IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 17, 2019, accepted April 22, 2019, date of publication April 29, 2019, date of current version May 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913678

An Improved Attack on the Basic Merkle–Hellman Knapsack Cryptosystem

JIAYANG LIU[1], JINGGUO BI[2,3], AND SONGYAN XU[2]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2]Beijing Research Institute of Telemetry, Beijing 100094, China
[3]Institute for Advanced Study, Tsinghua University, Beijing 100084, China

Corresponding author: Jingguo Bi (bijingguo-001@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502269, in part by The National Key Research and Development Program of China under Grant 2017YFA0303903, and in part by the Zhejiang Province Key Research and Development Project under Grant 2017C01062.

ABSTRACT Knapsack problem is a famous NP-complete problem, which is believed to be difficult to be solved even by a quantum computer. Hence, this type of cryptosystem is a good candidate for post-quantum cryptography. Recently, many new knapsack-based cryptosystems were proposed. The basic operations of all these cryptosystems are superincreasing sequences and modular multiplications, which is the same as the basic Merkle–Hellman cryptosystem. In this paper, we revisit and present an improved version of Shamir's attack on the basic Merkle–Hellman cryptosystem, this new idea would be helpful to estimate the security of the new knapsack-based cryptosystems. The main tool of our attack is the orthogonal lattice technique. More precisely, we first obtain a sublattice containing the private key vector by calculating the orthogonal lattice of the public key vector. Combining with the necessary conditions of the equivalent keys, we can easily recover several groups of equivalent keys. The time complexity of our new attack is lower than Shamir's. The feasibility of our attack is validated by the experimental data.

INDEX TERMS LLL algorithm, Merkle-Hellman knapsack cryptosystem, orthogonal lattice, post-quantum cryptography, public-key cryptosystem.

I. INTRODUCTION

In 1978, Merkle and Hellman proposed the first public key cryptosystem based on the knapsack problem [1]. The basic scheme proposed by Merkle and Hellman was based on superincreasing sequences and modular multiplications. In 1982, Shamir [2] proposed a polynomial time algorithm to break the basic Merkle-Hellman cryptosystem. This attack relied on the fact that the modular multiplication method does not disguise completely the easy knapsack which is the basis of the construction. For the general knapsack problem, the low-density attack was proposed [3], [4], which did not assume any particular structure in the knapsack. It is well known that knapsack problem is NP-complete, and accordingly it is considered to be quite hard in the worst case even by a quantum computer. In 2016, NIST announced the post-quantum standardization process to look towards standardization of post-quantum cryptography. Knapsack-based cryptosystem can be a good candidate for post-quantum cryptography. In recent years, new kinds of knapsack-based cryptosystems have been proposed [11]–[13]. In this paper, we revisit Shamir's attack on the basic Merkle-Hellman cryptosystem and present an improved one. This new idea would be helpful in the design of the new knapsack-based cryptosystems.

Lattices are classical objects of number theory, which have many applications in mathematics and computer science. A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$ and is the set $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of all integral linear combinations of $\mathbf{b}_i$. In this paper, we present a new attack on the Merkle-Hellman cryptosystem using the lattice theory. More precisely, our attack is based on the orthogonal lattice technique [5]. Considering the natural linear relationships between the secret key vector and the public key vector for knapsack-based cryptosystems, we have presented an equivalent key attack against a knapsack public-key cryptosystem [14]. For the Merkle-Hellman cryptosystem, we can obtain a sublattice which contains the secret key vector by calculating the orthogonal lattice $\mathcal{L}$ of the public key vector. Notice that the dimension of $\mathcal{L}$ is only 2, one can easily recover the coefficients of the secret key in $\mathcal{L}$ by considering the necessary conditions of the equivalent key. Under Lagarias' analysis [6], the time complexity of Shamir's algorithm is $O(n^{g+10}L\log L)$, which g is the number of variables during solving the integer program and L

The associate editor coordinating the review of this manuscript and approving it for publication was Rasheed Hussain.

is the input length. The most time-consuming part of our attack is calculating the orthogonal lattice $\mathcal{L}$ of the public key vector by using the classical LLL algorithm [7]. The time complexity is $O(n^6L^3)$, which can be reduced further if one invoking the L^2 algorithm [8]. In the basic Merkel-Hellman cryptosystem, taking $g = 5, L = O(n)$, our method obtains a $O(n^7)$ speed-up compared with Shamir's algorithm if one only invoke the classical LLL algorithm.

It is worth mentioning that we also present a clever idea to improve the core step of the original Shamir's attack. In Shamir's attack, the core step is to recover a intermediate parameter by solving $O(2n\log_2 n)$ times integer programming with g variables, the time complexity of solving the integer programming is $O(g^{9g}L\log L)$ [9], [10]. In this paper, we first propose a method to generate a lattice with small dimension, and one can recover this intermediate parameter by invoking the lattice reduction algorithm, which is much efficient than Shamir's attack. One can obtain a minor improvement of Shamir's attack by only taking replace of the core step by our method.

We organized the paper as follows. Section 2 shows some backgrounds about lattices and briefly introduces Merkle-Hellman cryptosystem. In Section 3, we present the new attack to recover the equivalent private keys. And we provide the experimental data of our attack. Finally, we conclude the paper in Section 4.

