applied sciences

MDPI

Article

Single Trace Analysis against HyMES by Exploitation of Joint Distributions of Leakages

ByeongGyu Park [1], Suhri Kim [1], Seokhie Hong [1], HeeSeok Kim [2] and Seog Chung Seo [3,*]

[1] Center for Information Security Technologies (CIST), Korea University, Seoul 02841, Korea; parkbg@korea.ac.kr (B.P.); suhrikim@gmail.com (S.K.); shhong@korea.ac.kr (S.H.)
[2] Department of Cyber Security, College of Science and Technology, Korea University, Sejong 30019, Korea; 80khs@korea.ac.kr
[3] Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul 02707, Korea
* Correspondence: scseo@kookmin.ac.kr; Tel.: +82-02-910-4742

Received: 21 January 2020; Accepted: 2 March 2020; Published: 6 March 2020

Abstract: Beginning with the proposal of the McEliece cryptosystem in 1978, code-based cryptography has positioned itself as one of main categories in post-quantum cryptography (PQC). To date, the algebraic security of certain variants of McEliece cryptosystems has been challenged many times, although some of the variants have remained secure. However, recent studies on code-based cryptography have focused on the side-channel resistance since previous studies have indicated that the existing algorithms were vulnerable to side-channel analysis. In this paper, we propose the first side-channel attack on the Hybrid McEliece Scheme (HyMES) using only a single power consumption trace. HyMES is a variant of the McEliece system that provides smaller keys, along with faster encryption and decryption speed. By exploiting joint distributions of nonlinear functions in the decryption process, we were able to recover the private key of HyMES. To the best of our knowledge, this is the first work proposing a side-channel analysis based on a joint distribution of the leakages on the public-key system.

Keywords: McEliece; HyMES; side-channel analysis; code-based cryptosystem; joint distribution; single trace analysis

1. Introduction

Currently, main public-key cryptosystems in use (such as Rivest–Shamir–Adleman (RSA) and elliptic-curve cryptography (ECC)) are based on the difficulty of number theoretic problems. For example, RSA is based on the difficulty of factoring large numbers, while ECC is based on the difficulty of solving discrete logarithm problems on elliptic curves. However, when a quantum computer is put into practical use running the Shor algorithm, these problems can be solved in polynomial time, thus making the RSA and ECC insecure [1]. Studies on post-quantum cryptography have been actively conducted worldwide in attempts to solve this problem. Post-quantum cryptography involves a cryptographic algorithm that runs on a classical computer, and it is believed to be immune to quantum attacks. There are several categories of post-quantum cryptography: multivariate-based, lattice-based, code-based, isogeny-based cryptography, and hash-based digital signature algorithms. Multivariate cryptography is based on the difficulty of solving multivariate equation systems. Lattice-based cryptography is based on the difficulty of solving several lattice problems. Code-based cryptography is based on the difficulty of decoding general linear codes. Isogeny-based cryptography is based on the hardness of finding isogeny between elliptic curves. Finally, hash-based digital signature algorithms are based on the security of cryptographic hash functions. Among these categories, code-based cryptography has been researched the most due to

its fast encryption and decryption speeds [2]. Additionally, code-based cryptosystems account for the second-most submitted categories to the National Institute of Standards and Technology (NIST) standardization project [3].

The first code-based cryptosystem was the McEliece cryptosystem, which was proposed by Robert McEliece in 1978 [4]. The McEliece cryptosystem was cryptographic scheme to use randomization in the encryption process. Since the main building block in implementing a McEliece cryptosystem is matrix multiplication, it provides faster encryption and decryption speeds than RSA [2]. However, one disadvantage of code-based cryptosystems is the key size involved in such systems. For example, the public key size of McEliece base on the Goppa code is about 437KB for 80-bit security [5]. Hybrid McEliece Scheme (HyMES) is a variant of the McEliece system that was proposed by B.Biswas in 2008 [6]. Since the public-key of HyMES uses a generator matrix represented in reduced row echelon form, it has a smaller key size than the classical McEliece. For 80-bit security, the public-key size of HyMES is 63KB, and it also provides faster speed than the classical McEliece [5].

Although the McEliece cryptosystem is theoretically secure, there could be certain vulnerabilities when implementing such a system [7–9]. For example, the side-channel analysis proposed by Kocher et al. is an attack based on information gained from the implementation rather than from the algorithm itself [10,11]. During the execution of an algorithm, additional information, such as time, power consumption, and electromagnetic information, can be used to reveal a secret key. Since the side-channel attack is now considered as a de facto standard, post-quantum cryptography (PQC) should similarly consider possible side-channel attacks in order to substitute RSA and ECC.

