EPiC Series in Computing
Volume 106, 2025, Pages 144–154
Proceedings of the 20th Asia Joint Conference on Information Security

A Smart Contract-Based Non-Transferable Signature Verification System using Nominative Signatures

Hinata Nishino[1], Kazumasa Omote[2], and Keita Emura[13]*

[1] Kanazawa University, Kanazawa, Ishikawa, Japan
[2] University of Tsukuba, Tsukuba, Ibaraki, Japan
[3] National Institute of Advanced Industrial Science and Technology, Koto, Tokyo, Japan.
k-emura@se.kanazawa-u.ac.jp

Abstract

Nominative signatures allow us to indicate who can verify a signature, and they can be employed to construct a non-transferable signature verification system that prevents the signature verification by a third party in unexpected situations. For example, this system can prevent IOU/loan certificate verification in unexpected situations. However, nominative signatures themselves do not allow the verifier to check whether the funds will be transferred in the future or have been transferred. It would be desirable to verify the fact simultaneously when the system involves a certain money transfer such as cryptocurrencies/cryptoassets. In this paper, we propose a smart contract-based non-transferable signature verification system using nominative signatures. We pay attention to the fact that the invisibility, which is a security requirement to be held for nominative signatures, allows us to publish nominative signatures on the blockchain. Our system can verify whether a money transfer actually will take place, in addition to indicating who can verify a signature. We evaluate the gas cost when a smart contract runs the verification algorithm of the Hanaoka-Schuldt nominative signature scheme (ACNS 2011, IEICE Trans. 2016).

1 Introduction

1.1 Background

There are many situations where it is necessary to verify who has issued certain information. Generally, using a digital signature scheme makes it possible to verify the issuer of the information. However, since the usual digital signature scheme allows for public verification, there is a possibility that the signature verification could be executed in unexpected situations for the person handling the information. For example, even information that one does not want to disclose to third parties, such as debts, could be verified for its validity.

Here, we introduce an advertisement of investment contracts as a specific example where the public verifiability of signatures becomes a problem (See Fig 1). Assume that investment contracts are made between a business operator who conducts business and an investor who

*Corresponding Author

makes investments. When a business operator seeks to receive more investments, it is effective to appeal to other capitalists that they have received investments from investors. When a business operator appeals that they have received investments, it is assumed that the business operator produces and publishes a signature on the contract so that the information can be verified by third-party capitalists as being issued by the business operator.

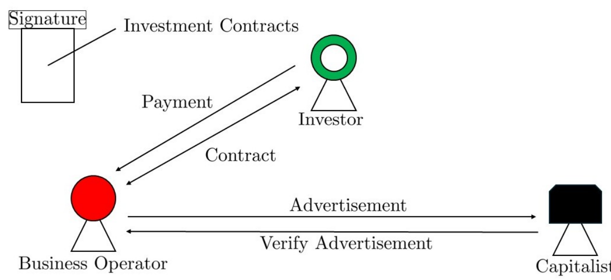

Figure 1: Advertisement of Investment Contracts

Specifically, by attaching a signature to a contract indicating that the business operator and the investor have entered into a financial contract, it serves as evidence of the contract between the business operator and the investor. When the business operator uses the contract to advertise to capitalists, the capitalists can indeed verify that the information was issued by the business operator.[1] At the first sight, the system seems to be feasible when the investor will complete the financial assistance. However, due to the public verifiability, anyone, who obtains the business operator's verification key and the signed contract, can verify the signature. This raises concerns that capitalists could use this investment information although the business operator does not know this fact. A capitalist could cause trouble by proving the validity of the contract to a third party without the business operator's and investor's awareness. For example, a capitalist might commit investment fraud based on the investment information. Therefore, it is necessary to

appropriately control who can verify the signature.

Additionally, the investor has an incentive to attract more investments to the business operator they are investing in, as it increases the likelihood of the business's success. Note that considering the possibility that the business operator might solicit funds through false advertising, the verifiability of the business operator's investment should also involve the investor.