II. PRELIMINARY

A. LATTICE

Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. A lattice $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$: there exist $n(\leq m)$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ s.t. $\mathcal{L}$ is the set $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of all integral linear combinations of $\mathbf{b}_i$,

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}.$$

Then the matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a *basis* of $\mathcal{L}$ and n is the *rank* (or *dimension*) of $\mathcal{L}$. The (co-)volume of $\mathcal{L}$ is $\mathrm{vol}(\mathcal{L}) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$ for any basis $\mathbf{B}$ of $\mathcal{L}$, where $\mathbf{B}^T$ denotes $\mathbf{B}$'s transpose. If $\mathbf{B}$ is square, then $\mathrm{vol}(\mathcal{L}) = |\det \mathbf{B}|$, and if $\mathbf{B}$ is further triangular, then $\mathrm{vol}(\mathcal{L})$ is simply the product of the diagonal entries of $\mathbf{B}$ in absolute value.

Definition 1 (Successive minima): Given a lattice $\mathcal{L}$ with rank n, the i-th minima $\lambda_i(\mathcal{L})$ is the radius of the smallest sphere centered in the origin containing i linearly independent lattice vectors, i.e., $\lambda_i(\mathcal{L}) = \inf\{r : \dim(\mathrm{span}(\mathcal{L} \cap B_n(r))) \geq i\}$, where $B_n(r)$ represents the n-dimension ball centered at the origin with radius r.

A non-zero vector with the smallest Euclidean norm in a lattice $\mathcal{L}$ is called a shortest vector of $\mathcal{L}$. The length of a shortest vector is $\lambda_i(\mathcal{L})$, $\lambda_i(\mathcal{L}) = \min_{\mathbf{v}\in\mathcal{L}\setminus\{\mathbf{0}\}} \| \mathbf{v} \|$. The Shortest Vector Problem(SVP) is a famous computational hard problem in lattice theory.

Definition 2 (SVP): Given a basis of a lattice $\mathcal{L}$, find a non-zero vector $\mathbf{u} \in \mathcal{L}$, such that $\| \mathbf{u} \| \leq \| \mathbf{v} \|$ for any vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$.

To find a short vector in a given lattice, the first polynomial algorithm is the celebrated LLL algorithm [7]: given a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of an integer lattice $L \subseteq \mathbb{Z}^m$, LLL algorithm outputs a non-zero $\vec{v} \in L$ s.t. $\|\vec{v}\| \leq 2^{\frac{n-1}{2}}\lambda_1$ in time $O(n^5\, mb^3)$ (resp. $n^3\, mb\tilde{O}(n)\tilde{O}(b)$) without (resp. with) fast integer arithmetic, where $b = \max_{1\leq i\leq n} \log \|\mathbf{b}_i\|$: strictly speaking, this vector is actually the first vector of the basis outputted by the algorithm.

Proposition 1: Let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be an LLL-reduced basis of a lattice $\mathcal{L}$. Then :

1) $vol(\mathcal{L}) \leq \prod_{i=1}^{n} \|\mathbf{b}_i\| \leq 2^{\frac{n(n-1)}{4}} vol(\mathcal{L})$.
2) $\|\mathbf{b}_1\| \leq 2^{\frac{n-1}{4}} (vol(\mathcal{L}))^{\frac{1}{n}}$.
3) $\forall 1 \leq i \leq n, \|\mathbf{b}_i\| \leq 2^{\frac{n-1}{2}} \lambda_i(\mathcal{L})$.

In this paper, we mostly use the following properties of orthogonal lattice [5].

Definition 3 (Orthogonal Lattice $\mathcal{L}^\perp$): Given a lattice $\mathcal{L} \subseteq \mathbb{Z}^m$. All bases of $\mathcal{L}$ span the same subspace of $\mathbb{Q}^m$, which we denote by $\mathbf{E}$. Let $\mathbf{F} = \mathbf{E}^\perp$ be the orthogonal vector subspace with respect to the inner product. We define the orthogonal lattice to be $\mathcal{L}^\perp = \mathbf{F} \cap \mathbb{Z}^m$. i.e. $\mathcal{L}^\perp = \{\mathbf{v} \in \mathbb{Z}^m | \mathbf{u} \in \mathcal{L}, < \mathbf{u}, \mathbf{v} >= 0\}$.

Proposition 2: A lattice $\mathcal{L} \subseteq \mathbb{Z}^m$, then $rank(\mathcal{L}) + rank(\mathcal{L}^\perp) = m$.

Theorem 1: [5] Given a basis $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ of a lattice $\mathcal{L}$ in $\mathbb{Z}^m$, there is a deterministic polynomial time algorithm with respect to the space dimension m, the lattice dimension n and any upper bound of the bit-length of the $\|\mathbf{b}_j\|$'s which computes an LLL-reduced basis of $\mathcal{L}^\perp$.

We propose the algorithm(Algorithm 1) [5] to compute the LLL-reduced basis of $\mathcal{L}^\perp$, and we use column representation for matrices in this algorithm. The core of this algorithm is choosing a suitable g to make sure the first k coordinates of the first $n-k$ vectors of the LLL-reduced basis are 0. Therefore, the last n coordinates of the first $n-k$ vectors of the LLL-reduced basis would construct the orthogonal lattice. For more details, please refer to [5].

