An efficient lattice-based threshold signature scheme using multi-stage secret sharing

Hossein Pilaram\textsuperscript{1} | Taraneh Eghlidos\textsuperscript{2} \* | Rahim Tolutee\textsuperscript{1}

\textsuperscript{1}Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
\textsuperscript{2}Electronics Research Institute, Sharif University of Technology, Tehran, Iran

Correspondence
Taraneh Eghlidos, Electronics Research Institute, Sharif University of Technology, Tehran, Iran. Email: teghlidos@sharif.edu

Funding information
Iran National Science Foundation (INSF), Grant/ Award Number: 94017742

Abstract
Secret sharing is a cryptographic technique used in many different applications such as cloud computing, multi-party computation and electronic voting. Security concerns in these applications are data privacy, availability, integrity and verifiability, where secret sharing provides proper solutions. The authors address some important features like verifiability and being multi-stage to make it usable in various field of application. Here, the authors propose an anonymous threshold signature scheme based on the trapdoor function introduced by Micciancio and Peikert by sharing the private key using a lattice-based threshold multi-stage secret sharing (TMSSS) scheme. Then, the authors improve the previously proposed TMSSS scheme, in such a way that less public values are required to publish on the bulletin board which makes the scheme more efficient while preserving the security of the scheme based on the lattice hard problems. The proposed scheme inherits the desired features such as anonymity and verifiability from the improved TMSSS scheme. Furthermore, both schemes use simple matrix operations, which additionally increase the efficiency.

1 | INTRODUCTION

Secret sharing schemes are cryptographic primitives to protect a secret by sharing it to some values and assign them to different users. These schemes are used in many different applications such as cloud computing \cite{1}, key management in sensor networks \cite{2}, electronic voting \cite{3} and electronic cash \cite{4}. Secret sharing provides solutions for data privacy, availability, integrity and verifiability, which makes it suitable for these applications. In a \((t, n)\) threshold secret sharing scheme (TSSS), a trusted third party named dealer, splits a secret among \(n\) parties, called participants, in such a way that at least \(t\) participants can recover the secret using their assigned values, called shares. The first TSSS was introduced by Blakley and Shamir, independently, in 1979 \cite{5,6}. According to the applications, some features have been added to the TSSS scheme such as being fault-tolerant, \cite{7,8}, verifiability of the shares \cite{9,10} and dynamic variation of the threshold and/or the group size \cite{11,12}.

Multi-secret sharing schemes are generalizations of the secret sharing schemes to share more than one secret by assigning one share to each participant \cite{13}. If the size of each share is equal to the size of each secret, these schemes provide only the computational security \cite{14}. Pang et al. \cite{15} proposed a multi-secret sharing scheme, in which when the participants pull their shares together, all of the secrets are disclosed simultaneously. He and Dawson \cite{16} proposed a \((t, n)\) threshold multi-stage secret sharing (TMSSS) scheme in 1994, in which the secrets are revealed stage by stage, that is in recovering a couple of secrets, the recovered secrets gives no information about the unreconstructed secrets. Geng et al. \cite{17} showed that their scheme is one-time-use and not immune to collusion attacks. Using a one-way hash function, they proposed a TMSSS scheme, in which the shares can be reused by some technical measures, when we wish to share new secrets. In multi-use or multi-stage secret sharing schemes, the following two general security requirements are needed: (1) While recovering the secret(s), the participants should not reveal the original shares. (2) The secrecy of the other unreconstructed secrets should be computationally independent from the reconstructed secrets. In a TMSSS scheme, the participants should provide pseudo-secret shares instead of the
original ones to the combiner, in which the pseudo-secret shares depend on the main shares and the desired secret which is to be recovered, while the shares cannot be obtained from the pseudo-secret shares. TMSSS schemes use two-variable one-way functions \[18,19\], one-way (hash) functions \[14,20,21\] and discrete logarithm problem \[22\] which is threatened by quantum computers.

Shamir's secret sharing scheme \[5\] is secure from information theoretic point of view, but in real applications, for distributing the shares securely among the participants, a public key cryptosystem is normally used. The security of conventional public key cryptosystems has been threatened, since Shor proposed a quantum algorithm for solving ‘integer factorization’ and ‘discrete logarithm’ problems in polynomial time \[23\]. Therefore, Shamir's secret sharing scheme cannot be practically used in a secure protocol. Ever since Ajtai’s introduction of lattice-based quantum resistant one-way functions \[24\], lattice-based cryptosystems play a significant role in the world of post-quantum cryptography. Ajtai proposed a family of one-way functions whose security is based on the worst-case hardness of the lattice problems such as shortest vector problem (SVP) known to be NP-hard \[25\]. Lattice-based cryptosystems are provably secure based on hardness of lattice problems in worst case. Furthermore, they use linear computations on relatively small integers. Since the secret sharing algorithm should be consistent with the lattice-based public key infrastructure, therefore, Shamir's secret sharing scheme should be replaced by a lattice-based secret sharing scheme.

In 2005, Regev proposed a public key cryptosystem based on the LWE problem \[26\]. The LWE problem was quantumly reduced by Regev to classical lattice problems, such as the shortest independent vector problem, in the worst case. It follows that LWE-based schemes are provably secure assuming the worst-case hardness of classical lattice problems. The LWE problem has been the source of great progress in the lattice-based cryptography.

