ECDSA-compatible Delegable Undeniable Signature

Sam Ng and Tomas Tauber
Crypto.com, Hong Kong, China

Keywords: Delegable Undeniable Signature, Designated Confirmer, Blockchain, Privacy.

Abstract: We present the first ECDSA-compatible delegable undeniable signature. Undeniable signature was first introduced by Chaum and Antwerpen. Such signatures cannot be verified without running a zero-knowledge protocol with the signer. Delegable undeniable signature extends this by allowing the signer to delegate the verification ability to a third party. An example use case for delegable undeniable signature is that a trusted party verifies a user's personal information, signs a message and then passes the signature back to the user. If a verifier needs to know that personal information (e.g. an online merchant selling alcohol needs to verify the user's age), the user can run the verification protocol as a delegate to prove the trusted party (e.g. the government) signed that personal information. The verifier will be convinced the signature is genuine, but will not be able to convince others. Our signature scheme is based on standard ECDSA, which is the most common signature scheme in blockchain technology. It is easy to construct (it involves two standard ECDSA signatures) and easy to verify (a simple two-round zero-knowledge protocol). We believe our signature scheme is useful especially in Self-Sovereign Digital Identity.

1 INTRODUCTION

Digital signatures can authenticate message origin, as changes inside signed content nullify the validity of the original digital signatures. However, this universal verification of digital signatures may not be desirable in some applications (e.g. disclosing personal information). Traditional digital signatures, hence, have an inherent conflict between authenticity and privacy.

To address this issue, Chaum and van Antwerpen (Chaum and van Antwerpen, 1989) introduced the notion of undeniable signatures. The key property of undeniable signatures lies in the fact that they require the signer's involvement in the signature verification. By doing so, the signer has control over whom the signed document is being disclosed to. Different variants of undeniable signature schemes have been proposed since the initial work of Chaum and van Antwerpen:

- *Convertible*: the signature can be converted to a publicly verifiable signature (Boyar et al., 1990; Damgård and Pedersen, 1996).
- *Designated Verifier*: the verification protocol can only convince a particular verifier, preventing the verifier acting as an intermediary and launching man-in-the-middle attacks (Jakobsson et al., 1996).
- *Designated Confirmer*: the validity of the signature can be confirmed by the signer or by another third party who knows the private key of a particular public key (Chaum, 1994; Okamoto, 1994).
- *Delegable*: a relaxed version of designated confirmer, the validity of the signature can be confirmed by the signer or by any third party who knows a particular secret value (Gennaro et al., 1997).

At the time of these works, the problem designated confirmer and delegable verification were trying to solve is to prevent the signer from refusing to engage in the verification process. In that case, the signer is not a "trusted" party and the message to be signed is around promises – e.g. something like "I will pay Alice 1 million dollars if X happens". The confirmer or the delegate act as arbitrators.

Recently, there is another emerging use case that makes designated confirmer and delegable verification very interesting: signing sensitive personal information. In this case, the signer is usually a trusted party such as a government office, a bank, or a doctor, who verifies a user's personal information and then signs the corresponding data. In this case, the confirmer or the delegate is actually the data owner. An example use case is to allow users to prove their government verified digital identity while signing-up for

Ng, S. and Tauber, T.
ECDSA-compatible Delegable Undeniable Signature.
DOI: 10.5220/0007949604700477
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 470-477
ISBN: 978-989-758-378-0

online bank accounts. While this setup does not prevent the bank from leaking its users' information, it prevents the bank from acting on behalf of its users and abusing their identities.

The motivating problem we aim to solve and implement is digital identity management in blockchain networks, and we will refer to this use case throughout the paper.

Our Contribution. We present a simple delegable undeniable signature scheme which is based on constraints of two related polynomials. Our signature scheme utilizes the standard ECDSA (Johnson and Menezes, 1998) operations which ease the effort of analyzing its security properties. In addition to that, our scheme can be directly applied to blockchain networks, such as Bitcoin and Ethereum, where ECDSA is employed. These properties make our signature scheme a very suitable candidate in the area of digital identity management in blockchain networks.

2 RELATED WORK

Designated confirmer undeniable signature scheme was first introduced by Chaum (Chaum, 1994). The main motivation was to prevent the signer from refusing or being unavailable to be engaged in the signature verification protocol. Following that, Okamoto presented a formal security model for DC-signature (Okamoto, 1994) which proved that "*designated confirmer signatures exist if and only if public-key encryption exists*".