From the perspective of the capitalist, it is desirable to decide whether the capitalist investments or not after confirming the investor has actually invested. It might be possible to verify this fact by some means after funds have actually been transferred from the investor to the business operator. However, a certain time lag is expected between the conclusion of the contract and the transfer of funds by considering the time required for the investor to prepare the funds. It is unreasonable to wait for the transfer before starting the advertisement when the business operator wants to advertise to attract further investments by using the fact of the transfer from the investor. Furthermore, since the signature is independent to the funds, and the transfer is conducted between the business operator and the investor, the capitalist cannot confirm the presence or absence of the transfer through signature verification. Therefore, even

[1]Strictly speaking, it is necessary to separately verify that the signature verification key belongs to the business operator using PKI.

if the investor will not transfer the funds contrary to the contract, the capitalist cannot verify this. This could result in the capitalist bearing the risk unilaterally. To solve these problems, a method that can prove that

the funds will be transferred in the future, even before the transfer,

is necessary. The usual publicly verifiable signature scheme does not meet these requirements.

1.2 A Naive Solution and Its Limitation

As a naive and simple solution, we consider to employ nominative signatures [13]. In a nominative signature scheme, a new entity called Nominee is defined. A signer and a nominee jointly generate a signature which is called a nominative signature, and the nominee proves the validity of the nominative signature through an interactive protocol with the verifier. Without the nominee, even the signer or a verifier who has once participated in verification cannot prove the verification result to a third party. By taking advantage of this property, we can expect to construct a system in which the business operator and the investor can control the verifiability of signed contracts. Specifically, the business operator is assigned as a signer and the investor as a nominee, and they jointly generate a nominative signature (on a contract). The business operator asks the investor to prove the validity of the contract to the capitalist, and the investor communicates with the capitalist to prove it. Due to the property of nominative signatures, the capitalist cannot prove the validity of the contract to a third party. The above system allows us to appropriately control who can verify the signatures. Note that nominative signatures themselves are still independent to the funds provided. Due to this reason, nominative signatures are not effective to prove that the funds will be transferred in the future before the funds are transferred.

1.3 Our Contribution

In this paper, we propose a non-transferable signature verification system.

1. In addition to a nominative signature scheme, we employ smart contracts to connect a signature with money transfers.
    - Concretely, we connect a signature with money transfers by the following procedure: A business operator and an investor jointly generate a nominative signature on *the program source code* of the smart contract.
2. We also evaluated the performance of the proposed system using pre-compiled contracts provided by smlXL.inc [19].
    - We employ the pairing-based Hanaoka-Schuldt nominative signature scheme [8] in our evaluation.

Smart contracts allow various processes to be executed according to pre-defined and publicly disclosed contracts. Therefore, it is expected that the system will be configured in such a way that the capitalist can verify the transfer as well. The proposed system is briefly explained as follows.

1. We introduce a smart contract that manages a transfer.

2. A business operator and an investor jointly generate a nominative signature on the program source code of the smart contract, and store the nominative signature on the smart contract.

3. An operation using a nominative signature in the smart contract is run after the investor is ready to transfer, which triggers the transfer.

A wallet on the blockchain can be viewed by anyone, so the capitalist can check whether or not the transfer has been made. Also, the capitalist can check whether the funds will be transferred in the future or not (by the trigger described in the program), even before the transfer, by checking the program source code of the smart contract. Note that it is easy to verify whether or not the transfer to the business operator has been done by checking the transaction after the transfer, and that the investor cannot illegally withdraw the funds from the business operator's wallet since the funds are locked by the wallet. In the proposed system, a nominative signature is stored on a smart contract and disclosed to the public. However, no information of the business operator and the investor is revealed due to the invisibility of the underlying nominative signature scheme. Moreover, due to the security of the underlying nominative signature scheme, it is guarantees that both the business operator and the investor agree on the creation of the nominative signature. It also employed to prevent the business operator to run the smart contract without the investor's approval, and to prevent the investor from proving to a capitalist that they are willing to make an investment.