B. BRIEF INTRODUCTION TO THE BASIC MH CRYPTOSYSTEM

Here we introduce the knapsack cryptosystem created by Merkle and Hellman in 1978 [1]. Without loss of generality, we set the upper bound of the bit length of s_1 as N.

Key generation:

1) Set $N = 100$. Create a superincreasing integer sequence $\mathbf{s} = (s_1, \ldots, s_n)^T$. For all $i = 1, \ldots, n$, s_i is chosen uniformly from the range $[(2^{i-1}-1)*2^{100} + 1, 2^{i-1}*2^{100}]$.
2) The modulus m is chosen uniformly from $[2^{201} + 1, 2^{202}-1]$. And w' is chosen uniformly from $[2, m-2]$ then divided by the greatest common divisor of w' and m to yield w.

Algorithm 1 Orthogonal Lattice $\mathcal{L}^{\perp}$

Input:A basis $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k)$ **of a lattice** $\mathcal{L}$ **in** $\mathbb{Z}^n$**.**
Output:The basis $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{n-k})$ **of** $\mathcal{L}^{\perp}$

1) **Select** $g = \lceil 2^{\frac{n-1}{2}+\frac{(n-k)(n-k-1)}{4}} \prod_{j=1}^{k} \|\mathbf{b}_k\| \rceil$**.**
2) **Compute the** $(n+k) \times n$ **integral matrix** $\widetilde{\mathbf{B}}$**.**

$$\widetilde{\mathbf{B}} = \begin{pmatrix} g \times b_{1,1} & g \times b_{1,2} & \ldots & g \times b_{1,n} \\ g \times b_{2,1} & g \times b_{2,2} & \ldots & g \times b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ g \times b_{k,1} & g \times b_{k,2} & \ldots & g \times b_{k,n} \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & 1 \end{pmatrix} \quad (1)$$

3) **Compute an LLL-reduced basis** $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ **of the lattice spanned by** $\widetilde{\mathbf{B}}$**.**
4) **Output an LLL-reduced basis of** $\mathcal{L}^{\perp}$**,** $\mathbf{y}_j \in \mathbb{Z}^n$**,** $\mathbf{y}_j$ **is the vector of the last** n **coordinates of** $\mathbf{x}_j$**,** $1 \le j \le n-k$**.**

3) Compute the trapdoor knapsack vector $\mathbf{a} = (a_1, \ldots, a_n)^T$, where $a_i = w * s_i \mod m$, for $i = 1, \ldots, n$.
4) Secret key: $s_1, s_2, \ldots, s_n, m, w$. Public key: $a_1, a_2, \ldots, a_n$.

Encryption:

1) Messages are encrypted by first being broken into blocks $(x_1, \ldots, x_n)$ of n binary digits, $x_i \in \{0, 1\}, i = 1, \ldots, n$.
2) Compute the ciphertext $C = \sum_{i=1}^{n} a_i x_i$.

Decryption:

1) Define C' as $C' = \sum_{i=1}^{n} s_i x_i$. Compute $C' = w^{-1}C \mod m$, where w^{-1} is the inverse element of w with the modulus m.
2) The plaintext $x_1, \ldots, x_n$ can be recovered by $C' = \sum_{i=1}^{n} s_i x_i$, because $\mathbf{s}$ is a superincreasing sequence.

C. THE RATIONALE OF SHAMIR' ATTACK

In 1982, Shamir developed a polynomial time attack to the basic Merkel-Hellman cryptosystem [2]. We omit the specific algorithm here for brevity. The rationale for the algorithm which was elaborated in [6] is as follows.

From the key generation algorithm, we have $s_i = w^{-1}a_i \mod m$, which is equivalent to the equality

$$\frac{s_i}{ma_i} = \frac{w^{-1}}{m} - \frac{k_i}{a_i}.$$

for non-negative integer $k_i, i = 1, \ldots, n$.

By explicit calculation, we obtain

$$|k_1 a_i - k_i a_1| \le 2^{-n+g} m, \quad \text{for } 1 \le i \le g. \quad (2)$$

Note that the bit lengths of k_i and a_i are about $2N$ bit, and the bit length of the right part of this equation is about $N + g$ bit. That is a very "unusual" phenomena. Shamir [2] proved that one can recover each possible solutions of Equation (2) by solving at most $O(2n \log_2 n)$ integer programmings. This is the core step in Shamir's attack. The time complexity of solving the integer programming is $O(g^{9g} L \log L)$ [9], [10]. Because the number of the variables in the integer programming is a constant integer g, the time complexity of solving the integer programming is polynomial.

For all possible solutions $(x_1^{(i)}, \ldots, x_g^{(i)})$ of the integer programming with $x_1^{(i)} = k_1$. We have

$$0 \le \frac{w^{-1}}{m} - \frac{k_1}{a_1} \le \frac{n^2}{2^n m}.$$

Examine the n^7 rationals $\frac{w_j^{-1}}{m_j} = \frac{x_1^{(i)}}{a_1} + \frac{j}{n^7 2^n m}, 1 \le j \le n^7$, in this case

$$\left|\frac{w_j^{-1}}{m_j} - \frac{k_1}{a_i}\right| \le \frac{1}{n^5 2^n m}.$$

Set (w^{-1*}, m^*) equal to (w_j^{-1}, m_j). Define λ, ϵ by $m^* = \lambda m, w^{-1*} = \lambda(w^{-1} + \epsilon), |\epsilon| \le n^{-5}2^{-n}$. It turns out that $s_i^* = s_i + \epsilon a_i$ will be a superincreasing sequence for "almost all" superincreasing sequences and hence (w^{-1*}, m^*) will then be the desired decryption pair.