In recent years, there have been some works on lattice-based threshold secret sharing. Georgescu \[27\] proposed an \((n, n)\) TSSS based on the hardness of the LWE problem. Bansarkhani et al. \[28\] proposed a verifiable \((n, n)\) TSSS using linear hash functions based on lattice in which each participant is able to verify the recovered secret and her/his share. The security of the scheme depends on the hardness of \(n^2\)-approximate SVP. Amini et al. \[29\] and Asaad et al. \[30\] proposed lattice-based \((t, n)\) TSSSs with asymptotic security, in 2014. Later, we have proposed an efficient lattice-based verifiable TMSSS scheme using the Ajtai's one-way function in 2015 \[31\].

In threshold cryptographic schemes, which are widely used in multi-party computation, a certain number of participants can decrypt or sign messages. Bendlin et al. proposed the first lattice based threshold encryption scheme \[32\]. They proposed a threshold decryption algorithm for the Regev’s LWE public key encryption scheme supporting one-bit messages. Frederiks extended this scheme for multi-bit messages \[33\]. Singh et al. \[34\] proposed an efficient lattice-based threshold public key encryption scheme based on Lindner's work \[35\]. In these schemes, Shamir's TSSS have been used to share the private key as an array. Hence, each entry of the private key is shared independently in these schemes. However, this method seems to be inefficient. In 2016, we have proposed a threshold decryption algorithm for the Lindner’s LWE-based public key encryption \[36\]. The authors used the lattice-based TMSSS scheme proposed in Ref. \[31\] to share the private key among the participants. Threshold signature is another example of threshold cryptography. This scheme must provide anonymity and succinctness. Anonymity means that the same signature is produced, no matter which subset of \(t\) participants is used. Succinctness means that the signature size can depend on the security parameter, but must be independent of \(n\) and \(t\) \[37\]. In 2010, Cayrel et al. \[38\] proposed a lattice-based threshold ring signature scheme using an identification scheme. In 2013, Battieb et al. \[39\] improved this scheme by generalizing the same identification scheme to obtain a more efficient threshold ring signature scheme with less signature size. Both schemes are interactive. A common goal for threshold systems is to minimize the amount of interactions in the system, and in particular, construct one-round schemes. In 2013, Bendlin et al. \[40\] proposed a sharing algorithm for a lattice-based trapdoor function. In 2017, Boneh et al. proposed a general framework for universally thresholdizing many (non-threshold) cryptographic schemes using a lattice-based fully homomorphic encryption (FHE) scheme \[37\]. Their reliance on FHE causes the resulting schemes to be slow in practice.

1.1 Contributions

We improve the previously proposed lattice-based TMSSS scheme, in such a way that less public values are published on the bulletin board by preserving the security of the scheme. Then, we propose a threshold signature scheme based on the trapdoor function introduced by Micciancio and Peikert \[41\]. Here, we share the private key using the proposed lattice-based TMSSS scheme. For threshold signing, each participant signs the message using their share and send the result to the combiner as partial signature. Because of additive homomorphic property of the TMSSS scheme and linear property of the signature scheme, the combiner can sign the message using the partial signatures. The proposed signature scheme inherits the desired features like anonymity and verifiability from the improved TMSSS scheme. Furthermore, since we use TMSSS scheme to share the private key, the size of the shares in the proposed signature scheme is much smaller than that of the only counterpart lattice-based threshold signature scheme \[40\], which makes it suitable for memory-constrained applications. Our proposed lattice based schemes are resistant against the quantum computers. Moreover, both schemes are efficient because simple matrix operations are used in the secret sharing and threshold signature protocols.

This paper is organized as follows: Section 2 provides a brief review of lattices, our previous multi-stage secret sharing scheme and the lattice-based trapdoor function proposed by
Micciancio and Peikert. Section 3 is dedicated to the improved lattice-based TMSSS scheme. The proposed threshold signature scheme is introduced in Section 4. In Sections 5 and 6, we discuss about the security and efficiency of the proposed schemes, respectively. Finally, we end up the paper with a brief conclusion in Section 7.

2 | PRELIMINARIES

This section deals with some basic definitions we need in the rest of the paper.

2.1 | Notations

In the following, we assume column vectors. Vectors and matrices are denoted by lowercase and uppercase letters, respectively. The matrices $I_n$ and $0_{m \times n}$ denote the $n \times n$ identity matrix and the zero matrix of size $m \times n$, respectively. The transpose of a rectangular matrix is denoted by $(\cdot)^T$. Furthermore, $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{Z}_q$ denote the sets of reals, integers and the finite field modulo $q$, respectively. We denote $\mathbb{S}^n$ and $\mathbb{S}^{m \times n}$ as the set of vectors of size $n$ and the set of $m \times n$ matrices, whose entries are chosen from a set of numbers $\mathbb{S}$. The operators $\lfloor \cdot \rfloor$ and $\lVert \cdot \rVert$ denote rounding operation to the nearest integer and an arbitrary norm, respectively. The $\ell_p$ norms are defined for any $p \geq 1$ and a vector $x \in \mathbb{R}^n$ as $\lVert x \rVert_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$. For classifying the growth of functions we use the standard big-$O$ and little-o notations. The function $\text{negl}(n)$ denotes a negligible function which is defined as $f(n) = o(n^{-c})$ for every fixed constant $c$.

2.2 | Lattices

We use lattices as a tool for constructing a TMSSS scheme and a threshold signature scheme. In the following, we give a formal definition of lattices.

Definition 1 [25] Let $b_1, b_2, \ldots, b_n$ be $n$ linearly independent vectors in vector space $\mathbb{R}^m$. We define $L(b_1, \ldots, b_n)$ as the set of all integer linear combinations of $b_1, b_2, \ldots, b_n$ as follows:

$$L = L(b_1, \ldots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \right\}$$

The set of vectors $\{b_1, \ldots, b_n\}$ is called a basis for the lattice $L$, and $n$ is called the rank of the lattice.