In Chaum's original paper, the signing protocol is a proof to convince the signature receiver not only the signature is valid, but also the designated confirmer is guaranteed to be able to verify[1] the signature. This assurance is important for their usage scenario because the signature receiver may need the designated confirmer to prove to a judge that the signature is valid when there is a dispute.

Delegable undeniable signature allows anyone who knows the secret value to be able to verify the signature, does not bind the confirmer to a public key. Delegable undeniable signature does not work in Chaum's usage scenario because (1) if the signature receiver receives the secret value, then the receiver can convert the signature into a publicly verifiable signature by disclosing the secret value, which is what the original signer is trying to avoid, (2) if the signature receiver does not receive the secret value, then there is no way (at least no simple way on its own) for the receiver to be assured that the designated confirmer has received and securely stored the secret value.

However, in our usage scenario – signing personal information – the undeniablility is not a benefit of the signer, but a benefit of the message owner. Therefore, the signer does not need to prove to the receiver that a confirmer will later be able to verify the signature because the receiver is the confirmer, and the receiver has no incentive to convert the signature into a publicly verifiable signature.

RSA-Based Undeniable Signature (Gennaro et al., 2000) is closely related to our work except their signature scheme is based on RSA while ours is based on ECDSA. In their scheme, both exponents of RSA are kept secret by the signer, while the public key consists of a composite modulus and a sample RSA signature. The verification process can be delegated to a third party by sharing the verification exponent with the delegate.

Nominative signature (Kim SJ, 1996) is another type of undeniable signature. While undeniable signature can only be verified with the aid of the signer, a nominative signature can only be verified and created with the aid of the nominee. In 2007, Liu et al. presented a nominative signature (Liu et al., 2007) using a witness indistinguishable proof (Feige and Shamir, 1990) with a construction similar to a 3-Move Undeniable Signature Scheme (Kurosawa and Heng, 2005). In 2012, Liu et al. followed up with a one-move Nominative Signature Scheme (Liu and Wong, 2012) but this advanced scheme requires the use of bilinear pairings.

The idea of "Undeniable Certificates" (Gennaro et al., 2001) is that a signer uses an undeniable signature scheme to sign his public key and, thereby, create an undeniable certificate which can be used to verify the signer's digital signature on any document signed using the signer's corresponding private key. Hence, once the undeniable certificate is received by the recipient, the recipient and the signer engage one time in a confirmation protocol or denial protocol to the satisfaction of the recipient that the undeniable certificate has in fact been signed by the signer and thus comprises the signer's certified public key. Thereafter, the recipient can use the certified public key to verify any documents signed by the signer. The problem for using undeniable certificate on digital identity is, if users pass their signed personal information to a malicious verifier, the malicious verifier can pass the signed documents to another verifier, as long as this verifier has previously run the verification protocol with the CA, it can be convinced about the authenticity of the personal data as well.

[1] There is no disavowal protocol in (Chaum, 1994).

Selectively disclosable digital certificates (Benaloh, 2003) enable certifying private data that can be per-field disclosed to selected third parties. They do so by generating a random key for each field and using these keys to encrypt data fields in issued certificates. Owners of certificates can disclose some of the keys to selected third parties and selected third parties can decrypt and verify only the disclosed data fields. Similarly to our work, CAs certify private data authenticity and only one certificate is issued. Selectively disclosable digital certificates are, however, fully transferable. If a third party leaks received random keys, anyone can decrypt and verify private data stored in a certificate. In contrast, our work is non-transferable, so that certified private data cannot be publicly leaked with a strong cryptographic proof.

Hyperledger Indy (West, 2016) is a project that centers around the idea of Self-Sovereign Identity on blockchain networks. The technology that underpins Hyperledger Indy, such as privacy-preserving attribute credentials and verifiable claims, was mainly prototyped in IBM Identity Mixer (Osborne et al., 2017). In Hyperledger Indy, a user obtains certified credentials from authorities (e.g. government or employers) and uses them to prove "claims" about their content to selected third parties (e.g. an insurance company). It employs zero-knowledge proofs (Goldwasser et al., 1989) in unlinkable "claims", such that certified data is not directly revealed to selected verifying third parties. In case of disputes, values are revealed to a trusted third party "inspector" which can decrypt the private data. It employs the CL signature scheme (Camenisch and Lysyanskaya, 2002). This signature scheme allows signing on cryptographic commitments which can be verified. One other similar digital signature scheme is U-Prove (Paquin and Zaverucha, 2011). The approach taken in Hyperledger Indy is different from our work, so we see both approaches as complimentary: standardization that Hyperledger Indy provides can be incorporated with our work, while ours can simplify the workflow where private data needs to be revealed in a claim (e.g. cross-hospital medical record transfer) and the CA / issuer needs to be the same entity as the "inspector".

