An Efficient Lattice-Based Secret Sharing Construction

Rachid El Bansarkhani[1] and Mohammed Meziani[2]

[1] Technische Universität Darmstadt
Fachbereich Informatik
Kryptographie und Computeralgebra,
Hochschulstraße 10, 64289 Darmstadt, Germany
elbansarkhani@cdc.informatik.tu-darmstadt.de

[2] CASED – Center for Advanced Security Research Darmstadt,
Mornewegstrasse 32, 64293 Darmstadt, Germany
mohammed.meziani@cased.de

Abstract. This paper presents a new construction of a lattice-based verifiable secret sharing scheme. Our proposal is based on lattices and the usage of linear hash functions to enable each participant to verify its received secret share. The security of this scheme relies on the hardness of some well known approximation problems in lattices such as n^{c}-approximate SVP. Different to protocols proposed by Pedersen this scheme uses efficient matrix vector operations instead of exponentiation to verify the secret shares.

Keywords: Secret Sharing, Lattice-based Cryptography, Shortest Vector Problem, Hash Functions.

1 Introduction

It is known from [12] that quantum algorithms could break most of number-theory based cryptosystems used in practice (e.g., RSA, DL). Indeed, such algorithms can solve both the factoring problem and the discrete log problem in finite fields and on elliptic curves in polynomial time. It may therefore be desirable to design new cryptographic primitives, such as public-key cryptosystems and secret sharing schemes, whose security depends upon problems that are not vulnerable to quantum attacks. Lattice-based systems offer a promising alternative to number-theoretic ones, and they are believed to be secure against quantum attacks since their security is based on lattice problems that in their general form are well-known NP-hard. In addition to post-quantum security, they are easy to implement and utilize only basic operations.

Motivated by the need for protecting cryptographic keys by more than one party, secret sharing schemes were first introduced independently in 1979 by Shamir [11] and Blakley [3]. The Shamir's proposal relies on polynomials with n coefficients where n is the number of parties involved to reconstruct the coefficients of polynomials with degrees at most $\mathsf{n} - 1$. Blakley, instead, makes use of the fact that different hyperplanes can intersect in one point. And a certain amount of these hyperplanes is only required to specify that point. Since then several other secret sharing schemes have been developed, for example those that are based on the Chinese Remainder Theorem ([7] and [2]) or hybrid constructions [8] using the discrete logarithm problem for verification. Our construction follows a new simple approach since it uses only simple operations from linear algebra. Compared to the previous schemes it recovers a vector rather than a single value. Furthermore, the verification of the secret shares as well as the secret vector can be done via lattice-based hash functions whose security is reducible to some NP-hard problems in lattices. Previous schemes, instead, rely on the discrete logarithm problem for the verification. The main idea behind our scheme is to recover a basismatrix at first which is then required to compute the secret vector that in turn could be used as a secret key for the lattice-based LWE encryption scheme [10]. Secret sharing systems can be found in many real applications such as secure multi-party computations [4], e-voting [14], and sensor networks [6]. In the basic model of secret sharing we differ at least two major protocols: the distribution protocol in which the dealer forwards the secret shares to the participants, and the combination or reconstruction protocol in which the secret is recovered by pooling the shares of a qualified subset of the participants.

A system is called a (t, n) threshold secret sharing scheme with $\mathsf{t} \leq \mathsf{n}$, when at least t participants are required to recover the secret key, where n is the number of participants obtaining a secret share from the dealer. We present a (n, n) secret sharing scheme based on the difficulty to solve n^{c}-approximate SVP in lattices for some constant c.

2 Preliminaries

The following briefly introduces the definitions and notations used in the present paper.

2.1 Notations

The following notations and definitions will be used throughout the rest of the paper.