Definition 2 [42] A $q$-ary lattice is a lattice $\Lambda$ satisfying $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$ for some (possibly prime) integer $q$.

The well-known instance of $q$-ary lattices is $\Lambda_q^\perp(A) = \{ x \in \mathbb{Z}^n \mid Ax = 0 \pmod{q} \}$, where $A$ is an integer matrix from $\mathbb{Z}^{m \times n}$ and $q$ is the modulus. $\Lambda_q^\perp(A)$ is a lattice of dimension $m$.

Lattice-based cryptographic schemes are based on the hardness of lattice problems such as the well-known SVP and CVP, which are secure against quantum computers.

Definition 3 SVP problem [25] Let a lattice basis $B \in \mathbb{Z}^{m \times n}$ be given. The solution is a nonzero lattice vector $Bx$ (with $x \in \mathbb{Z}^n \setminus \{0\}$) such that $\lVert Bx \rVert \leq \lVert By \rVert$ for any other $y \in \mathbb{Z}^n \setminus \{0\}$, with respect to any norm.

Definition 4 CVP problem [25] Let a lattice basis $B \in \mathbb{Z}^{m \times n}$ and a target vector $t \in \mathbb{Z}^m$ be given. The solution is a lattice vector $Bx$ closest to the target $t$, that is finding an integer vector $x \in \mathbb{Z}^n$ such that $\lVert Bx - t \rVert \leq \lVert By - t \rVert$ for any other $y \in \mathbb{Z}^n$, with respect to any norm.

Definition 5 Approximate SVP problem [25] Let a lattice basis $B \in \mathbb{Z}^{m \times n}$ be given. The solution is a nonzero lattice vector $Bx$ (with $x \in \mathbb{Z}^n \setminus \{0\}$) such that $\lVert Bx \rVert \leq \gamma \lVert By \rVert$ for any other $y \in \mathbb{Z}^n \setminus \{0\}$, with respect to any norm.

In 1996, Ajtai introduced a lattice-based one-way function whose security is based on the worst-case hardness of $n^\epsilon$-approximate SVP, which is resistant against quantum computers in polynomial time. This function is described by $f_A(x) = Ax \pmod{q}$, where $A \in \mathbb{Z}_q^{m \times n}$ and $x \in \{0,1\}^\star$. The inversion of this function leads to the following problem:

- Parameters: $n, m, q \in \mathbb{N}$ s.t. $m > n \log q$, and $q = O(n^\epsilon)$ for some constant $c$.
- Input: A uniformly random matrix $A \in \mathbb{Z}_q^{m \times n}$ and a vector $y = Ax$ for some random vector $x \in \{0,1\}^m$.
- Output: A vector $x \in \{0,1\}^m$ s.t. $Ax = y \pmod{q}$.

This problem is a special case of the Inhomogeneous Small Integer Solution problem, in which the condition $x \in \{0,1\}^m$ is replaced by $\lVert x \rVert \leq \beta$ for a real parameter $\beta$ [43].

Definition 6 LWE problem [42] Let $n, m$ and $q$ be integers and $\chi$ be a probability distribution on $\mathbb{Z}_q$, typically taken to be a rounded normal distribution. The input is a pair $(A, v)$, where $A \in \mathbb{Z}_q^{m \times n}$ is uniformly distributed, and $v = As + e$ for a uniformly chosen $s \in \mathbb{Z}_q^n$ and a vector $e \in \mathbb{Z}_q^m$ chosen according to $\chi$. The goal is finding $s$ with some non-negligible probability.

Regev showed that the approximate SVP can be quantumly reduced to the LWE problem and hence the hardness of LWE problem is based on the worst-case quantum hardness of approximate SVP [26].

Definition 7 Gaussians on Lattices [43] For any $s > 0$, define the Gaussian function on $\mathbb{R}^n$ centred at $c$ with parameter $s$:
\[ \forall x \in \mathbb{R}^n, \rho_{x,c}(x) = \exp(-\pi\|x - c\|^2/s^2). \] (2)

For any \( c \in \mathbb{R}^n \), real \( s > 0 \), and \( n \)-dimensional lattice \( \Lambda \), define the discrete Gaussian distribution over \( \Lambda \) as:

\[ \forall x \in \Lambda, D_{\Lambda,s,c}(x) = \rho_{x,c}(x)/\rho_{x,c}(\Lambda) \] (3)

### 2.3 Secret sharing

In a secret sharing scheme, a secret is shared among a set of parties, called participants, denoted by \( \mathcal{P} \). Each participant is assigned a confidential value, named share, by a trusted third party, called dealer. Only some subsets of participants, the authorized subsets, can reconstruct the secret by executing a predefined algorithm. An access structure is defined as the set of all authorized subsets. In this paper, we deal with a threshold access structure, which, for a given parameter \( t \), authorized subsets consist of at least \( t \) participants.

In a secret sharing scheme we consider two phases:

- **Share distribution:** The shares are computed using a pre-specified algorithm by the dealer who sends the shares to the participants through a secure channel.
- **Secret recovery:** A server, called combiner, receives the shares from the authorized participants and recover the secret by running the algorithm.

The above-defined protocol is simple and cannot be used in real world applications directly. Depending on different features, TSSSs can be extended as follows:

**Verifiable TSSS**

In this scheme, the participants can verify the validity of the shares received from the dealer and the recovered secrets by the combiner [21].

**Threshold multi-secret sharing scheme**

In this scheme, we wish to share multiple secrets among the participants. For sake of efficiency, we desire to distribute only one share among the participants for reconstructing all the secrets [13].