3 PRELIMINARIES

The core of our signature scheme relies on the standard Elliptic Curve Cryptography (Koblitz, 1987; Miller, 1985) and ECDSA (Johnson and Menezes, 1998) signature scheme. In this section, we briefly present the basic ideas and operations of ECDSA for a later reference in our discussion.

3.1 Quick Review on ECDSA

3.1.1 Signature Generation

If Alice, with private key d_A and public key $Q_A = d_A \times \mathcal{G}$, wants to sign a message $z = hash(m)$ using ECDSA with $\mathcal{G}$ as the base point of the curve, n as the order of the curve, she needs to generate a random number $k \in \{1, n-1\}$ and publish (r,s) where $r,s \neq 0$ and

$$\left.\begin{aligned} (x,y) &= k \times \mathcal{G} \\ r &= x \ (mod\ n) \\ s &= k^{-1}(z + rd_A) \ (mod\ n) \end{aligned}\right\} \quad (1)$$

3.1.2 Signature Verification

Bob wants to verify that a message m is signed with Alice's signature (r,s). He calculates $z = hash(m)$ and computes the following:

$$\left.\begin{aligned} & zs^{-1} \times \mathcal{G} + rs^{-1} \times Q_A \\ &= zk(z+rd_A)^{-1} \times \mathcal{G} + rk(z+rd_A)^{-1} \times Q_A \\ &= zk(z+rd_A)^{-1} \times \mathcal{G} + rk(z+rd_A)^{-1} d_A \times \mathcal{G} \\ &= k(z+rd_A)(z+rd_A)^{-1} \times \mathcal{G} \\ &= k \times \mathcal{G} \\ &= (x,y) \end{aligned}\right\} \quad (2)$$

Now, if $x \ (mod\ n) = r$, then the signature is verified.

4 OUR SIGNATURE SCHEME

4.1 Signature Generation Protocol

The signature generated in our scheme is based on the standard ECDSA with the following modifications:

1. Instead of providing one signature, the signer signs two values, using two random numbers k_1 and k_2 respectively.
2. Instead of signing $z = hash(m)$, the first signature is performed on $z + \alpha$ where $\alpha \in \{1, n-1\}$ is a random number where $z + \alpha \neq 0$ and $z + \alpha^2 \neq 0$, the signer shares the value of α with the prover (data owner) via a secure channel[2].
3. The second signature is performed on $z + \beta$ with $\beta = \alpha^2$.

[2]The secret value can be exchanged face-to-face, by using a USB token, S/MIME, ECIES, NFC, etc. It is out of the scope of this paper.

4. The value of x, instead of $r = x \bmod n$, is published.
5. The value of y is published as well.

In summary, the signer publishes two signatures (x_1, y_1, s_1) and (x_2, y_2, s_2) which are generated according to Eq. 3 and 4, and sends the value of α to the prover (data owner) in a secure and encrypted channel.

$$\left.\begin{aligned}(x_1, y_1) &= k_1 \times \mathcal{G} \\ s_1 &= k_1^{-1}(z + \alpha + x_1 d_A)\end{aligned}\right\} \quad (3)$$

$$\left.\begin{aligned}(x_2, y_2) &= k_2 \times \mathcal{G} \\ s_2 &= k_2^{-1}(z + \alpha^2 + x_2 d_A)\end{aligned}\right\} \quad (4)$$

4.2 Confirmation Protocol

Upon receiving the signature, the verifier calculates $\alpha \times \mathcal{G}$ and $\beta \times \mathcal{G}$ by using Eq. 2:

$$\begin{aligned}&zs_1^{-1} \times \mathcal{G} + x_1 s_1^{-1} \times Q_{\mathcal{A}} \\ &= (z + x_1 d_A) s_1^{-1} \times \mathcal{G} \\ &= (z + x_1 d_A) k_1 (z + \alpha + x_1 d_A)^{-1} \times \mathcal{G} \\ &= (k_1 - k_1 \alpha (z + \alpha + x_1 d_A)^{-1}) \times \mathcal{G} \\ &= k_1 \times \mathcal{G} - \alpha s^{-1} \times \mathcal{G} \\ &= (x_1, y_1) - \alpha s^{-1} \times \mathcal{G}\end{aligned}$$

