Cryptanalysis of the Randomized Version of a Lattice-Based Signature Scheme from PKC'08

Haoyu LiRenzhang Liu[1], Abderrahmane Nitaj[2], and Yanbin Pan[1] [*]

[1] Key Laboratory of Mathematics Mechanization, NCMIS,
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
Beijing 100190, China
[2] Laboratoire de Mathématiques Nicolas Oresme
Université de Caen Basse Normandie, France

Abstract. In PKC'08, Plantard, Susilo and Win proposed a lattice-based signature scheme, whose security is based on the hardness of the closest vector problem with the infinity norm (CVP_∞). This signature scheme was proposed as a countermeasure against the Nguyen-Regev attack, which improves the security and the efficiency of the Goldreich, Goldwasser and Halevi scheme (GGH). Furthermore, to resist potential side channel attacks, the authors suggested modifying the deterministic signing algorithm to be randomized. In this paper, we propose a chosen message attack against the randomized version. Note that the randomized signing algorithm will generate different signature vectors in a relatively small cube for the same message, so the difference of any two signature vectors will be relatively short lattice vector. Once collecting enough such short difference vectors, we can recover the whole or the partial secret key by lattice reduction algorithms, which implies that the randomized version is insecure under the chosen message attack.

Keywords: Lattice-based cryptography, Signature schemes, Lattice reduction

1 Introduction

It is well known that classical cryptography is vulnerable to quantum computers since Shor's algorithm [19] will solve the integer factorization and the logarithm discrete problems efficiently. This has motivated the development of post-quantum cryptography, especially lattice-based cryptosystems. In general, the security of lattice-based cryptosystems is always related to some hard computational problems in lattices, such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP).

As important cryptographic primitives, several lattice-based digital signature schemes have been proposed in recent years, such as [8, 9, 17, 7, 5]. In 1997, Goldreich, Goldwasser and Halevi [8] proposed the GGH signature scheme based on lattices, whose security is related to the hardness of approximate CVP. In fact, GGH is not only a concrete signature scheme, but also a general framework to construct lattice-based digital signature schemes. The GGH framework consists of a good lattice basis G, a bad basis B for the same lattice and a reduction algorithm as the signing algorithm. Usually, the good basis is used as the secret key, with which the reduction algorithm can efficiently output an approximation for the closest vector of a target vector corresponding to the message. Such approximation is the signature of the message. The bad basis is published as the public key, with which one can check if the signature is in the lattice and close enough to the target vector. In GGH scheme, they used a nearly orthogonal basis G as the good basis, a random basis as the bad basis B, and Babai's rounding-off algorithm [2] as the reduction algorithm.

Based on GGH framework, Hoffstein, Howgrave-Graham, Pipher, Silverman, and Whyte [10] presented the NTRUSign as a more efficient lattice-based signature scheme. They used some special short basis as a good basis, a "random" basis as the bad basis B, and Babai's rounding-off algorithm as the reduction algorithm.

However, Nguyen and Regev [16] proposed a clever method to recover the secret key of the GGH signature scheme and NTRUSign by studying the parallelepiped of the lattice. More precisely, by collecting enough message-signature pairs, they can obtain many samples uniformly distributed in the parallelepiped due to Babai's rounding-off algorithm employed as reduction algorithm in this two signature schemes. Then with these samples, they can finally recover the parallelepiped which leaks the good basis. They also pointed out that even taking Babai's nearest plane algorithm [2] as the signing algorithm, these two schemes are still insecure. Later, Ducas and Nguyen [6] proposed some method to analyze some countermeasures against the Nguyen-Regev attack.

By the Nguyen-Regev attack, it seems that the security of GGH type signature schemes depends heavily on the reduction algorithms. To resist such attack, at least two different reduction algorithms have been proposed. In 2008, Gentry, Peikert and Vaikuntanathan [7] presented a Gaussian sample algorithm similar to [11]. Based on such a random vector-sampling algorithm, Gentry, Peikert and Vaikuntanathan constructed a signature scheme, with a short trap-door basis as the private key and a

long basis as the public key. Since the lattice vectors outputted by the new sampling algorithm do not reveal the trap-door, the signature scheme of Gentry, Peikert and Vaikuntanathan can be proved to be secure under the chosen message attack (CMA).