**Threshold multi-stage secret sharing scheme**

In this multi-secret sharing scheme, the secrets can be recovered in multiple stages and the unrecovered secrets cannot be computed from the recovered secret(s) in polynomial time, despite information leakage between the recovered and unrecovered secrets [20,44].

### 2.4 Lattice-based TMSSS scheme

In this section, we introduce the lattice-based \((t,n)\) TMSSS scheme [31], where for recovering each of the secrets at least \( t \) participants should take part. In this scheme, multiple secrets can be recovered separately. It is worth noting that the recovered secret(s) do not leak information to the unrecovered secrets in polynomial time. In this scheme, there are \( m \) secrets \( s_i \in \mathbb{Z}_q^t \), \( i = 1, \ldots, m \), where \( t \) is the threshold and \( q \) is a prime number. A random vector \( v \in \mathbb{Z}_q^t \) is selected by the dealer and published. Then, a private lattice basis \( B_i \) is found for each secret \( s_i \), such that

\[ s_i = B_i v, i = 1, \ldots, m \] (4)

where \( B_i \in \mathbb{Z}_q^{t \times n} \) forms a basis for a lattice of dimension \( t \).

Then, \( n \) public vectors \( h_j \in \mathbb{Z}_q^{t \times 1} \), \( j = 1, \ldots, n \) are chosen by the dealer, in such a way that every \( t \) of these vectors are linearly independent. Then, the dealer finds some public matrices \( E_i \in \mathbb{Z}_q^{n \times r}, i = 1, \ldots, m \) and some private shares \( c_j \in \{0,1\}^r \), \( j = 1, \ldots, n \), in such a way that the relation \( E_i c_j = B_i h_j \) holds, for \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \), where \( r \geq \text{max}\{t \log t, n\} \).

Therefore, \( n \) shares \( c_j \) from \( \{0,1\}^r \) are chosen randomly by the dealer and then the dealer solves a system of linear equations to obtain the matrices \( E_i, i = 1, \ldots, m \).

For share verification by the participants, a random matrix \( F \in \mathbb{Z}_q^{n \times r} \) is chosen by the dealer and published together with the hash values of the shares as the vectors of \( h_j = F c_j, j = 1, \ldots, n \). Moreover, the dealer uses a public hash function \( H(\cdot) \) and publishes \( H(s_i), i = 1, \ldots, m \) on the bulletin board, by means of which the participants can verify the recovered secrets.

### 3 THE IMPROVED LATTICE-BASED TMSSS SCHEME

In this section, we improve the lattice-based TMSSS scheme, introduced in Section 2.4, in such a way that less public values are published. Here, for saving the memory needed for storing the public matrices \( E_i, i = 1, \ldots, m \), the dealer chooses them as \( E_i = E P_i, i = 1, \ldots, m \), where \( E \) is chosen uniformly at random from \( \mathbb{Z}_q^{n \times r} \) and \( P \) is a permutation matrix of size \( r \). To hold the equations \( E c_j = B_i h_j, i = 1, \ldots, m, j = 1, \ldots, n \), the dealer chooses the share vectors \( c_j \in \{0,1\}^r, j = 1, \ldots, n \) uniformly at random and the public vectors \( h_j \in \mathbb{Z}_q^{t \times 1}, j = 1, \ldots, n \), such that any \( t \) out of \( n \) vectors are linearly independent, as before. Then the matrices \( B_i \) are obtained by solving the linear equations \( E c_j = B_i h_j, i = 1, \ldots, m; j = 1, \ldots, n \). The dealer chooses the vectors \( v_i \in \mathbb{Z}_q^t, i = 1, \ldots, m \) in such a way that the equations \( s_i = B_i v_i, i = 1, \ldots, m \) hold. Finally, he sends the share vectors \( c_j, j = 1, \ldots, n \) securely to the participants and the matrices \( E \) and \( P \) and the vectors \( h_j, j = 1, \ldots, n \) and \( v_i, i = 1, \ldots, m \) are published on the bulletin board. Note that in the improved TMSSS scheme, the
size of the public information is reduced. The improved scheme consists of the following phases:

**Distribution phase**
- Each participant \( P_j \) receives the share vector \( c_j \) securely and the dealer publishes the matrices \( E_i, i = 1, \ldots, m \) and the vectors \( \lambda_j, j = 1, \ldots, n \) on the bulletin board.
- The participant \( P_j \) checks if \( FC_j \neq h_j \).

**Combination phase**

Here, the secrets are recovered separately. Suppose that a \( t \)-subset \( \{ j_1, \ldots, j_t \} \subset \{ 1, \ldots, n \} \) of the participants wish to recover the secret \( s_i, i \in \{ 1, \ldots, m \} \). To do this, the participant \( P_j \), \( l = 1, \ldots, t \) computes the vector \( d_{ji} = E_i c_{ji}, l = 1, \ldots, t \) as his pseudo-secret share and sends it securely to the combiner. The combiner constructs the matrix \( D_l = \left[ d_{j_1}^l \ldots d_{j_t}^l \right] \) and then recovers the secret \( s_i \) by computing \( s_i = D_l \lambda^T \psi_i \), where \( \Lambda = \left[ \lambda_{j_1}^T \ldots \lambda_{j_t}^T \right] \). The participants are able to verify the reconstructed secret by computing its hash value \( H(s_i) \) and comparing it with that published on the bulletin board.

### 4 | THRESHOLD SIGNATURE SCHEME

In this paper, we propose a threshold signature scheme.