Therefore

$$\mathcal{A} = \alpha \times \mathcal{G} = s_1 \times (x_1, y_1) - z \times \mathcal{G} - x_1 \times Q_{\mathcal{A}} \quad (5)$$

Similarly

$$\mathcal{B} = \beta \times \mathcal{G} = s_2 \times (x_2, y_2) - z \times \mathcal{G} - x_2 \times Q_{\mathcal{A}} \quad (6)$$

Without knowing the exact value of α and β, the verifier cannot verify if the signature is valid or not. However, no matter what is the exact value of α, as long as the prover can prove $\beta = \alpha^2$, then it becomes very difficult to cheat because

$$z + \alpha = c_1 \quad (7a)$$

$$z + \alpha^2 = c_2 \quad (7b)$$

where c_1 and c_2 are constants

Equation 7a is a straight line while 7b is a quadratic equation, therefore, there are at most 2 intersection points. In other words, if $\beta = \alpha^2$ then there exists only two possible (z, α) pairs that can satisfy Eq. 3 and 4 at the same time.

Table 1: Expected input/output values for the zk-proof.

Challenge	Expected Response
$r \times \mathcal{G}$	$r \times \mathcal{A}$
$r \times \mathcal{A}$	$r \times \mathcal{B}$

4.2.1 Shadow Answer

With c_1 and c_2, it is easy to find the other intersection point (Eq. 8), and we call it the shadow answer denoted by $(\hat{z}, \hat{\alpha})$.

$$\left.\begin{aligned}\hat{\alpha} &= 1 - \alpha \\ \hat{z} &= 2\alpha + z - 1 \\ \hat{z} &= hash(\hat{m})\end{aligned}\right\} \quad (8)$$

If a malicious prover finds the $\hat{m}$, then he can prove to the verifier that the signer signed m or $\hat{m}$ without being detected. However, while finding $\hat{z}$ is easy, finding $\hat{m}$ requires breaking the hash function which is one of basic cryptographic assumption that almost all digital signature schemes rely upon.

4.2.2 Zero-knowledge Proof for $\beta = \alpha^2$

To prove $\beta = \alpha^2$, the prover proves the point $\mathcal{A}$ (Eq. 5) is $\alpha \times \mathcal{G}$ for some value of α. And the point $\mathcal{B}$ (Eq. 6) is $\alpha^2 \times \mathcal{G}$ for the *same value* of α. Equations 7a and 7b already ensure that if this relationship holds, i.e. $\beta = \alpha^2$, then m is either the real m, i.e. the value signed by the signer, or the shadow $\hat{m}$.

The protocol for proving $\beta = \alpha^2$:

1. The verifier chooses a random number $r \in \{1, n-1\}$ and use it as a blinding factor.
2. The verifier randomly either sends $r \times \mathcal{G}$ or $r \times \mathcal{A}$ to the prover, we call this challenge $\mathcal{P}$.
3. The prover returns $\alpha \times \mathcal{P}$ as long as $\mathcal{P}$ is a valid curve point.
4. The verifier checks the result according to Table 1.

4.2.3 Zero-knowledgeness of the Protocol

Consider a simulator, who does not have the knowledge of α but has access to the verifier's random oracle, i.e. the simulator knows the blinding factor r and whether the challenge is derived from $\mathcal{G}$ or $\mathcal{A}$. Obviously, the simulator can convince the verifier $\beta = \alpha^2$ by constructing the expected response according to Table 1. Since a convincing simulator can be constructed without the knowledge of α, the zero-knowledgeness is proved.