1.4 Related Work

Signature Schemes with Controllable Verifiability. In addition to nominative signatures, many other signature schemes with controllable verifiability have been proposed. Undeniable signatures [4] require that the verifier needs to run an interactive protocol with the signer, and it can prevent signatures from being verified without the signer's knowledge or consent. Furthermore, the signer can claim that they produced a signature, but cannot claim that they did not produce a signature when they have produced the signature. In some cases, it is desirable to be able to use an undeniable signature together with a conventional (i.e., publicly verifiable) signature. Therefore, a convertible undeniable signature scheme [2] has also been proposed where the signer can convert a previously issued undeniable signature to a publicly verifiable signature. In undeniable signatures, the signer is required to be always involved to the verification process that increases the workload of the signer. To solve this problem, confirmer signatures have been proposed [3] that introduce a third entity called a confirmer who runs the interactive verification protocol with the verifier. Online untransferable signatures [14] have also been proposed as a method to prevent a third party from verifying the validity/invalidity of a signature by observing the interactive protocol run between a signer and a verifier online. In designated verifier signatures [11], the signer designates a verifier, and only the designated verifier can verify the signature, and the signer is not involved in the signature verification process itself. Furthermore, the designated verifier cannot convince a third party of the validity/invalidity of the signature.

As a kind of these signatures with controllable verifiability, nominative signatures have been proposed [13]. In nominative signatures, which are the dual relationship with undeniable signatures [4], the signature holder called nominee can prove the validity/non-authenticity of the signature to a third party. Several nominative signature schemes have been proposed so far [9, 10, 21]. Schuldt and Hanaoka [18] formalized a security definition of nominative signatures (we mainly refer to the full version [8]). As an application of nominative signatures, a privacy-enhanced access log management mechanism in single-sign on (SSO) systems has been proposed [15, 16]. The system employs a nominative signature as an access log stored on the system. Due to the invisibility of the underlying nominative signature scheme, no information of access user is reveled from the log whereas users can prove that they have accessed the system. The proposed system is inspired by the SSO system because the invisibility is attractive to preserve privacy in blockchain, especially in a public blockchain where anyone can observe information stored on the blockchain.

2 Roles of ECDSA in Ethereum

In Ethereum, ECDSA signatures are required for the transfer of cryptocurrencies/cryptoassets. Note that the underlying ECDSA signature scheme in Ethereum is not the usual one and is called recoverable ECDSA in Ethereum Yellow Paper [20] where it provides the key recovery property: the verification key is recovered from a signature and a message. The following is a brief overview of the ECDSA signature verification process in Ethereum. There are two entities: a sender and a receiver of the funds. The sender generates an ECDSA signature on a transaction M using own secret signing key where (hash value of) the public verification key is the address of its wallet, and sends the transaction with the ECDSA signature to the receiver. The receiver recovers the verification key from the signature and the message. If (the hash value of) the recovered verification key matches the sender's address, it is accepted as a valid signature.

In the proposed system, ECDSA signatures are generated when funds are transferred from an investor's wallet. To avoid any confusion, we do not explicitly specify the recovery phase in the proposed system and employ the following syntax. Let ($\mathsf{ECDSA.KeyGen}$, $\mathsf{ECDSA.Sign}$, $\mathsf{ECDSA.Verify}$) be the ECDSA scheme. The key generation algorithm is denoted as $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{ECDSA.KeyGen}(1^\lambda)$ where $\lambda \in \mathbb{N}$ is a security parameter, vk is a verification key, and sk is a signing key. The signing algorithm is denoted as $\sigma_{\mathsf{ECDSA}} \leftarrow \mathsf{ECDSA.Sign}(\mathsf{sk}, M)$ where M is a message (transaction) to be signed and σ_{ECDSA} is a ECDSA signature. The verification algorithm is denoted as $1/0 \leftarrow \mathsf{ECDSA.Verify}(\mathsf{vk}, \sigma_{\mathsf{ECDSA}}, M)$.

3 Nominative Signatures

In this section, we give the definition of a nominative signature scheme given by Hanaoka and Schuldt [8].

3.1 Syntax

Definition 1 (Syntax of Nominative Signatures [8]).