Therefore, the time complexity of Shamir's attack is $O(n^{g+10} L \log L)$ [2], [6], which g is the number of variables during solving the integer program and L is the input length.

III. CRYPTANALYSIS

A. OBSERVATION OF SHORT VECTORS

Note that $a_i = w * s_i \mod m, i = 1, \ldots, n$, we have $s_i = w^{-1} * a_i \mod m, i = 1, \ldots, n$. Let $k_i \in \mathbb{Z}$ be the quotient such that $s_i = w^{-1}a_i + mk_i$. The equations imply that

$$\frac{s_1 a_2 - s_2 a_1}{m} = k_1 a_2 - k_2 a_1 \quad (3)$$

$$\frac{s_1 a_i - s_i a_1}{m} = k_1 a_i - k_i a_1. \quad (4)$$

Take $N = 100$. Consider the bit length of the first few integers of the sequence s_i, which is less than $99 + i$, it is far less than the bit length of a_i, which is about 200. We try to recover the first five integers $\frac{s_1 a_i - s_i a_1}{m}, i = 2, \ldots, 6$. By the right part of the above equations, we define $\mathbf{b_1} = (a_2, \ldots, a_6)^T$, $\mathbf{b_2} = (-a_1, \ldots, 0)^T, \ldots, \mathbf{b_6} = (0, \ldots, -a_1)^T$. Then we apply LLL algorithm on the lattice $\mathcal{L}(\mathbf{b_1}, \ldots, \mathbf{b_6})$.

$$\begin{pmatrix} a_2 & -a_1 & \cdots & 0 \\ a_3 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_6 & 0 & \cdots & -a_1 \end{pmatrix} \xrightarrow{\text{LLL}} \begin{pmatrix} 0 & d_1 & \cdots \\ 0 & d_2 & \cdots \\ \vdots & \vdots & \ddots \\ 0 & d_5 & \cdots \end{pmatrix}$$

Obviously, $\mathbf{b_1}, \ldots, \mathbf{b_6}$ is not a basis of the lattice and we can get the first short vector of the output of LLL algorithm is a zero vector. Heuristically, the second short vector of the output is the vector $(\frac{s_1a_2-s_2a_1}{m}, \frac{s_1a_3-s_3a_1}{m}, \frac{s_1a_4-s_4a_1}{m}, \frac{s_1a_5-s_5a_1}{m}, \frac{s_1a_6-s_6a_1}{m})$, and the other four vectors are much longer than the second one. The experiment results confirmed our assumption. And we note that the short vector is $(d_1, \ldots, d_5)^T$. Then we have

$$s_1a_{i+1} - s_{i+1}a_1 = d_im, \quad i = 1, \ldots, 5. \tag{5}$$

Remark 1: In most cases, the second short vector is $(-d_1, \ldots, -d_5)^T$. We can get the transformation matrix of LLL algorithm and one of k_i or $-k_i$ must appear in the matrix. Bacause $k_i > 0$, we can determine which one of k_i and $-k_i$ appears. So we can get the right vector $(d_1, \ldots, d_5)^T$ through the output as well as $k_1, \ldots, k_6$.

Remark 2: In Shamir's attack, the core step is recovering the k_1 by solving $O(2n \log_2 n)$ times integer programming with g variables, which is super-exponential of g. In this subsection, we can recover k_1 by invoking LLL algorithm. Note that the dimension of the lattice is very small, the time complexity is negligible. In this point of view, we obtain a minor improvement of Shamir's attack by only taking replace of the core step by our method.

B. COMPUTE THE ORTHOGONAL LATTICE

Consider that $a_i = ws_i \mod m$ for $i = 1, \ldots, n$. Let $h_i \in \mathbb{Z}$ be the quotient such that $a_i = ws_i + mh_i$. Let $\mathbf{a} = (a_1, \ldots, a_n)^T, \mathbf{s} = (s_1, \ldots, s_n)^T, \mathbf{h} = (h_1, \ldots, h_n)^T$, we have

$$\mathbf{a} = w\mathbf{s} + m\mathbf{h} \tag{6}$$

Let $\mathbf{t}$ be a vector in $\mathcal{L}^{\perp}(\mathbf{a})$, then we have $w < \mathbf{s}, \mathbf{t} > + m < \mathbf{h}, \mathbf{t} > = < \mathbf{a}, \mathbf{t} > = 0$. In this subsection, we will show that if $< \mathbf{a}, \mathbf{t} > = 0$, then $\|\mathbf{t}\|$ will be relatively longer in $\mathcal{L}^{\perp}(\mathbf{a})$ or orthogonal to the vectors $\mathbf{s}, \mathbf{h}$.

Constructing a lattice

$$\mathcal{L}_1 = \{\mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}) | < \mathbf{s}, \mathbf{t} > = 0\}.$$

Theorem 2: There exists a vector $\mathbf{t}_0 \in \mathcal{L}^{\perp}(\mathbf{a})$, such that $< \mathbf{s}, \mathbf{t_0} > \neq 0$, then $\mathcal{L}^{\perp}(\mathbf{a}) = \mathcal{L}_1 \oplus \mathcal{L}(\mathbf{t}_0)$.