In 2008, Plantard, Susilo, and Win [17] proposed another signature scheme at PKC'08 to resist the Nguyen-Regev attack. They employed a special type of lattices as the good basis which has a basis that can be written into the sum of a diagonal matrix and a ternary random matrix. With such a basis, they proposed a reduction algorithm to reduce any vector into a small cube. Since the cube is public and it seems hard to recover the private basis from the cube, the authors claimed that their scheme can resist the Nguyen-Regev attack well.

As pointed out by Plantard, Susilo, and Win, since their reduction algorithm is deterministic, the scheme may suffer some potential side channel attacks. To make the scheme more secure, they modified their reduction algorithm to be randomized.

In this paper, we show that the randomized version of the PSW signature scheme is insecure under the CMA model. Simply speaking, note that when we query the signing oracle with the single message m for many times, we will usually obtain different signature vectors $w_1, w_2, \cdots, w_k$ with $k \geq 2$. Denote by $\mathcal{H}(m)$ the hash vector of the message m. Note that, in the PSW scheme, the difference $w_i - \mathcal{H}(m)$, $1 \leq i \leq k$ are all in the given lattice. It is easy to see that $w_i - w_j$, $1 \leq i < j \leq k$ are all in the lattice. Note that each signature w_i is contained in a relatively small cube, then their difference vectors $w_i - w_j$ are relatively short. Once we obtain many such difference vectors, the $\mathbb{Z}$-linear combinations of these vectors will span the given lattice with high probability. By using the lattice reduction algorithms such as LLL [12] and BKZ [18, 4] to these short difference vectors, we could obtain a much shorter basis, which may leak the good basis in this signature scheme. In fact, we find that for dimension less than 400, BKZ-20 will recover all or partial rows of the good basis in our experiments.

To fix the randomized version of the PSW signature scheme, we will give two methods as presented in [7]. The first method is to store the message-signature pairs locally. When signing a message, we first check whether the message is in storage or not. If the message is in storage, we output the stored corresponding signature, otherwise, we apply the randomized reduction algorithm to generate a signature. The second method is using the randomized reduction algorithm to generate the signature for

the hash value of a message and some additional random number instead of the hash value of just the message.

Roadmap. The remainder of the paper is organized as follows. First we present some notations and preliminaries on lattices and hard problems in Section 2. Then we describe the Plantard, Susilo, and Win signature scheme in Section 3. Finally we describe our attacks and some experimental results in detail in Section 4, and some strategies to fix the randomized version of PSW signature scheme are discussed in Section 5.

2   Preliminaries

Denote by $\mathbb{R}$, $\mathbb{Z}$ the real number field and the integer ring respectively. For a vector $\boldsymbol{v} = (v_1, v_2, \cdots, v_n) \in \mathbb{R}^n$, denote by v_i its i-th component and denote by $\|\boldsymbol{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$ its length.

2.1   Lattices

A lattice Λ is a discrete subgroup of $\mathbb{R}^n$. Equivalently, a lattice is a $\mathbb{Z}$-linear combinations of m linearly independent vectors in $\mathbb{R}^n$. The set of these linearly independent vectors is called a basis of Λ. Given a matrix $\boldsymbol{B} \in \mathbb{Z}^{m \times n}$, we denote by $\Lambda(\boldsymbol{B})$ the lattice spanned by the row vectors of $\boldsymbol{B}$. That is,

$$\Lambda(\boldsymbol{B}) = \Big\{ \sum_{i=1}^{m} x_i \boldsymbol{b}_i | x_i \in \mathbb{Z}, 1 \leq i \leq m \Big\},$$

where $\boldsymbol{b}_i$ is the i-th row of $\boldsymbol{B}$. If the rows of $\boldsymbol{B}$ are linearly independent, we call $\boldsymbol{B}$ a basis of $\Lambda(\boldsymbol{B})$. For a basis $\boldsymbol{B}$, we denote by $\det(\Lambda(\boldsymbol{B}))$ the determinant of the lattice $\Lambda(\boldsymbol{B})$ as $\sqrt{\det(\boldsymbol{B}\boldsymbol{B}^T)}$.