The first proposed side-channel attack on McEliece is the timing attack, which was proposed by Strenzke in 2008 [7]. In Reference [7], they revealed the secret key by exploiting the fact that the time difference information is related to the error bits of the codeword. Since the attack proposed by Strenzke, constant-time implementations have been considered when constructing and implementing code-based cryptosystems [12]. In 2010, Heyse et al. proposed the SPA (Simple Power Analysis) on McEliece [9]. Their attack involved analyzing the power consumption trace obtained during the decoding process. However, due to the structure of the decryption algorithm, the SPA proposed in Reference [9] cannot be applied to HyMES. On the other hand, Y. Linge proposed a side-channel analysis technique with which to identify a secret key by using the probability distribution of the input and the output of a nonlinear function varying with the secret key value [13]. The joint distribution is a probability distribution considering two or more random variables. The side-channel analysis using the joint distribution has only been investigated in the symmetric key cryptography, and several traces have been used.

In this paper, we present our proposed side-channel analysis on HyMES using a single power consumption trace. The main contributions of this paper are as follows.

- We propose the first side-channel analysis on HyMES using the joint distributions of leakages. We target the non-linearity of the pre-computation table in HyMES implementation [6]. By analyzing the leakage that occurs while calculating the parity-check matrix, we are able to recover the Goppa polynomial $g(z)$ and the support L_{sec}, which are the secret keys of HyMES. The proposed method only uses one power consumption trace, and it is the first joint distribution based analysis for public-key cryptography. Moreover, we demonstrate that any other public-key cryptosystem using nonlinear operation can be vulnerable to our attack. The details of our attack are presented in Section 3.
- We present an experimental result validating the efficiency of the proposed attack. We confirm that the proposed method works as expected by using simulated power traces. These simulated traces are collected by adding noise to the Hamming weight model. The attack success rate is 100% up to the noise standard deviation of 1.3. Further, we show that the performance of the proposed attack increase when several traces are used. When 10 traces are used, the attack success rate is 100% up to the noise standard deviation of 2.1.

The remainder of this paper is organized as follows. Section 2 provides background on code-based cryptography and discusses the side-channel attack used for the analysis. Section 3 provides a high-level description of the proposed attack. Section 4 shows the experimental results using the simulation traces. Finally, the conclusion is drawn in Section 5.

2. Related Works

In this section, we describe the backgrounds of concepts to be used throughout this paper. First, we briefly introduce binary Goppa code, which is one of the error correcting codes used in HyMES. We introduce the basic concepts in code-based cryptography and structures of HyMES. Finally, we introduce the side-channel analysis exploiting the joint distributions of leakages, which is the key attack method in the proposed analysis.

2.1. Binary Goppa Code

This paper only discusses basic information about the binary Goppa code over the finite field F_2. The Goppa code consists of the Goppa polynomial $g(z)$ and support L_{sec}. The $g(z)$ and L_{sec} are defined in Definition 3.

Definition 1. *For positive integer m, t, Goppa polynomial $g(z)$ and support L are as follows.*

$$g(z) = \sum_{i=0}^{t} g_i z^i \in F_{2^m}[z]$$

$$L = \{\alpha_0, ..., \alpha_{n-1}\} \in F_{2^m}^n,\ g(\alpha_j) \neq 0,\ \forall 0 \leq j \leq n-1.$$

The syndrome $S_{\hat{c}}$ required to decode the Goppa code is denoted in Equation (1). The binary $[n, k, d_c]$-Goppa code is defined as the set of all $\hat{c}$ that have the syndrome $S_{\hat{c}}$ at zero in Equation (2). Since the parity check matrix H spans the null space of C, as shown in Definition 2, we can derive H from the syndrome $S_{\hat{c}}$ as in Equation (3).

Definition 2. *For the Goppa polynomial $g(z)$ and the support $L = \{\alpha_0, ..., \alpha_{n-1}\}$, the syndrome $S_{\hat{c}}$ is as follows.*

$$S_{\hat{c}} = -\sum_{i=0}^{n-1} \frac{\hat{c}_i}{g(\alpha_i)} \frac{g(z) - g(\alpha_i)}{z - \alpha_i} \equiv \sum_{i=0}^{n-1} \frac{\hat{c}_i}{z - \alpha_i} \mod g(z), \tag{1}$$

$$Goppa(L, g(z)) = \{\hat{c} \in F_{2^m}^n \mid S_{\hat{c}} = \sum_{i=0}^{n-1} \frac{\hat{c}_i}{z - \alpha_i} \equiv 0 \mod g(z)\}, \tag{2}$$

