}_i, \mathsf{pk}_i)$ for $i \in [1, q_k]$, as well as the randomness ρ used to run Setup.
2) The adversary can adaptively issue a sign oracle query on message m_j and pk_{i_j}, for $j \in [1, q_s]$. The challenger answers by $\sigma_j \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{i_j}, m_j)$.
3) Finally, the adversary outputs an index $i^* \in [1, q_k]$, a message m^* and a signature σ^*.

The adversary wins the game if $(1, \cdot) \leftarrow \mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}_{i^*}, m^*, \sigma^*)$ and $(i^*, m^*) \neq (i_j, m_j)$ for all $j \in [1, q_s]$.

Definition 1. An address-based signature scheme is TMU-EUF-CMA if no PPT adversary can win the above game with non-negligible probability, with at most q_s queries to the Sign oracle.

C. Collision Resistance

The second security requirement is the collision resistance of the address. We require that it is difficult to find two secret keys which are represented by the same address. The requirement of collision resistance is also explicitly mentioned in XIA [1].

The notion of *collision resistance* is defined as the following game between a challenger and an adversary.

1) The challenger runs $\mathsf{mpk} \leftarrow \mathsf{Setup}(1^\lambda; \rho)$. The adversary is given mpk, as well as the randomness ρ used to run Setup.
2) Finally, the adversary outputs signatures σ_b^*, messages m_b^* and an address addr^*, for $b = 0/1$.

The adversary wins the game if $(1, \mathsf{pk}_b^*) \leftarrow \mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}^*, m_b^*, \sigma_b^*)$ for $b = 0/1$ and $\mathsf{pk}_0^* \neq \mathsf{pk}_1^*$.

Definition 2. An address-based signature scheme is collision resistant if no PPT adversary can win the above game with non-negligible probability.

IV. Address-based ECDSA Signature

Some of the classical digital signature schemes allow *public key recovery* and can be used as address-based signature directly. One example is ECDSA [15]. Consider x as a secret key and g as a generator of a multiplicative ECC group $\mathbb{G}$, the public key is $X = g^x$. Denote $H : \mathbb{G} \to \{0,1\}^{160}$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$ as cryptographic hash functions and m as a message. The address is $A = H(X)$.

The SEC standard proposed a ECDSA scheme with public key recovery [6]. The verification of the ECDSA signature does not require the use of public key, at a price of repeating the verification computation for up to 4 times (about 49.99% for one time, 49.99% for two times). Ethereum used an address-based ECDSA signature. They used an extra byte to store the information about the different cases in verification, such as using odd or even y-coordinates.

Ethereum used ECDSA as an address-based signature directly. However, there are some well-known drawbacks of the ECDSA signature. Firstly, there is no known security proof of ECDSA in the standard model or in the random oracle model. The EUF-CMA security of ECDSA is only proven in the generic group model [5], or in the bijective random oracle model [9]. Secondly, ECDSA signature is known to be malleable: if (r, s) is a valid signature from address A, then $(r, -s)$ is also a valid signature. The ECDSA malleability is one of the causes of transaction malleability in the Bitcoin system, and a number of related attacks are found [7]. A common trick to deal with it is to use the smaller of s and $-s$ mod p only during the signing phase. Thirdly, as shown in the

SEC standard [6], the public key recovery process is repeated since there are four possible choices of address. Therefore, the computation complexity in verification is 1.5 times higher on average (since the last two cases are rare). In Ethereum, they used this address-based ECDSA and they solved the third issue by having an extra byte of information about the x and y coordinate of K.

V. Compact and Secure Address-based Signature

Since the address-based ECDSA and Schnorr signatures have their own problems when used in blockchain, we would like to consider if we can construct a secure and efficient address-based signature. In this section, we give our construction and prove the security of our scheme in the random oracle model. Then we give the construction of our address-based signature.

A. Our Construction

Our construction can be regarded as the middle ground between Schnorr signature and ECDSA. We remove the extraction of x-coordinate in ECDSA, which makes ECDSA hard to have a standard security proof. On the other hand, we have to prevent the strong known related-key attack [17] in Schnorr signature, while preserving the public key recovery property (unlike the key-prefixed Schnorr signature). As a result, we remove the linear structure of the Schnorr signature $z = r + cx$ and change it to $z = r^{-1}(c + x)$. The randomness r^{-1} is multiplied with the secret key x to prevent the strong known related-key attack.

