The ISC Int'l Journal of Information Security
ISeCure

July 2013, Volume 5, Number 2 (pp. 189–208)
http://www.isecure-journal.org

Convertible Limited (Multi-) Verifier Signature: New Constructions and Applications[☆]

Sepideh Avizheh[1,*], Maryam Rajabzadeh Assar[1], and Mahmoud Salmasizadeh[2]

[1] *Sharif University of Technology, Department of Electrical Engineering, Tehran, Iran*
[2] *Sharif University of Technology, Electronics Research Institute, Tehran, Iran*

A R T I C L E   I N F O.

Article history:
Received: 9 February 2013
Revised: 2 September 2013
Accepted: 14 September 2013
Published Online: 20 March 2014

Keywords:
Convertible Limited (Multi-)Verifier Signature, Universal Designated (Multi-)Verifier Signature, Deniable Authentication, Bilinear Pairing, Electronic Voting.

A B S T R A C T

A convertible limited (multi-) verifier signature (CL(M)VS) provides controlled verifiability and preserves the privacy of the signer. Furthermore, limited verifier(s) can designate the signature to a third party or convert it into a publicly verifiable signature upon necessity. In this proposal, we first present a generic construction of convertible limited verifier signature (CLVS) into which the existing secure CLVS schemes fit. Afterwards, we extend this generic construction to address the unsolved question of designing an efficient construction with more than two limited verifiers. To this effect, two generic CLMVS constructions are presented, which are proven to be efficient in that they generate a unique signature for more than two limited verifiers. Given the first generic construction, each limited verifier checks the validity of the signature solely, while in the second, cooperation of all limited verifiers is imperative. Thereupon, on the ground of our second generic construction, we present the first pairing-based CLMVS scheme secure in the standard model, which is of a strong confirmation property as well. Finally, we employ the proposed CLMVS scheme for one limited verifier (CLVS) so as to design a new electronic voting protocol.

1 Introduction

Digital signatures were created to satisfy the integrity and non-repudiation of messages. However, in these schemes, privacy of the signer is not preserved since everyone can verify the signature. Necessity to preserve the privacy of the signer during the verification process termed as "controlled verifiability", has inspired researchers to introduce different types of signatures. Firstly, the concept of undeniable signatures was introduced by Chaum and Antwerpen [1]. Although the verification process is controlled in their scheme, the verifier has to interact with the signer to verify the signature. To avoid the interactive nature of undeniable signatures, the researchers then attempted to design designated verifier signatures (DVS) [2], whereby the designated verifier can check the validity of the signature using a non-interactive approach. Nevertheless, the designated verifier cannot transfer the signature to a third party to convince them that the signer has indeed signed the message. Therefore,

☆ Part of this paper has been presented at 9th Int. ISC Conf. on Information Security and Cryptology, Tabriz, Iran, Sep. 2012, entitled "Two efficient generic patterns for convertible limited multi-verifier signature".
* Corresponding author.

Email addresses: avizheh@alum.sharif.edu (S. Avizheh), asaar@ee.sharif.edu (M. Rajabzadeh Assar), salmasi@sharif.edu (M. Salmasizadeh).

ISSN: 2008-2045 © 2013 ISC. All rights reserved.

a DVS scheme was found not to be appropriate for scenarios where dispute possibly occurs and a trusted third party such as a judge is required to interfere.

In 1999, Araki *et al.* [3] introduced the concept of limited verifier signatures (LVS) to rectify the aforementioned shortcomings. In a LVS scheme, only a limited verifier, who is considered to be trustee to preserve the privacy of the signer, can check the validity of the signature, and he or she can in turn designate the signature to a third party upon necessity. Araki *et al.* also presented the concept of convertible limited verifier signatures (CLVS) in the same study [3]. A CLVS scheme is identical to the LVS scheme in which the limited verifier and the signer can convert the signature to a publicly verifiable signature. A CLVS scheme is applicable to official and patent documents, each of which needs a LVS signature that is highly likely to become public after a period of time.

Araki *et al.* commenced the development of CL(M)VS schemes by proposing the first CLVS scheme [3], which requires the cooperation of the signer in converting the signature to a publicly verifiable one, although Zhang and Kim [4] proposed a universal forgery attack on it later in 2003. In 2004, Chen *et al.* [5] introduced two CLVS schemes acting non-interactively in the conversion process. Then, they generalized their second scheme using the universal designated verifier signatures (UDVS). In 2005, Zhang *et al.* [6] constructed a provable secure CLVS scheme without using the random oracle model. The security of their scheme was based on the CDH assumption, even though in 2006, Shao [7] showed that their proofs have been established on the basis of several logic errors. In 2006, certificateless CLVS scheme was put forth by Wu *et al.* [8] based on certificateless cryptography. They extended their scheme for two limited verifiers and declared that constructing a convertible limited multi-verifier signature (CLMVS) with more than two limited verifiers with a unique signature would be an open problem. In 2008, Wang *et al.* [9] proposed an identity-based CLVS scheme which was then extended to a CLMVS scheme. This scheme, however, appeared to be inefficient due to the fact that for each limited verifier, a particular signature needed to be generated. In 2008, Tso *et al.* [10] presented a CLVS scheme which was efficient in terms of computation and communication cost. And finally, it was in 2011 that the first provably secure identity-based CLVS scheme was introduced by Shen and Ming [11] without using random oracle model.

As mentioned earlier, the CLMVS scheme proposed by Wang *et al.* [9] is inefficient since the signer has to generate a particular signature for each limited verifier, while in an efficient CLMVS scheme, a unique signature should be generated for all limited verifiers. In this regard, in the present paper, we first propose a "generic construction" into which all the previous secure CLVS schemes fit. This generic construction uses a UDVS scheme as its building block. Additionally, it will be shown that the generalized scheme proposed by Chen *et al.* [5] is also a specific form of our generic construction. Then, we extend the generic construction to two efficient generic CLMVS constructions, whose efficiency is guaranteed by virtue of generating a unique signature for more than two limited verifiers. In the first scheme, each limited verifier can verify the validity of the signature solely, and in the second scheme, limited verifiers should gather and aggregate their private keys to verify the signature. Afterwards, we present the first pairing-based CLMVS scheme secure in the standard model on the ground of our second generic construction. Being used for one limited verifier, i.e. a CLVS, this scheme will be then comparable with the one porposed by Shen and Ming [11]. It has to be noted though that our scheme does not encounter the key escrow problem and it needs the private key of the third party to verify the confirmation proof (strong confirmation property).

In addition, the CLVS scheme proposed in the present paper is of valuable properties such as provable security in the standard model and non-transferability. These properties make this concrete scheme suitable to be employed as a building block of an electronic voting protocol. To the best of our knowledge, CLVS schemes have not yet been used in voting protocols so far. Therefore, we have also designed a new electronic voting protocol based on the proposed concrete scheme. The presented voting protocol is considered to fall into the category of semi-hidden voter with semi-hidden vote approach which also resists against coercion of the tally authority.

The remainder of this paper is thus organized as follows. In Section 2 the preliminaries required for constructing the schemes will be described followed by a generic construction of CLVS scheme presented in Section 3; in Section 4, the existing schemes are classified based on our generic construction. Two generic constructions of CLMVS scheme are then being proposed in Section 5; Section 6 introduces the first pairing-based CLMVS scheme secure in the standard model. Comparisons are then made between the proposed scheme and the one generated by Wang *et al.* [9] in terms of efficiency in Section 7. In Section 8, a new voting protocol is presented and finally the paper will be concluded in Section 9.

2 Preliminaries

In this section, detailed descriptions and definitions are being provided for a number of basic concepts used in the remaining of this paper.

2.1 Framework of UD(M)VS

A universal designated (multi-) verifier signature scheme (UD(M)VS) involves three parties: a signer, a signature holder and a designated verifier.

Definition 1. A universal designated (multi-) verifier signature (UD(M)VS) scheme is a collection of six algorithms, as described below:

1) System Parameters Generation (SPG):This probabilistic polynomial-time algorithm takes security parameter k as its input, and outputs the system parameters SP.
2) Key Generation (KG): For each entity u , this probabilistic polynomial-time algorithm takes SP as input, and outputs a private/public key pair (Sk_u, Pk_u).
3) Sign (SIG): This probabilistic polynomial-time algorithm takes as input the private key of the signer and a message m, and outputs the signature S on the message m.
4) Public Verification (PV): This deterministic polynomial algorithm takes the message/signature pair as input, and outputs a decision bit $b \in \{0, 1\}$. The signature is accepted if $b = 1$ and rejected otherwise.
5) Designation (DS): This probabilistic polynomial-time algorithm takes the message/signature pair, the public key of the designated verifier(s) and the public key of the signer as input, and outputs the designated signature for message m.
6) Designated Verification (DV): This deterministic polynomial-time algorithm takes the message/designated verifier signature pair, the private key(s) of the designated verifier(s) and the public key of the signer as inputs, and outputs a decision bit $b \in \{0, 1\}$. The signature is accepted if $b = 1$ and rejected otherwise.

2.2 Framework of CLVS

A CLVS scheme involves three parties: namely a signer, a limited verifier and a certain trusted third party such as a Judge.

Definition 2. A convertible limited verifier signature scheme can be described as a collection of the following seven algorithms:

1) System Parameters Generation (SPG): This probabilistic polynomial-time algorithm takes security parameter k as input, and outputs the system parameters SP.
2) Key Generation (KG): For each entity u, this probabilistic polynomial-time algorithm takes SP as input, and outputs a private/public key pair (Sk_u, Pk_u).
3) Limited Verifier Signing (LVS): This probabilistic polynomial-time algorithm takes the private key of the signer, the public key of the limited verifier and a message m as inputs, and outputs a message/signature pair (m, σ) which can be verified exclusively by the limited verifier.
4) Limited Verifier Verification (LVV): This deterministic polynomial-time algorithm takes the public key of the signer, the private key of the limited verifier and the message/limited verifier signature pair as inputs, and outputs a decision bit $b \in \{0, 1\}$. The signature is accepted if $b = 1$ and rejected otherwise.
5) Confirmation Proof (CP): This probabilistic polynomial-time algorithm takes the private/public key pair of the limited verifier, the public key of the third party and the message/limited verifier signature pair as inputs, and outputs a signature δ. The third party can verify δ, yet he or she cannot transfer the signature to convince another party.
6) Conversion Algorithm (CA): This deterministic polynomial-time algorithm takes the public key of the signer, the private/public key pair (Sk_{lv}, Pk_{lv}) of the limited verifier and the message/limited verifier signature pair as inputs, and outputs a converted signature σ_{pub}, which is publicly verifiable.
7) Public Verification (PV): This deterministic polynomial-time algorithm takes the public key of the signer and the message/converted signature pair as inputs, and outputs a decision bit $b \in \{0, 1\}$. The signature is accepted if $b = 1$ and rejected otherwise.

2.3 Security Requirements of CLVS Schemes

Completeness: A properly formed CLVS scheme must pass the verification algorithm. Formally, a CLVS scheme satisfies completeness if for any message m, any signer with the key pair (Sk_s, Pk_s) and any verifier with the key pair (Sk_v, Pk_v) (1) holds true.

$$Pr[ver(m, Sk_v, Pk_v, Pk_s, \sigma = Sign(m, Sk_s, Pk_s, Pk_v)) = 1] \quad (1)$$

Unforgeability: In a CLVS scheme, we assume that the limited verifier is trusted to preserve the signer's privacy, yet the limited verifier may forge a signature. Thus, unforgeability is regarded as a necessary property in CLVS schemes. According to Chen *et al.*'s [5] formal definition of unforgeability, there are three kinds of forgers: namely an insider adversary, an outsider adversary and the limited verifier. Chen *et al.* [5] consider the strongest adversarial model in which the limited verifier is the forger. The limited verifier may forge the LVS signature or the confirmation proof. From among forging the confirmation proof is of no use for the limited verifier. As a result, unforgeability in CLVS schemes refers merely to existential unforgeability of the LVS signature against adaptive chosen message attack.

In order to shed more light on this concept, let's assume the following game between the challenger and the adversary:

Stage 1. Challenger C runs the SPG algorithm on a security parameter k and sends SP to the adversary F.

Stage 2. C runs the KG algorithm to obtain the signer's and the limited verifier's keys. Then, C gives F the public key Pk_s of the signer and the private/public key pair (Sk_{lv}, Pk_{lv}) of the limited verifier.

Stage 3. F is allowed to make queries into signing oracle, O_{lvs}, to request a LVS signature σ_i on message m_i. O_{lvs} responds with the corresponding signature through running the LVS algorithm.

Stage 4. After enough queries, F outputs a forgery (m^*, σ^*) and wins the game with advantage A_F if $LVV(m^*, Sk_{lv}, Pk_{lv}, Pk_s, \sigma^*) = 1$.