4.2.4 Soundness of the Protocol

Consider, instead of $m \ni z = hash(m)$, the prover tells the verifier the message is $m' \ni z' = hash(m')$ where m' is not the real m nor the shadow $\hat{m}$. Then when the verifier calculates the values of $\mathcal{A}$ and $\mathcal{B}$ by the following

$$\begin{aligned}\mathcal{A}' &= s_1 \times (x_1, y_1) - z' \times \mathcal{G} - x_1 \times Q_{\mathcal{A}} \\ &= s_1 \times (x_1, y_1) - z \times \mathcal{G} - x_1 \times Q_{\mathcal{A}} + z \times \mathcal{G} \\ &\quad - z' \times \mathcal{G} \\ &= \alpha \times \mathcal{G} + z \times \mathcal{G} - z' \times \mathcal{G} \\ &= (\alpha + z - z') \times \mathcal{G} \\ \Rightarrow \alpha' &= \alpha + z - z'\end{aligned}$$

$$\begin{aligned}\mathcal{B}' &= s_2 \times (x_2, y_2) - z' \times \mathcal{G} - x_2 \times Q_{\mathcal{A}} \\ &= s_2 \times (x_2, y_2) - z \times \mathcal{G} - x_2 \times Q_{\mathcal{A}} + z \times \mathcal{G} \\ &\quad - z' \times \mathcal{G} \\ &= \beta \times \mathcal{G} + z \times \mathcal{G} - z' \times \mathcal{G} \\ &= (\beta + z - z') \times \mathcal{G} \\ \Rightarrow \beta' &= \beta + z - z'\end{aligned}$$

Note, (1) $\beta' \neq \alpha'^2$ in this case, (2) upon receiving the challenge point $\mathcal{P}$, the prover cannot distinguish whether $\mathcal{P}$ is a blinded $\mathcal{G}$ or $\mathcal{A}'$. The prover can make a guess and he can guess the correct answer with 50% chance. Assuming the prover can make the correct guess, then the prover can

$$return \begin{cases} \alpha' \times \mathcal{P}, & \text{if } \mathcal{P} \text{ is a blinded } \mathcal{G} \\ \alpha'^{-1}\beta' \times \mathcal{P}, & \text{if } \mathcal{P} \text{ is a blinded } \mathcal{A}' \end{cases}$$

Repeat the above process, for instance, 80 times, the chance for the prover cheating without being detected is 2^{-80} which is negligible.

Furthermore, because these 80 challenges can be verified in parallel, it can be sent as one single array. Therefore, the zero-knowledge proof requires only one request and one response only, with each message size around $33 \times 80 = 2640$ *bytes* on a 256-bit ECC curve with the (x, y) coordinate compression.

4.3 Security Requirements

In this section, we summarize the basic security properties that we require from our work. Formal definitions with more goals in a key-only attack model will be presented in the extended version of this paper. Our work scheme, carried out by a certificate authority C, a data owner A, and a verifier (message recipient) B, is secure if the following requirements are satisfied:

- Three modifications (publishing x, y, and signing offset hashed value) of standard ECDSA signature scheme do not affect security properties of C's individual signatures.
- *Unforgeability*: attacker cannot forge a valid signature for a given message
- *Non-transferability*: A's disclosed data and proof to B cannot be transferred to a third party.

5 SECURITY PROPERTIES

In this section, we outline the proofs of the underlying security properties as described in Section 4.3. We divide these proofs into the ones relevant to individual signatures (Section 5.1) where we wish to retain security properties of the standard ECDSA and the ones relevant to joint signatures (Section 5.2) where their joint properties and non-transferability of our scheme are shown.

5.1 Properties of Individual Signatures

In this section, we show that modifications in our scheme made for individual signatures do not affect the overall security model and assumptions of ECDSA (Vaudenay, 2003). If we compare the unmodified ECDSA signature generation (1) and our signature generation, as shown in (3) and (4), we find three differences:

- The value of x, instead of $r = x \ (mod \ n)$, is published.
- The value of y is published.
- The signature is performed on $hash(m) + offset$ instead of $hash(m)$.

In the rest of this section, we look at these modifications and show they do not affect the security properties of ECDSA.

Lemma 5.1. *Publishing x and y does not affect the security model of ECDSA.*

Proof (Sketch). We refer to Eq. 2 where the value $(x, y) = k \times \mathcal{G}$ is calculated as a part of the ECDSA signature verification process. Let's call Eq. 2 as v, i.e. $v(r, s) = (x, y)$.