Setup: The setup algorithm takes a security parameter 1^λ as input, and outputs a public parameter par.

$\mathsf{KeyGen_S}$: The signer's key generation algorithm takes par as input, and outputs a public/secret key pair $(\mathsf{pk_S}, \mathsf{sk_S})$.

$\mathsf{KeyGen_N}$: The nominee's key generation algorithm takes par as input, and outputs a public/secret key pair $(\mathsf{pk_N}, \mathsf{sk_N})$.

Sign**:** The signing algorithm takes par, $\mathsf{pk_N}$, a message to be signed m, and $\mathsf{sk_S}$ as input, and outputs a signature generation message δ. This algorithm is run by the signer who has $\mathsf{sk_S}$.

$\mathsf{Receive}$**:** The nominative signature generation algorithm takes par, $\mathsf{pk_S}$, m, δ, and $\mathsf{sk_N}$ as input, and outputs a nominative signature σ. This algorithm is run by the nominee who has $\mathsf{sk_N}$.

$\mathsf{Convert}$**:** The conversion algorithm takes par, $\mathsf{pk_S}$, m, σ, and $\mathsf{sk_N}$ as input, and outputs a verification token tk_σ. This algorithm is run by the nominee who has $\mathsf{sk_N}$.

$\mathsf{TkVerify}$**:** The token verification algorithm takes par, $\mathsf{pk_S}$, $\mathsf{pk_N}$, m, σ, and tk as input, and outputs either accept or reject. Anyone can run the algorithm because it does not take a secret key as input.

$(\mathsf{Comfirm}, \mathsf{V_C})$**:** The interactive protocol for nominative signature confirmation takes as input par, $\mathsf{pk_S}$, $\mathsf{pk_N}$, m, and σ as common input, the $\mathsf{Comfirm}$ algorithm takes $\mathsf{sk_N}$ as input, and outputs either accept or reject. This protocol is run by the nominee who has $\mathsf{sk_N}$ and the verifier.

$(\mathsf{Disavow}, \mathsf{V_D})$**:** The interactive protocol for nominative signature disavowal takes par, $\mathsf{pk_S}$, $\mathsf{pk_N}$, m, and σ as common input, the $\mathsf{Disavow}$ algorithm takes $\mathsf{sk_N}$ as input, and outputs either accept or reject. This protocol is run by the nominee who has $\mathsf{sk_N}$ and the verifier.

An outline of a nominative signature scheme is as follows (See Fig. 2). Let $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$. A signer runs $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{par})$. A nominee runs $(\mathsf{pk_N}, \mathsf{sk_N}) \leftarrow \mathsf{KeyGen_N}(\mathsf{par})$. A signer and a nominee collaboratively generate a nominative signature on a message m as follows. The signer generates a signature generation message $\delta \leftarrow \mathsf{Sign}(\mathsf{par}, \mathsf{pk_N}, m, \mathsf{sk_S})$, and sends m and δ to the nominee. The nominee generates a nominative signature $\sigma \leftarrow \mathsf{Receive}(\mathsf{par}, \mathsf{pk_S}, \mathsf{m}, \delta, \mathsf{sk_N})$. To verify (m, σ), the nominee and a verifier run $\{\mathsf{Comfirm}(\mathsf{sk_N}), \mathsf{V_C}\}(\mathsf{par}, \mathsf{pk_S}, \mathsf{pk_N}, m, \sigma)$ or $\{\mathsf{Disavow}(\mathsf{sk_N}), \mathsf{V_D}\}(\mathsf{par}, \mathsf{pk_S}, \mathsf{pk_N}, m, \sigma)$. The nominee can convert σ to a publicly verifiable signature (called a token) $\mathsf{tk}_\sigma \leftarrow \mathsf{Convert}(\mathsf{par}, \mathsf{pk_S}, m, \sigma, \mathsf{sk_N})$. Anyone can verify tk_σ by running $\mathsf{TkVerify}(\mathsf{par}, \mathsf{pk_S}, \mathsf{pk_N}, m, \sigma, \mathsf{tk}_\sigma)$.