Proof: Because $\gcd(m, w) = 1$, we have $m | < \mathbf{s}, \mathbf{t} >$. We can take $\mathbf{t_1} \in \mathcal{L}^{\perp}(\mathbf{a}), < \mathbf{s}, \mathbf{t_1} > = b_1m, b_1 \in \mathbb{Z}, b_1 \neq 0$. If $\exists \mathbf{t}_2 \in \mathcal{L}^{\perp}(\mathbf{a}), < \mathbf{s}, \mathbf{t_2} > = b_2m$, there greatest common divisor $\gcd(|b_1|, |b_2|) = 1$. Set $b_0 = 1$. Else if $\forall \mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}), < \mathbf{s}, \mathbf{t} > = bm$, we get that $\gcd(|b_1|, |b|) \neq 1$. In this case, because $|b_1|$ has limited divisors, note that $b_0 = \min_b\{\gcd(|b_1|, |b|)\}$. After computing b_0 by Euclidean algorithm, we can get $\mathbf{t}_0$ such that $\mathbf{t}_0 \in \mathcal{L}^{\perp}(\mathbf{a}), < \mathbf{s}, \mathbf{t_0} > = b_0m, b_0 \neq 0$.

Now we only need to prove that $\mathcal{L}^{\perp}(\mathbf{a}) = \mathcal{L}_1 \oplus \mathcal{L}(\mathbf{t}_0)$. Obviously, $\mathcal{L}_1 = \{\mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}) | < \mathbf{s}, \mathbf{t} > = 0\}$ is a sublattice of $\mathcal{L}^{\perp}(\mathbf{a})$. The basis of $\mathcal{L}_1$ can be extended to the basis of $\mathcal{L}^{\perp}(\mathbf{a})$. $\forall \mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}), \mathbf{t} \notin \mathcal{L}_1$, we know $< \mathbf{s}, \mathbf{t} > \neq 0$. Note that $< \mathbf{s}, \mathbf{t} > = bm$. We have $< \mathbf{s}, b_0\mathbf{t} > = < \mathbf{s}, b\mathbf{t_0} >, b_0\mathbf{t} - b\mathbf{t_0} \in \mathcal{L}_1$. If $b_0 = 1, \mathbf{t} - b\mathbf{t_0} \in \mathcal{L}_1$. If $b_0 \neq 1$, we can prove that $\gcd(b_0, |b|) = b_0$. Because it is contradictory to the definition of $b_0 = \min_b\{\gcd(|b_1|, |b|)\}$ if $\gcd(b_0, |b|) \neq b_0$. In this case, $\mathbf{t} - \frac{b}{b_0}\mathbf{t_0} \in \mathcal{L}_1, \frac{b}{b_0} \in \mathbb{Z}$.

To summarise, $\forall \mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}), \mathbf{t} \notin \mathcal{L}_1$, we can get $\mathbf{t} - \frac{b}{b_0}\mathbf{t_0} \in \mathcal{L}_1, \frac{b}{b_0} \in \mathbb{Z}$. That is, $\forall \mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a}), \mathbf{t} \in \mathcal{L}_1 \oplus \mathcal{L}(\mathbf{t}_0)$. On the other hand, $\forall \mathbf{t} \in \mathcal{L}_1 \oplus \mathcal{L}(\mathbf{t}_0)$, we can get $\mathbf{t} \in \mathcal{L}^{\perp}(\mathbf{a})$ directly because $\mathcal{L}_1 \subset \mathcal{L}^{\perp}(\mathbf{a})$ and $\mathbf{t}_0 \in \mathcal{L}^{\perp}(\mathbf{a})$. In sum, we have proved that $\mathcal{L}^{\perp}(\mathbf{a}) = \mathcal{L}_1 \oplus \mathcal{L}(\mathbf{t}_0)$. □

We compute the LLL-reduced basis of $\mathcal{L}^{\perp}(\mathbf{a})$ by Algorithm 1 and note the basis as $(\mathbf{t}_1, \ldots, \mathbf{t}_{n-1})$. Rearrange these $n-1$ vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{n-1}$ in ascending order of their lengths, we know that only one vector of the basis satisfy $< \mathbf{s}, \mathbf{t} > \neq 0$ according to Theorem 2. Now we claim that $< \mathbf{s}, \mathbf{t}_{n-1} > \neq 0$.

Remark 3: Let $\mathbf{a}^j = (a_1, \ldots, a_j)^T, \mathbf{s}^j = (s_1, \ldots, s_j)^T, j = n, \ldots, n_0$, take $n_0 = \lfloor \frac{n}{2} \rfloor$. We know Theorem 2 still holds for all $\mathbf{a}^j, \mathbf{s}^j$. If $\|\mathbf{t}^j\| \leq 2^{n-j+1}, | < \mathbf{s}^j, \mathbf{t}^j > | \leq \|\mathbf{s}^j\|\|\mathbf{t}^j\| < 2^{N+j}2^{n-j+1} < m, < \mathbf{s}^j, \mathbf{t}^j > = 0$. Because $< \mathbf{s}^j, \mathbf{t_0}^j > \neq 0$, we have $\|\mathbf{t}_0^j\| > 2^{n-j+1}$. From experiment results, we know that when $n = 100, N = 100$, the bit length of $\|\mathbf{t}_i^j\|, i \neq j-1$ in each basis is about 5. When $j < n-4$, we can easily distinguish $\mathbf{t}_0^j$ from $\mathbf{t}_i^j$. That is, $\|\mathbf{t}_{j-1}^j\| \gg \|\mathbf{t}_i^j\|, i \neq j-1, \mathbf{t}_{j-1}^j = \mathbf{t}_0^j$. When $j \geq n-4$, the bit length of $\|\mathbf{t}_{j-1}^j\|$ is only a little longer than others. In this case, we need to consider other properties rather that the bit lengths of the norms.