A lattice $\Lambda(\boldsymbol{B})$ may have many bases. If $\boldsymbol{B}$ is a nonsingular square matrix with all entries in $\mathbb{Z}$, then $\Lambda(\boldsymbol{B})$ has a special basis in Hermite Normal Form. In general, a nonsingular square matrix $\boldsymbol{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ is in Hermite Normal Form if

1) $h_{ij} = 0$ for $1 \leq j < i \leq n$;
2) $h_{ii} > 0$ for $1 \leq i \leq n$;
3) $0 \leq h_{ij} < h_{jj}$ for $1 \leq i < j \leq n$.

Hermite Normal Form of any integer matrix can be computed in polynomial time, and Micciancio [14] suggested publishing the Hermite Normal Form as the public key which will improve the security of some lattice-based cryptosystems.

2.2 Lattice problems and algorithms

In lattice theory, the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) are two famous computational problems which have been proved to be NP-hard [3, 1]. Given a lattice basis $\boldsymbol{B} \in \mathbb{Z}^{m \times n}$, the shortest vector problem aims to find a nonzero shortest vector in $\Lambda(\boldsymbol{B})$, and the closest vector problem aims to find the closest vector to a target vector $\boldsymbol{t} \in \mathbb{Z}^n$. We denote by $\lambda_1(\Lambda(\boldsymbol{B}))$ the length of the shortest nonzero lattice vectors in the lattice $\Lambda(\boldsymbol{B})$.

The approximation versions of SVP and CVP are usually used to evaluate the security for lattice-based schemes. For the approximation of SVP, we need to find a lattice vector $\boldsymbol{v}$ such that $\|\boldsymbol{v}\| \leq \gamma \lambda_1$, and for the approximation of CVP, our aim is to find a lattice vector $\boldsymbol{w}$ satisfying $\|\boldsymbol{w} - \boldsymbol{t}\| \leq \gamma \min_{\boldsymbol{v} \in \Lambda(\boldsymbol{B})} \|\boldsymbol{v} - \boldsymbol{t}\|$ with $\gamma \geq 1$.

Some polynomial-time algorithms have been presented to solve approximate SVP and approximate CVP with exponentially large factor γ, such as LLL [12], BKZ [18, 4] for the approximate SVP and Babai's nearest plane algorithm [2] for approximate CVP.

LLL algorithm is a polynomial-time lattice reduction algorithm which was presented in [12]. An important property of this algorithm is the output vectors are relatively short. Furthermore, in practice, the output of LLL algorithm is much better than the theoretical analysis.

Blockwise Korkine-Zolotarev (BKZ) algorithm [18, 4] is also a widely used lattice reduction algorithm in the analysis for lattice-based cryptosystems. In general, BKZ algorithm has an additional parameter $\beta \geq 2$ as the block size. In the process of BKZ algorithm, a subalgorithm which finds the shortest vector of the projective lattice with dimension β is called at each iteration. Generally speaking, BKZ algorithm will cost more time than LLL, but the output will be much shorter than that of LLL when β becomes larger.

3 The PSW digital signature scheme

In PKC'08, Plantard, Susilo, and Win [17] proposed a new digital signature based on CVP_∞, which was claimed to be a countermeasure against the Nguyen-Regev attack.