Considering this requirement of the address-based system, the actual signature scheme is as follows.

Setup. On input the system parameter λ, it generates a cyclic group $\mathbb{G}$ of prime order p. with a generator $g \in \mathbb{G}$. Suppose $H : \mathbb{G} \to \{0,1\}^*$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$ are cryptographic hash functions. It sets $\mathsf{mpk} = (p, \mathbb{G}, g, H, H_{zp})$.

KeyGen. On input mpk, it picks a random $x \in \mathbb{Z}_p$ and computes $X = g^x$. It outputs the secret key x, the public key X and the address $A = H(X)$.

Sign. On input mpk, the secret key x and a message m, it picks a random number $r \in \mathbb{Z}_p$ and computes:

$$R = g^r, \quad c = H_{zp}(R, m), \quad z = r^{-1}(c + x) \mod p.$$

It outputs the signature $\sigma = (R, z)$.

Verify. On input mpk, the address A, a message M and a signature $\sigma = (R, z)$, it computes $c = H_{zp}(R, M)$, $X' = R^z g^{-c}$ and outputs $(1, X')$ if $A = H(X')$. Otherwise, it outputs $(0, \perp)$.

Theorem 1. *Our address-based signature is collision resistant if H is a collision resistant hash function.*

Theorem 2. *Our address-based signature is TMU-EUF-CMA secure if the DL assumption holds in the random oracle model and H is an always second-preimage resistant hash function.*

Proof. Suppose $\mathcal{B}$ is given a DL problem X^* and $SI = (\mathbb{G}, p, g)$. $\mathcal{B}$ tries to solve the DL by using a TMU-EUF-CMA adversary $\mathcal{A}$.

Setup. $\mathcal{B}$ picks cryptographic hash function H_{zp} and H using randomness ρ. $\mathcal{B}$ prepares an empty list $\mathcal{H}_{zp}$. $\mathcal{B}$ sets $\mathsf{mpk} = (p, \mathbb{G}, g, H, H_{zp})$. $\mathcal{B}$ picks a random index $i' \in [1, q_k]$. $\mathcal{B}$ runs $(x_i, X_i, A_i) = \mathbf{KeyGen}(\mathsf{mpk})$ for all $i \in [1, q_k]$ except i'. $\mathcal{B}$ sets $X_{i'} = X^*$ and $A_{i'} = H(X^*)$. $\mathcal{B}$ sends mpk, (X_i, A_i) for $i \in [1, q_k]$ and ρ to the adversary $\mathcal{A}$.

Oracle Query. $\mathcal{B}$ answers the oracle queries as follows:

- Sign: On input a message m_j and public key X_{i_j}, if $i_j \neq i'$, $\mathcal{B}$ returns $\sigma = \mathbf{Sign}(\mathsf{mpk}, x_{i_j}, m_j)$.
  If $i_j = i'$, $\mathcal{B}$ picks some random $z, c \in \mathbb{Z}_p$ and computes $R = (g^c X^*)^{1/z}$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$. (If the value of c is already set in $\mathcal{H}_{zp}$, $\mathcal{B}$ picks another c and repeats the previous step.) $\mathcal{B}$ returns $\sigma = (R, z)$.
- H_{zp}: On input (R, m), if $(c, (R, m))$ is in the list $\mathcal{H}_{zp}$, $\mathcal{B}$ returns c. Otherwise, $\mathcal{B}$ picks a random $c \in \mathbb{Z}_p$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$ and returns c.

Output. Finally, $\mathcal{A}$ outputs an index $i^* \in [1, q_k]$, a message m^* and a signature $\sigma^* = (R^*, z^*)$. If $i' \neq i^*$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ can compute $c^* = H_{zp}(R^*, m^*)$ such that

$$X' = {R^*}^{z^*} g^{-c^*}, \quad A_{i^*} = H(X').$$

If the *always second-preimage resistance* property holds for H, then $X' = X^*$ for the random choice of ρ. Then:

$${R^*}^{z^*} = g^{c^*} X^*.$$

$\mathcal{B}$ rewinds H_{zp} to the point that (R^*, m^*) was queried, and returns a different $c' \neq c^*$. $\mathcal{B}$ eventually obtains another forgery (R^*, z') from $\mathcal{A}$. Therefore, we have

$$(g^{c^*} X^*)^{1/z^*} = (g^{c'} X^*)^{1/z'}.$$

It implies ${X^*}^{z' - z^*} = g^{c' z^* - c^* z'}$.