Definition 3. A CLVS scheme is (t, q_{lvs}, ϵ)–existentially unforgeable under adaptive chosen message attack if no (t, q_{lvs}, ϵ)-forger exists; an adversary F is said to be (t, q_{lvs}, ϵ)-forger if F can win the above game with a probability of at least $\epsilon = A_F$, running in time at most t and issuing at most q_{lvs} queries.

Non-transferability: In CLVS schemes, the limited verifier can generate a proof for a third party J and convince him or her that the signer has indeed generated the signature. Non-transferability ensures that the judge is able to generate an indistinguishable proof p'. Hence, no one can be persuaded that the signature is originated from the signer, even if J reveals his or her private key.

Definition 4. A CLVS scheme is non-transferable if the output of a probabilistic polynomial-time simulation algorithm on Sk_J, Pk_J, and Pk_{lv} and a message m is indistinguishable from a proof having been generated by the CP algorithm. Formally, for a CLVS scheme to be non-transferable, (2) must hold true.

Limited Verifiability: According to [8], A CLVS scheme is said to be of limited verifiability as no one but the signer and the limited verifier can verify the signature or generate a confirmation proof or a converted signature to convince a third party.

$$\left| Pr\begin{bmatrix} \delta_0 \leftarrow CP((m,\delta), Sk_{lv}, Pk_{lv}, Pk_J), \\ \delta_1 \leftarrow Sim(m, Pk_{lv}, Sk_J, Pk_J), \\ b \leftarrow \{0,1\}, \\ b' \leftarrow F(m, Sk_{lv}, Pk_{lv}, Sk_J, Pk_J, \delta_b), \\ : b' = b \end{bmatrix} - \frac{1}{2} \right| < \epsilon(k) \tag{2}$$

3 A Generic Construction of CLVS

Taking into account all the aforementioned information, we now propose a generic CLVS construction into which all the previous CLVS schemes fit. A UDVS scheme is regarded as the building block of this construction, in which the message and the signature space are identical to those of the UDVS scheme. The proposed generic construction is described below:

The generic construction of CLVS scheme is a collection of seven algorithms SPG, KG, LVS, LVV, CP, CA, and PV.

- SPG and KG algorithms are the same as SPG and KG algorithms from the UDVS scheme.
- Limited Verifier Signing (LVS): this algorithm has three stages for generating LVS signature, as follows:
  1) On inputs (m, Sk_s) the LVS algorithm runs the SIG algorithm from the UDVS scheme, and outputs a publicly verifiable signature S on the message m.
  2) On inputs Sk_s and Pk_{lv} the LVS algorithm generates a "simulatable transcript", $SimTr$, which is a common secret transcript between the signer and the limited verifier.
  3) The LVS algorithm applies a function, $Func$, on inputs S and $SimTr$, and outputs a limited verifier signature σ on the message m. $Func$ is an invertible function and can be a simple xor or a simple multiply operation or a more complicated function. $InvFunc$ denotes inverted function of $Func$.

$$\sigma = Func(S, SimTr) \tag{3}$$

- Limited Verifier Verification (LVV): This algorithm takes as inputs the public key Pk_s of the signer and the private key Sk_{lv} of the limited verifier to generate $SimTr$. Next, it obtains signature S using $InvFunc$. Then, it runs the PV algorithm from the UDVS scheme on the message/publicly verifier signature (m, S). Finally, the PV algorithm outputs a decision bit $b \in \{0,1\}$. The signature is accepted if $b = 1$ and rejected otherwise.
- Confirmation Proof (CP): This algorithm runs the DS algorithm from the UDVS scheme on inputs the signature S, the public key Pk_s of the

signer and the public key Pk_J of the third party to generate a signature δ. δ is the confirmation proof for the signature S. To verify the proof, the CP algorithm runs the DV algorithm from the UDVS scheme.
- Conversion Algorithm (CA): Both the limited verifier and the signer can publish S , where S is only known by the limited verifier and the signer.
- Public Verification (PV): This algorithm is the same as the PV algorithm of the UDVS scheme.

Note: One thing may come to the mind is that the limited verifier can reveal $SimTr$ to a malicious party, yet we emphasize that in this kind of signature, the limited verifier is considered as being reliable for preserving the privacy of the signer on account of his or her benefit, and therefore, he or she will not reveal $SimTr$.

3.1 Security Analysis

In this section, we present evidences for the security of our generic construction of CLVS.

3.1.1 Unforgeability

In the following, existential unforgeability of the proposed generic construction of CLVS is studied.

Lemma 1. *The presented CLVS scheme is (t, q_{lvs}, ϵ)–existentially unforgeable under an adaptive chosen message attack if the publicly verifiable signature S is $(t' = t + t_{SimTr} + t_{InvFunc}, q_{lvs}, \epsilon)$–existentially unforgeable under an adaptive chosen message attack.*

Proof. We prove the lemma using contradiction. Suppose that the publicly verifiable signature S is (t', q_{lvs}, ϵ)–existentially unforgeable under an adaptive chosen message attack. Below, we will show that if there exists a (t, q_{lvs}, ϵ) -forger F which can forge the LVS signature σ, then there exists another algorithm which can forge S in time t' with q_{lvs} queries and with advantage ϵ (a contradiction).

We assume that there exists a probabilistic polynomial-time algorithm F which can forge the LVS signature σ with advantage ϵ in time t after q_{lvs} queries to the LVS oracle O_{lvs}. Then, we construct another forging algorithm F' which runs the following game with the challenger C. F' tries to forge S.

Stage 1. C runs the SPG algorithm on the security parameter k and sends SP to F'.

Stage 2. C runs the KG algorithm to obtain the key pairs of the signer and the limited verifier and sends the public key Pk_s of the signer and the private/public key pair (Sk_{lv}, Pk_{lv}) of the limited verifier to F'. Subsequently, F' computes $SimTr$ using Sk_{lv} and Pk_s.

Stage 3. F' runs F; F issues queries, adaptively, to LVS oracle, O_{lvs}, for an LVS signature on a message m_i, where $1 \leq i \leq N$. O_{lvs} runs LVS algorithm and returns σ_i to F. Finally, F outputs a forgery σ^* on message m^*.

Stage 4. F' computes $S^* = Func^{-1}(\sigma^*, SimTr)$ and outputs (m^*, S^*) as its forgery.

F' wins the game and forges the signature S with non-negligible advantage $A^S_{F'} = Pr[Ver(m^*, S^*) = 1] = \epsilon$ in time $t' = t + t_{SimTr} + t_{InvFunc}$ with at most q_{lvs} queries; This deduction contradicts the first assumption that S is (t', q_{lvs}, ϵ)–existentially unforgeable under an adaptive chosen message attack. Therefore, the presented CLVS scheme is (t, q_{lvs}, ϵ)–existentially unforgeable under an adaptive chosen message attack. □

3.1.2 Non-transferability

In the following, non-transferability of the proposed generic construction of CLVS is studied.

Lemma 2. *The CLVS scheme is $(\hat{t}, q_t, \hat{\epsilon})$–non-transferable under an adaptive chosen message attack if the UDVS scheme is $(t'' = \hat{t}, q_t, \hat{\epsilon})$–non-transferable under an adaptive chosen message attack.*

Proof. Note that the confirmation proof in our generic pattern is identical to the designated signature in the UDVS scheme. Thus, the UDVS scheme should be non-transferable to satisfy non-transferability of our scheme. In other words, the CLVS scheme is non-transferable if the UDVS scheme is non-transferable.

We prove Lemma 2 by contradiction. Suppose that the UDVS scheme is $(t'', q_t, \hat{\epsilon})$–non-transferable under an adaptive chosen message attack:

$$\left| Pr\begin{bmatrix} \delta_0 \leftarrow DS((m, Sk_s, Pk_s, Pk_{dv}), \\ \delta_1 \leftarrow Sim(m, Pk_s, Sk_{dv}, Pk_{dv}), \\ b \leftarrow \{0,1\}, \\ b' \leftarrow F(m, Sk_s, Pk_s, Sk_{dv}, Pk_{dv}, \delta_b), \\ : b' = b \end{bmatrix} - \frac{1}{2} \right| < \hat{\epsilon} \quad (4)$$

We assume that there exists a probabilistic polynomial-time distinguisher D which can distinguish the confirmation proof from its simulation in the CLVS scheme with non-negligible advantage A^{CLVS}_D in time $\hat{t}$ after q_t queries to the CLVS oracle O_{CLVS}. Then, we construct another probabilistic polynomial-time distinguisher D' which runs the following game with the challenger C. D' tries to distinguish the designated verifier signature DS from its simulation Sim.

Stage 1. C runs the SPG algorithm on the security parameter k and sends SP to D'.

Stage 2. C runs the KG algorithm to obtain the key pair of the signer and the designated verifier and sends the public key pair Pk_s of the signer and the public key Pk_{dv} of the designated verifier to D'.

Stage 3. C chooses an arbitrary message m and picks a random bit $b \in \{0,1\}$; if $b = 0$ then C runs the DS algorithm in the UDVS scheme and returns δ_0 to D'. Otherwise, if $b = 1$, then C runs the Sim algorithm and returns δ_1 to D'.

Stage 4. D' runs D with the challenge (m, δ_0, δ_1); D issues N queries, adaptively, to CLVS oracle on messages $m_i, 1 \leq i \leq N$, $Pk_{lv} = Pk_s$ and $Pk_J = Pk_{dv}$. O_{CLVS} runs LVS, CA and CP algorithms and returns the confirmation proof δ_i^* to D. Finally, D outputs a guess $b' \in \{0,1\}$.

Stage 5. D' outputs $b' \in \{0,1\}$ as his guess.

D' wins the game with non-negligible advantage $A_{D'}^{UDVS} = |Pr[b' = b] - 1/2| = A_D^{CLVS} \gg \hat{\epsilon}$ in time $t'' = \hat{t}$, with at most q_t queries. Thus,

$$\left| Pr \begin{bmatrix} \delta_0 \leftarrow DS((m, Sk_s, Pk_s, Pk_{dv}), \\ \delta_1 \leftarrow Sim(m, Pk_s, Sk_{dv}, Pk_{dv}), \\ b \leftarrow \{0,1\}, \\ b' \leftarrow F(m, Sk_s, Pk_s, Sk_{dv}, Pk_{dv}, \delta_b), \\ : b' = b \end{bmatrix} - \frac{1}{2} \right| \gg \hat{\epsilon} \quad (5)$$

This deduction contradicts the assumption that the UDVS scheme is $(t'', q_t, \hat{\epsilon})$–non-transferable; Thus, the CLVS scheme is $(\hat{t}, q_t, \hat{\epsilon})$–non-transferable under an adaptive chosen message attack. □

3.1.3 Discussion on Limited Verifiability

Limited verifiability is intrinsically a composition of three separated properties, and yet a formal definition of the term is sadly lacking due to its complexity. Thus, presenting a formal proof for the lemmas made on the ground of limited verifiability is impossible until the presentation of a formal definition. In this section, the general property of limited verifiability is broken into its composing features and through using a formal framework, it will also be shown that how the presented CLVS scheme relates to these parcels conversely. Having this in mind, we present three conditions interwoven with the concept of limited verifiability, each of which is essential since if one fails, the security of our scheme will remove. However, it should be highlighted that the security of our scheme will not be assuredly guaranteed even if all of these conditions are satisfied. These conditions are elaborated upon, and the impacts of each one on the violation of the limited verifiability will also be studied afterwards.

By definition, a scheme is said to be of limited verifiability when no one, except the signer and the limited verifier, can verify the signature or generate a confirmation proof or a converted signature to convince a third party. First, if $SimTr$ is secure, then no one, except the signer and the limited verifiers, can obtain S and verify the limited verifier signature or run the confirmation proof or the conversion algorithm. Second, if S is unforgeable then no one, except the signer and the limited verifiers, can generate a publicly verifiable signature or verify the limited verifier signature. Finally, since the confirmation proof is the same as the DS algorithm from the UDVS scheme, if the UDVS scheme is unforgeable then no one, except the signer and the limited verifiers, can generate a valid confirmation proof. Correspondingly, the following conditions should be observed:

1) $SimTr$ should be kept secret to anyone except the signer and the limited verifier. Since in CLVS schemes it is assumed that the limited verifier is trusted for preserving $SimTr$, this condition reduces to a mere evaluation of $SimTr$'s unforgeability.
2) The publicly verifiable signature S should be an unforgeable signature.
3) The UDVS scheme should be an unforgeable scheme.

In the following, we study the violating effect of the aforementioned conditions on the properties composing limited verifiability.