3.1 The original signature scheme

The original PSW signature scheme consists of three main steps as the following:

Setup

1. Choose an integer n.
2. Compute a random matrix $\boldsymbol{M} \in \{-1, 0, 1\}^{n \times n}$.
3. Compute $d = \lfloor 2\rho(\boldsymbol{M}) + 1 \rfloor$ and $\boldsymbol{D} = dI_n$, where $\rho(\boldsymbol{M})$ is the maximum of the absolute value of the eigenvalues of $\boldsymbol{M}$.
4. Compute the Hermite Normal Form $\boldsymbol{H}$ of the basis $\boldsymbol{D} - \boldsymbol{M}$.
5. The public key is $(\boldsymbol{D}, \boldsymbol{H})$, and the secret key is $\boldsymbol{M}$.

To sign a message $\boldsymbol{m} \in \{0, 1\}^*$, one does the following.

Sign

1. Compute the vector $\boldsymbol{v} = \mathcal{H}(\boldsymbol{m}) \in \mathbb{Z}^n$ where $\mathcal{H}$ is a hash function which maps $\boldsymbol{m}$ to $\{\boldsymbol{x} \in \mathbb{Z}^n | |x_i| < d^2, 1 \leq i \leq n\}$.
2. By Algorithm 1, compute $\boldsymbol{w}$ as the signature of $\boldsymbol{m}$.

Algorithm 1 Signing algorithm

Input: A vector $\boldsymbol{v} \in \mathbb{Z}^n$, the matrix $\boldsymbol{D}$ and $\boldsymbol{M}$ obtained in the Setup step.
Output: A vector $\boldsymbol{w} \in \mathbb{Z}^n$ such that $\boldsymbol{w} \equiv \boldsymbol{v} \pmod{\Lambda(\boldsymbol{D} - \boldsymbol{M})}$ and $|w_i| < d$ for all $i = 1, 2, \cdots, n$.

1: $\boldsymbol{w} \leftarrow \boldsymbol{v}$
2: $i \leftarrow 1$
3: $k \leftarrow 0$
4: **while** $k < n$ **do**
5:    $k \leftarrow 0$
6:    $q \leftarrow \lceil \frac{w_i}{d} \rfloor$;
7:    $w_i \leftarrow w_i - qd$
8:    **for** $j \leftarrow 1$ to n **do**
9:        $w_{i+j \bmod n} \leftarrow w_{i+j \bmod n} + q\boldsymbol{M}_{i, i+j \bmod n}$
10:        $|w_{i+j \bmod n}| < d$
11:        $k \leftarrow k + 1$
12:    **end for**
13:    $i \leftarrow i + 1 \bmod n$
14: **end while**
15: **return** $\boldsymbol{w}$

To verify a message-signature pair $(\boldsymbol{m}, \boldsymbol{w})$, one does the following.

Verify

1. Check if $|w_i| < d$, $1 \leq i \leq n$.
2. Compute the vector $\mathcal{H}(\boldsymbol{m}) \in \mathbb{Z}^n$.
3. Check if the vector $\mathcal{H}(\boldsymbol{m}) - \boldsymbol{w}$ is in the lattice of basis $\boldsymbol{H}$.

3.2 The randomized version of PSW signature scheme

As pointed out by Plantard, Susilo, and Win, since the reduction algorithm is deterministic, the original PSW scheme may suffer some potential side channel attacks. To resist the potential side channel attacks, they suggest using the following randomized algorithm (Algorithm 2) as the signing algorithm.

Algorithm 2 Randomized signing algorithm

Input: A vector $\boldsymbol{v} \in \mathbb{Z}^n$, the matrix $\boldsymbol{D}$ and $\boldsymbol{M}$ obtained in the Setup step.
Output: A vector $\boldsymbol{w} \in \mathbb{Z}^n$ such that $\boldsymbol{w} \equiv \boldsymbol{v} \pmod{\Lambda(\boldsymbol{D} - \boldsymbol{M})}$ and $|w_i| < d$ for all
    $i = 1, 2, \cdots, n$.