Assume there is an attack method $T(x, y)$ on our scheme that relies on x and y, then it can be used to attack ECDSA by $T(v(r, s))$. Therefore, disclosing x, y

does not affect the security model because if such attack exists, it is not specific to our signature scheme and can be applied to ECDSA as well. □

Lemma 5.2. *Signing 'hash(m) + offset' instead of 'hash(m)' does not affect the security model of ECDSA.*

Proof (Sketch). First, assume there is a cryptographic hash function $hash_1$ and another hash function $hash_2 = hash_1 + offset$. Assume there exists an attack method T on the hash function $hash_2$ with input x and output y, i.e. the following can reveal some information:

$$T(hash_2, x, y = hash_2(x)).$$

Obviously, T can also be used to attack $hash_1$ by

$$T(hash_1 + offset, x, y = hash_1(x) + offset).$$

Therefore, adding or subtracting an offset does not affect the security properties of a hash function.

The "s" equations in ECDSA and our signature scheme can be defined as follows:

$$s = k^{-1}(hash_1(m) + rd_a) \longleftarrow \text{ECDSA}$$
$$s = k^{-1}(hash_2(m) + rd_a) \longleftarrow \text{Our scheme}$$
$$\text{where } hash_2 = hash_1 + offset$$

Since adding or subtracting an offset does not affect the security properties of a hash function, signing $hash(m) + \alpha$ and $hash(m) + \beta$ do not affect the security model of ECDSA.

□

Theorem 5.3. *Modifications of ECDSA signature generation in our scheme signature do not affect the overall security model of ECDSA.*

Proof (Sketch). The proof follows from Lemmas 5.1 and 5.2. □

5.2 Properties of Joint Signatures

In this section, we look at the properties that are relevant to publishing two signatures and a verification procedure using them.

Lemma 5.4. *Unforgeability: attacker cannot forge a valid signature for a given message.*

Proof (Sketch). Assuming an attacker has a list of n existing valid signatures, along with the plain text m and the corresponding α value. By using m and α, the attacker can calculate $c = hash(m) + \alpha$. This c value is the input to the standard ECDSA. In other words, the attacker has a list of $(c, sign(c))$ pair.

To forge a signature in our scheme for message m_f, the attacker can just randomly pick one valid signature, set $\alpha_f = c_1 - hash(m_f)$. Now if the attacker can find another $c_2 = hash(m_f) + \alpha_f^2$, then the attacker can forge a valid signature for m_f, and he/she can simply lookup this c_2 value from the list.

Assuming this is a 256-bit ECC curve, and the CA signs 1000 signatures per second for 50 years, then there will be 2^{41} ECDSA signatures available, the chance for being able to successfully forge a signature in our scheme by reusing these signatures is[3]

$$2^{-256} \times 2^{41} \times 2^{41} = 2^{-174}$$

which is much smaller than randomly choosing two signatures and then running the zero-knowledge proof by cheating (assuming 80 rounds of zk-messages).

On the other hand, if we randomly pick any two (c_1, c_2), the pair is a valid signature of some z_f and α_f (if a solution exist):

$$z_f + \alpha_f = c_1 \tag{9}$$
$$z_f + \alpha_f^2 = c_2 \tag{10}$$

However, since it is computational infeasible to find $m_f \ni hash(m_f) = z_f$, we conclude it it is not possible to find an existential signature by combining two existing signatures.

□

Lemma 5.5. *Non-transferability: except for the certificate owner and issuer, no one can prove to another party that the certificate owner produced a signature proof.*

Proof (Sketch). Consider the case if the verifier wants to disclose all data obtained from the prover during the interactive zk-proof: as shown in Section 4.2.3, a simulator with access to the verifier's random oracle, i.e. the verifier himself, can generate the correct responses for the challenges. Therefore, the recorded data of the zk communication, without the interaction and a secure random oracle, will not convince anyone.

Consider the case if the verifier (message recipient) wants to run the interactive zk-proof to another party: as shown in Section 4.2.4, without the correct values of α and β, the verifier can only cheat with a negligible chance. We conclude this is not possible.

□

[3]To be more accurate, the term 2^{-256} should be replaced with the order of the curve which is a little bit smaller.

Theorem 5.6. *Assuming the ECDSA security assumptions and those modifications of ECDSA signature generation in our scheme do not affect the overall security model of ECDSA, our scheme satisfies the properties of unforgeability and non-transferability.*

Proof (Sketch). The proof follows from Theorem 5.3 and Lemmas 5.4 and 5.5. □

6 KNOWN LIMITATION

6.1 No Disavowal Protocol

In a standard undeniable signature scheme, a failed confirmation protocol does not imply signature invalidation. Signature invalidation has to be performed by the disavowal protocol. For example, if Alice claims Susan, using an undeniable signature scheme, signed "Susan will pay Alice one million dollars if X happens," Susan can prove to the judge that the signature is not signed on this message, without disclosing what it really is.