Particularly, when $|t_n| \leq 4, |s_nt_n| \leq 2^{n+N+1} < m$. And we claim that under stochastic assumption the output of LLL-reduced basis have the property that the bit lengths of t_i are almost equal. That is, when $|t_n| \leq 4, | < \mathbf{s}, \mathbf{t} > | \approx |s_nt_n| < m$. The experiment data show that $|t_{n-1,n}| > 4$ and $|t_{i,n}| \leq 4, i = 1, \ldots, n-2$. Then from Theorem 2, we can get $< \mathbf{s}, \mathbf{t}_0 > \neq 0$, then $\mathbf{t}_{n-1} = \mathbf{t}_0, < \mathbf{s}, \mathbf{t}_{n-1} > \neq 0$.

It is easy to compute a LLL-reduced basis $(\mathbf{u}, \mathbf{v})$ of $\mathcal{L}_1^{\perp} = \mathcal{L}^{\perp}(\mathbf{t}_1, \ldots, \mathbf{t}_{n-2})$ by using Algorithm 1 one more time.

According to Theorem 2 and $\mathbf{t}_{n-1} = \mathbf{t}_0$, we confirm that $< \mathbf{s}, \mathbf{t}_i > = 0, < \mathbf{h}, \mathbf{t}_i > = 0, i = 1, \ldots, n-2, \mathbf{s}, \mathbf{h} \in \mathcal{L}_1^{\perp}$.

So the (equivalent) private keys $\mathbf{s}$ satisfy the following equations

$$\mathbf{s} = x\mathbf{u} + y\mathbf{v}, \tag{7}$$

$$\mathbf{h} = x'\mathbf{u} + y'\mathbf{v}, \tag{8}$$

where $x, y, x', y' \in \mathbb{Z}$.

C. RECOVER THE EQUIVALENT PRIVATE KEYS

The decryption of the scheme implies that

$$w^{-1}C = w^{-1}\sum_{i=1}^{n} a_ix_i \mod m$$

$$= \sum_{i=1}^{n} s_ix_i \mod m.$$

So the foremost condition to guarantee the decryption successfully is that $s_1, s_2, \ldots, s_n$ is a positive superincreasing sequence and $\sum_{i=1}^{n} s_i x_i \mod m = \sum_{i=1}^{n} s_i x_i$, i.e. $m > \sum_{i=1}^{n} s_i$.

In conclusion, to recover the a group of equivalent private keys $s_1, s_2, \ldots, s_n, m, w$, we only need to find out the keys which satisfy these conditions as follows:

1) $a_i = w s_i \mod m$, for $i = 1, 2, \ldots, n$.
2) $\gcd(m, w) = 1$ and $m > \sum_{i=1}^{n} s_i$.
3) $s_1, s_2, \ldots, s_n$ is a positive superincreasing sequence.

Consider Equation (7), the third condition can be translated into computing a narrow feasible region between a group of extremely close values for x, y. That is,

$$\begin{aligned} s_1 &> 0 \\ s_2 &> s_1 \\ s_i &> \sum_{j=1}^{i-1} s_j, \ i = 2, \ldots, n \\ &\Downarrow \\ u_1 x + v_1 y &> 0 \qquad (9) \\ (v_2 - v_1) y &> (u_1 - u_2) x \\ (v_i - \sum_{j=1}^{i-1} v_j) y &> (\sum_{j=1}^{i-1} u_j - u_i) x. \qquad (10) \end{aligned}$$

We combine Equation (7) with Equation (5) and take $i = 1$, then we can get

$$(u_1 a_2 - u_2 a_1) x + (v_1 a_2 - v_2 a_1) y = d_1 m. \qquad (11)$$

From this equation, the latter part of the second condition can be added to restrict the feasible region as well.

$$\frac{(u_1 a_2 - u_2 a_1) x + (v_1 a_2 - v_2 a_1) y}{d_1} > \sum_{j=1}^{n} u_j x + \sum_{j=1}^{n} v_j y. \qquad (12)$$

To find the reasonable coefficients x, y, We need to compute the feasible region defined by the inequations (Equation (9), Equation (10),Equation (12)). Note that these inequations only have two variables and the straight lines all pass through the origin. The feasible region is just a region between two straight lines which pass through the origin. Actually, we only need to compute all the slopes $\frac{y}{x}$, and select the closest pair of slopes which have different direction. Then these two lines form a narrow feasible region and the direction of each line is determined by the coefficient of x.