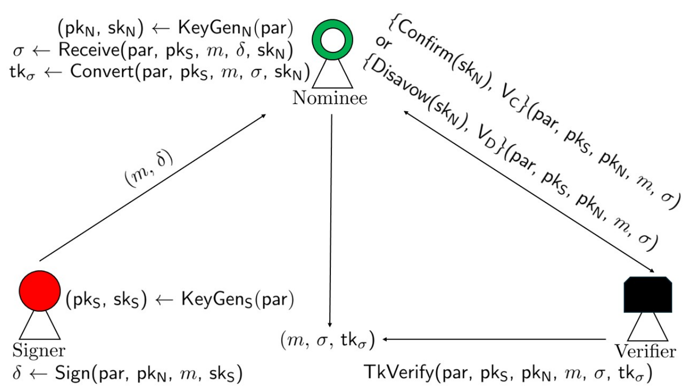

Figure 2: Nominative Signatures

3.2 Security of Nominative Signatures

Here, we briefly introduce the security of nominative signatures (See [8] for more details) and briefly introduce the roles of each security in the proposed system.

Invisibility. It guarantees that even a malicious signer cannot distinguish between an honestly generated nominative signature and a random value. Therefore, an adversary who has the signer's secret key sk_S and other information, and even a verifier who has once participated in the verification cannot know the correspondence between the message and the nominative signature. In the proposed system, this security is employed to guarantee that the information of business operator and investor is not leaked when nominative signatures are stored on the blockchain.

Unforgeability. It guarantees that a legitimate nominative signature can be obtained only through a signer. In other words, even a malicious nominee cannot generate a nominative signature without communicating with the signer. In the proposed system, this security is employed to ensure that the investor cannot independently generate legitimate nominative signatures associated with the business operator, i.e., it guarantees that both the business operator and the investor have agreed to produce nominative signatures (together with Security against malicious signers as described below).

Security against malicious signers. It guarantees that a legitimate nominative signature is produced only when the nominee is participated in. In other words, even a malicious signer cannot generate a legitimate nominative signature without communicating with the nominee. In addition, it guarantees that the signer cannot generate legitimate verification tokens and cannot prove the verification result to a third party through an interactive protocol. In the proposed system, this security is employed to ensure that the business operator cannot generate a legitimate nominative signature related to the investor by itself.

Protocol Security. It also was referred as non-transferability. It guarantees that even a verifier who has executed an interactive protocol with the nominee and has verified the signature cannot prove the verification result to a third party. To satisfy the property, the interactive protocol needs to be zero-knowledge. In the proposed system, this security guarantees that a capitalist whose investor certifies the verification result of a nominative signature cannot prove the verification result to a third party.

4 Proposed System

In this section, we propose a smart contract-based non-transferable signature verification system using nominative signatures. In the proposed system, the business operator acts as the signer and the investor acts as the nominee. They generate a nominative signature σ on the program source code of the smart contract, and store σ in a smart contract. Due to the invisibility, information of the business operator and the investor is not leaked even σ is stored on the public blockchain. Note that the verifier needs to choose a random number in the interactive verification protocols of the underlying nominative signature scheme. Thus, the smart contact is not allowed to run these protocols as the verifier because the random number is disclosed when the smart contract runs these protocols. Thus, we employ these interactive protocols