1: $\boldsymbol{w} \leftarrow \boldsymbol{v}$
2: $i \xleftarrow{\$} \{1, 2, \cdots, n\}$
3: $k \leftarrow 0$
4: **while** $k < n$ **do**
5:     $k \leftarrow 0$
6:     $q \leftarrow \lceil \frac{w_i}{d} \rfloor$;
7:     $w_i \leftarrow w_i - qd$
8:     **for** $j \leftarrow 1$ to n **do**
9:         $w_{i+j \bmod n} \leftarrow w_{i+j \bmod n} + q\boldsymbol{M}_{i,i+j \bmod n}$
10:         $|w_{i+j \bmod n}| < d$
11:         $k \leftarrow k + 1$
12:     **end for**
13:     $i \leftarrow i + 1 \bmod n$
14: **end while**
15: **return** $\boldsymbol{w}$

---

4 The chosen message attack against the randomized version of PSW scheme

4.1 Key idea of our chosen message attack

As we can see, in the randomized version of the PSW signature scheme, the signature vectors for the same message may not be unique. Therefore, in the CMA model, if we query the randomized signing oracle with the same message $\boldsymbol{m}$, we may obtain different signature vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_k$ where $k \geq 2$. Note that $\boldsymbol{w}_i - \mathcal{H}(\boldsymbol{m})$, $1 \leq i \leq k$ are all in the lattice, and so are their difference vectors

$$(\boldsymbol{w}_i - \mathcal{H}(\boldsymbol{m})) - (\boldsymbol{w}_j - \mathcal{H}(\boldsymbol{m})) = \boldsymbol{w}_i - \boldsymbol{w}_j,$$

where $1 \leq i \leq j \leq k$.

Since each component of $\boldsymbol{w}_i$ is in $(-d, d)$, we know that each component of $\boldsymbol{w}_i - \boldsymbol{w}_j$ is in $(-2d, 2d)$. Since $d \in \Theta(\sqrt{n})$ as stated in [17], the lattice vectors $\boldsymbol{w}_i - \boldsymbol{w}_j$'s are very short.

Once we obtain many such short difference vectors, the $\mathbb{Z}$-linear combinations of these vectors will span the lattice $\Lambda(\boldsymbol{D} - \boldsymbol{M})$. By using the lattice reduction algorithms such as LLL and BKZ to the set of short generators, we expect to obtain a much shorter basis, which may leak the private key.

We present the framework of our attack as the following:

1. Generate some messages $\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots$ randomly;
2. For any message $\boldsymbol{m}_j \in \{\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots\}$, querying the signing oracle for several times to obtain many different signatures $\{\boldsymbol{w}_{j1}, \boldsymbol{w}_{j2}, \cdots, \boldsymbol{w}_{jk}\}$ with $k \geq 2$;
3. Collect enough difference vectors $\boldsymbol{w}_{ji} - \boldsymbol{w}_{j1}$'s such that they can span the lattice $\Lambda(\boldsymbol{D} - \boldsymbol{M})$. Denote by $\boldsymbol{L}$ the set of these $\boldsymbol{w}_{ji} - \boldsymbol{w}_{j1}$'s;
4. Use lattice basis reduction algorithm to $\boldsymbol{L}$ to output a square matrix $\boldsymbol{LL}$, and expect to obtain some information about the private key.

4.2   Our strategy to collect the difference vectors

To collect the difference vectors, we have to decide how many messages we will choose in Step 1 and how many signatures for one message we will query with the oracle in Step 2. Below we give a very simple but efficient strategy, that is, for one message we query as many different signatures as possible and we choose as few messages as possible to satisfy Step 3.