Next, we argue that $z^* \neq z'$. Suppose on the contrary $z^* = z'$. Then $g^{c^*} X^* = g^{c'} X^*$. It leads to a contradiction with the setting that $c \neq c'$. Therefore we have $z^* \neq z'$.

From ${X^*}^{z' - z^*} = g^{c' z^* - c^* z'}$ and $z^* \neq z'$, we can extract $\log_g X^* = \frac{c' z^* - c^* z'}{z' - z^*}$ as the answer to the DL problem instance. ☐

It is known that the Fiat-Shamir transform has non-malleability in the random oracle model [8]. Therefore, it is straightforward that our signature scheme has non-malleability. Finally, we show the security against strong known relate-key attack under the security model defined in [17].

Theorem 3. *Our address-based signature is Φ^{aff}-EUF-CM-sRKA secure for the class of affine function Φ^{aff} if the DL assumption holds in the random oracle model and H is an always second-preimage resistant hash function.*

Proof. The proof is similar to the proof of TMU-EUF-CMA. We sketch the differences below. In the setup phase, $\mathcal{B}$ samples a number of random affine functions $\phi_k(x) = a_k x + b_k$ for some $a_k, b_k \in_R \mathbb{Z}_p$. For the signing oracle query:

- Sign: On input a message m_j, public key X_{i_j} and index k, if $i_j \neq i'$, $\mathcal{B}$ returns $\sigma = \textbf{Sign}(\mathsf{mpk}, a_k x_{i_j} + b_k, m_j)$. If $i_j = i'$, $\mathcal{B}$ picks some random $z, c \in \mathbb{Z}_p$ and computes $R = (g^c (X^*)^{a_k} g^{b_k})^{1/z}$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$. $\mathcal{B}$ returns $\sigma = (R, z)$.

The output phase is almost the same. □

VI. Hash Functions for Address-based Signatures

From the security proofs in the previous sections, we showed that our new address-based signature requires both the collision resistant property and the always second-preimage resistant property of the hash function H. Theoretically, these two properties are not always the same [12]. In practice, there are a few possible instantiations of H. In this section, we show that there are some differences when we instantiate the hash functions differently in Bitcoin and in Ethereum.

In Ethereum, it first computes a keccak256 hash of the uncompressed X and stores the rightmost 20 bytes. The 20 bytes string is usually expressed as a hexadecimal string for Ethereum address. Ethereum's address does not involve any prefix or key in the computation of the hash function. For unkeyed hash function, the always second-preimage resistant property is equivalent to the traditional second-preimage resistant property. The latter is implied by the collision resistant property [12]. Therefore, only the collision resistant property is required for the security of address-based signature with Ethereum's H function.

In Bitcoin, the public key $X = g^x$ can be represented as a compressed or uncompressed format. One byte of prefix (0x02/0x03/0x04) is put in front of the public key (0x04 represents using uncompressed X as the public key; 0x02 represents using x coordinate of X and y coordinate is even; 0x03 represents using x coordinate of X and y coordinate is odd). The resulting 65/33 bytes are hashed by SHA256 first and then by RIPEMD160 to obtain a 20 bytes string. One byte of prefix is then added to denote the network ID (e.g., 0x00 represents Bitcoin main network, 0x6f represents a test network). It is then appended with 4 bytes of checksum on the previous 21 bytes. The 25 bytes string is usually expressed as a base58 string for Bitcoin address. For the case of Bitcoin, the choice of the prefix representing the type of public key and the prefix representing the network ID are selected by the Bitcoin core developers. Therefore the security of address-based signature with Bitcoin's H function requires both the collision resistant property and the always second-preimage resistant property of H.

From a practical point of view, the function H constructed from popular hash functions (e.g., SHA256 or keccak256) should satisfy both the collision resistant property and the always second-preimage resistant property. However, from a theoretical point of view, the Bitcoin setting requires a slightly stronger assumption.