- $\star$ $|x|$ is the length in bits of a string x.
- $\star$ The Hamming weight of a string x is the number of its non-null coordinates and denoted by $\mathtt{wt}(x)$.
- $\star$ $[\mathsf{v}_1, ..., \mathsf{v}_\mathsf{n}]$ is the matrix composed by the column vectors $\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n}$.
- $\star$ By $\mathsf{t}_1 = (\mathsf{t}_1^{(1)}, ..., \mathsf{t}_1^{(\mathsf{n})})$ we denote a vector consisting of n elements, where $\mathsf{t}_\mathsf{j}^{(\mathsf{i})}$ identifies the i-th element of the vector t_j.
- $\star$ For a finite set S, we denote by $x \overset{\$}{\leftarrow} S$ the experiment of uniformly choosing an element x from S and assigning it to x.
- $\star$ With $\lceil r \rceil$ we round a real number r up to the next integer.
- $\star$ $x^\top$ is the transpose of a vector x.
- $\star$ $\mathsf{rot}(x)$ cyclically rotates the vector x, e.g. $\mathsf{rot}((1, 2, 3, 4)) = (4, 1, 2, 3)$.

2.2 Basics on Lattices

A lattice L is a discrete abelian subgroup of $\mathbb{R}^{\mathsf{m}}$. A basis $\mathsf{B} = [\mathsf{b}_1, .., \mathsf{b}_\mathsf{n}] \in \mathbb{R}^{\mathsf{m}\times\mathsf{n}}$ for the lattice L consists of $\mathsf{d} \leq \mathsf{n}$ linearly independent vectors b_i with $\mathcal{L}(\mathsf{B}) = \left\{ \sum_{\mathsf{i}=1}^{\mathsf{n}} \mathsf{t}_\mathsf{i}\mathsf{b}_\mathsf{i} \mid \mathsf{t}_1, ..., \mathsf{t}_\mathsf{n} \in \mathbb{Z} \right\} \subset \mathbb{R}^{\mathsf{m}}$, i.e. $\mathcal{L}$ is the set of all integer combinations of the vectors in B. The minimum distance of a lattice $\mathcal{L}(\mathsf{B})$, denoted $\lambda(\mathcal{L}(\mathsf{B}))$, is the minimum distance between any two distinct lattice points, and equals the length of a nonzero shortest lattice vector.

$$\lambda(\mathcal{L}(\mathsf{B})) = \min\{\|\mathsf{x}\| : \mathsf{x} \in \mathcal{L}(\mathsf{B})\} \qquad (1)$$

Definition 1. ***(Shortest Vector Problem - SVP)***
Let $\mathcal{L}(\mathsf{B}) \subset \mathbb{R}^{\mathsf{n}}$ be a d-dimensional lattice and $\mathsf{B} \in \mathbb{Z}^{\mathsf{n}\times\mathsf{d}}$ the corresponding basis matrix. An input to SVP is a lattice basis B and the goal is to find a vector x in $\mathcal{L}(\mathsf{B})$ such that $\|\mathsf{x}\| = \lambda(\mathcal{L}(\mathsf{B}))$

Definition 2. ***(Approximate Shortest Vector Problem - γ-SVP)***
Let $\mathcal{L}(\mathsf{B}) \subset \mathbb{R}^{\mathsf{n}}$ be a d-dimensional lattice and $\mathsf{B} \in \mathbb{Z}^{\mathsf{n}\times\mathsf{d}}$ the corresponding basis matrix. An input to SVP is a lattice basis B and the goal is to find a vector x in $\mathcal{L}(\mathsf{B})$ such that $\|\mathsf{x}\| = \gamma \cdot \lambda(\mathcal{L}(\mathsf{B}))$

3 Our Construction

In this section, we present a non-interactive verifiable lattice-based secret sharing protocol that requires all parties to participate with their shares in order to recover a secret, while at the same time enabling them to verify their secret share. In this scheme, a dealer selects different vectors of length n and weight $\mathsf{m} > 1$ and produces secret shares of length n , where n is a positive integer. A higher weight is chosen in order to prevent the participants to get information about the basis vectors from their secret shares, because the secret share is a linear combination of the private basis vectors.

3.1 Description

This scheme works in non-interactive manner, meaning that each participant can verify its own secret share without communicating with other participants. It consists of two steps: the distribution and combination phase. On the other hand, its security relies on the hardness of solving SVP in lattices. Furthermore, it is very fast as it uses only matrix-vector multiplication to recover the secret.