Note that for every message, the signing algorithm (Algorithm 2) will generate at most n different signatures since there are n choices for the index i. Assume there were exactly n different signatures, then it is natural to ask how many times we query the signing oracle to collect all these signatures. Since every signature is uniformly randomly returned by the oracle, by the classical result for Coupon Collector's Problem [15], it can be easily concluded that the expectation of this number is

$$n(1 + \frac{1}{2} + \cdots + \frac{1}{n}) = n \ln n + \gamma n + \frac{1}{2} + O(\frac{1}{n}),$$

where $\gamma \approx 0.5772156649$ is the Euler's constant. Hence, we can query one message for $\lceil n \log n \rceil$ times, and then we know that the probability of collecting all the n signatures is greater than $1 - n^{-\frac{1}{\ln 2} + 1}$ [15]. When $n \geq 100$, this value is greater than 0.85, which is acceptable.

Therefore, in our attack we query $\lceil n \log n \rceil$ signatures for each message, and choose random messages until we collect enough difference vectors, then applying LLL and BKZ to obtain a short basis for the lattice.

We present the attack as Algorithm 3.

Algorithm 3 Chosen message attack against the randomized version of PSW scheme

---

Input: The public key $\boldsymbol{H}$, the randomized signing oracle $\mathcal{O}$ and a message generator $\mathcal{G}$ to generate the messages randomly.
Output: A set of short basis for $\Lambda(\boldsymbol{H})$.
 1: Let $\boldsymbol{LL}$ be a zero matrix of $n \times n$
 2: **while** $\det \boldsymbol{LL}/\det \boldsymbol{H} \;! = 1$ and $\det \boldsymbol{LL}/\det \boldsymbol{H} \;! = -1$ **do**
 3:     $W = \{\}$
 4:     $\boldsymbol{m} \leftarrow \mathcal{G}$
 5:     **for** $i \leftarrow 1$ to $\lceil n \log n \rceil$ **do**
 6:         $\boldsymbol{w} \leftarrow \mathcal{O}(\boldsymbol{m})$
 7:         If $\boldsymbol{w}$ is not in W, append $\boldsymbol{w}$ to W
 8:     **end for**
 9:     Collect all $\boldsymbol{w}_1 - \boldsymbol{w}_i$, $1 \leq i \leq |W|$ to append to the matrix $\boldsymbol{LL}$
10:     $\boldsymbol{LL} \leftarrow$ the last n rows of $LLL(\boldsymbol{LL})$ (since LLL algorithm puts linearly independent vectors in the last rows)
11: **end while**
12: $\boldsymbol{B} \leftarrow LatticeReduction(\boldsymbol{LL})$
13: Check whether $\boldsymbol{B}$ leaks the private key or not.

4.3 Experimental Results

In our experiments, we used SageMath 7.5.1 [21] to implement our attacks, and the LLL's parameter is set to the default value. For BKZ algorithm, we set the parameter "algorithm" as "NTL" to call the NTL library [20] to implement this algorithm. All experiments were run on a machine with Intel(R) Xeon(R) CPU E5-2620 v4 @2.1GHz.

We chose the dimension n to be 200, 300, 400, and for any dimension we chose 5 randomized generated instances. For the lattice reduction algorithms, we used LLL algorithm, BKZ-10, and BKZ-20 respectively. The results are listed in Table 1.

Table 1. Experimental Results for Our Attack

dim	200					300					400				
num	3	2	3	2	3	3	2	2	2	2	3	4	3	2	3
LLL	A	P(22)	A	A	P(32)	N	N	N	N	N	N	N	N	N	N
BKZ-10	A	A	A	A	A	N	N	P(3)	N	N	N	N	N	N	N
BKZ-20	A	A	A	A	A	A	N	A	P(4)	P(22)	N	P(2)	N	N	N

[1] dim: The dimension of the lattice $\Lambda(\boldsymbol{D}-\boldsymbol{M})$;

[2] num: The number of messages we need to span the lattice $\Lambda(\boldsymbol{D}-\boldsymbol{M})$;

[3] N: The lattice reduction algorithm can not recover any rows of the matrix $\boldsymbol{D}-\boldsymbol{M}$;

[4] A: The lattice reduction algorithm can recover all rows of the matrix $\boldsymbol{D}-\boldsymbol{M}$;

[5] P: The lattice reduction algorithm can recover partial rows of the matrix $\boldsymbol{D}-\boldsymbol{M}$, and the number in the bracket is the number of rows we recovered.