#### 4.1 | Models

**Signature scheme**

A signature scheme \( S \) is a triple \((KGen, Sign, Verify)\) of efficient algorithms. \( KGen \) is the key generation algorithm, which on input of the security parameter \( 1^t \), outputs a pair \((pk, sk)\), such that \( pk \) is the public key and \( sk \) is the secret key of the signature scheme. \( Sign \) is the signing algorithm: on input a message \( m \) and the secret key \( sk \), outputs the signature \( s \) on the message \( m \). \( Verify \) is the verification algorithm: on input a message \( m \), the public key \( pk \) and \( s \), checks whether \( s \) is a valid signature on \( m \) or not.

**Threshold signature schemes**

Let \( S = (KGen, Sign, Verify) \) be a signature scheme. A \((t, n)\) threshold signature scheme \( TS \) for \( S \) is a pair of protocols \((TKGen, TSign)\) for the set of participants \( P_1, \ldots, P_n \). \( TKGen \) is a key generation protocol which outputs the public key \( pk \) and private shares \( c_j, j = 1, \ldots, n \) for the participant \( P_j \). \( TSign \) is a distributed algorithm which outputs a signature \( s \) on inputs, a message \( m \) and the \( t \) private shares.

**Definition 8** We say that a \((t, n)\) threshold signature scheme \( TS = (TKGen, TSign) \) is unforgeable, if no adversary who corrupts up to \( t-1 \) participants can produce the signature on any new message \( m \), given the threshold signatures on input messages \( m_1, \ldots, m_k \).

### 4.2 | The proposed threshold signature scheme

We propose a threshold signature scheme in which the signed message is computed based on the trapdoor function, proposed by Micciancio and Peikert [41], in a distributed manner. According to Section 4.1 our scheme consists of the following algorithms:

**TKGen**

The parameters are the integers \( l, q, \overline{m} \) and the so-called gadget vector \( g = (1, 2, 4, \ldots, 2^{k-1})^T \in \mathbb{Z}_q^k \), where \( k = \lceil \log q \rceil \). To generate a random matrix \( A \) with a trapdoor, they build the matrix \( A \) using two sub-matrices as:

\[
A = [AG - \overline{AR}] = [AG] \begin{bmatrix} I_m & -R \\ 0 & I_{lk} \end{bmatrix} \in \mathbb{Z}_q^{l \times m},
\]

where \( \overline{A} \) is chosen uniformly at random from \( \mathbb{Z}_q^{l \times m} \), \( m = m \) and \( R \in \mathbb{Z}_q^{m \times lk} \) is the trapdoor. The gadget matrix \( G \) is defined as follows:

\[
G = I_n \otimes g^T
\]

\[
= \begin{bmatrix} g^T \\ \vdots \\ g^T \end{bmatrix} \in \mathbb{Z}_q^{l \times lk}
\]

(5)
We share the vectors \( r_i, i = 1, \ldots, m \) using the improved TMSSS scheme with the threshold \( t = \lfloor k \rfloor \), where \( r_i^T \) is the \( i \)th row of the matrix \( R \).

\[
\begin{align*}
    r_i &= B_i \psi_i, i = 1, \ldots, m, \\
    E_i c_j &= B_i \lambda_j, i = 1, \ldots, m, j = 1, \ldots, n, 
\end{align*}
\]

where \( n \) is the number of the participants.

The matrices \( E_i \in \mathbb{Z}_q^{x^T r}, i = 1, \ldots, m, r \geq \max \{ t \log t, n \} \), the vectors \( \lambda_j \in \mathbb{Z}_q^r, j = 1, \ldots, n \) and \( \psi_i \in \mathbb{Z}_q^r, i = 1, \ldots, m \) are considered as public parameter \( pk \) and published on the bulletin board. The vectors \( c_j \in \{ 0,1 \}^r, j = 1, \ldots, n \) are the private shares and the dealer sends them securely to the participants.

**TSign**

Each participant from the set \( \{ j_1, \ldots, j_t \} \) solves the equation \( Gz = H(m) \) for \( z \) which is equivalent to \( z \leftarrow f^\frac{1}{G}(H(m)) \), where \( H(z) \) is a public hash function. Note that there are efficient algorithms to sample from a discrete Gaussian distribution over \( \Lambda_q^2[H(m)](G) = \{ z \in \mathbb{Z}_q^{t_k} : Gz = H(m) \mod q \} \), for any syndrome \( H(m) \in \mathbb{Z}_q^r \). This operation can be performed by a server computer and then the result can be sent to the participants. The participant \( j_k, k = 1, \ldots, t \) computes the partial signature \( d_k = \begin{bmatrix} z^T E_i c_{j_k} \\ z^T E_m c_{j_k} \end{bmatrix} \) and sends it to the combiner.

The combiner computes \( w \) as the main diagonal of matrix \( D \Lambda^{-1} V \) using the received partial signatures, where \( D = [d_1 \ldots d_t], \Lambda = [\lambda_1 \ldots \lambda_t] \) and \( V = [\psi_1 \ldots \psi_m] \).

Then, he obtains the signature \( y = \begin{bmatrix} w \\ z \end{bmatrix} \).