After computing the feasible region for x, y, for each feasible pair of x, y, we can recover m by Equation (11) and get $w^{-1} = \frac{s_1 - m k_1}{a_1}$. In most cases, $m, w^{-1} \in \mathbb{Z}$. If $m, w^{-1} \notin \mathbb{Z}$, we can get $m, w^{-1} \in \mathbb{Z}$ through multiplying x, y by the same factor.

Remark 4: Equation (7) and Equation (5) can generate 5 equations. But no matter we take $i = 2, \ldots, 5$, the equation is a scalar multiple of Equation (11). We can not get x, y, m by solving linear equations directly.

We must examine that $\gcd(m, w^{-1}) = 1$. Actually, that means $\forall i, a_i$ and m don't have any common factor which is larger than 1 from Equation (6). We know the public key a_i would be indistinguishable from uniformly distributed sequence. So the greatest common factor of $a_1, \ldots, a_n$ and m should be quite small. From experiment data, more than one-fifth of the pairs (m, w^{-1}) meet the condition $\gcd(m, w^{-1}) = 1$.

Till now, we recover more than one group of equivalent private keys $s_1, s_2, \ldots, s_n, m, w$ which satisfy all the three conditions.

D. OUR ATTACK AND COMPLEXITY

Here we formalize the complete algorithm to attack the basic Merkle-Hellman knapsack cryptosystem (Algorithm 2).

Algorithm 2 The New Attack

Input: Public key $a_1, a_2, \ldots, a_n$.
Output: Groups of equivalent private keys $s_1, s_2, \ldots, s_n, m, w$.

1) **Note that** $s_1 = w^{-1} a_1 + m k_1$. **Define** $\mathbf{b_1} = (a_2, \ldots, a_6)^T, \mathbf{b_2} = (-a_1, \ldots, 0)^T, \ldots, \mathbf{b_6} = (0, \ldots, -a_1)^T$. **Apply LLL algorithm on the lattice** $\mathcal{L}(\mathbf{b_1}, \ldots, \mathbf{b_6})$. **Remove the zero vector and get the short vector** $(d_1, \ldots, d_5)^T$ **as well as** k_1. **We have** $s_1 a_2 - s_2 a_1 = d_1 m$.
2) **Let a** $= (a_1, \ldots, a_n)^T, \mathbf{s} = (s_1, \ldots, s_n)^T$. **Using Algorithm 1, compute the orthogonal lattice** $\mathcal{L}^{\perp}(\mathbf{a}) = \mathcal{L}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{n-1})$.
3) **Using Algorithm 1, compute** $\mathcal{L}_1^{\perp}$ **as the orthogonal lattice of** $\mathcal{L}_1 = \mathcal{L}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{n-2})$, **denote** $\mathcal{L}_1^{\perp} = \mathcal{L}(\mathbf{u}, \mathbf{v})$.
4) **Set** $\mathbf{s} = x\mathbf{u} + y\mathbf{v}$. **Get the feasible region for** x, y **by the inequations (Equation (9), Equation (10),Equation (12)).**
5) **Calculate** $m = (u_1 a_2 - u_2 a_1) x / d_1 + (v_1 a_2 - v_2 a_1) y / d_1$, **and** $w^{-1} = (s_1 - m k_1)/a_1$.
   If $\gcd(m, w^{-1}) = 1$, **then calculate** $w = InvMod(w^{-1}, m)$.
   Else choose another x, y **from the feasible region.**
6) **Calculate** $\mathbf{s} = x\mathbf{u} + y\mathbf{v}$, **then** $s_1, s_2, \ldots, s_n, m, w$ **is a group of equivalent private keys.**

From Algorithm 2, the main time-consuming part is calculating the orthogonal lattice of **a**. The time complexity of LLL algorithm is $O(n^6 L^3)$ if we invoke the classical LLL algorithm [7], where L is the input length. If we use L^2 algorithm [8], the complexity can be reduced to $O(n^5(n + L)L)$. As we know, the time complexity of Shamir's algorithm is $O(n^{g+10} L \log L)$. In the basic Merkel-Hellman cryotosystem, taking $g = 5, L = n + N = 2n$, our method obtains a $O(n^7)$ speed-up compared with Shamir's algorithm if we only invoke the classical LLL algorithm.

TABLE 1. Experiment results.

$N\backslash n$	50			100		
	step1	step2,3	step4,5	step1	step2,3	step4,5
50	1.9	419	0.04	2.7	1707	0.07
100	4.5	899	0.05	4.8	4124	0.09

E. EXPERIMENT RESULTS

In our experiment, we use Shoups NTL library [15] and need to invoke LLL algorithm three times. We use Inter(R) Xeon(R) CPU E5620@2.40GHz with 4 cores in the whole process.

We claim in the subsection 3.1 that the elements of the short vector is equal to $\frac{s_1 a_i - s_i a_1}{m}$, $i = 2, \ldots, 6$ and in the subsection 3.2 that $\mathbf{t}_{n-1} = \mathbf{t}_0$. Experiment results shows that our assumption is reasonable because step 1 and step 3 fails with a negligible probability. In the subsection 3.3, we have to make sure that $\gcd(m, w^{-1}) = 1$ in Algorithm 4. More than one-fifth of the output m, w^{-1} during the operation of procedure satisfy this condition. In particular, all of m, w^{-1} are integers. So the experimental data validate our assumptions and our algorithm is feasible and efficient.