Luckily, the disavowal protocol is not a critical feature in our potential use case – signing personal information – because the message is not a commitment on what the signer will perform in the future.

6.2 No Designated Verifier

Undeniable signature can be vulnerable to the man-in-the-middle attack if the proof is not designated to a particular verifier. Generally speaking, an interactive zero-knowledge proof can convince the one who generates the random numbers. Therefore, if the verifier acts as a proxy and is controlled by another party, the proof will be convincing to this party as well. Therefore, for the use case of signing personal information, the prover would want to disclose the information to authenticated parties only, but this feature is missing in the current version of our work.

7 CONCLUSION

Our signature scheme is a simple undeniable signature scheme for issuing certificates about private data without publicly disclosing them. It enables data owners to selectively disclose their private data in a non-transferable way. For future work, besides extending our analysis, we plan to improve the algorithm to be a designated verifier signature and reduce the communication data size.

REFERENCES

Benaloh, J. (2003). Selectively disclosable digital certificates.

Boyar, J., Chaum, D., Damgård, I., and Pedersen, T. (1990). Convertible undeniable signatures. In *Advances in Cryptology-CRYPT0' 90*, pages 189–205. Springer Berlin Heidelberg.

Camenisch, J. and Lysyanskaya, A. (2002). A signature scheme with efficient protocols. In *Security in communication networks*, pages 268–289. Springer.

Chaum, D. (1994). Designated confirmer signatures. In *Advances in Cryptology — EUROCRYPT'94*, pages 86–91. Springer Berlin Heidelberg.

Chaum, D. and van Antwerpen, H. (1989). Undeniable signatures. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 212–216.

Damgård, I. and Pedersen, T. (1996). New convertible undeniable signature schemes. In *Advances in Cryptology — EUROCRYPT '96*, pages 372–386. Springer Berlin Heidelberg.

Feige, U. and Shamir, A. (1990). Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing - STOC '90*. ACM Press.

Gennaro, R., Krawczyk, H., and Rabin, T. (1997). RSA-based undeniable signatures. In *Advances in Cryptology — CRYPTO '97*, pages 132–149. Springer Berlin Heidelberg.

Gennaro, R., Krawczyk, H. M., and Rabin, T. D. (2001). Undeniable certificates for digital signature verification.

Gennaro, R., Rabin, T., and Krawczyk, H. (2000). RSA-based undeniable signatures. *Journal of Cryptology*, 13(4):397–416.

Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208.

Jakobsson, M., Sako, K., and Impagliazzo, R. (1996). Designated verifier proofs and their applications. In *Advances in Cryptology — EUROCRYPT '96*, pages 143–154. Springer Berlin Heidelberg.

Johnson, D. B. and Menezes, A. J. (1998). Elliptic curve dsa (ecsda): An enhanced dsa. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 13–13.

Kim SJ, Park SJ, W. D. (1996). Zero-knowledge nominative signatures. *Pragocrypt*, pages 380–392.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–203.

Kurosawa, K. and Heng, S.-H. (2005). 3-move undeniable signature scheme. In *Lecture Notes in Computer Science*, pages 181–197. Springer Berlin Heidelberg.

Liu, D. Y. W. and Wong, D. S. (2012). One-move convertible nominative signature in the standard model. In *Provable Security*, pages 2–20. Springer Berlin Heidelberg.

Liu, D. Y. W., Wong, D. S., Huang, X., Wang, G., Huang, Q., Mu, Y., and Susilo, W. (2007). Formal definition

and construction of nominative signature. In *Information and Communications Security*, pages 57–68. Springer Berlin Heidelberg.

Miller, V. S. (1985). Use of elliptic curves in cryptography. In *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg.

Okamoto, T. (1994). Designated confirmer signatures and public-key encryption are equivalent. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 61–74, London, UK, UK. Springer-Verlag.

Osborne, M., Lehmann, A., Baentsch, Michael, V., Tamas, and Towa, P. (2017). Ibm identity mixer.

Paquin, C. and Zaverucha, G. (2011). U-prove cryptographic specification v1. 1. *Technical Report, Microsoft Corporation*.

Vaudenay, S. (2003). The security of dsa and ecdsa. In *International Workshop on Public Key Cryptography*, pages 309–323. Springer.

West, R. (2016). Indy documentation index.