Algorithm 2: Generic Construction 1

```
1  Function Setup($1^\lambda$)
2      pick $H$ as the encoding function for the public key
         space and $H^{-1}$ is the reverse of $H$;
3      return mpk = $(1^\lambda, H, H^{-1})$;
4  Function KeyGen(mpk)
5      $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$;
6      $A = H(\mathsf{pk})$;
7      return $(\mathsf{sk}, \mathsf{addr} = A)$;
8  Function Sign(mpk, sk, $m$)
9      return $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, m)$;
10 Function Verify(mpk, addr = $A, m, \sigma$)
11     pk = $H^{-1}(A)$;
12     if $1 \leftarrow \mathsf{S.Verify}(\mathsf{pk}, m, \sigma)$ then
13         stop with $(1, \mathsf{pk})$;
14     stop with $(0, \perp)$;
```

VII. Generic Construction of Address-based Signatures

In this section, we give two generic construction of address-based signatures from standard signatures. Then we analyze the signature size of these schemes when instantiation from other signatures, such as EdDSA, BLS and BB signatures.

A. Generic Construction 1

A trivial way to construct address-based signature is to treat the encoding of the public key as address. Suppose (S.KeyGen, S.Sign, S.Verify) is a EUF-CMA secure signature scheme, in the multi-user setting. Therefore, the address function H is a simple encoding function. The first generic construction is given in Algorithm 2. For the security, the collision resistance of this generic address-based signature is trivial: it holds if H is collision resistant. The TMU-EUF-CMA security of this generic address-based signature is based on the TMU-EUF-CMA security of the underlying signature scheme. The major disadvantage of this scheme is that the size of the address is at least as long as the public key.

B. Generic Construction 2

Another generic construction of address-based signature is to send the classical signature σ_c and together with the signer public key pk to the verifier. The address is the hash of pk. The verifier checks the validity of (1) the classical signature σ_c with respect to pk and (2) pk with respect to the signer's address. Therefore, the actual data to be stored by a blockchain includes the signer's signature (σ_c, pk) and the recipient's address. The second generic construction is given in Algorithm 3. For the security, the collision resistance of the second generic address-based signature is trivial: it holds if H is collision resistant. The TMU-EUF-CMA security of this generic address-based signature is based on the TMU-EUF-CMA security of the underlying signature scheme, and also the always second-preimage resistant property of H. The proof is similar to the

Algorithm 3: Generic Construction 2

```
Function SETUP($1^\lambda$)
    pick hash function $H$ for the public key space;
    return $\mathsf{mpk} = (1^\lambda, H)$;
Function KEYGEN($\mathsf{mpk}$)
    $(\mathsf{pk}, \mathsf{sk}') \leftarrow \mathsf{S.KeyGen}(1^\lambda)$;
    $A = H(\mathsf{pk})$;
    return $(\mathsf{sk} = (\mathsf{pk}, \mathsf{sk}'), \mathsf{addr} = A)$;
Function SIGN($\mathsf{mpk}, \mathsf{sk} = (\mathsf{pk}, \mathsf{sk}'), m$)
    $\sigma_c \leftarrow \mathsf{S.Sign}(\mathsf{sk}', m)$;
    return $\sigma = (\sigma_c, \mathsf{pk})$;
Function VERIFY($\mathsf{mpk}, \mathsf{addr} = A, m, \sigma = (\sigma_c, \mathsf{pk})$)
    if $1 \leftarrow \mathsf{S.Verify}(\mathsf{pk}, m, \sigma_c)$ and $A = H(\mathsf{pk})$ then
        stop with $(1, \mathsf{pk})$;
    stop with $(0, \perp)$;
```

proof of our construction. The major drawback of this scheme is that the public key pk gives an extra overhead and induces higher transaction fee.

The case of Bitcoin. In Bitcoin, the most common type of transaction is P2PKH. In a P2PKH transaction, the signature script contains an secp256k1 signature (Sig) and a full public key (PubKey), and it is concatenated to the pubkey script as:

<Sig> <PubKey> OP_DUP OP_HASH160 <PubkeyHash> OP_EQUALVERIFY OP_CHECKSIG

Therefore, the spender's public key (PubKey) is sent as a part of the input signature script [3]. It is the same as our generic construction 2.

C. Candidate Signature Schemes and their Drawbacks

We review some popular digital signature schemes (other than Schnorr and ECDSA) which can be used with the generic constructions. It helps us to evaluate the overall performance of all the combinations with generic constructions.

The key-prefixed variant of Schnorr signature does not allow public key recovery as in the standard Schnorr signatures, since the computation of $c = H(R, M, X)$ requires the knowledge of the public key X. The same holds for the EdDSA signature. The key-prefixed variant of Schnorr signature is shown in Algorithm 4. Without public key recovery, the key-prefixed variant of Schnorr signature has a larger communication cost no matter combining with generic construction 1 and 2, as shown in Table I.