We would like to point out a natural attempt to recover the rows of $\boldsymbol{D}-\boldsymbol{M}$ is by applying lattice basis reduction algorithm on the public key $\boldsymbol{H}$ directly, since every row of $\boldsymbol{D}-\boldsymbol{M}$ is very short. However, for just dimension $n = 165$ in our experiments, we could not recover any row of $\boldsymbol{D}-\boldsymbol{M}$ when we even applied BKZ-20 on the public key $\boldsymbol{H}$ directly.

In contrast, with our attack, for the dimension $n = 200$, LLL algorithm could recover all (or partial) rows of $\boldsymbol{D}-\boldsymbol{M}$, and BKZ-10 could recover all the rows of $\boldsymbol{D}-\boldsymbol{M}$ for our instances. For the dimension $n = 300$, we could recover all rows of $\boldsymbol{D}-\boldsymbol{M}$ in 2 instances and partial rows in 2 instances when BKZ-20 was used.

For the dimension $n = 400$, we just obtain partial rows in $\boldsymbol{D}-\boldsymbol{M}$ for only one instance with BKZ-20 algorithm. Employing BKZ algorithm with bigger blocksize, we may obtain more rows.

However, we would like to point out that even only partial rows are recovered, the randomized version of the PSW signature scheme is not secure. Since the messages are all generated randomly, we may expect to recover all the rows of the matrix $\boldsymbol{D}-\boldsymbol{M}$ by repeating our attack several times.

Remark 1. Once obtaining a short basis, we can also recover the matrix $\boldsymbol{M}$ by finding some lattice vector close to $(0, \cdots, d, \cdots, 0)$. Using some strategies in [13] to solve the Bounded Distance Decoding (BDD) problem may improve our results.

Remark 2. We would like to point out that the strategy to collect the difference vectors also plays an important role in our attack. Another natural strategy is to query the signing oracle just twice for each message and collect enough difference vectors to mount the attack. However, the new stategy did not work so well as Algorithm 3. For dimension $n = 200$ and larger dimensions, we could never recover any rows of the matrix $\boldsymbol{D} - \boldsymbol{M}$ by using this strategy in our experiments.

5   Possible ways to fix the randomized version

There are two possible ways to fix the randomized version similar to the strategies in [7].

The first way is to store the message-signature pairs locally, which seems a bit impractical. In detail, once given a message $\boldsymbol{m}$, we will modify the Sign step as the following:

Sign
1. Check whether $\boldsymbol{m}$ has been signed or not.
2. If $\boldsymbol{m}$ is stored locally, return the locally stored signature $\boldsymbol{w}$ corresponding to $\boldsymbol{m}$.
3. Otherwise, use Algorithm 2 to output a signature $\boldsymbol{w}$ and store $(\boldsymbol{m}, \boldsymbol{w})$ locally.

The second way is to add some random number to the hash function. This strategy is usually used in the hash-then-sign schemes. Since the original PSW scheme has no security proof and we do not know the exact hardness of CVP_∞ over the PSW instances, we can not present some formal security proof for this fixed version, but just present it as the following:

Sign
1. Choose $\boldsymbol{r} \leftarrow \{0, 1\}^n$ at random.
2. Compute the vector $\boldsymbol{v} = \mathcal{H}(\boldsymbol{m}||\boldsymbol{r})$, where $\mathcal{H}$ maps $(\boldsymbol{m}||\boldsymbol{r})$ to the area $(-d^2, d^2)^n$.
3. Applying Algorithm 2, compute the signature $\boldsymbol{w}$.

Once given the signature $(\boldsymbol{m}, \boldsymbol{r}, \boldsymbol{w})$, we will modify the Verify step as below.