Lemma 3. *The presented generic construction of the CLVS is $(t' = t + t_{InvFunc}, q_{SimTr}, \tilde{\epsilon_1} = |\epsilon_1 - 1/2|)$–unsecure (i.e. it does not provide limited verifiability) if $SimTr$ is $(t, q_{SimTr}, \epsilon_1)$–forger and the publicly verifiable signature S is $(t', q_{SimTr}, \epsilon_2)$–existentially unforgeable under adaptive chosen message attack.*

Proof. Now, we show directly that if there exists an algorithm which can forge $SimTr$, then there exists another algorithm that can verify the limited verifier signature, and thus break the limited verifiability with advantage $A_{F'}^{verify}$ in time t', using at most q_{SimTr} queries.

We assume that there exists a probabilistic polynomial-time algorithm F which can break $SimTr$ with advantage ϵ_1 in time t, after q_{SimTr} queries to the $SimTr$ oracle O_{SimTr}. Then, we construct another probabilistic polynomial-time algorithm F'

which runs the following game with the challenger C. F' tries to verify the limited verifier signature (LV-signature) σ.

Stage 1. C runs the SPG algorithm on the security parameter k and sends SP to F'.

Stage 2. C runs the KG algorithm to obtain the key pairs of the signer and the limited verifier and sends the public key Pk_s of the signer and the public key Pk_{lv} of the limited verifier to F'.

Stage 3. C chooses an arbitrary message m and picks a random bit b; if $b = 0$ then C chooses a random number $R \in G_\sigma$ and sends (m, R) to F' as a challenge, else if $b = 1$ then C runs the LVS algorithm to obtain a LV-signature $\sigma = Func(S, SimTr)$ on the message m and sends (m, σ) to F' as a challenge.

Stage 4. F' runs F with the challenge $\{Pk_s, Pk_{lv}\}$; F issues N queries, adaptively to $SimTr$ oracle, O_{SimTr}. O_{SimTr} runs $SimTr$ algorithm and returns $SimTr_i, 1 \le i \le N$, to F. Finally, F outputs $SimTr^*$ on $\{Pk_s, Pk_{lv}\}$.

Stage 5. F' computes $S^* = Func^{-1}(\sigma, SimTr^*)$ and verifies (m, S^*). F' outputs a guess on b, b'. If $Ver(m, S^*) = 1$, it outputs $b' = 1$, and $b' = 0$ otherwise.

Having assumed that S is $(t', q_{SimTr}, \epsilon_2)$–unforgeable, F' wins the game if only $SimTr^*$ is a correct forge. In other words, F' verifies the signature σ with advantage $Pr[b' = b] = \epsilon_1(1 - \epsilon_2) \approx \epsilon_1$ and thus, $A_{F'}^{verify} = \tilde{\epsilon_1} = |Pr[b' = b] - 1/2| = |\epsilon_1 - 1/2|$ in time $t' = t + t_{InvFunc}$, with at most q_{SimTr} queries. □

Lemma 4. *The presented generic construction of the CLVS is $(t' = t + t_{InvFunc}, q_S, \tilde{\epsilon_3} = |\epsilon_3 - 1/2|)$–unsecure (i.e. it does not provide limited verifiability) if the publicly verifiable signature S is (t', q_S, ϵ_3)–forger and the $SimTr$ is (t', q_S, ϵ_4)–existentially unforgeable under adaptive chosen message attack.*

Proof. Below, we show directly that if there exists an algorithm which can forge the publicly verifiable signature S, then there exists another algorithm which can verify the limited verifier signature and thus break limited verifiability with advantage $A_{F'}^{verify}$ in time t' and using at most q_S queries.

We assume that there exists a probabilistic polynomial-time algorithm F which can forge S with advantage ϵ_3 in time t, after q_S queries to the SIG oracle O_S. Then, we construct another probabilistic polynomial-time algorithm F' which runs the following game with the challenger C. F' tries to verify the limited verifier signature (LV-signature) σ.

Stage 1. C runs the SPG algorithm on the security parameter k and sends SP to F'.

Stage 2. C runs the KG algorithm to obtain key pairs of the signer and the limited verifier and sends the public key Pk_s of the signer and the public key Pk_{lv} of the limited verifier to F'.

Stage 3. C chooses an arbitrary message m and picks a random bit b; if $b = 0$, then C chooses a random number $R \in G_\sigma$ and sends (m, R) to F' as a challenge, else if $b = 1$, then C runs the LVS algorithm to obtain a LV-signature $\sigma = Func(S, SimTr)$ on the message m and sends (m, σ) to F' as a challenge.

Stage 4. F' runs F; F issues N queries, adaptively to SIG oracle, O_S, for a Publicly verifier signature S_i on the message m_i. O_S runs SIG algorithm and returns S_i, $1 \le i \le N$, to F. Finally, F outputs a forgery (m, S^*).

Stage 5. F' computes $SimTr^* = Func^{-1}(\sigma, S^*)$ and verifies $SimTr^*$. F' outputs a guess on b, b'. If $SimTr^*$ is correct, it outputs $b' = 1$, and $b' = 0$ otherwise.

Given the assumption that $SimTr$ is (t', q_S, ϵ_4)–unforgeable, F' wins the game if only S^* is a correct forge. In other words, F' verifies the signature σ with advantage $Pr[b' = b] = \epsilon_3(1 - \epsilon_4) \approx \epsilon_3$ and thus, $A_{F'}^{verify} = \tilde{\epsilon_3} = |Pr[b' = b] - 1/2| = |\epsilon_3 - 1/2|$ in time $t' = t + t_{InvFunc}$, with at most q_S queries. □

Lemma 5. *The presented generic construction of the CLVS is $(t, q_{UDVS}, \epsilon_5)$–unsecure (i.e. it does not provide limited verifiability) if the UDVS scheme is $(t, q_{UDVS}, \epsilon_5)$–forger under adaptive chosen message attack.*

Proof. It will be explained in the following that if there exists an algorithm which can forge the UDVS scheme, then there exists another algorithm which can forge the confirmation proof in the CLVS scheme and thus break the limited verifiability.

We assume that there exists a probabilistic polynomial-time algorithm F which can forge the UDVS scheme with advantage ϵ_5 in time t, after q_{UDVS} queries to the $UDVS$ oracle O_{UDVS}. Then, we construct another probabilistic polynomial-time algorithm F' which runs the following game with the challenger C. F' tries to forge the confirmation proof (CP) δ.

Stage 1. C runs the SPG algorithm on the security parameter k and sends SP to F'.

Stage 2. C runs the KG algorithm to obtain the key pairs of the signer and the limited verifier and sends the public key Pk_s of the signer, the public key Pk_J

of the Judge and the public key Pk_{lv} of the limited verifier to F'.

Stage 3. C chooses an arbitrary message m and runs the LVS algorithm to obtain a LV-signature $\sigma = Func(S, SimTr)$ on the message m and sends (m, σ) to F'.

Stage 4. F' runs F with the challenge (m, Pk_J); F issues queries, adaptively to $UDVS$ oracle, O_{UDVS}, for a UDVS-signature. Subsequently, O_{UDVS} runs the UDVS algorithm and returns the signature (m_i, δ_i), $1 \leq i \leq N$,to F. Finally, F outputs the signature δ^* on the message m.

Stage 5. F' outputs (m, δ^*) as his forgery. F' wins the game, if δ^* is a correct forge, with advantage $A_{F'}^{CP} = Pr[Ver(m, \delta^*) = 1] = \epsilon_5$ in time t, with at most q_{UDVS} queries. □

4 Classification of the Existing Secure Schemes Based on our CLVS Generic Construction

In Table 1, the existing secure CLVS schemes are classified based on the UDVS scheme, $SimTr$ and the function used in their schemes. In this table, multiply operation is denoted with *times*, xor operation with *oplus* , hash operation with H and pairing operation with e.

It can be viewed from Table 1 that the existing schemes are compatible with our generic construction. In the following, more details are being provided regarding some of the statements of the table.

As can be observed in the table, Chen *et al.* 1's scheme [5] uses a combination of the Hess signature [12] and the designated verifier scheme of Laguillaumie and Vergnaud [13] for the UDVS scheme.

Chen *et al.*'s generalized scheme [5] which is based on UDVS schemes employs a public key encryption algorithm as $Func$ and the public/private key of the limited verifier as $SimTr$.

Wu *et al.*'s scheme [8] uses $SimTr$ as one of the parameters of the UDVS scheme; hence, it is not essential to obtain the public verifiable signature from the limited verifier signature to generate the confirmation proof or the public conversion. The confirmation proof is not presented in the protocol and we suggest using Ming *et al.*'s scheme [15] to complete the protocol. However, in Wu *et al.*'s scheme [8], it is necessary to publish $SimTr$ for public verification since verification is performed on the limited verifier signature.

5 Two Generic Constructions for CLMVS

In this section, using the aforementioned generic CLVS scheme, two efficient CLMVS schemes are being constructed as described below in details.

5.1 The First Generic Construction of CLMVS

In this section, we propose a CLMVS scheme in which each limited verifier can verify, confirm and convert the signature solely.

The first generic construction of CLMVS is a collection of eight algorithms SPG, KG, GKG, LMVS, LMVV, CP, CA, and PV. Among these algorithms, GKG, LMVS and LMVV need to be redefined while the others are identical to the generic construction of the CLVS scheme. A brief description of these algorithms is presented below:

- Group Key Generation (GKG): This secure algorithm takes as inputs the keys of the signer and the keys of n limited verifiers, and outputs a group key which is denoted by $SimTr$. $SimTr$ is only known to the signer and the limited verifiers.
- Limited Multi-Verifier Signing (LMVS): This algorithm is of two stages:
  1) The LMVS algorithm runs the SIG algorithm from the UDVS scheme, on a message m and the private key Sk_s of the signer as inputs, and outputs a signature S.
  2) The LMVS algorithm applies a function, $Func$, similar to the one in the previous section, on S and $SimTr$, and outputs σ.

$$\sigma = Func(S, SimTr) \tag{6}$$

- Limited Multi-verifier Verification (LMVV): Having the message/signature pair (m, σ) as its inputs, each verifier uses $SimTr$, σ and $InvFunc$ to obtain a signature S. Then the verifier runs the PV algorithm from the UDVS scheme, and checks the validity of the signature.

Note that each of the limited verifiers can obtain the signature S; hence, they can run the CP or CV algorithms solely.

5.2 The Second Generic Construction of CLMVS

In this section, a generic CLMVS scheme is being constructed using an efficient UDMVS scheme, which needs the private keys of the recipients for verification. A UDMVS scheme is efficient if its output signature is unique for all verifiers.

Table 1. Classification of the existing secure CLVS schemes

Scheme	Security Model	UDVS Scheme	SimTr	Function
Chen *et al.* 1 [5]	Random oracle	[12],[13]	$H_2(e(r_AQ, r_BP))^{-1}$	$\times$
Chen *et al.* 2 [5]	Random oracle	[14]	$H_3(c)$	$\oplus$
Chen *et al.* generalized [5]	Generic	Generic	Private/public key pair of the limited verifier	Encryption
Wu *et al.* [8]	Random oracle	[15]	$H_2(m, w)$	Nonlinear
Wang *et al.* [9]	Random oracle	[16]	$H_2(e(Q_B, V))$	$\oplus$
Tso *et al.* [10]	Random oracle	[14]	$H_0(bV_a + R, m)$	$\times$
Shen and Ming [11]	Standard	[17]	$e(g_1, g_2)^s$	$\oplus$

Note: In this section, the term "UDMVS scheme" is being used to refer to "an efficient UDMVS with the mentioned property" to avoid illegibility of the text.

The second generic construction of CLMVS is a collection of seven algorithms SPG, KG, LMVS, LMVV, CP, CA, and PV. Among these algorithms, LMVS and LMVV algorithms are being redefined whereas the other ones are the same as the generic CLVS scheme.

- Limited Multi-Verifier Signing (LMVS): this algorithm is of three stages, as discussed below:
  1) The LMVS algorithm runs the SIG algorithm from the UDMVS scheme on a message m and the private key Sk_s of the signer as inputs, and outputs a verifiable signature S on the message m.
  2) The LMVS algorithm runs the DS algorithm of the UDMVS on S and the public keys $\{Pk_{lv1}, Pk_{lv2}, \ldots, Pk_{lvn}\}$ of the n limited verifiers, and outputs a signature V. Then, the LMVS algorithm applies a hash function on V, for mapping to S space, and outputs $SimTr$.
  3) The LMVS algorithm applies a function $Func$ on inputs S and $SimTr$ to compute σ as presented in (7).

$$\sigma = Func(S, SimTr) = Func(S, H(V)) \quad (7)$$

- Limited Multi-Verifier Verification (LMVV): This algorithm runs the simulation algorithm of the UDMVS scheme on the public key Pk_s of the signer, the private keys $\{Sk_{lv1}, Sk_{lv2}, \ldots, Sk_{lvn}\}$ of the limited verifiers as inputs, and outputs $SimTr$. Then, applying $InvFunc$ on $SimTr$ and σ it outputs the signature S. To verify the signature, the LMVV algorithm runs the PV algorithm from the UDMVS scheme on S. The signature is accepted if $b = 1$ and rejected otherwise.