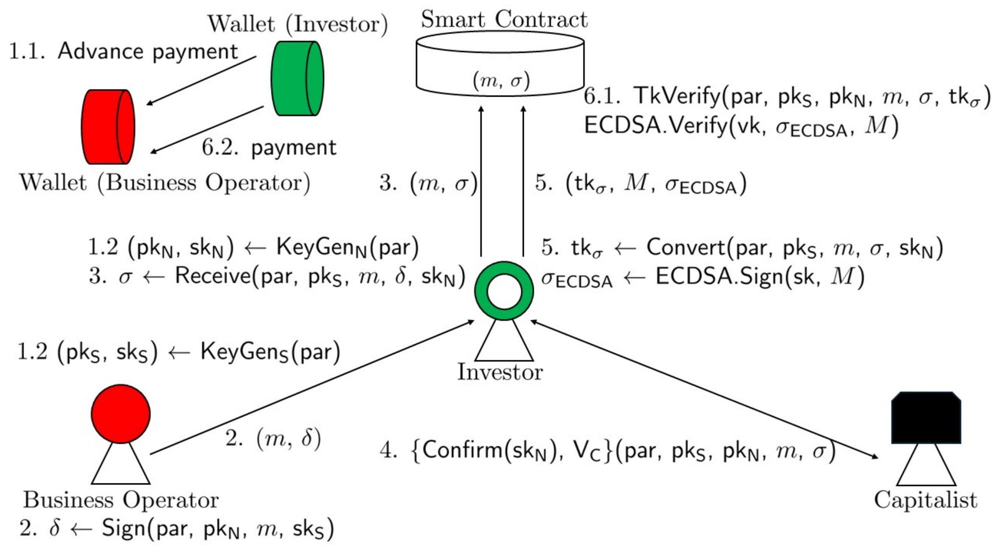

Figure 3: Proposed System

when the investor proves the validity of σ to the capitalist off-chain. Tu run the smart contract, the investor converts σ to tk_σ via the Convert algorithm and sends tk_σ to the smart contract that can be regarded as the trigger of the contract because tk_σ is a publicly verifiable. Note that a ECDSA signature is required for executing a transaction. Thus, the investor sends a ECDSA signature σ_{ECDSA} together with tk_σ.

Before giving the proposed system, we consider the cases that the investor does not follow the procedure. First of all, we need to consider the case that the investor does not send $(\mathsf{tk}_\sigma, \sigma_{\mathsf{ECDSA}})$ to the smart contract (or the case that $(\mathsf{tk}_\sigma, \sigma_{\mathsf{ECDSA}})$ sent to the smart contract is invalid). Then, the smart contract does not transfer the investment to the business operator. To capture the case, a portion of the investment amount is paid in advance in the proposed system. The investor makes an advance payment to the wallet of the business operator, which is confirmed by the business operator, who then generates the signature generation message δ and sends δ to the inverter. Here, we do not consider the case that tk_σ is invalid but σ_{ECDSA} is valid (then, the investment is transferred without employing the smart contact by anyone who obtains σ_{ECDSA}) because there is no merit of the investor.

4.1 System Description

We give the proposed non-Transferable signature verification system using smart contracts and nominative signatures as follows (See Fig. 3). We assume that the business operator and the investor manage own wallet. Moreover, we assume that they have agreed with the description of the program source code m of the smart contact and the smart contract has already been deployed by the business operator.

1. Setup.

1.1. Advance Payment. The investor pays a portion of the investment amount in advance to the business operator's wallet. The business operator will not process any further transactions if the advance payment is not transferred or is insufficient amount.

1.2. Key Generation. . Let $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$. The business operator runs $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{par})$, and the investor runs $(\mathsf{pk_N}, \mathsf{sk_N}) \leftarrow \mathsf{KeyGen_N}(\mathsf{par})$.

2. Signing by the business operator. The business operator runs $\delta \leftarrow \mathsf{Sign}(\mathsf{par}, \mathsf{pk_N}, m, \mathsf{sk_S})$, and then sends the signature generation message δ to the investor.

3. Signing by the investor. The investor runs $\sigma \leftarrow \mathsf{Receive}(\mathsf{par}, \mathsf{pk_S}, \mathsf{m}, \delta, \mathsf{sk_N})$, and then sends σ to the smart contract.

4. Advertisement. The business operator requests the investor to advertise the investment. The investor and the capitalist run $\{\mathsf{Comfirm}(\mathsf{sk_N}), \mathsf{V_C}\}(\mathsf{par}, \mathsf{pk_S}, \mathsf{pk_N}, m, \sigma)$ where the investor acts as the nominee and the capitalist acts as the verifier.