The principle of our proposal is the following: the dealer generates a private lattice basis $\mathsf{B} \in \mathbb{Z}^{\mathsf{n}\times\mathsf{n}}$ with low orthogonal defect and selects linearly independent vectors $\lambda_\mathsf{i} \in \{\mathsf{x} \in \mathbb{F}_2^\mathsf{n} : \mathsf{wt}(\mathsf{x}) = \mathsf{m} > 1\}$ and publishes them together with the hashed (encrypted) secret and basis vectors. The dealer also reveals the vector $\mathsf{v} \in \mathbb{Z}^\mathsf{n}$ which is needed to compute the secret as $\mathsf{s} := \mathsf{B} \cdot \mathsf{v}$. Here a linear hash function is used, such as Ajtai's hash function [1] or more efficient ones proposed by Lyubashevsky and Micciancio [5] or Peikert and Rosen [9], to enable each participant to verify its secret share in a non-interactive manner. It is also possible to take a semi-homomorphic lattice-based encryption scheme with additive property to achieve the same results. As next step, the dealer secretly distributes the vectors $\mathsf{c_i} = \mathsf{B} \cdot \lambda_\mathsf{i}$ to each participant $\mathsf{P_i}$. By doing so, n participants would be able to recover the private basis and compute the secret. In what follows, we describe the distribution and combination phase in details.

Distribution Phase

1. D chooses a private lattice Basis B and selects linearly independent binary vectors λ_i of length n and weight $\mathsf{m} > 1$. Then he computes the secret shares $\mathsf{c_i} = \mathsf{B} \cdot \lambda_\mathsf{i}$ and secretly forwards them to the participants.
2. Upon receipt of the own secret share, each participant P checks the hash value, in particular he verifies whether the hash value of its secret share can be written as a linear combination of the hashed basis vectors, i.e., $\mathsf{H}(\mathsf{c_i}) \overset{?}{=} \sum \lambda_{ij} \cdot \mathsf{H}(\mathsf{b_j})$, since the λ_i and $\mathsf{H}(\mathsf{b_j})$ as well as the secret generating vector v are public. If the hash value is valid, the participant sends its acknowledgment to the dealer back.

   Since many hash functions are fed with binary strings of a special length we can hash each component of the received vector without losing the linearity of the hash function. In this way we obtain for every input vector a matrix as hash value.

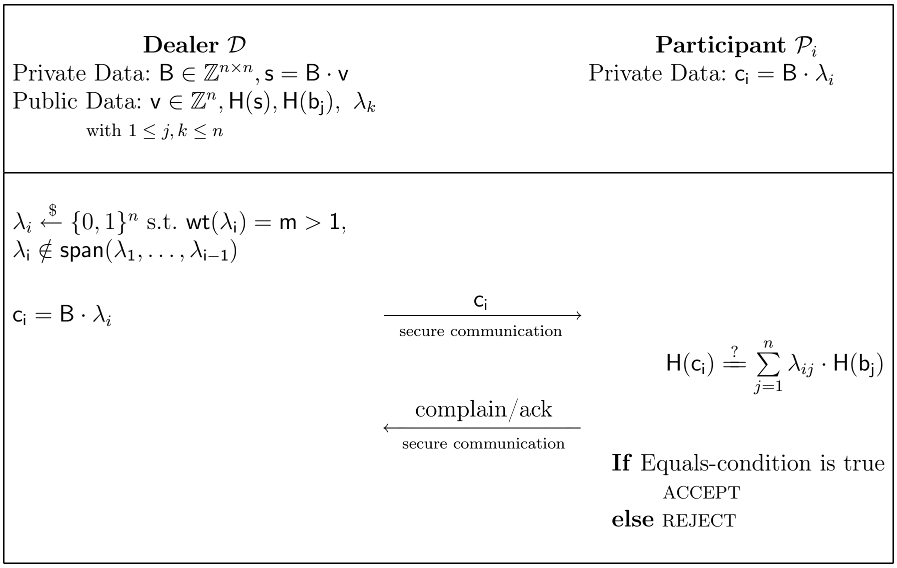
Dealer $\mathcal{D}$
Private Data: $\mathsf{B} \in \mathbb{Z}^{n\times n}, \mathsf{s} = \mathsf{B} \cdot \mathsf{v}$
Public Data: $\mathsf{v} \in \mathbb{Z}^n, \mathsf{H}(\mathsf{s}), \mathsf{H}(\mathsf{b_j}),\ \lambda_k$ with $1 \leq j, k \leq n$
Participant $\mathcal{P}_i$
Private Data: $\mathsf{c_i} = \mathsf{B} \cdot \lambda_i$
$\lambda_i \xleftarrow{\$} \{0,1\}^n$ s.t. $\mathsf{wt}(\lambda_\mathsf{i}) = \mathsf{m} > 1$,
$\lambda_\mathsf{i} \notin \mathsf{span}(\lambda_1, \ldots, \lambda_{\mathsf{i}-1})$
$\mathsf{c_i} = \mathsf{B} \cdot \lambda_i$
$\xrightarrow[\text{secure communication}]{\mathsf{c_i}}$
$\mathsf{H}(\mathsf{c_i}) \overset{?}{=} \sum_{j=1}^{n} \lambda_{ij} \cdot \mathsf{H}(\mathsf{b_j})$
$\xleftarrow[\text{secure communication}]{\text{complain/ack}}$
If Equals-condition is true
ACCEPT
else REJECT