6 A Pairing-Based CLMVS Scheme Secure in the Standard Model

In this section, we propose the first pairing-based CLMVS scheme secure in the standard model which is of a strong confirmation. This scheme is based on our second generic CLMVS construction, and uses Waters' signature [18].

6.1 Preliminaries

In this section, mathematical preliminaries of our concrete scheme is being defined.

Bilinear Pairings: Let G_1 and G_2 be two multiplicative cyclic groups of large prime order q and g denotes the generator of G_1. There exists an admissible bilinear map $e : G_1 \times G_1 \to G_2$ iff the following properties are satisfied.

- Bilinearity: for all $a, b \in Z_q^*$, $e(g^a, g^b) = e(g, g)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1_{G_2}$.
- Computability: there exists an efficient algorithm for computing $e(g, g)$.

Decisional Bilinear Diffie-Hellman (DBDH) Problem in $(\mathbf{G_1}, \mathbf{G_2})$: Given $g, g^a, g^b, g^c \in G_1, e(g, g)^h \in G_2$ for randomly chosen $a, b, c, h \in Z_q^*$, the DBDH problem in (G_1, G_2) is to decide whether $e(g, g)^{abc} = e(g, g)^h$.

Gap Bilinear Diffie-Hellman (GBDH) Problem in $(\mathbf{G_1}, \mathbf{G_2})$: Given $g, g^a, g^b, g^c \in G_1$ for unknown $a, b, c \in Z_q^*$, the GBDH problem in (G_1, G_2) is to compute $e(g, g)^{abc} \in G_2$ with the help of the DBDH oracle O_{DBDH}.

The DBDH oracle O_{DBDH} takes $g, g^x, g^y, g^z \in G_1$ and $e(g, g)^h \in G_2$ as input and outputs 1 if $e(g, g)^h = e(g, g)^{xyz}$ and 0 otherwise.

Weak Gap Bilinear Diffie-Hellman (wGBDH)

Problem in $(\mathbf{G_1}, \mathbf{G_2})$: Given $g, g^a, g^b, g^c \in G_1$ for unknown $a, b, c \in Z_q^*$, the wGBDH problem in (G_1, G_2) is to compute $e(g,g)^{abc} \in G_2$ with the help of the restricted DBDH oracle O_{DBDH-r}.

The restricted DBDH oracle O_{DBDH-r} takes $g^z \in G_1$ and $e(g,g)^h \in G_2$ as input and outputs 1 if $e(g,g)^h = e(g,g)^{abz}$ and 0 otherwise.

6.2 Our Concrete Scheme

Given our proposed concrete scheme, we assume that all messages are bit strings with a length of n_m. For the sake of flexibility, we can use arbitrary lengths and apply a collision resistant hash function to convert messages to a specific length, (i.e. $H_m : \{0,1\}^* \to \{0,1\}^{n_m}$). The scheme is further described below:

1) System Parameters Generation: A trusted authority (TA) is responsible for running the system parameters generation algorithm. TA runs the algorithm with a security parameter k as input and returns system parameters $SP = \{G_1, G_2, e, q, g, g_2, H_m, H, m', M'\}$ where G_1 and G_2 are two multiplicative cyclic groups with a large prime order q. g is a generator of G_1, $e : G_1 \times G_1 \to G_2$ describes an admissible mapping, $H : G_2 \to G_1$ is a collision resistant hash function and G_1 is of a sufficiently large length, m' is a random element in G_1, $M' = (m_j)$ with length n_m is a vector whose entries are random elements from G_1, and $g_2 = g^\beta, \beta \in_R Z_q^*$.
2) Key Generation: Each entity u chooses a random number $x_u \in_R Z_q^*$ as his/her private key and computes the public key $Pk_u = g^{x_u}$. (x_s, Pk_s) denotes the signer's key pair, $(x_i, Pk_i), i = 1, \dots, n$ denotes the key pair for the n limited verifiers and (x_J, Pk_J) is the judge's key pair.
3) Limited Multi-Verifier Signing: To sign a message $m \in \{0,1\}^{n_m}$ for n limited verifiers with $\{Pk_1, Pk_2, \dots, Pk_n\}$ public keys, the signer constructs $M \subseteq \{1, \dots, n_m\}$ which is the set of all j: $m[j] = 1$ and $m[j]$ is jth bit of m. Then, the signer generates the signature using a random parameter $r_m \in_R Z_q^*$ as follows:

$$S = g_2^{x_s}(m' \prod_{j\in M} m_j)^{r_m}, R = g^{r_m} \qquad (8)$$

$$V = H(e(S, \prod_{i=1}^{n} Pk_i)) \in G_1 \qquad (9)$$

$$\sigma = S \oplus V \qquad (10)$$

The signer then sends (m, σ, R) as a signature triple to the limited verifiers.
4) Limited Multi-Verifier Verification: given a signature triple (m, σ, R) the limited verifiers compute

$$V = H(e(g_2, Pk_s)^{\sum_{i=1}^n x_i} e(m' \prod_{j\in M} m_j, R)^{\sum_{i=1}^n x_i})$$

and $S = \sigma \oplus V$.The limited verifiers accept the signature iff $e(S,g) = e(g_2, Pk_s)e(m' \prod_{j\in M} m_j, R)$. Completeness:

$$\begin{aligned} V &= H(e(S, \textstyle\prod_{i=1}^n Pk_i)) \\ &= H(e(g_2^{x_s}, \textstyle\prod_{i=1}^n g^{x_i})e((m' \prod_{j\in M} m_j)^{r_m}, \prod_{i=1}^n g^{x_i})) \\ &= H(e(g_2^{x_s}, g^{\sum_{i=1}^n x_i})e((m' \textstyle\prod_{j\in M} m_j)^{r_m}, g^{\sum_{i=1}^n x_i})) \\ &= H(e(g_2, g^{x_s})^{\sum_{i=1}^n x_i} e((m' \textstyle\prod_{j\in M} m_j, g^{r_m})^{\sum_{i=1}^n x_i})) \\ &= H(e(g_2, Pk_s)^{\sum_{i=1}^n x_i} e((m' \textstyle\prod_{j\in M} m_j, R)^{\sum_{i=1}^n x_i})). \end{aligned} \qquad (11)$$

$$\begin{aligned} e(S,g) &= e(g_2^{x_s}(m' \textstyle\prod_{j\in M} m_j)^{r_m}, g) \\ &= e(g_2^{x_s}, g)e((m' \textstyle\prod_{j\in M} m_j)^{r_m}, g) \\ &= e(g_2, g^{x_s})e(m' \textstyle\prod_{j\in M} m_j, g^{r_m}) \\ &= e(g_2, Pk_s)e(m' \textstyle\prod_{j\in M} m_j, R). \end{aligned} \qquad (12)$$

5) Confirmation proof: To designate the signature to a judge, the limited verifiers compute $\delta = e(S, Pk_J)$ and send (m, δ, R) to the judge. The judge accepts the proof if and only if (13) holds:

$$\delta = e(g_2, Pk_s)^{x_J} e(m' \prod_{j\in M} m_j, R)^{x_J}. \qquad (13)$$

Completeness:

$$\begin{aligned} \delta &= e(S, Pk_J) = e(g_2^{x_s}(m' \textstyle\prod_{j\in M} m_j)^{r_m}, g^{x_J}) \\ &= e(g_2^{x_s}, g^{x_J})e((m' \textstyle\prod_{j\in M} m_j)^{r_m}, g^{x_J}) \\ &= e(g_2, g^{x_s})^{x_J} e(m' \textstyle\prod_{j\in M} m_j, g^{r_m})^{x_J} \\ &= e(g_2, Pk_s)^{x_J} e(m' \textstyle\prod_{j\in M} m_j, R)^{x_J}. \end{aligned} \qquad (14)$$

6) Conversion Algorithm: Both the signer and the limited verifiers can convert the signature into a publicly verifiable signature (m, S, R).
7) Public Verification: Any party can thus check the validity of the signature through (15).

$$e(S,g) = e(g_2, Pk_s)e(m' \prod_{j\in M} m_j, R). \qquad (15)$$

6.3 Security Analysis

In this section, the security requirements of our concrete scheme are being proven.

Unforgeability: According to Lemma 1 from Section 3.1.1, our concrete scheme is unforgeable if the signature S is unforgeable. The signature S is Waters' signature [18] and it is (t, q, ϵ)–existentially unforgeable against the adaptive chosen message attack in the standard model, assuming the decisional $(t, \frac{\epsilon}{16(n_m+1)q})$–BDH problem is intractable.

Hence, our concrete scheme is existentially unforgeable against the adaptive chosen message attack with the same assumption in the standard model.

Non-transferability: According to Lemma 2 from Section 3.1.2, our concrete scheme is non-transferable if the UDVS scheme is non-transferable. The UDVS scheme used here is the Laguillaumie *et al.*'s scheme [19]. Based on [19], the UDVS scheme is non-transferable and thus our proposed scheme is non-transferable as well.

Limited Verifiability: As was observed in Section 3.1.3, V, σ and S should be unforgeable and according to Section 5.2, the UDMVS scheme should use the private key of the recipients for verification.

Proof 1: based on the scheme used here, V needs the private key of the recipients for verification, and as proved in the Appendix A, with the help of technique used in [20], it is unforgeable with the probability shown in (16) and in time t', assuming that the GBDH problem is intractable.

$$Pr[succ(forge)] \geq \frac{\epsilon}{8(q_{ps}+q_{ds})(n_m+1)}$$
$$t' \leq t + O((q_{ps}n_m + q_{ds}(n_m+n) + q_{dv}(n_m+n))t_1$$
$$+q_{dv}t_2 + (q_{ps}+q_{ds}+q_{dv})T_1 + q_{dv}T_2 + (q_{ds}+q_{dv})t_e) \tag{16}$$

where t_1 and t_2 are the times for multiplication in G_1 and G_2 respectively; T_1 and T_2 are the times for an exponentiation in G_1 and G_2 respectively; and t_e is the time for a pairing computation in (G_1, G_2).

Therefore, V is unforgeable in the standard model assuming the GBDH problem is intractable.

Proof 2: Based on [19], δ is unforgeable assuming that the wGBDH problem is intractable.

Proof 3: Based on [18], S is unforgeable assuming that the DBDH problem is intractable.

Therefore, our concrete scheme is proven to be limited verifier in the standard model.

6.4 Discussion

If the concrete scheme is being used for one limited verifier, a CLVS scheme, it will be comparable with the Shen and Ming's CLVS scheme [11] which is also secure in the standard model. In this regard, our presented CLVS scheme is the first pairing-based CLVS scheme secure in the standard model while the Shen and Ming's scheme [11] is identity-based and encounters the key escrow problem. Note that our concrete scheme is pairing-based and thus the key escrow is not an issue. In Shen and Ming's scheme [11], the private key of the third party is not required for the confirmation proof, yet in our concrete scheme the private key of the third party is necessary in this respect (strong confirmation property).

7 Efficiency

In this section, comparisons are being made between our presented CLMVS scheme and the one proposed by Wang *et al.* [9] in terms of communication and computation cost.

Suppose that the arbitrary length of the message m is l_1, the length of a point in G_1 is l_2, the length of an element of G_2 is l_3 and the number of limited verifiers is n. Based on [9], it is assumed that the signature is published via a broadcast communication channel. This assumption is fair since in the CLMVS schemes no one can verify the signature except the limited verifiers. Table 2 presents the comparison of communication costs for the Wang *et al.*'s scheme [9] and our concrete scheme.

It is well observable from Table 2 that our concrete scheme is more efficient than the Wang *et al.* scheme [9] since its communication cost remains constant for any number of limited verifiers, while in the Wang *et al.*'s scheme [9], the communication cost increases in proportion to n. Therefore, our scheme is efficient in that it can generate a unique signature for n limited verifiers. This conclusion is also true for our generic constructions of CLMVS as their communication cost also remains constant.

Table 3 presents the computation cost of the Wang *et al.*'s scheme [9] and our concrete scheme. In this table, e denotes the pairing operation, H is the hash operation, M stands for multiplication and E implies exponentiation. These operations are the most time-consuming operations. n_m also denotes the length of message m (Section 6.2).