There are a few short signature schemes based on pairings. Denote $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ as a pairing, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p. Denote g_1 and g_2 as the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Consider x as a secret key, the public key is $X = g_1^x$. Denote $H_{G2} : \{0,1\}^* \rightarrow \mathbb{G}_2$ as a cryptographic hash function. For the BLS signature [4]

[3]https://developer.bitcoin.org/devguide/transactions.html

Algorithm 4: Key-prefix Schnorr Signature

```
Function SIGN($x, m$)
    $r \leftarrow_s \mathbb{Z}_p$;
    $R = g^r$;
    $X = g^x$;
    $c = H_{zp}(R, m, X)$;
    $z = r + cx \mod p$;
    return $\sigma = (c, z)$;
Function VERIFY($X, m, \sigma = (c, z)$)
    $R = g^z X^{-c}$;
    if $c = H_{zp}(R, m, X)$ then
        stop with 1;
    stop with 0;
```

and the BB signature [3], the verification of both signatures requires the public key.

We can define $\mathbb{G}_2$ as an ECC group of 256-bit order p and obtains a compact signature of 256 bits only. However, the public key will then be defined as an ECC group of 512-bit order and the public key size will become 512 bits. As shown in Table I, the larger size of public key in BLS/BB offsets the saving in the signature size, no matter combining with generic construction 1 and 2.

VIII. Efficiency Analysis

We implement a number of different classical and address-based signatures by Rust in a MacBook with Intel Core i5 1.4GHz, 16GB RAM. The results are the median running time for running > 300 times. For ECC-based schemes, they all use the same curve secp256k1, which is the curve used in Bitcoin. For pairing-based scheme, they use the BN curves in the bn library in Rust.

Classical Signatures. We compare them with our scheme and the classical signatures in Fig. 1. Our scheme is more efficient than ECDSA in both signing and verification. Schnorr signature is the most efficient in terms of signing, since no modular inverse computation is involved. The verification time of our scheme is similar to that of Schnorr signature. As shown in Table I, there are some disadvantages of using the Schnorr signature or ECDSA. Therefore, our scheme is a better alternative as compared with the Schnorr signature or ECDSA.

Address-based Signatures. We implemented a few address-based signatures based on our generic constructions. The comparison with our scheme is shown in Fig. 2. We also include the address-based Schnorr signature (given in the full version of the paper) for comparison. Generally speaking, the computational performance of our scheme is similar to the family of Schnorr signature. In particular, the ECDSA following the SEC standard for public kye recovery is slower because of the repetition in the verification process.

The schemes constructed from BB and BLS signatures are much slower and hence we do not include them in the figure. The running time for BB signature is 2083 μs and 12503 μs for signing and verification, respectively. The running time for BLS signature is 4610 μs and 13433 μs for signing and verification, respectively. The address-based version (by applying GC1 or GC2) of BB and BLS signatures are slightly

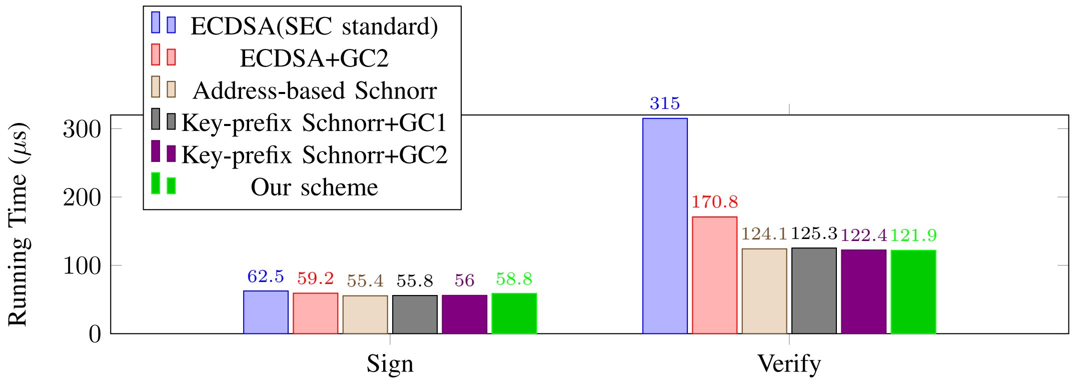

Fig. 2. Comparison of running time of address-based signature.

slower. Hence, they are at least 30 times slower than other schemes in signing and 40 times slower in verification.