Fig. 1. Distribution Phase

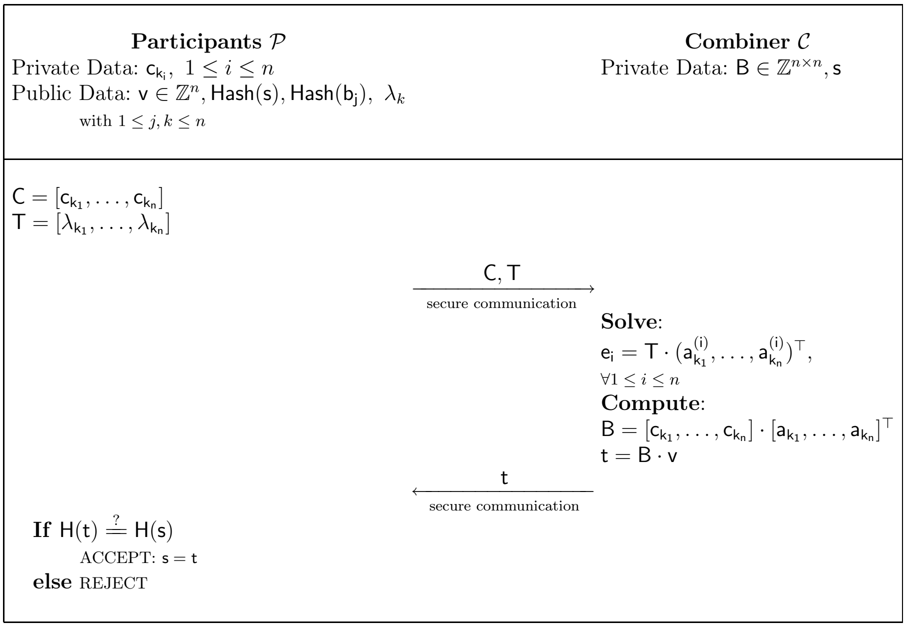
Participants $\mathcal{P}$
Private Data: $\mathsf{c}_{\mathsf{k}_\mathsf{i}},\ 1 \le i \le n$
Public Data: $\mathsf{v} \in \mathbb{Z}^n, \mathsf{Hash}(\mathsf{s}), \mathsf{Hash}(\mathsf{b_j}),\ \lambda_k$ with $1 \le j, k \le n$
Combiner $\mathcal{C}$
Private Data: $\mathsf{B} \in \mathbb{Z}^{n \times n}, \mathsf{s}$
$\mathsf{C} = [\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}]$
$\mathsf{T} = [\lambda_{\mathsf{k}_1}, \ldots, \lambda_{\mathsf{k}_\mathsf{n}}]$
C, T
secure communication
Solve:
$\mathsf{e_i} = \mathsf{T} \cdot (\mathsf{a}^{(\mathsf{i})}_{\mathsf{k}_1}, \ldots, \mathsf{a}^{(\mathsf{i})}_{\mathsf{k}_\mathsf{n}})^\top,$
$\forall 1 \le i \le n$
Compute:
$\mathsf{B} = [\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}] \cdot [\mathsf{a}_{\mathsf{k}_1}, \ldots, \mathsf{a}_{\mathsf{k}_\mathsf{n}}]^\top$
$\mathsf{t} = \mathsf{B} \cdot \mathsf{v}$
t
secure communication
If $\mathsf{H}(\mathsf{t}) \overset{?}{=} \mathsf{H}(\mathsf{s})$
ACCEPT: $\mathsf{s} = \mathsf{t}$
else REJECT