From Table 3, at the first glance, one may take the view that the concrete scheme is not efficient. However, our goal in proposing this concrete scheme is provable security in the standard model not computation efficiency. One of the main contributions of our proposal is presenting a concrete CLMVS scheme in the standard model due to the criticism to random oracle model. This is the reason behind selecting the Laguillaumie *et al.*'s scheme [19] as a building block. Note that every instantiation for generic construction is possible. If a more efficient scheme is desired compared to the one proposed by Wang *et al.* [9], we could employ some building blocks in random oracle model without pairings.

To fairly compare our generic constructions with the Wang *et al.*'s scheme [9] and in order to show how the computation efficiency increases, we have extended our pairing-based CLVS scheme to a CLMVS scheme adopting the method used in the Wang *et al.*'s scheme [9] and then comparisons were drawn with our scheme. The results are presented in Table 4.

Table 2. Comparison of communication cost

Schemes	Wang *et al.* scheme [9]	Our concrete scheme
From LMV-signing to LMV- verification	$l_1 + (n+1)l$	$l_1 + 2l_2$
From LMV- verifierication to confirmation	$l_1 + l_2 + l_3$	$l_1 + l_2 + l_3$
From confirmation to public verification	$l_1 + 2l_2$	$l_1 + 2l_2$

Table 3. Comparison of computation cost

Schemes	Security model	e	H	MinG$_1$	MinG$_2$	EinG$_1$	EinG$_2$
Wang *et al.* scheme [9]	Random oracle	$3n+4$	$3n+3$	$n+5$	0	0	0
Our concrete scheme	Standard	$5n+7$	$n+1$	$n+4n_m$	$4n+2$	2	$2n+2$

Table 4. Comparison of computation cost in LMV-signing stage

Schemes	Security model	e	H	MinG$_1$	EinG$_1$
The generated pairing-based CLMVS scheme using the method in the Wang *et al.* scheme [9]	Standard	n	n	n_m	2
Our concrete scheme	Standard	1	1	$n+n_m$	2

One can see from Table 4 that the method used in our construction is more efficient regarding the computation cost in LMV-signing stage (the computation cost in other stages are identical in both schemes). Hence, our method could be said to be more efficient in comparison with the one presented by Wang *et al.* [9].

8 Application of the Presented CLVS Scheme in Electronic Voting

A secret voting scheme is classified into four categories: hidden voter, hidden vote, hidden voter with hidden vote [21] and semi-hidden voter with semi-hidden vote.

In the hidden voter approach, the voter remains anonymous while the vote is submitted to the tally authority plainly. In the hidden vote, the voter is obvious while the vote is encrypted. The hidden voter with hidden vote (HVHV) is a hybrid of two previous classes and benefits from both approaches [21]. We have used the term semi-hidden voter with semi-hidden vote (semi-HVHV) to refer to the recent techniques [22, 23] used in voting protocol designing in which the voter uses the HVHV approach to send the vote to the tally authority, while the voter and the plain vote are known to the tally authority (i.e. semi).

In the semi-HVHV approach, the use of HVHV helps provide fairness and preserves the privacy of the voter against eavesdroppers; however, the information leaks to the tally authority decreases the reliability of the scheme. Therefore, it is essential to prevent the tally authority from misusing the information she/he knows. This property is called non-transferability and can be addressed with deniable authentication schemes [24] and any other scheme which provides non-transferability (i.e. semi-HVHV). By doing so, the semi-HVHV can strengthen the voting protocol against the authorities' coercion or collusion, a feature which was lacking in the previous approaches. However, a drawback of the semi-HVHV is that the non-transferability subjects the protocol to the "simulation attack", or more informally, "ballot forging". To overcome this attack, it is necessary to assume that the tally authority is trusted. In practice, the trust assumption can be implemented by distributing trust among several authorities or through the technique presented in [23].

In the following, our proposed pairing-based CLVS scheme is employed to introduce a new voting protocol. To the best of the authors' knowledge, this is the first time that a CLVS scheme is used in a voting protocol. This protocol is based on the semi-HVHV approach since the CLVS scheme provides non-transferability. Additionally, the presented CLVS scheme is secure in the standard model which can enrich the security of our protocol in practice.

8.1 Security Requirements

An electronic voting protocol should satisfy the following security requirements:

- Completeness: All valid votes are counted in the final poll.

- Uniqueness: One vote per each voter is counted in the final poll.
- Privacy: No one can find any links between the vote and the voter which leads into misuses.
- Soundness: No dishonest voter can disrupt the voting process.
- Eligibility: Only eligible voters can cast their vote.
- Fairness: No partially information leaks during the voting phase which can in turn change the final result.
- Verifiability: Every third party can verify the counting results, which is called universal verifiability, and each voter can verify that his vote has been counted, which is called individual verifiability.
- Receipt-freeness: No one, even the voter, can generate a receipt which shows how the voter has voted.
- Uncoercibility: No one can coerce a voter to cast a ballot against his/her intention.
- Non-transferability: The receiver cannot transfer the information he/she knows to a third party since he/she can simulate the received information.

In this proposal, we integrate the eligibility and non-transferability properties and call the new property "deniable authenticity".

8.2 Protocol Structure

Our proposed scheme involves three participants, as described below:

Voter. Denoted by v_j, the voter has a private/public key pair(x_{v_j}, Pk_{v_j}) and a valid certificate for her key pair.

Registration Authority. For simplicity, only one registration authority is used and denoted by R. R is responsible for publishing the necessary parameters of the building blocks which are used in the voting process. R has a private/public key pair (x_s, Pk_s) and he also generates a private/public key pair (x_{pub}, Pk_{pub}) for public verification and publishes Pk_{pub}, while keeping x_{pub} secret until the end of the voting phase.

Tally Authority. For simplicity, only one tally authority is used which is denoted by T. T is responsible for tallying the valid votes and publishing the results on the bulletin board. (x_T, Pk_T) denotes the private/public key pair of T.

8.2.1 Assumptions

In order to shed light on the application of the proposed CLVS scheme in a voting protocol, the following six assumptions are made to simplify the voting protocol:

A1. The registration phase is trustworthy.

A2. Each entity keeps his/her private key secret.

A3. Each voter casts one ballot.

A4. The voter is honest. (This assumption can be met using a smart card, for instance.)

A5. The tally authority is trusted scheme for tallying the valid votes.

A6. The tally authority would not forge a ballot instead of any voter.

8.2.2 The Scheme

In the process of constructing the scheme, the following four phases are to be followed:

Setup Phase. This phase is carried out few days or few weeks before the voting phase. The registration authority R establishes the necessary parameters of the CLVS scheme as proposed in Section 6, a secure signature scheme, $Sign$, and a secure public key encryption algorithm, E. R also publishes the list of eligible voters, his public key Pk_s and the verification public key Pk_{pub}.

Registration Phase. v_j encrypts his voting identity $ident_j$ along with his signature on $ident_j$ and sends $E_{Pk_R}(ident_j, Sign(ident_j))$ to R via an anonymous channel. R decrypts the received message and checks $ident_j$. If it belongs to an eligible voter, then he verifies the signature and if the signature is valid, then he signs $ident_j$ using the proposed CLVS scheme as follows. He,

1) Chooses r_m randomly and computes $R = g^{r_m}$.
2) Generates $S = g_2^{x_s}(m' \prod_{j \in IDENT_j} m_j)^{r_m}$ and $V = H(e(S, Pk_j))$; where m' is a random number; $M' = (m_j)$ is a random vector with the length of n_m and $IDENT_j \subseteq \{1, \ldots, n_m\}$ is composed of all i which satisfy $ident_j[i] = 1$.
3) Sends (σ, R) to v_j.

v_j computes:

$$V = H(g_2, Pk_s)^{x_j} e(m' \prod_{j \in IDENT_j} m_j, R)^{x_j}$$

and $S = \sigma \oplus V$. Next, she verifies S and if it was valid then, she accepts it, otherwise S is rejected and reported to R.

Voting Phase. This phase is started immediately after the registration phase. v_j chooses a random number $r_b \in Z_q^*$ and his favorite candidate b^t, and then she performs the following steps. She,

1) Computes $S' = S.(u' \prod_{j \in B^t} u_j)^{r_b}$ and $R_b = g^{r_b}$; where u' is a random number; $U' = (u_j)$ is a random vector with a length of n_u and $B^t \subseteq \{1, \ldots, n_u\}$ is composed of all $i$es which satisfy $b^t[i] = 1$.
2) Computes $\delta = e(S', Pk_T)$ and $\gamma = e(S, Pk_{pub})$.

3) Sends $E_{Pk_T}(ident_j, b^t, \gamma, R, R_b)$ to T via an anonymous channel.

Tallying Phase. T decrypts the received ballots and removes duplicate of ballots with the same $ident_j$ and b^t. Next, he verifies δ using (17).

$$\delta = e(g_2, Pk_s)^{x_T} e(m' \prod_{j \in IDENT_j} m_j)^{x_T} \dots e(u' \prod_{j \in B^t} u_j, R_b)^{x_T} \tag{17}$$

Completeness:

$$\begin{aligned}
\delta &= e(S', Pk_T) \\
&= e(S, Pk_T) e(u' \prod_{j \in B^t} u_j)^{r_b}, Pk_T) \\
&= e(g_2^{x_s}, g^{x_T}) e((m' \prod_{j \in IDENT_j} m_j)^{r_m}, Pk_T) \\
&\quad \dots e((u' \prod_{j \in B^t} u_j)^{r_b}, Pk_T) \\
&= e(g_2, g^{x_s})^{x_T} e(m' \prod_{j \in IDENT_j} m_j, g^{r_m})^{x_T} \\
&\quad e(u' \prod_{j \in B^t} u_j, g^{r_b})^{x_T} \\
&= e(g_2, Pk_s)^{x_T} e(m' \prod_{j \in IDENT_j} m_j, R)^{x_T} \\
&\quad e(u' \prod_{j \in B^t} u_j, R_b)^{x_T}
\end{aligned} \tag{18}$$

Then, T counts b^t in total poll and sends b' and $(ident_j, \gamma, R)$ to the bulletin board separately.

As mentioned before, R reveals x_{pub} for public verification after the tallying phase is terminated.

8.3 Security Analysis

In this section, the security requirements of the presented voting protocol are being explained.

- Completeness: T is assumed to be our concrete scheme in verifying and tallying the ballots and all algorithms are perfect. Thus, all valid votes are counted in the final poll.
- Uniqueness: T removes the duplicate of ballots with the same $ident_j$ and b^t. Additionally, each voter casts one ballot. Thus, one vote per each voter is counted in the final poll.
- Privacy: The registration and voting phases are carried out via the anonymous channel and the messages containing $ident_j$ are encrypted; hence the link between the vote and the voter is not obvious for an eavesdropper. However, this link is intentionally known to T. Since the CLVS scheme is non-transferable, T can simulate a ballot and no one would be convinced. Furthermore, when the results are shown, b^t is separated from $(ident_j, \gamma, R)$. Therefore, no one can find and misuse the linkage between the vote and the voter.
- Soundness: This property is provided using the assumption A4.
- Fairness: The Ballots are encrypted and the tallying phase is completed after the voting phase. Thus, no partial information would leak during the voting phase.
- Verifiability: Each voter can see her $(ident_j, \gamma, R)$ on the bulletin board. Moreover, T is trusted in tallying the ballots and he would not forge ballots. Thus, the scheme provides individual verifiability. Additionally, after revealing x_{pub} by R, everyone can verify $(ident_j, \gamma, R)$, and therefore, the scheme provides universal verifiability as well.
- Receipt-freeness: b^t and $(ident_j, \gamma, R)$ are listed separately and the voter casts her ballot anonymously. Furthermore, T cannot transfer his knowledge to a third party. This will be discussed in deniable authenticity property.
- Uncoercibility:
  - ⋄ Comparing transmitted ballots [25]: The voter encrypts her ballot and sends it via an anonymous channel. Thus, the adversary cannot control the behavior of the voter through comparing the transmitted ballots.
- Deniable Authenticity: R checks the eligibility of the voter before signing. Furthermore,

1) $ident_j, b^t, R, R_b, \delta = e(s', Pk_T)$ is a deniable authentication scheme:
   a) Unforgeability: $e(s', Pk_T)$ is the hierarchical extension of [19] and it is unforgeable assuming that the GBDH problem is intractable (as shown in Appendix B, through serving some techniques from [18, 26]), with probability (19) and in time t':

$$\begin{aligned}
&Pr[succ(forge)] \geq \\
&\frac{\epsilon}{2^{q_{dv}+3}(q_{ps}+q_{ds})^{q_{dv}+2}(n_m+1)(n_u+1)} \\
&t' \leq t + O((q_{ps} n_m + q_{ds}(n_m + n_u) \\
&+q_{dv}(n_m))t_1 + q_{dv} t_2 + (q_{ps} + q_{ds} + q_{dv})T_1 \\
&+q_{dv} T_2 + (q_{ds} + q_{dv}) t_e)
\end{aligned} \tag{19}$$

   where t_1 and t_2 are the times for multiplication in G_1 and G_2 respectively; T_1 and T_2 are the times for an exponentiation in G_1 and G_2 respectively; and t_e is the time for a pairing computation in (G_1, G_2).

   b) Non-transferability: This property is proven in appendix C.