5. Trigger Generation. Let M be the transaction. The investor runs $\mathsf{tk}_\sigma \leftarrow \mathsf{Convert}(\mathsf{par}, \mathsf{pk_S}, m, \sigma, \mathsf{sk_N})$ and $\sigma_{\mathsf{ECDSA}} \leftarrow \mathsf{ECDSA.Sign}(\mathsf{sk}, M)$, and sends tk_σ and $(M, \sigma_{\mathsf{ECDSA}})$ to the management smart contract.

6. Investment

6.1. Trigger Verification $\mathsf{TkVerify}(\mathsf{par}, \mathsf{pk_S}, \mathsf{pk_N}, m, \sigma, \mathsf{tk}_\sigma)$ and $\mathsf{ECDSA.Verify}(\mathsf{vk}, \sigma_{\mathsf{ECDSA}}, M)$ are run in the management smart contract.

6.2. Execution If $(\mathsf{tk}_\sigma, \sigma_{\mathsf{ECDSA}})$ is valid, the funds are transferred from the investor's wallet to the business operator's wallet according to the program m and transaction M If either tk_σ or σ_{ECDSA} is not valid, the smart contract does not transfer funds.

4.2 Security of Proposed System

Due to the unforgeability of the underlying nominative signature scheme, the proposed system guarantees that investors cannot generate legitimate nominative signatures without communicating with the business operator. Moreover, due to the security against malicious signers, the business operator cannot generate legitimate nominative signatures without communicating with the investor. Therefore, it is guaranteed that it is impossible to forge a legitimate nominative signature without the agreement of both parties. Due to the invisibility, the information of the business operator and the investor cannot be leaked from the nominative signature σ stored on the public blockchain. Moreover, due to the non-transferability, the capitalist who has known the verification result cannot prove the result to a third party.

5 Performance Evaluation

Gas Cost. In this section, we estimate the gas cost of the proposed system when the $\mathsf{TkVerify}$ algorithm is run by the smart contract. First, we need to select the underlying nominative signature scheme. Due to the progress of quantum computers, one may think that we should employ a lattice-based nominative signature scheme, e.g., [12]. However, to the best of our knowledge, no pre-compiled contract providing a lattice-based cryptographic scheme has been published so far. Moreover, currently, ECDSA is necessary to issue a transaction which is secure under the discrete-logarithm problem over elliptic curves [5,7] and is not a post-quantum cryptography (PQC). Thus, replacing the underlying nominative signature scheme to be PQC does not affect the post-quantum security of the proposed system. Thus, we employ the pairing-based Hanaoka-Schuldt nominative signature scheme [8] in our evaluation.

We employed a pre-compiled contract provided by smlXL.Inc [19] that allows us to run the smart contract efficiently. It provides elliptic curve operations and pairing computation on Barreto-Naehrig (BN) curves [1] (bn128). Note that the Hanaoka-Schuldt nominative signature scheme is constructed on symmetric pairings,[2] while the pre-compiled contract provides asymmetric pairings. This is expected to cause a difference between the performance evaluation in this paper and the actual performance. We also note that asymmetric pairings provide more efficient implementation compared to that of symmetric pairings [6].

First, we introduce a benchmark of pairing computations and additions over elliptic curves in Table 1.

Table 1: Process and gas costs

Process	Gas cost (Unit)
Pairing	$45,000 + 34,000 * n$
Addition over elliptic curves	150

Here, n is the number of pairing computations. Precisely, n increases for every 192 bytes of input size. In the Hanaoka-Schuldt nominative signature scheme, eight pairing computations are required in the TkVerify algorithm. By considering the gas cost in table 1, the gas cost for running the TkVerify algorithm is estimated to be 317,000 + 150 * 256 = 355,400 Units (0.00629058 ETH (11.2 U.S. dollars) by the rate on March 14, 2025)). The gas cost for verifying a ECDSA signature on the pre-compiled contract is just 3,000 Units (when the algorithm that recovers the verification key from the ECDSA signature and the message, called ecRecover, is executed). It should be noted that running the ecRecover algorithm is an essential procedure in Ethereum, and it can be assumed that there is a consensus that this level of gas cost is acceptable to run smart contracts.