Fig. 2. Combination Phase

Combination Phase

The combination phase involves three steps to recover the secret:

1. n participants $\mathsf{P}_{\mathsf{k}_1}, \ldots, \mathsf{P}_{\mathsf{k}_\mathsf{n}}$ secretly send their received secret shares $\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}$ to the combiner. These secret shares $\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}$ can be represented as square matrix of size $\mathsf{n} \times \mathsf{n}$, i.e., $\mathsf{C} = [\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}]$. Then the combiner solves the following equations for the unkonwn vectors $\mathsf{a}_{\mathsf{k}_1}, \ldots, \mathsf{a}_{\mathsf{k}_\mathsf{n}}$, where $\mathsf{e_i}$ denotes the i-th n-dimension unit vector:

$$\mathsf{e_i} = [\lambda_{\mathsf{k}_1}, \ldots, \lambda_{\mathsf{k}_\mathsf{n}}] \cdot (\mathsf{a}^{(\mathsf{i})}_{\mathsf{k}_1}, \ldots, \mathsf{a}^{(\mathsf{i})}_{\mathsf{k}_\mathsf{n}})^\top,\ 1 \le \mathsf{i} \le \mathsf{n} \tag{2}$$

   We then obtain a matrix $\mathsf{A} = [\mathsf{a}_{\mathsf{k}_1}, \ldots, \mathsf{a}_{\mathsf{k}_1}]$ in order to compute the private lattice basis B. The following equation shows that the private Basis can be expressed as a function of the matrices C and A:

$$\begin{aligned} \mathsf{B} &= [\mathsf{c}_{\mathsf{k}_1}, \ldots, \mathsf{c}_{\mathsf{k}_\mathsf{n}}] \cdot [\mathsf{a}_{\mathsf{k}_1}, \ldots, \mathsf{a}_{\mathsf{k}_\mathsf{n}}]^\top \\ &= [\mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{n}] \cdot [\lambda_{\mathsf{k}_1}, \ldots, \lambda_{\mathsf{k}_\mathsf{n}}] \cdot [\mathsf{a}_{\mathsf{k}_1}, \ldots, \mathsf{a}_{\mathsf{k}_\mathsf{n}}]^\top \\ &= [\mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{n}] \cdot [\mathsf{e}_1, \ldots, \mathsf{e}_\mathsf{n}] \end{aligned} \tag{3}$$

2. The combiner computes the secret by simple matrix vector multiplication.

$$\mathsf{s} = \mathsf{B} \cdot \mathsf{v}, \tag{4}$$

   The secret is then sent via a secure communication channel to each participant back.

3. The participants can then verify the secret produced by the combiner. To this end, the participant compares the hash of the received value with the hashed secret provided by the dealer which is also publicly available.

Example: n=4, $\mathsf{wt} = 3$

$$[\lambda_1, \ldots, \lambda_4] = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}, [b_1, \ldots, b_4] = \begin{pmatrix} 7 & 9 & 8 & 4 \\ 6 & 3 & 3 & 0 \\ 5 & 9 & 0 & 5 \\ 4 & 4 & 2 & 3 \end{pmatrix},$$

$$[c_1, \ldots, c_4] = \begin{pmatrix} 15 & 22 & 10 & 12 \\ 17 & 16 & 13 & 7 \\ 18 & 21 & 11 & 9 \\ 15 & 16 & 5 & 8 \end{pmatrix}$$

3.2 Security and Performance

The security of our construction is based on the underlying hash function which is used for hiding the basis vectors as well as the secret. It is also possible to use any encryption function instead, which however needs to satisfy the additive property. The security of Ajtai's hash function (see Algorithm 1) or more efficient ones constructed by Lyubashevsky and Micciancio [5] or Peikert and Rosen [9] can be reduced to the hardness of solving approximate n^{c}-SVP.