We take $N, n = 50, 100$ in the experiments. Experiment results are presented in Table 1. Running times are given in seconds.

As we analysis in the subsection 3.4, the main part of the computational complexity of our attack is $O(n^6 L^3)$ in step 2. The operational time of other steps is negligible compare with the operational time of step 2. The performance of step 5 which we search for proper x, y is even better than our estimation. The data of the running time in Table 1 confirm the estimation for the computational complexity of our algorithm.

From Remark 3, if we take $n_0 = \lfloor \frac{n}{2} \rfloor$ or even more small, Theorem 2 still holds. Then Algorithm 2 can remain valid after a tiny adjustment. In this case, $s_1, s_2, \ldots, s_{n_0}$ is a positive superincreasing sequence, and we need to prove that $s_1, s_2, \ldots, s_n$ is a positive superincreasing sequence. The proof may be not obvious. We don't find a appropriate proof in this paper and the rationale may be similar as the rationale present in subsection 2.3 [2], [6]. The experiment results also confirm our assumption that $s_1, s_2, \ldots, s_n$ is also a superincreasing sequence and we can get groups of equivalent private keys. The computational complexity of computing the orthogonal lattice is $O(n_0^6 L^3)$, which is much smaller when we take n_0 rather than n.

IV. CONCLUSION

In this paper, we present a new attack to the basic Merkle-Hellman knapsack cryptosystem. We utilize the lattice theory and our algorithm is quite different from Shamir's famous attack. At the same time, we hardly run search procedure and reduce the computational complexity to $O(n^6 L^3)$. The main contribution of our attack is to provide a new feasible algorithm to recover the equivalent private keys in seconds and the new algorithm make full use of several recent works in cryptography technology. The experimental data validate the efficiency of our attack.

REFERENCES

[1] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 5, pp. 525–530, Sep. 1978.

[2] A. Shamir, "A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem," in *Advances in Cryptology*. Berlin, Germany: Springer, 1982, pp. 279–288.

[3] J. C. Lagarias and A. M. Odlyzko, "Solving low-density subset sum problems," *J. Assoc. Comp. Math.*, vol. 32, no. 1, pp. 229–246, Jan. 1985.

[4] M. J. Coster, B. A. LaMacchia, A. M. Odlyzko, and C. P. Schnorr, "An improved low-density subset sum algorithm," in *Computational Complexity*. Berlin, Germany: Springer-Verlag, 1991, pp. 54–67.

[5] P. Nguyen and J. Stern, "Merkle-Hellman revisited: A cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations," in *Proc. Annu. Int. Cryptol. Conf.*, 1997, pp. 198–212, 1997.

[6] J. C. Lagarias, "Performance analysis of Shamir's attack on the basic Merkle-Hellman knapsack cryptosystem," in *Proc. Int. Colloq. Automata, Lang., Program.*, 1984, pp. 312–323.

[7] A. K. Lenstra, H. W. Lenstra Jr, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Ann.*, vol. 261, no. 4, pp. 515–534, 1982.

[8] P. Nguyen and D. Stehlé, "An LLL Algorithm with quadratic complexity," *SIAM J. Comput.*, vol. 39, no. 3, pp. 874–903, 2009.

[9] H. W. Lenstra, "Integer programming with a fixed number of variables," *Math. Oper. Res.*, vol. 8, no. 4, pp. 538–548, Nov. 1983.

[10] R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proc. ACM Symp. Theory Comput.*, Dec. 1983, pp. 193–206.

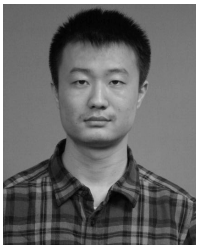
[11] K. Kobayashi, K. Tadaki, M. Kasahara, and S. Tsujii, "A knapsack cryptosystem based on multiple knapsacks," in *Proc. Int. Symp. Inf. Theory Its Appl.*, Oct. 2010, pp. 428–432.

[12] W. Zhang, B. Wang, and Y. Hu, "A new knapsack public-key cryptosystem," in *Proc. Int. Conf. Inf. Assurance Secur. (IAS)*, vol. 2, Aug. 2009, pp. 53–56.

[13] Y. Murakami, S. Hamasho, and M. Kasahara, "A public-key cryptosystem based on decision version of subset sum problem," in *Proc. ISITA*, Sep. 2012, pp. 735–739, 2012.

[14] J. Liu and J. Bi, "Equivalent key attack against a public-key cryptosystem based on subset sum problem," *IET Inf. Secur.*, vol. 12, no. 6, pp. 498–501, 2018.

[15] V. Shoup. (2018). *Number Theory C++ Library*. [Online]. Available: https://www.shoup.net/ntl/



JIAYANG LIU received the B.S. degree in mathematics from Tsinghua University, in 2012, where he has been studying information security with the Department of Computer Science and Technology, since 2012.

JINGGUO BI received the B.Sc. and Ph.D. degrees in information security from Shandong University, in 2007 and 2012, respectively. He was an Associate Researcher with Tsinghua University, Beijing, China. He is currently a Researcher with the Beijing Research Institute of Telemetry, China. His research interest includes public key cryptosystems.

SONGYAN XU is currently a Researcher with the Beijing Research Institute of Telemetry, China. Her main research interest includes information security.