Verify
1. Check if $|w_i| < d$ for $1 \leq i \leq n$.
2. Compute the vector $\mathcal{H}(\boldsymbol{m}||\boldsymbol{r})$.
3. Check whether the vector $\mathcal{H}(\boldsymbol{m}||\boldsymbol{r}) - \boldsymbol{w} \in \Lambda(\boldsymbol{H})$ or not.

6   Conclusions

In this paper, we show that the randomized PSW signature scheme is not secure under the chosen message attack at least for dimension less than or equal to 400. However, for the scheme with bigger dimension which becomes less efficient apparently, it seems that we need the BKZ algorithm with bigger blocksize to recover the private key. In fact, our attack reveals that the storage of previous signature or the use of random nonce employed in the randomized signature scheme is crucial.

References

1. Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. pp. 10–19. STOC '98, ACM, New York, NY, USA (1998), http://doi.acm.org/10.1145/276698.276705
2. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (Mar 1986), https://doi.org/10.1007/BF02579403
3. Boas, P.V.E.: Another NP-complete problem and the complexity of computing short vectors in lattices. Math. Dept. Report 81-04. Univ. of Amsterdam (1981)
4. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Lecture Notes in Computer Science, vol. 7073, pp. 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), https://doi.org/10.1007/978-3-642-25385-0_1
5. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer, Berlin, Heidelberg (Aug 2013), https://doi.org/10.1007/978-3-642-40041-4_3
6. Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 433–450. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_27
7. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374376.1374407
8. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski, B.S. (ed.) Advances in Cryptology - CRYPTO '97: 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997. Proceedings. Lecture Notes in Computer Science, vol. 1294, pp. 112–131. Springer Berlin Heidelberg, Berlin, Heidelberg (1997), https://doi.org/10.1007/BFb0052231

9. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSign: Digital signatures using the NTRU lattice. In: Joye, M. (ed.) Topics in Cryptology - CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003 Proceedings. Lecture Notes in Computer Science, vol. 2612, pp. 122–140. Springer Berlin Heidelberg, Berlin, Heidelberg (2003), `https://doi.org/10.1007/3-540-36563-X_9`

10. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21-25, 1998 Proceedings. Lecture Notes in Computer Science, vol. 1423, pp. 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg (1998), `https://doi.org/10.1007/BFb0054868`

11. Klein, P.: Finding the closest lattice vector when it's unusually close. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 937–941. SODA '00, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2000), `http://dl.acm.org/citation.cfm?id=338219.338661`

12. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (Dec 1982), `https://doi.org/10.1007/BF01457454`

13. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Dawson, E. (ed.) Topics in Cryptology – CT-RSA 2013: The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7779, pp. 293–309. Springer, Berlin, Heidelberg (2013), `https://doi.org/10.1007/978-3-642-36095-4_19`

14. Micciancio, D.: Improving lattice based cryptosystems using the hermite normal form. In: Silverman, J.H. (ed.) Cryptography and Lattices: International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers. Lecture Notes in Computer Science, vol. 2146, pp. 126–145. Springer Berlin Heidelberg, Berlin, Heidelberg (Mar 2001), `https://doi.org/10.1007/3-540-44670-2_11`

15. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (Sep 1995), `http://doi.acm.org/10.1145/211542.606546`

16. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 271–288. Springer Berlin Heidelberg, Berlin, Heidelberg (2006), `https://doi.org/10.1007/11761679_17`

17. Plantard, T., Susilo, W., Win, K.T.: A digital signature scheme based on CVP_∞. In: Cramer, R. (ed.) Public Key Cryptography - PKC 2008: 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4939, pp. 288–307. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), `https://doi.org/10.1007/978-3-540-78440-1_17`

18. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming 66(1-3), 181–199 (Aug 1994), `https://doi.org/10.1007/BF01581144`

19. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE, Santa Fe, NM, USA (Nov 1994)

20. Shoup, V.: NTL: A library for doing number theory (2001), `http://www.shoup.net/ntl`

21. Stein, W., et al.: Sage Mathematics Software Version 7.5.1. The Sage Development Team (2017), `http://www.sagemath.org`