Advantages over existing CA approach. In the existing Internet architecture, a trusted certificate authority (CA) is used to issue certificates to websites for secure SSL/TLS connection. The certificate binds the address of the website with its public key. There are real world attacks that malicious CAs or intermediate CAs issue certificates to hackers. The chain of trust problem is difficult to eradicate in the current Internet architecture. By using the hash of public key in XIA, we can have intrinsic security without relying on trusted third party issuing certificates.

From the efficiency point of view, the overhead of using certificate with standard signature is large. A typical X.509 certificate is around 2-4 Kb, which is much larger than a single signature ($\approx$ 512 bits for ECDSA/Schnorr signature). By using address-based signature in XIA, we can save a lot of overhead regarding certificates.

IX. Conclusion

We propose the notion of *address-based signature* to capture the systems that use a short address to authenticate users. Systems like XIA and blockchain use the hash of public key as the address. We formalize the security models for address-based signature. We analyze a number of constructions, such as an address-based signature from Schnorr signature and two generic constructions. We propose a new address-based signature in this paper, which is better than the above schemes in terms of security or total output size (i.e., signature and address size).

Acknowledgment

The second author is supported by the HKU Project #201910159277.

References

[1] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste, “Xia: An architecture for an evolvable and trustworthy internet,” in *HotNets-X*. ACM, 2011, pp. 2:1–2:6.

[2] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, “Accountable internet protocol (aip),” in *ACM SIGCOMM 2008*, V. Bahl, D. Wetherall, S. Savage, and I. Stoica, Eds. ACM, 2008, pp. 339–350.

[3] D. Boneh and X. Boyen, “Short signatures without random oracles,” in *EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 56–73.

[4] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *ASIACRYPT 2001*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., vol. 2248. Springer, 2001, pp. 514–532.

[5] D. R. L. Brown, “Generic groups, collision resistance, and ECDSA,” *Des. Codes Cryptography*, vol. 35, no. 1, pp. 119–152, 2005.

[6] ——, “Standards for efficient cryptography. sec 1: Elliptic curve cryptography,” 2009.

[7] C. Decker and R. Wattenhofer, “Bitcoin transaction malleability and mtgox,” in *ESORICS 2014*, ser. Lecture Notes in Computer Science, M. Kutylowski and J. Vaidya, Eds., vol. 8713. Springer, 2014, pp. 313–326.

[8] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi, “On the non-malleability of the fiat-shamir transform,” in *INDOCRYPT 2012*, ser. Lecture Notes in Computer Science, S. D. Galbraith and M. Nandi, Eds., vol. 7668. Springer, 2012, pp. 60–79.

[9] M. Fersch, E. Kiltz, and B. Poettering, “On the provable security of (EC)DSA signatures,” in *CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 1651–1662.

[10] E. Kiltz, D. Masny, and J. Pan, “Optimal security proofs for signatures from identification schemes,” in *CRYPTO 2016*, ser. Lecture Notes in Computer Science, M. Robshaw and J. Katz, Eds., vol. 9815. Springer, 2016, pp. 33–61.

[11] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[12] P. Rogaway and T. Shrimpton, “Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance,” in *FSE 2004*, ser. Lecture Notes in Computer Science, B. K. Roy and W. Meier, Eds., vol. 3017. Springer, 2004, pp. 371–388.

[13] A. Sadeghi and M. Steiner, “Assumptions related to discrete logarithms: Why subtleties make a real difference,” in *EUROCRYPT 2001*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Springer, 2001, pp. 244–261.

[14] A. Shamir, “Identity-based cryptosystems and signature schemes,” in *CRYPTO ’84*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 47–53.

[15] A. Vanstone, “Responses to NIST’s proposa,” *Communications of the ACM*, vol. 35, pp. 50–52, 1992.

[16] P. Wuille, “bip-schnorr,” 2018, https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki.

[17] T. H. Yuen and S. Yiu, “Strong known related-key attacks and the security of ECDSA,” in *NSS 2019*, ser. Lecture Notes in Computer Science, J. K. Liu and X. Huang, Eds., vol. 11928. Springer, 2019, pp. 130–145.