**Verify:** This algorithm checks if the vector \( y \) is short enough and \( Ay = H(m) \).

**Correctness** In the following we show that the vector \( y \) computed by the combiner lies in the equation \( Ay = H(m) \).

\[
\begin{align*}
    D \Lambda^{-1} V &= [d_1 \ldots d_t] \Lambda^{-1} V \\
    &= \begin{bmatrix} z^T E_i c_{j_1} & \cdots & z^T E_i c_{j_t} \\ z^T E_m c_{j_1} & \cdots & z^T E_m c_{j_t} \end{bmatrix} \Lambda^{-1} V \\
    &= \begin{bmatrix} z^T B_1 \lambda_{i_1} & \cdots & z^T B_1 \lambda_{i_t} \\ \vdots & \vdots & \vdots \\ z^T B_m \lambda_{j_1} & \cdots & z^T B_m \lambda_{j_t} \end{bmatrix} \Lambda^{-1} V \\
    &= \begin{bmatrix} z^T B_1 \Lambda \\ \vdots \\ z^T B_m \Lambda \end{bmatrix} \Lambda^{-1} [\psi_1 \ldots \psi_m] \\
    \Rightarrow w &= \begin{bmatrix} z^T r_1 \\ z^T r_m \end{bmatrix} = Rz \\
    A y &= \begin{bmatrix} w \\ z \end{bmatrix} = A Rz = A \begin{bmatrix} R \\ I \end{bmatrix} z = Gz = H(m) 
\end{align*}
\]

The Equations (7) and (8) imply the correctness of our threshold signature scheme which outputs the correct signature through the distributed method.

As mentioned in Ref. [41], since \( y = \begin{bmatrix} R \\ I \end{bmatrix} z \), the distribution of \( y \) depends on the private key \( R \), leading to an attack similarly to the one given in Ref. [46]. To sample from a spherical Gaussian over \( \Lambda_q^2[H(m)](A) \), the authors have used the convolution technique from Ref. [47] to dissolve this problem. Specifically, they have chosen a Gaussian Permutation \( p \in \mathbb{Z}_q^m \) with the covariance matrix \( s^2 - \begin{bmatrix} R \\ I \end{bmatrix} \Sigma G \begin{bmatrix} R^T \\ I \end{bmatrix} \). Then they sample \( y = \begin{bmatrix} R \\ I \end{bmatrix} z \) for the syndrome \( w = H(m) - Ap \) similar to the above-mentioned signing algorithm and output \( x = y + p \). Here, we discuss about the perturbation vector \( p \) in our threshold signature scheme. Assume that some \( p \)'s with the covariance matrix \( s^2 - \begin{bmatrix} R \\ I \end{bmatrix} \Sigma G \begin{bmatrix} R^T \\ I \end{bmatrix} \) are shared among the participants. The authorized set of participants \( j_1, \ldots, j_t \) sends \( Ap_1, \ldots, Ap_t \) to the combiner. He combines them to obtain \( Ap \) and send the result to authorized set of participants. The participant \( j_k \) computes the \( w = H(m) - Ap \) and do the partial signature using the syndrome \( w \) and add the partial perturbation \( p_k \) to it and sends the result to the combiner. He combines the partial signatures to obtain the signature \( x = y + p \).

### 5 SECURITY

In secret sharing schemes, any \( t-1 \) subset or less of participants should not be able to recover the secret. In the TMSSS scheme proposed in Section 3, no \( t-1 \) participants are able to obtain the basis \( B_i \) since they face with a system of \( (t-1) \times t \) linear equations and \( t^2 \) unknowns: \( E_i c_{j_k} = B_i \lambda_{j_k}, k = 1, \ldots, t-1 \), having \( t \) degrees of freedom.

In Theorem 1, we prove that the pseudo-secret shares corresponding to different secrets are not computable in polynomial time from each other.

**Theorem 1** Let \( E \) be a randomly chosen matrix from \( \mathbb{Z}_{q^x}^{r^x}, r \geq \max \{ t \log t, n \} \), \( P \) be a random permutation matrix and \( x \in \{ 0,1 \}^r \) be a random vector. Then, knowledge of \( EPx \) does not help to compute \( EPx \) in polynomial time, where \( 1 \leq k \neq i \leq m \).

**Proof:**

We argue by contradiction. Let we have an algorithm \( \mathcal{N} \) which returns \( EPx \) on inputs \( E, P \) and \( EPx \), with a non-negligible probability in polynomial time, where \( E \) is chosen randomly from \( \mathbb{Z}_{q^x}^{r^x}, r \geq \max \{ t \log t, n \} \), \( P \) is a random permutation matrix and \( x \) is a random vector from \( \{ 0,1 \}^r \). By
applying \( \mathcal{A} \), we design another algorithm \( \mathcal{B} \) which returns \( x \) on inputs \( B \) and \( Bx \), meaning inversion of Ajtai’s function, where \( B \) is chosen at random from \( \mathbb{Z}_q^{r \times s} \) and \( x \) is randomly chosen from \( \{0,1\}^s \). We observe that the algorithm \( \mathcal{A} \) can output \( E'x \) on inputs \( E' \), \( Q \) and \( E'x \) by defining \( E' \triangleq EP \) and \( Q \triangleq P^{-i} \). Like \( E \), \( E' \) is a uniformly random matrix and \( Q \) is also a random permutation matrix similar to \( P \). The algorithm \( \mathcal{B} \) chooses two permutation matrices \( P_1 \) and \( P_2 \) in such a way that they only differ in two columns. For example, we assume that the permutation matrices \( P_1 \) and \( P_2 \) are different in two columns \( i \) and \( j \), that is, \( P_{1i} = e_i, P_{2j} = e_j \) and \( P_{1j} = e_j, P_{2i} = e_i \), where \( e_i \) is the \( i \)th standard unit vector. The algorithm \( \mathcal{A} \) outputs \( c_1 = BP_1x \) and \( c_2 = BP_2x \) on inputs \( B, Bx \) and the matrices \( P_1 \) and \( P_2 \). The algorithm \( \mathcal{B} \) computes \( d = c_1 - c_2 = B(P_1 - P_2)x \). It can easily be seen that \( d = x_i(b_1 - b_2) + x_j(b_2 - b_1) \), where \( b_i \) is the \( i \)th column of \( B \) and \( x_i \) is the \( i \)th entry of the vector \( x \). Therefore, the algorithm \( \mathcal{B} \) can obtain \( x_i \) and \( x_j \) by comparing \( d \) to binary linear combinations of the vectors \( b_1 - b_2 \) and \( b_2 - b_1 \). Other entries of the vector \( x \) are obtained with similar approach using different permutation matrices, that contradicts with the one-wayness property of Ajtai’s function. \( \square \)