Algorithm 1 Calculate A hash function following Ajtai's construction

```
1. Parameters: integers $\mathsf{n}, \mathsf{k}, \mathsf{p}, \mathsf{d} \geq 1$ with $\mathsf{n} > \frac{\mathsf{k} \cdot \log(\mathsf{p})}{\log(\mathsf{d})}$
2. Key: a matrix $\mathsf{B}$ chosen uniformly from $\mathbb{Z}_{\mathsf{p}}^{\mathsf{k} \times \mathsf{n}}$
3. Hash function: $\mathsf{f_A} : \{0, \ldots, \mathsf{d}-1\}^{\mathsf{n}} \rightarrow \mathbb{Z}_{\mathsf{p}}^{\mathsf{k}}$ given by $\mathsf{f_A}(\mathsf{x}) = \mathsf{A} \cdot \mathsf{x} \bmod \mathsf{p}$
```

As already mentioned above many hash functions are fed with a binary input string and produce a vector. Since we operate on n dimensional vector spaces we would have to hash each component of a vector leading to a $\mathsf{n} \times \mathsf{n}$-matrix as a hash value without affecting the additive property. In the distribution phase every participant verifies its secret share. The needed number of operations and transferred bits of each protocol step are depicted in the following tables. When $\mathbb{Z}^n$ is replaced by $\mathbb{Z}_q^n$, the bit size of each vector component is bounded by $\log(\mathsf{q})$ and the solution is unique if the determinant of $[\lambda_{\mathsf{k}_1}, \ldots, \lambda_{\mathsf{k}_\mathsf{n}}]$ is coprime to q as shown in [13].

In a (n, n) scheme the dealer can efficiently generate n linearly independent vectors having length n by setting the weight to $\mathsf{n} - 1$. If we use the vectors

Table 1. Theoretical performance in the distribution phase

	Dealer	Participant
Computation (bops)	$\mathsf{n}^2\mathsf{m} \cdot \log(\mathsf{q})$	$\frac{\mathsf{n}^2 \cdot \log(\mathsf{q})}{\log(\mathsf{p})}$
Communication (bits)	$\mathsf{n} \cdot \log(\mathsf{q})$	1

Table 2. Theoretical performance in the combination phase

	Participants	Combiner
Computation (bops)		$\mathsf{n}^2 \cdot (\mathsf{n}^2 + \mathsf{n}^{0.373} + \log(\mathsf{q}))$
Communication (bits)	$\mathsf{n}^2 \cdot (\log(\mathsf{q}) + 1)$	$\mathsf{n} \cdot \log(\mathsf{q})$

$\lambda_i = (\sum_{j=1}^{n} e_j) - e_i$ for $1 \leq i \leq n$ it can easily be shown that these vectors form a base due to the fact that each vector represents a vertice in a n-dimensional hypercube. These vertices are on different axial planes perpendicular to each other. So any checks in terms of linearly independence can be neglected.

Example: $n = 6$, $wt = 5$

$$[\lambda_1, \ldots, \lambda_6] = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

From linear algebra we know that fewer than n participants cannot recover the private basis and thus get no information about the secret s because the secret satisfies equation (4). It is also possible to show that no unit vector can be computed in the first step of the combination phase when the number of participants is less than n. So any private basis vector remains hidden. This advantageous side effect comes from the fact that $n - 1$ of the chosen vectors λ_k span a $(n - 1)$-dimensional hyperplane connecting the $n - 1$ vertices and the origin in a hypercube. So every unit vector is outside the hyperplane.

Our secret sharing scheme has been implemented on a 2.5 GHz Intel Core i5-2520M, running Linux (Ubuntu 11.04) 32 Bit. The performance results of the implementation are reported in Table 3.