   Simulation algorithm:

$$\begin{aligned}
&ident_j, b_2^t, R', R_b' \\
&\delta' = e(g_2, Pk_s)^{x_T} e(m' \prod_{j \in IDENT_j} m_j, R')^{x_T} \\
&\dots e(u' \prod_{j \in B_2^t} u_j, R_b')^{x_T}
\end{aligned} \tag{20}$$

2) $\gamma = e(S, Pk_{pub})$ provides deniable authentication.

a) Unforgeability: Based on [19], γ is unforgeable assuming the wGBDH problem is intractable.
b) Non-transferability: Once the tallying phase is terminated, x_{pub} is revealed and everyone can simulate γ based on [19]; hence γ is non-transferable.

Note that in the prevalent signature-based deniable authentication protocols, the sender (at this juncture, the voter) is the signer herself, while the signature is generated by the registration authority in our scheme. Consequently, if the sender is a member of an eligible group endorsed by the registration authority, then the message and the identity of the sender are deniably authenticated.

8.4 Discussion

Disregarding assumption A5, in the proposed voting protocol, T can only forge ballots instead of the present voters, since he does not know x_{pub} and cannot simulate $(ident_j, \gamma, R)$ instead of the absent voters.

In order to improve the scheme for the cases where assumption A5 is not met, as mentioned before, we can distribute trust among several authorities or use the technique presented in [23] to protect the voter against ballot forging. Based on [23], we can add an observer, such as a judge, to the scheme who controls the tallying phase and interferes when a forgery takes place. Moreover, an unforgeable part is added to the ballot which can be $P = E_K(ident_j, b^t, \delta, R, R_b)$; where K is a secure common key between the voter and the judge. The judge can find the forgery if the signature is not validated after decrypting P.

9 Conclusions

In the presented paper, a generic construction was proposed for CLVS scheme and it was indicated that all the previous secure CLVS schemes fit into it. The UDVS scheme was regarded as the building block of this construction. In addition, for comparison purposes, we classified the existing secure schemes based on the UDVS scheme, $SimTr$ and the function having been used in them.

Subsequently, this construction was extended into two efficient generic CLMVS constructions with a unique signature for more than two limited verifiers. In the first construction, each limited verifier can verify, confirm or convert the signature solely while in the second construction verification is done with the cooperation of all the limited verifiers, yet the confirmation or conversion of the signature could be done solely.

Finally, we proposed the first pairing-based CLMVS scheme based on Waters' signature. This scheme was proven to be secure in the standard model and is of a strong confirmation property. These properties has made the presented CLMVS scheme, with one limited verifier (CLVS), suitable to be employed as a building block in a voting protocol. Consequently, we employed the proposed CLVS scheme to design a new voting protocol within a semi-hidden voter with semi-hidden vote approach.

10 Acknowledgment

This research was supported by the Deputy of Science and Technology through grant no. 19435/11.

References

[1] D. Chaum and H. V. Antwerpen, "Undeniable signatures", Advances in Cryptology –CRYPTO'89, LNCS 435, pp. 212–216, 1989.
[2] M. Jakobsson, K. Sako and R. Impagliazzo, "Designated verifier proofs and their applications", Advances in Cryptology –EUROCRYPT'96, LNCS 1070, pp. 143–154, 1996.
[3] S. Araki, S. Uehara and K. Imamura, "The limited verifier signature and its application", IEICE Trans. Fundamentals, vol.E82-A, no.1, pp. 63–68, 1999.
[4] F. Zhang and K. Kim, "A Universal forgery on Araki *et al.*'s convertible limited verifier signature scheme", IEICE Trans. Fundamentals, vol. E86-A, no.2, pp.515–516, 2003.
[5] X. Chen, F. Zhang and K. Kim, "Limited verifier signature from bilinear pairings", in Proc. ACNS'04, Berlin: Springer, LNCS 3089, pp. 135–148, 2004.
[6] J. Zhang, H. Li and J. Wang, "A Convertible limited verifier signature scheme", WAIM 2005, Berlin: Springer, LNCS 3739, pp. 638–644, 2005.
[7] Z.-H. Shao, "Comment on a convertible limited verifier signature scheme", Journal of Zhejiang University of Science and Technology, vol.18, no.4, pp. 262–267, 2006.
[8] C. Wu and Z. Huang, "Certificateless convertible limited verifier signature scheme", TECON'06, IEEE, Hong Kong, China, 2006, pp.1–4.
[9] X. Wang, L. Cao, S. Wang and Y. Zhang, "Id-based convertible limited (multi-) verifier signature scheme", ICCSSE'08, IEEE CS, Wuhan, Hubei, 2008, pp. 774–777.
[10] R. Tso, X. Yi, T. Okamoto and E. Okamoto, "Efficient convertible limited verifier signatures", ISIT'08, IEEE , Toronto, Canada, 2008, pp. 230–234.

[11] X.Q. Shen and Y. Ming, “Identity-based convertible limited verifier signature scheme in the standard model”, Journal of Applied Mechanics and Materials, Trans. Tech. Publications, vols.48–49, pp. 599–602, 2011.

[12] F. Hess. “Efficient identity based signature schemes based on pairings”, SAC 2002, K. Nyberg and H. Heys Eds., Berlin: Springer, LNCS 2595, pp. 310–324, 2003.

[13] F. Laguillaumie and D. Vergnaud, “Designated verifier signatures: anonymity and efficient construction from any bilinear map”, SCN’04, Berlin: Springer, LNCS 3352, pp. 105–119, 2004.

[14] C.Y. Ng, W. Susilo and Y. Mu, “Universal designated multi verifier signature schemes”, in Proc. 11th Int. Conf. Parallel and Distributed Systems (ICPADS’05), 2005, vol.2, pp. 305–309.

[15] Y. Ming, X.Q. Shen and Y.M. Wang, “Certificateless universal designated verifier signature schemes”, Journal of China Universities of Posts and Telecommunications, vol.14, no.3, pp.85–90, 2007.

[16] S.-H. Seo, J.Y. Hwang. K.Y. Choi and D.H. Lee, “Identity-based universal designated multi-verifiers signature schemes”, Journal of Computer Standards and Interfaces, Elsevier Inc., vol.30, pp. 288–295, 2008.

[17] F. Cao and Z. Cao, “An identity based universal designated verifier signature scheme secure in the standard model”, Journal of Systems and Software, Elsevier Inc., vol.82, pp. 643–649, 2009.

[18] B. Waters, “Efficient identity-based encryption without random oracles”, Advances in Cryptology –EUROCRYPT’05, Berlin: Springer, LNCS 3494, pp. 114–127, 2005.

[19] F. Laguillaumie, B. Libert and J.J. Quisquater, “Universal designated verifier signatures without random oracles or non-black box assumptions”, in Proc. 5th Int. Conf. Security and Cryptography for Networks (SCN 2006), Berlin: Springer, LNCS 4116, pp. 63–77, 2006.

[20] Y. Ming and Y. Wang, “Universal designated multi verifier signature scheme without random oracles”, Journal of Natural Sciences, Wuhan University, vol.13, no.6, pp. 685–491, 2008.

[21] K. Sampigethaya and R. Poovendran, “A framework and taxonomy for comparison of electronic voting schemes”, Computers and Security, vol.25, no.2, pp. 137–153, 2006.

[22] C.T. Li, M.S. Hwang and C.Y. Liu, “An electronic voting protocol with deniable authentication for mobile ad hoc networks”, Journal of Computer Communications, Elsevier Inc.,vol.31, pp.2534–2540, 2008.

[23] S. Avizheh, M. Rajabzadeh Asaar and M. Salmasizadeh, “A new internet voting protocol with voter’s protection based on deniable authentication”, in Proc. 9th Int. ISC Conf. Information Security and Cryptology, Tabriz, Iran, 2012.

[24] C. Dwork, M. Naor and A. Sahai, “Concurrent zero-knowledge”, in Proc. 30th Annu. ACM Symp. Theory of Computing, Dallas TX, USA, 1998, pp. 409–418.

[25] Y.F. Chung and Z.Y. Wu, “Approach to designing bribery-free and coercion-free electronic voting scheme”, Journal of Systems and Software, Elsevier Inc., vol. 82, pp. 2081–2090, 2009.

[26] K.G. Paterson and J.C.N. Schuldt, “Efficient identity-based signature secure in the standard model”, In: Batten, L.M., Safavi-Naini, R. (Eds.), ACISP 2006, Berlin: Springer, LNCS 4058, pp. 207–222, 2006.

Appendix A. Theorem 1

V is unforgeable with the following probability and in time t', assuming that the GBDH problem is intractable.

$$\begin{aligned}&Pr[succ(forge)] \geq \tfrac{\epsilon}{8(q_{ps}+q_{ds})(n_m+1)}\\ &t' \leq t + O((q_{ps}n_m + q_{ds}(n_m+n) + q_{dv}(n_m+n))t_1\\ &+q_{dv}t_2 + (q_{ps}+q_{ds}+q_{dv})T_1 + q_{dv}T_2 + (q_{ds}+q_{dv})t_e)\end{aligned} \tag{21}$$

where t_1 and t_2 are the times for multiplication in G_1 and G_2 respectively; T_1 and T_2 are the times for an exponentiation in G_1 and G_2 respectively; and t_e is the time for a pairing computation in (G_1, G_2).

Proof. Let’s assume that $\lambda = e(S, \prod_{i=1}^{n} Pk_i)$ and $V = H(\lambda)$. In order to prove the unforgeability of V, we first take F as a forger who forges λ after enough queries within time t and with probability ϵ, identical to [20]. F uses another algorithm B to break the GBDH assumption within time t' with probability ϵ'. B responds to F’s queries as follow:

Setup: The challenge (g, g^a, g^b, g^c) is given to B and B tries to solve the GBDH problem with the help of DBDH oracle. B picks $k \in \{0, \ldots, n_m\}$ randomly and sets an integer $l = 2(q_{ps} + q_{ds})$, where $l(n_m + 1) < q$ and $0 \leq kl < q$. B then chooses a value $x' \overset{R}{\leftarrow} Z_l$ and a random n_m-vector $(x_1, \ldots, x_{n_m})$, where $\{x_j \in Z_l, \forall j\}$. Additionally, B randomly picks a value $y' \overset{R}{\leftarrow} Z_q$ and a random n_m-vector $(y_1, \ldots, y_{n_m})$, where $\{y_j \in Z_q, \forall j\}$. These values are kept internal to B. For a message m, $M \subseteq \{1, \ldots, n_m\}$is composed of all $j$es which satisfy $m[j] = 1$. To simplify the analysis, we form three functions $J(m)$, $K(m)$ and $F(m)$ identical to the ones in Waters’ scheme [18].

$$\begin{aligned} J(m) &= x' + \textstyle\sum_{j \in M} x_j - kl \\ K(m) &= y' + \textstyle\sum_{j \in M} y_j \\ F(m) &= h^{J(m)} g^{K(m)} \end{aligned} \tag{22}$$

Then, B assigns the public keys of entities and the common parameters as follows:

1. B sets the public key of the signer as $Pk_s = g^a$, the public parameter $h = g_2 = g^b$ and the public keys of n designated verifiers as $Pk_i = g^{d_i c}$, where $d_i \in_R Z_q^*, i \in \{1, \ldots, n\}$.
2. B sets $m' = h^{x'-kl} g_{y'}$, $m_j = h^{x_j} g^{y_j}$ and $\boldsymbol{m} = \{m_1, m_2, \ldots, m_{n_m}\}$.

Under these assignments, we have $m' \prod_{j \in M} m_j = h^{J(m)} g^{K(m)} = F(m)$. Afterwards, B returns $(G, G_1, e, q, g, m', \boldsymbol{m}, h)$ to the forger F.