**Theorem 2** The proposed threshold signature scheme is existentially unforgeable under a chosen-message attack.

**Proof:**

The proof is by contradiction. Assume that there is an adversary \( \mathcal{A} \) that breaks the existential unforgeability of the threshold signature scheme in polynomial time with non-negligible probability \( \delta(n) \). We propose a polynomial time adversary \( \mathcal{B} \) that finds collision in the trapdoor collision-resistant hash function with a non-negligible probability \( \delta(n) \). Using a public key matrix \( A \) and the function \( f_A(x) = Ax \), \( \mathcal{B} \) takes \( \mathcal{A} \) into service on the public key \( A \), and performs the following simulation using the signing oracle and the random oracle \( H \): \( \mathcal{B} \) samples \( \sigma_m \) using \( \text{SampleDom}(1^n) \) and sends \( f_A(\sigma_m) \) to \( \mathcal{A} \), where \( \text{SampleDom}(1^n) \) samples some \( x \) from a distribution, for which \( f_A(x) \) is uniformly distributed. Whenever \( \mathcal{A} \) makes a signing query on \( m \), \( \mathcal{B} \) looks up \( (m, \sigma_m) \) in its local storage and returns \( \sigma_m \) as the signature. \( \mathcal{A} \) queries \( H \) on \( m^* \) and produces \( (m^*, \sigma^*) \) as a forge. Then, \( \mathcal{B} \) looks up \( (m^*, \sigma_{m^*}) \) in its local storage and outputs \( (\sigma^*, \sigma_{m^*}) \) as a collision in \( f_A \). Now, we show that \( \sigma^* \neq \sigma_{m^*} \), which makes them a correct collision. To do so, we consider two cases: Case 1: If \( \mathcal{A} \) had made a signing query on \( m^* \), it had received the signature \( \sigma_{m^*} \). Since \( \mathcal{A} \) outputs a forge, it should produce a different valid signature \( \sigma^* \) meaning \( \sigma^* \neq \sigma_{m^*} \). Case 2: If \( \mathcal{A} \) had not made a signing query on \( m^* \), \( \mathcal{A} \) only views \( f_A(\sigma_{m^*}) \) and because of preimage min-entropy property of the hash family, \( \sigma^* \neq \sigma_{m^*} \), with overwhelming probability \( 1 - 2^{\omega(\log n)} \). Hence, we conclude that \( \mathcal{B} \) outputs a collision in \( f_A \) with negligible probability close to \( \delta(n) \). \( \square \)

**Proposition 1** Let \( c_1,\ldots,c_t \) be the shares corresponding to the private key \( R \). Then, using the pseudo-secret shares \( d_k = [d_{k1},\ldots,d_{km}] \), \( k = 1,\ldots,t \), the private key \( R \) cannot be revealed in polynomial time, where \( d_k = z^TE_i\epsilon_{ij}, i = 1,\ldots,m, k = 1,\ldots,t \), where \( E_i \in \mathbb{Z}_q^{r \times s} \) and \( z \in \mathbb{Z}_q^{n} \). Furthermore, the anonymity of the participants is preserved.

**Proof:**

Because of one-wayness of the Ajtai’s functions \( f_E(x) = E_ix, i = 1,\ldots,m \), one cannot compute the shares \( c_{ij}, k = 1,\ldots,t \) from the \( E_i \epsilon_{ij} \) and consequently from the pseudo-secret shares \( d_k = z^TE_i\epsilon_{ij} \). Hence, the private key \( R \) cannot be revealed in polynomial time.

Consider that \( M \in \{0,1\}^* \) is a message to be signed and \( t_{k1}, t_{k2} \) are two sets of the participants. The signatures \( \sigma_{t_{k1}} \) and \( \sigma_{t_{k2}} \) are the vectors having the same distribution of the domain in \( f_A \) within negligible statistical distance of \( D_{\chi_{\text{PILARM}}}[\mathcal{A},s] \), which implies that \( \sigma_{t_{k1}} \) and \( \sigma_{t_{k2}} \) are computationally indistinguishable [43]. Therefore, the anonymity of the participants taking part in the threshold signature process is preserved, enabling the proposed scheme to be a threshold anonymous signature scheme. \( \square \)

### Table 1 Memory usage for different TMSSS schemes

| Scheme                  | Size of public values per secret size | Size of each share per secret size |
|-------------------------|--------------------------------------|-----------------------------------|
| He & Dawson [16]        | \( m \times n \)                      | 1                                 |
| Ham [48]                | \( m \times (n - t) \)                | 1                                 |
| Chang [20]              | \( m \times n \)                      | 1                                 |
| Das [21]                | \( m \times t \times n \)             | 1                                 |
| Ham [49]                | \( m^2(n + 1) \) / \( t \)            | \( m \)                            |
| Pilaram [31]            | \( m \times r \)                      | \( r / (t \log q) \approx 0.5 \) |
| The proposed TMSSS scheme | \( r + n \)                          | \( r / (t \log q) \approx 0.5 \) |