Table 3. Performance of the secret sharing scheme

Number of users (dimension)	Distribution timing (sec)	Combination timing (sec)	Verification timing (sec)
n=128	0.005864	0.006625	0.000085
n=256	0.035171	0.042033	0.00023
n=512	0.195757	0.273959	0.000676
n=1024	1.212575	1.922552	0.002577

This scheme can easily be extended to a protocol where at most n out of $\binom{\mathsf{n}}{\mathsf{m}}$ participants are required to recover the secret. By this means, the chosen weight would maximize the binomial coefficient and therefore the number of potential secret share holders. This is the case when m equals to $\lceil\frac{\mathsf{n}}{2}\rceil$. To this end, the dealer selects all possible vectors with weight $\mathsf{wt} = \lceil\frac{\mathsf{n}}{2}\rceil$ and computes the corresponding secret shares. But the reconstruction of the secret is computational infeasible for less than $\lceil\frac{\mathsf{n}}{2}\rceil + 1$ parties because fewer than $\lceil\frac{\mathsf{n}}{2}\rceil + 1$ of the selected vectors cannot span a $(\lceil\frac{\mathsf{n}}{2}\rceil + 1)$-dimensional subspace and with the same argument as above no unit vector can be computed. However, every particpant can determine which other secret share holder could involve for the purpose of computing the secret since λ_{i} of all participants are publicly available. As n vectors form the basis matrix B, which is not necessarily consisting of linearly independent vectors, it is required to know at least $\lceil\frac{\mathsf{n}}{2}\rceil$ vectors in order to recover all basis vectors. One method is to use cyclic rotations of $\frac{\mathsf{n}}{2}$ basis vectors.

Example:

1. Dimension: $\mathsf{n} = 100, (\mathsf{n} = 1024)$
2. Weight: $\mathsf{n} - 2 = 98, (1022)$
3. Nr. of particpiants: $\binom{\mathsf{n}}{\mathsf{n}-2} = \frac{\mathsf{n}\cdot(\mathsf{n}-1)}{2} = 4950, (523776)$
4. Min. number of particpants to recover the secret: $\mathsf{n} - 1 = 99, (1023)$
5. Max. number of particpants to recover the secret: $\mathsf{n} = 100, (1024)$
6. $\mathsf{B} = [\mathsf{b}_1, \ldots, \mathsf{b}_{\mathsf{n}-1}, \mathsf{rot}(\mathsf{b}_{\mathsf{n}-1})]$

4 Conclusion

In this work we present a new and efficient construction of a secret sharing scheme by making use of a lattice based hash function for the protection of the secret. Every secret share holder is enabled to verify its share due to the linearity of the employed hash function. We showed that breaking this scheme by attacking the hash function is at least as hard as solving approximate SVP. We also showed that in the (n, n)-scheme all parties have to participate in order to recover the secret. In addition, it is computationally infeasible for less than n participants to compute the basis matrix and hence the secret. Furthermore, our scheme can be extended into n out of $\binom{\mathsf{n}}{\lceil\frac{\mathsf{n}}{2}\rceil}$ secret sharing scheme in order to increase the number of potential secret share holders.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, pp. 99–108. ACM (1996)
2. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory 29(2), 208–210 (1983)
3. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceedings of the National Computer Conference, pp. 313–317. American Federation of Information Processing Societies (1979)
4. Cramer, R., Damgård, I., Maurer, U.: General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
5. Lyubashevsky, V., Micciancio, D.: Generalized Compact Knapsacks Are Collision Resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006)
6. Mahimkar, A.: Securedav: A secure data aggregation and verification protocol for sensor networks. In: Proceedings of the IEEE Global Telecommunications Conference, pp. 2175–2179 (2004)
7. Mignotte, M.: How to Share a Secret? In: Beth, T. (ed.) Cryptography - EUROCRYPT 1982. LNCS, vol. 149, pp. 371–375. Springer, Heidelberg (1983)
8. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
9. Peikert, C., Rosen, A.: Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006)
10. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 34:1–34:40 (2009)
11. Shamir, A.: How to share a secret. Commun. ACM 22, 612–613 (1979)
12. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26, 1484–1509 (1997)
13. Smith, H.J.S.: On systems of linear indeterminate equations and congruences. Philosophical Transactions of the Royal Society of London 151, 293–326 (1861)
14. Sorin, Iftene: General secret sharing based on the chinese remainder theorem with applications in e-voting. Electronic Notes in Theoretical Computer Science 186, 67–84 (2007)