Sign queries (PS): Suppose that F asks for a signature on the message $m \in \{0,1\}^{n_m}$. Then, B terminates the simulation and reports failure if $J(m) = 0 \pmod q$. Otherwise, B chooses $r \overset{R}{\leftarrow} Z_q$ at random and computes the signature as follows [20]:

$$\sigma = (\sigma_1, \sigma_2) = (Pk_s^{\frac{-K(m)}{J(m)}} F(m)^r, h^{\frac{-1}{J(m)}} g^r) \tag{23}$$

Completeness:

$$\begin{aligned} \sigma_1 &= Pk_s^{\frac{-K(m)}{J(m)}} F(m)^r \\ &= h^a (h^{J(m)} g^{K(m)})^{-\frac{a}{J(m)}} (h^{J(m)} g^{K(m)})^r \\ &= h^a (h^{J(m)} g^{K(m)})^{r - \frac{a}{J(m)}} \\ &= h^a (h^{J(m)} g^{K(m)})^{\tilde{r}} \end{aligned} \tag{24}$$

Considering $\tilde{r} = r - \frac{a}{J(m)}$, we have:

$$\sigma = (\sigma_1, \sigma_2) = (h^a (h^{J(m)} g^{K(m)})^{\tilde{r}}, \tilde{r}) \tag{25}$$

Which is a valid signature.

Designated verifier signature queries (DS): Suppose that F asks for a signature on the message $m \in \{0,1\}^{n_m}$. Then, B terminates the simulation and reports failure if $J(m) = 0 \pmod q$. Otherwise, B issues a query to PS and computes the designated verifier's signature using the response coming from PS [20].

Designated verifier verification queries (DV): Suppose that F asks for a designated verifier verification on the message/designated verifier signature $(m, \bar{\sigma})$. Then, B responds as follows [20]:

1. If $J(m) \neq 0 \pmod q$, B computes another designated verifier signature $\hat{\sigma} = (\hat{\sigma_1}, \hat{\sigma_2})$ on the message m using the technique used in DS, and then sends:

$$(g, h^{J(m)} g^{K(m)}, \prod_{i=1}^{n} Pk_i, \frac{\bar{\sigma_2}}{\hat{\sigma_2}}, \frac{\bar{\sigma_1}}{\hat{\sigma_1}})$$

to the DBDH oracle O_{DBDH}. O_{DBDH} outputs $''1''$ if the received tuple is the BDH tuple, and $''0''$ otherwise.

Completeness: Let $\bar{\sigma} = (\bar{\sigma_1}, \bar{\sigma_2})$ be a valid signature.

$$\begin{aligned} \bar{\sigma_1} &= e(g^b, g^a)^{\sum_{i=1}^n d_i c} e(h^{J(m)} g^{K(m)}, \textstyle\prod_{i=1}^n Pk_i)^r \\ \bar{\sigma_2} &= g^r \end{aligned} \tag{26}$$

$\hat{\sigma} = (\hat{\sigma_1}, \hat{\sigma_2})$ is a valid signature:

$$\begin{aligned} \hat{\sigma_1} &= e(g^b, g^a)^{\sum_{i=1}^n d_i c} e(h^{J(m)} g^{K(m)}, \textstyle\prod_{i=1}^n Pk_i)^{r'} \\ \hat{\sigma_2} &= g^{r'} \end{aligned} \tag{27}$$

As a result:

$$\frac{\bar{\sigma_2}}{\hat{\sigma_2}} = g^{r-r'}, \frac{\bar{\sigma_1}}{\hat{\sigma_1}} = e(h^{J(m)} g^{K(m)}, \prod_{i=1}^{n} Pk_i)^{r-r'} \tag{28}$$

$(g, h^{J(m)} g^{K(m)}, \prod_{i=1}^n Pk_i, \frac{\bar{\sigma_2}}{\hat{\sigma_2}}, \frac{\bar{\sigma_1}}{\hat{\sigma_1}})$ is a valid BDH tuple.

2. If $J(m) = 0 \pmod q$, B sends (29) to O_{DBDH}. O_{DBDH} outputs $''1''$ if the received tuple is the BDH tuple, and $''0''$ otherwise.

$$\left(g, g^a, g^b, \prod_{i=1}^{n} Pk_i, \frac{\bar{\sigma_1}}{e(\prod_{i=1}^n Pk_i, \bar{\sigma_2}^{K(m)})} \right) \tag{29}$$

Probability Analysis: The forger issues enough queries and outputs the signature $\lambda^* = (\lambda_1^*, \lambda_2^*)$ on a new message m^*. In this case, there are two possible modes:

Mode 1: λ^* is a valid signature on the message m^* : if $J(m) \neq 0 \pmod q$, B reports failure. Otherwise, B solves the GBDH problem as follows [20]:

$$\begin{aligned} &\left(\frac{\lambda_1^*}{e(\prod_{i=1}^n Pk_i, \lambda_2^{*K(m^*)})} \right)^{(\sum_{i=1}^n d_i)^{-1}} \\ &= \left(\frac{e(g^b, g^a)^{\sum_1^n d_i c} e(h^{J(m^*)} g^{K(m^*)}, \lambda_2^*)^{\sum_1^n d_i c}}{e(\prod_{i=1}^n Pk_i, \lambda_2^{*K(m)})} \right)^{(\sum_1^n d_i)^{-1}} \\ &= \left(\frac{e(g^b, g^a)^{c \sum_{i=1}^n d_i} e(g^{K(m^*)}, \lambda_2^*)^{c \sum_{i=1}^n d_i}}{e(g^{\sum_{i=1}^n d_i c}, \lambda_2^{*K(m^*)})} \right)^{(\sum_{i=1}^n d_i)^{-1}} \\ &= \left(\frac{e(g^b, g^a)^{c \sum_{i=1}^n d_i} e(g^{K(m^*)}, \lambda_2^*)^{c \sum_{i=1}^n d_i}}{e(g^{K(m^*)}, \lambda_2^*)^{c \sum_{i=1}^n d_i}} \right)^{(\sum_{i=1}^n d_i)^{-1}} \\ &= e(g, g)^{abc} \end{aligned} \tag{30}$$

Success Probability: Let $A : J(m^*) = 0 \pmod q$ and $A_i : J(m_i) \neq 0 \pmod l$. $Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i \wedge A\right].\epsilon$ is:

$$\begin{aligned} &Pr\left[\textstyle\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i\right] = 1 - Pr\left[\textstyle\bigvee_{i=1}^{q_{ps}+q_{ds}} A_i\right] \\ &= 1 - Pr\left[\textstyle\bigvee_{i=1}^{q_{ps}+q_{ds}} J(m_i) = 0 \pmod l\right] \\ &\geq 1 - \textstyle\sum_{i=1}^{q_{ps}+q_{ds}} J(m_i) = 0 \pmod l = 1 - \frac{q_{ps}+q_{ds}}{l} \end{aligned} \tag{31}$$

$$\begin{aligned} &Pr[A] = Pr[J(m^*) = 0 \pmod q] \\ &= Pr[J(m^*) = 0 \pmod l] \ldots \\ &Pr[J(m^*) = 0 \pmod q | J(m^*) = 0 \pmod l] \\ &= \frac{1}{l} \frac{1}{n_m + 1} \end{aligned} \tag{32}$$

$$\begin{aligned}&Pr\left[succ\right] \geq Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i \wedge A\right].\epsilon \\ &= Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i\right] Pr\left[A\right].\epsilon \\ &= (1-\tfrac{q_{ps}+q_{ds}}{l})\left(\tfrac{1}{l}\tfrac{1}{n_m+1}\right).\epsilon \\ &= \tfrac{1}{4(q_{ps}+q_{ds})(n_m+1)}\end{aligned} \tag{33}$$

Thus [20],

$$Pr\left[solvingGBDH\right] \geq \frac{\epsilon}{4(q_{ps}+q_{ds})(n_m+1)} \tag{34}$$

Mode 2: λ^* is not a valid signature on the message m^*, however, $V' = H(\lambda^*)$ is equal to $\tilde{V} = H(\tilde{\lambda})$ while $\tilde{\lambda}$ is a valid signature on message $\tilde{m}$ which was queried before. In this case, V' is a collision for $\tilde{V}$. The probability of finding a collision is:

$$\begin{aligned}&Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i \wedge v' = \tilde{V}\right] \\ &= Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i\right] Pr\left[V' = \tilde{V}\right] \\ &= (1-\tfrac{q_{ps}+q_{ds}}{l})\tfrac{1}{l}\tfrac{1}{2^{le+1}} = \tfrac{1}{2^{le+1}}\end{aligned} \tag{35}$$

where $le = length(G_1), l = 2(q_{ps}+q_{ds})$.

$$\begin{aligned}&Pr\left[succ(forge)\right] \geq \left[\tfrac{1}{2}(\tfrac{\epsilon}{4(q_{ps}+q_{ds})(n_m+1)}) + \tfrac{1}{2^{le+1}}\right) \\ &= \tfrac{\epsilon}{8(q_{ps}+q_{ds})(n_m+1)}\end{aligned} \tag{36}$$

Hence, V is unforgeable in the standard model assuming the GBDH problem is intractable.

□

Appendix B. Theorem 2

$\delta = e(S', Pk_T)$ is a hierarchical extension of [19] and it is unforgeable with the following probability in time t' assuming the GBDH problem is intractable.

$$\begin{aligned}&Pr[succ(forge)] \geq \\ &\tfrac{\epsilon}{2^{q_{dv}+3}(q_{ps}+q_{ds})^{q_{dv}+2}(n_m+1)(n_u+1)} \\ &t' \leq t + O((q_{ps}n_m + q_{ds}(n_m+n_u) + q_{dv}(n_m))t_1 \\ &+ q_{dv}t_2 + (q_{ps}+q_{ds}+q_{dv})T_1 + q_{dv}T_2 + (q_{ds}+q_{dv})t_e)\end{aligned} \tag{37}$$

where t_1 and t_2 are the times for multiplication in G_1 and G_2 respectively; T_1 and T_2 are the times for an exponentiation in G_1 and G_2 respectively; and t_e is the time for a pairing computation in (G_1, G_2).

Proof. Let F be a forger who forges δ after enough queries within time t and with probability ϵ. F uses another algorithm B to break the GBDH assumption within time t' with probability ϵ'. B responds to F's queries as follows:

Setup: The challenge (g, g^a, g^b, g^c) is given to B and B tries to solve the GBDH problem with the help of DBDH oracle. B picks $k_m \in \{0,\ldots,n_m\}$ and $k_u \in \{0,\ldots,n_u\}$ at random and sets integers $l_m = l_u = 2(q_{ps}+q_{ds}+q_{dv})$, where $l_m(n_m+1) < q$, $0 \leq k_m l_m < q$, $l_u(n_u+1) < q$ and $0 \leq k_u l_u < q$. B then chooses values $x' \overset{R}{\leftarrow} Z_{l_m}$, $z' \overset{R}{\leftarrow} Z_{l_u}$ and two random vectors $(x_1,\ldots,x_{n_m})$ and $(z_1,\ldots,z_{n_u})$, where $\{x_j \in Z_{l_m}, z_j \in Z_{l_u} \forall j\}$. Additionally, B randomly picks the values $y', w' \overset{R}{\leftarrow} Z_q$ and two random n-vectors $(y_1,\ldots,y_{n_m})$ and $(w_1,\ldots,w_{n_u})$, where $\{y_j, w_j \in Z_q, \forall j\}$. These values are kept internal to B. For a message m, $M \subseteq \{1,\ldots,n_m\}$ is composed of all j's satisfying $m[j] = 1$ and similarly, for a message u, $U \subseteq \{1,\ldots,n_u\}$ is composed of all j's which satisfy $u[j] = 1$. To simplify the analysis, we form four functions $J(m)$, $K(m)$, $L(u)$ and $F(u)$ identical to the ones in [18, 26].

$$\begin{aligned}&J(m) = x' + \textstyle\sum_{j\in M} x_j - k_m l_m \\ &K(m) = y' + \textstyle\sum_{j\in M} y_j \\ &L(u) = z' + \textstyle\sum_{j\in U} z_j - k_u l_u \\ &F(u) = w' + \textstyle\sum_{j\in U} w_j\end{aligned} \tag{38}$$

Then, B assigns the public keys of entities and the common parameters as follows:

1. B sets the public key of the signer as $Pk_s = g^a$, the public parameter $h = g_2 = g^b$ and the public keys of the designated verifier as $Pk_v = g^c$.
2. B sets $m' = h^{x'-k_m l_m} g_{y'}$, $m_j = h^{x_j} g^{y_j}$, and $\boldsymbol{m} = \{m_1, m_2, \ldots, m_{n_m}\}$. Similarly, $u' = h^{z'-k_u l_u} g_{w'}$, $u_j = h^{z_j} g^{w_j}$, and $\boldsymbol{u} = \{u_1, u_2, \ldots, u_{n_u}\}$.

Under this assignment, we have $m' \prod_{j\in M} m_j = h^{J(m)} g^{K(m)}$ and $u' \prod_{j\in U} u_j = h^{L(u)} g^{F(u)}$. Afterwards, B returns $(G, G_1, e, q, g, m', \boldsymbol{m}, u', \boldsymbol{u}, h)$ to the forger F.

Note that we have used different vectors for the messages m and u since they have different formats in our voting protocol.