Our estimation indicates that the execution of the TkVerify algorithm requires about 120 times higher gas cost than that of the key recovery process of ECDSA, and we cannot say that the proposed system is efficient in practice. Nevertheless, we claim that our proposal is meaningful to demonstrate the feasibility of a smart contract-based system when enhanced cryptographic primitives such as nominative signatures are employed.

6 Conclusion

In this paper, we proposed a non-transferable signature verification system. We employed both the smart contract and a nominative signature scheme, and estimate the gas cost for running the system. Currently, the proposed system is not sufficiently practical due to the number of pairing computations of the underlying nominative signature scheme. Proposing an efficient nominative signature scheme is also an important future work since eight pairing computations are dominant of the gas cost. In addition, it is also a future work to enable lattice-based nominative signatures to be handled on smart contracts that may reduce the gas cost.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number JP25H01106.

[2]In the full version of this paper [17], we converted the Hanaoka-Schuldt nominative signature scheme constructed on symmetric pairings to a scheme constructed on asymmetric pairings.

References

[1] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[2] Joan Boyar, David Chaum, Ivan Damgård, and Torben P. Pedersen. Convertible undeniable signatures. In *CRYPTO*, pages 189–205, 1990.

[3] David Chaum. Designated confirmer signatures. In *EUROCRYPT*, pages 86–91, 1994.

[4] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, pages 212–216, 1989.

[5] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In *ACM CCS*, 2016.

[6] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008.

[7] Jens Groth and Victor Shoup. On the security of ECDSA with additive key derivation and presignatures. In *EUROCRYPT*, pages 365–396, 2022.

[8] Goichiro Hanaoka and Jacob C. N. Schuldt. Convertible nominative signatures from standard assumptions without random oracles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 99-A(6):1107–1121, 2016.

[9] Qiong Huang, Dennis Y. W. Liu, and Duncan S. Wong. An efficient one-move nominative signature scheme. *International Journal of Applied Cryptography*, 1(2):133–143, 2008.

[10] Zhenjie Huang and Yumin Wang. Convertible nominative signatures. In *ACISP*, pages 348–357, 2004.

[11] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.

[12] Meenakshi Kansal, Ratna Dutta, and Sourav Mukhopadhyay. Lattice-based nominative signature using pseudorandom function. *IET Information Security*, 15(4):317–332, 2021.

[13] Seungjoo Kim, Sungjun Park, and Dongho Won. Zero-knowledge nominative signatures. In *PragoCrypt*, pages 380–392, 1996.

[14] Moses Liskov and Silvio Micali. Online-untransferable signatures. In *Public Key Cryptography*, pages 248–267, 2008.

[15] Sanami Nakagawa, Keita Emura, Goichiro Hanaoka, Akihisa Kodate, Takashi Nishide, Eiji Okamoto, and Yusuke Sakai. A privacy-enhanced access log management mechanism in SSO systems from nominative signatures. In *IEEE TrustCom*, pages 565–574, 2014.

[16] Sanami Nakagawa, Takashi Nishide, Eiji Okamoto, Keita Emura, Goichiro Hanaoka, Yusuke Sakai, and Akihisa Kodate. A privacy-enhanced access log management mechanism in SSO systems from nominative signatures. *International Journal of Applied Cryptography*, 3(4):394–406, 2017.

[17] Hinata Nishino, Kazumasa Omote, and Keita Emura. A smart contract-based non-transferable signature verification system using nominative signatures. arXiv (https://arxiv.org/abs/2506.17504)), 2025.

[18] Jacob C. N. Schuldt and Goichiro Hanaoka. Non-transferable user certification secure against authority information leaks and impersonation attacks. In *ACNS*, pages 413–430, 2011.

[19] smlXL.Inc. Evm codes - precompiled contracts. https://www.evm.codes/precompiled?fork=shanghai.

[20] Gavin Wood. Ethereum yellow paper (Shanghai version 9fde3f4-2024-09-02), 2024. https://ethereum.github.io/yellowpaper/paper.pdf.

[21] Wei Zhao and Dingfeng Ye. Pairing-based nominative signatures with selective and universal convertibility. In *Inscrypt*, pages 60–74, 2009.