**Abbreviation:** TMSSS, threshold multi-stage secret sharing.

### 6 PERFORMANCE ANALYSIS

In this section, we investigate the performance of the proposed schemes in view of memory usage and time complexity.
From the viewpoint of memory consumption, the matrix $E$, the permutation matrix $P$ and the vectors $\lambda_j, j = 1, \ldots, n$ used in the improved TMSSS scheme, proposed in Section 3, are published on the bulletin board. The shares $e_j, j = 1, \ldots, n$ are sent to the participants via a secure channel. The amount of memory needed to save these matrices are given in Table 1. $m$ is the number of secrets, $n$ is the number of participants and $t$ is the threshold. In the improved scheme, the size of public parameters is independent from the number of the secrets, significantly outperforming the previous schemes. The ratio of public values in our proposed TMSSS scheme compared to Ref. [31] is equal to $\frac{m+n}{m+n} \approx 2/m$, if $t \approx n$. This ratio is very low for large values of $m$ as the number of secrets. It should be noted that the size of each share per secret is equal to $r/(t \log q)$, which approximately equals to 0.5, if $r = \max\{t \log t, n\} \approx t \log t$.

From the viewpoint of complexity, secret recovery consists of two steps:

1. The participants' side: In this step, each participant computes the pseudo-secret share from his share. In computing the pseudo-secret shares, we need simple column addition in matrix $E_i$, since the shares are binary vectors. This process has the complexity of $O(t)$ for each participant which is less than the complexity of modular exponentiation used in other schemes. Therefore, this scheme is suitable for the low-complexity applications.

2. The combiner's side: The complexity of this step is $O(t^3)$, which includes one matrix inversion and one matrix multiplication.

The share verification by the participants has complexity of $O(t^3)$ for each participant, since the shares are binary vectors, which has lower complexity than that of the other schemes using modular exponentiation.

Here, we compare our proposed threshold signature scheme with that proposed in Ref. [40], in which the Shamir's scheme is applied to each entry of the private key matrix. The size of shares for each participant in [40] is equal to the number of entries in the private matrix $R$ that is equal to $m \times k$, where $m = m - lk$, $k \in \left[ \log q \right]$, and $l, q$ are security parameters. In our proposed scheme, the share size is decreased to $\max\{l \log (lk), n/k\} \approx l \log (lk)$, where $n$ is the number of the participants. In Table 2, we compare the size of each share in the proposed scheme, using the parameters suggested in Ref. [41], with that proposed in Ref. [40]. We observe that the share size is much smaller than that of Bendlin's [40], which makes it suitable for memory-constrained applications such as smart cards.

### CONCLUSIONS

We have improved the previously proposed TMSSS scheme, in such a way that less public values are published on the bulletin board while preserving the security. Furthermore, we have proposed a threshold signature scheme based on the trapdoor function introduced by Micciancio and Peikert. Here, we share the private key using the already improved lattice-based TMSSS scheme. For threshold signing, the participants do partial signature on the plaintext using their shares and send the results to the combiner. The proposed threshold signature scheme inherits the desired features such as anonymity and verifiability from the improved TMSSS scheme. Moreover, both schemes are efficient because simple matrix operations are used in the secret sharing and threshold signature protocols. The security of the proposed schemes is based on the lattice hard problems which are believed to resist against quantum computers.

### ACKNOWLEDGEMENT

This work is supported by Iran National Science Foundation (INSF) under grant number 94017742.

### ORCID

Taranee Eghlidos

https://orcid.org/0000-0002-3182-0277

### REFERENCES

1. Kanai, A., Tanimoto, S., Sato, H.: Performance evaluation on data management approach for multiple clouds using secret sharing scheme. In: IEEE International Conference on Consumer Electronics (ICCE), pp. 471–473 (2016)

2. Chunying, W., Shundong, L., Yiying, Z.: Key management scheme based on secret sharing for wireless sensor network. In: Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on, pp. 574–578 (2013)

3. Yuan, L., et al.: A verifiable E-voting scheme with secret sharing. In: IEEE 16th International Conference on Communication Technology (ICCT), pp. 304–308 (2015)

4. Chang, C.C., Chen, W.Y., Chang, S.C.: A highly efficient and secure electronic cash system based on secret sharing in cloud environment. Secur. Commun. Net. 9(14), 2476–2483 (2016), https://doi.org/10.1002/sec.1517

5. Shamir, A.: How to share a secret. Commun. ACM. 22(11), 612–613 (1979), http://doi.acm.org/10.1145/359168.359176

6. Blakley, G.R.: Safeguarding cryptographic keys. Proc. 1979 AFIPS Nat. Comput. Conf. 48, 313–317 (1979)

7. Brickell, E.F., Stinson, D.R.: The detection of cheaters in threshold schemes. In: Goldwasser, S. (ed.) Advances in Cryptology – CRYPTO 88, vol. 403 of LNCS, pp. 564–577. Springer, New York (1990). https://doi.org/10.1007/3-540-46839-1

8. Mahmoud, Q.A.: A novel verifiable secret sharing with detection and identification of cheaters' group. Int. J. Math. Sci. Comput. 2(2), 1–13 (2016)

9. Studler, M.: Publicly verifiable secret sharing. In: Maurer, U. (ed.) Advances in Cryptology – EUROCRYPT ’96, vol. 1070 of LNCS, pp. 190–199. Springer, Berlin Heidelberg (1996). https://doi.org/10.1007/3-540-68339

### TABLE 2

| $\ell$ | $L$ | $K$ | Size of private key | Size of share [40] | Size of share (the proposed scheme) |
|-------|-----|-----|---------------------|-------------------|-----------------------------------|
| $13,812$ | $284$ | $24$ | $47684736$ | $47684736$ | $3617$ |
| $446,644$ | $436$ | $32$ | $6036918784$ | $6036918784$ | $6003$ |