Sign queries (PS): Suppose that F asks for a signature on the message $m \in \{0,1\}^{n_m}$. If $J(m) = 0 \pmod q$, B terminates the simulation and reports failure. Otherwise, B chooses $r_1 \overset{R}{\leftarrow} Z_q$ at random and computes the signature as follows:

$$\sigma = (\sigma_1, \sigma_2) = (Pk_s^{\frac{-K(m)}{J(m)}}(h^{J(m)} g^{K(m)})^{r_1}, h^{\frac{-1}{J(m)}} g^{r_1}) \tag{39}$$

Completeness:

$$\begin{aligned}\sigma_1 &= Pk_s^{\frac{-K(m)}{J(m)}}(h^{J(m)}g^{K(m)})^{r_1}\\ &= h^a(h^{J(m)}g^{K(m)})^{-\frac{a}{J(m)}}(h^{J(m)}g^{K(m)})^{r_1}\\ &= h^a(h^{J(m)}g^{K(m)})^{r_1-\frac{a}{J(m)}}\\ &= h^a(h^{J(m)}g^{K(m)})^{\tilde{r}}\end{aligned} \tag{40}$$

Considering $\tilde{r} = r_1 - \frac{a}{J(m)}$, we have:

$$\sigma = (\sigma_1, \sigma_2) = (h^a(h^{J(m)}g^{K(m)})^{\tilde{r}}, \tilde{r}) \tag{41}$$

which is a valid signature.

Designated verifier signature queries (DS): Suppose that F asks for a signature on the messages $m \in \{0,1\}^{n_m}$ and $u \in \{0,1\}^{n_u}$. Then, we have:

1. if $J(m) \neq 0 \pmod q$, then B issues a query to PS, chooses $r_2 \xleftarrow{R} Z_q$ at random and computes the designated verifier signature using the response of PS as follows:

$$\begin{aligned}&\delta_1 = e(Pk_s^{\frac{-K(m)}{J(m)}}(h^{J(m)}g^{K(m)})^{r_1}(h^{L(u)}g^{F(u)})^{r_2}, Pk_v)\\ &\delta_2 = h^{\frac{-1}{J(m)}}g^{r_1}\\ &\delta_3 = g^{r_2}\end{aligned} \tag{42}$$

2. if $J(m) = 0 \pmod q$, then B terminates the simulation and reports failure.

Note: if $J(m) = 0 \pmod q$ and $L(u) \neq 0 \pmod q$, we cannot issue PS queries on m, since, based on our voting protocol, m and u have different formats and cannot thus be used interchangeably.

Designated verifier verification queries (DV): Suppose that F asks for a designated verifier verification on the message/designated verifier signature $(m, u, \delta = (\delta_1, \delta_2, \delta_3))$. Then, B responds as follows:

1. If $J(m) \neq 0 \pmod q$ and $L(u) \neq 0 \pmod q$, B aborts.
2. If $J(m) \neq 0 \pmod q$ and $L(u) = 0 \pmod q$, B computes another designated verifier signature $\hat{\delta} = (\hat{\delta_1}, \hat{\delta_2}, \hat{\delta_3}) = \bar{\delta_3})$ on the message m using the technique used in DS and then sends $(g, h^{J(m)}g^{K(m)}, \frac{\bar{\delta_2}}{\hat{\delta_2}}, g^c, \frac{\bar{\delta_1}}{\hat{\delta_1}})$ to the DBDH oracle O_{DBDH}. O_{DBDH} outputs "1" if the received tuple is the BDH tuple, and "0" otherwise.

   Completeness: Let $\bar{\delta} = (\bar{\delta_1}, \bar{\delta_2}, \bar{\delta_3}))$ be a valid signature:

$$\begin{aligned}&\bar{\delta_1} = e(g^b, g^a)^c e(h^{J(m)}g^{K(m)}, Pk_v)^{r_1} e(\bar{\delta_3}, Pk_v)^{F(u)}\\ &\bar{\delta_2} = g^{r_1}, \bar{\delta_3} = g^{r_2}\end{aligned} \tag{43}$$

$\hat{\delta} = (\hat{\delta_1}, \hat{\delta_2}, \hat{\delta_3}))$ is a valid signature:

$$\begin{aligned}&\hat{\delta_1} = e(g^b, g^a)^c e(h^{J(m)}g^{K(m)}, Pk_v)^{r'_1} e(\hat{\delta_3}, Pk_v)^{F(u)}\\ &\hat{\delta_2} = g^{r'_1}, \hat{\delta_3} = g^{r_2}\end{aligned} \tag{44}$$

As a result:

$$\frac{\bar{\delta_2}}{\hat{\delta_2}} = g^{r_2-r'_2}, \frac{\bar{\delta_1}}{\hat{\delta_1}} = e(h^{L(u)}g^{F(u)}, Pk_v)^{r_2-r'_2} \tag{45}$$

$(g, h^{J(m)}g^{K(m)}, \frac{\bar{\delta_2}}{\hat{\delta_2}}, g^c, \frac{\bar{\delta_1}}{\hat{\delta_1}})$ is a valid BDH tuple.

3. If $J(m) = 0 \pmod q$ and $L(u) \neq 0 \pmod q$, B aborts.
4. If $J(m) = 0 \pmod q$ and $L(u) = 0 \pmod q$, B sends (44) to O_{DBDH}. O_{DBDH} outputs "1" if the received tuple is the BDH tuple, and "0" otherwise.

$$\left(g, g^a, g^b, g^c, \frac{\bar{\delta_1}}{e(Pk_v, \bar{\delta_2}^{K(m)})e(Pk_v, \bar{\delta_3}^{F(u)})}\right) \tag{46}$$

Probability Analysis: if B does not abort, F outputs the signature $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*)$ on messages u^*, m^* such that:

a) $\delta^* = (\delta_1^*, \delta_2^*, \delta_3^*)$ is a valid signature.
b) u^* has not been queried before.
c) (u^*, m^*, δ^*) is not among triples (u_i, m_i, δ_i) during queries.

Success Probability: Let $A : J(m^*) = 0 \pmod q$, $B : L(u^*) = 0 \pmod l$, $A_i : J(m_i) \neq 0 \pmod{l_m}$ and $B_i : L(u_i) = 0 \pmod{l_u}$; $Pr\left[succ\right] = Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i \wedge A \wedge \bigwedge_{i=1}^{q_{dv}} B_i \wedge B\right].\epsilon$ is:

$$\begin{aligned}&Pr\left[A\right] = Pr\left[J(m^*) = 0 \pmod q\right]\\ &= Pr\left[J(m^*) = 0 \pmod{l_m}\right]\ldots\\ &Pr\left[J(m^*) = 0 \pmod q | J(m^*) = 0 \pmod{l_m}\right]\\ &= \frac{1}{l_m}\frac{1}{n_m+1}\end{aligned} \tag{47}$$

$$\begin{aligned}&Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i | A\right] = 1 - Pr\left[\bigvee_{i=1}^{q_{ps}+q_{ds}} A_i\right]\\ &= 1 - Pr\left[\bigvee_{i=1}^{q_{ps}+q_{ds}} J(m_i) = 0 \pmod{l_m}\right]\\ &\geq 1 - \sum_{i=1}^{q_{ps}+q_{ds}} Pr\left[J(m_i) = 0 \pmod{l_m}\right]\\ &= 1 - \frac{q_{ps}+q_{ds}}{l_m)}\end{aligned} \tag{48}$$

$$\begin{aligned}&Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i \wedge A\right] = 1 - Pr\left[A\right]Pr\left[\bigwedge_{i=1}^{q_{ps}+q_{ds}} A_i | A\right]\\ &\geq (\frac{1}{l_m}\frac{1}{n_m+1})(1 - \frac{q_{ps}+q_{ds}}{l_m)})\end{aligned} \tag{49}$$

Similarly,

$$\begin{aligned}&Pr\left[\bigwedge_{i=1}^{q_{dv}} B_i \wedge B\right] = Pr\left[B\right]Pr\left[\bigwedge_{i=1}^{q_{dv}} B_i | B\right]\\ &= (\frac{1}{l_u}\frac{1}{n_u+1})(\frac{1}{l_u})^{q_{dv}}\end{aligned} \tag{50}$$

Thus,

$$\begin{aligned}&Pr\left[succ(forge)\right] \geq\\ &(\frac{1}{l_m}\frac{1}{n_m+1})(1 - \frac{q_{ps}+q_{ds}}{l_m)})(\frac{1}{l_u}\frac{1}{n_u+1})(\frac{1}{l_u})^{q_{dv}}\\ &= \frac{\epsilon}{2^{q_{dv}+3}(q_{ps}+q_{ds})^{q_{dv}+2}(n_m+1)(n_u+1)}\end{aligned} \tag{51}$$

where $l_u = l_m = 2(q_{ps} + q_{ds})$.

Hence, δ is unforgeable in the standard model assuming the GBDH problem is intractable.

□

Appendix C. Theorem 3

The proposal is non-transferable.

Proof. In order to prove the Theorem, we need to prove that the two following equations have the same distribution.

$$\delta = (\delta_1, \delta_2, \delta_3) = \begin{cases} \delta_1 = e(g_2^{x_s}(m' \prod_{j \in IDENT_j} m_j)^{r_m} \ldots \\ \times (u' \prod_{j \in B^t} u_j)^{r_b}, Pk_T) \\ \delta_2 = g^{r_m}, r_m \in Z_q^* \\ \delta_3 = g^{r_b}, r_b \in Z_q^* \end{cases} \quad (52)$$

$$\delta = (\delta'_1, \delta'_2, \delta'_3) = \begin{cases} \delta'_1 = e(g_2, Pk_s)^{x_T} e(m' \prod_{j \in IDENT_j} m_j, R')^{x_T} \ldots \\ e(u' \prod_{j \in B^t} u_j, R'_b) \\ \delta'_2 = g^{r'_m}, r'_m \in Z_q^* \\ \delta'_3 = g^{r'_b}, r'_b \in Z_q^* \end{cases} \quad (53)$$

Suppose that $\bar{\delta} = (\bar{\delta}_1, \bar{\delta}_2, \bar{\delta}_3)$ is a valid signature which is randomly chosen among the entire valid signer's signatures proposed to the verifier. Thus, the probability of the distributions is as (54) and (55).

$$Pr_\delta = Pr\left[(\delta_1, \delta_2, \delta_3) = (\bar{\delta}_1, \bar{\delta}_2, \bar{\delta}_3)\right] = Pr_{r_m, r_b \in_R Z_q^*}\begin{bmatrix} \delta_1 = \bar{\delta}_1 \\ \delta_2 = \bar{\delta}_2 \\ \delta_3 = \bar{\delta}_3 \end{bmatrix} = \frac{1}{(q-1)^2} \quad (54)$$

$$Pr_\delta = Pr\left[(\delta'_1, \delta'_2, \delta'_3) = (\bar{\delta}_1, \bar{\delta}_2, \bar{\delta}_3)\right] = Pr_{r'_m, r'_b \in_R Z_q^*}\begin{bmatrix} \delta'_1 = \bar{\delta}_1 \\ \delta'_2 = \bar{\delta}_2 \\ \delta'_3 = \bar{\delta}_3 \end{bmatrix} = \frac{1}{(q-1)^2} \quad (55)$$

Since both distributions have the same probability, our proposal could be concluded to be non-transferable. □

Sepideh Avizheh received her B.S. degree in Electrical Engineering from Dr. Shariaty College of Technology, Tehran, Iran, in 2009, holding the first rank among all graduates. She also received her M.S. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran, in 2012. Currently she is an adjunct instructor in Electronic Engineering program at Dr. Shariaty College of Technology, Tehran, Iran. Her research interests include Digital Signatures, Design and Cryptanalysis of Cryptographic Protocols and Network Security.

Maryam Rajabzadeh Asaar received her B.S. degree in Electrical Engineering from Shahid Bahonar University of Kerman, Kerman, Iran, in 2004 and her M.S. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 2007. She is now a Ph.D. candidate in Electrical Engineering Department of Sharif University of Technology. Her research interests include Provable Security, Digital Signatures, Design and Analysis of Cryptographic Protocols and Network Security.

Mahmoud Salmasizadeh received his B.S. and M.S. degrees in Electrical Engineering from Sharif University of Technology in Iran, in 1972 and 1989, respectively. He also received the Ph.D. degree in Information Technology from Queensland University of Technology in Australia, in 1997. Currently he is an associate professor in Electronics Research Institute and adjunct associate professor in Electrical Engineering Department at Sharif University of Technology, Tehran, Iran. His research interests include Information Theoretic Secrecy, Design and Cryptanalysis of Cryptographic Algorithms and Protocols and E-commerce Security. He is a founding member of Iranian Society of Cryptology.