$$H = \begin{pmatrix} \frac{g_t}{g(\alpha_0)} & \frac{g_t}{g(\alpha_1)} & \cdots & \frac{g_t}{g(\alpha_{n-1})} \\ \frac{g_t\alpha_0 + g_{t-1}}{g(\alpha_0)} & \frac{g_t\alpha_0 + g_{t-1}}{g(\alpha_1)} & \cdots & \frac{g_t\alpha_0 + g_{t-1}}{g(\alpha_{n-1})} \\ \cdots & \cdots & & \cdots \\ \frac{g_t\alpha_0^{t-1} + \ldots + g_2\alpha + g_1}{g(\alpha_0)} & \frac{g_t\alpha_0^{t-1} + \ldots + g_2\alpha + g_1}{g(\alpha_1)} & \cdots & \frac{g_t\alpha_0^{t-1} + \ldots + g_2\alpha + g_1}{g(\alpha_{n-1})} \end{pmatrix}. \tag{3}$$

The H in Equation (3) can be simplified as shown in Equation (4), and the $\hat{H}$ can be used as the parity check matrix because the determinant of H_g is not 0.

$$H = \begin{pmatrix} g_t & 0 & \ldots & 0 \\ g_{t-1} & g_t & \cdots & 0 \\ \cdots & \cdots & & \cdots \\ g_1 & g_2 & \ldots & g_t \end{pmatrix} \times \begin{pmatrix} \frac{1}{g(\alpha_0)} & \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \frac{\alpha_0}{g(\alpha_0)} & \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}}{g(\alpha_{n-1})} \\ \cdots & \cdots & & \cdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{pmatrix} = H_g \times \hat{H}. \tag{4}$$

2.2. Code-Based Cryptography

Error correction codes were first developed in 1947 by Richard Hamming. The technique was developed to enable the reliable delivery of digital data over unreliable communication channels. In this technique, the sender encodes the data using an error-correcting code prior to transmission. The additional information added by the code is used by the receiver to recover the original data. However, decoding is only efficient for linear codes with efficient decoding algorithms. The code-based cryptosystems make use of the fact that decoding the syndrome of a general linear code is known to be NP-hard, while efficient algorithms exist for the decoding of specific linear codes. The Goppa codes are an example of efficient correcting codes that can be turned into a secure coding scheme when the decoding functions are kept secret. Only an attacker in possession of the secret decoding function can remove the secret mapping and recover the plaintext. Before describing the Goppa codes, the definition of binary linear code is presented in Definition 1. The definitions of the generator matrix G and the parity check matrix H are as denoted in Definition 2.

Definition 3. *Binary linear code $[n,k]-C$ is a one-to-one function that takes a k-bit string message and outputs a n-bit string codeword.*

Definition 4. *If the row vector space of $k \times n$ matrix G spans C over finite field F and the dimension of row vector space is k, then matrix G is the generator matrix of $[n,k]-C$ over F. For the $(n-k) \times n$ matrix H, if the column vectors of H^T are the basis of the null space of G, H is the parity check matrix of C.*

2.2.1. McEliece Cryptosystem

The McEliece cryptosystem was the first code-based public-key cryptosystem proposed by McEliece in 1978 [4]. Although the algorithm states that any error-correcting codes can be used, only McEliece with Goppa codes has resisted cryptanalysis to date [14,15]. In this paper, we define classical McEliece as McEliece with Goppa code. The encryption and decryption processes of McEliece are respectively described in Algorithms 1 and 2. In step 2, the generator matrix G corresponding to the code C is a matrix consisting of the basis sequences necessary for generating C. Therefore, k represents the dimension of code C. G is computed by using Goppa polynomial of degree t, which is the primary part of the public key. Scrambling matrix S and a permutation matrix P need to hide the algebraic structure of G. These are generated randomly and multiplied with G [4]. $HW(e)$ is the number of 1 when e is expressed in binary.

Algorithm 1 Classical McEliece : Encryption

Input M, $K_{pub} = (\hat{G} = SGP, t)$
Output Ciphertext c
1: Represent the message M as k-bit vector m
2: Generate a random n-bit vector e. $HW(e) \leq t$
3: return $c = m \times \hat{G} + e$

Algorithm 2 Classical McEliece : Decryption

Input c, $K_{sec} = (G, S^{-1}, P^{-1})$
Output Message M
1: Compute the $\hat{c} = c \times P^{-1}$
2: Compute the syndrome $S_{\hat{c}}$
3: Obtain a k-bit $\hat{m} = m \times S$ from $S_{\hat{c}}$ using decoding algorithm $D_{Goppa}(\hat{c})$
4: Compute the $m = \hat{m} \times S^{-1}$
5: Represent the m as a message M
6: return M

2.2.2. HyMES (Hybrid McEliece Scheme)

The HyMES was proposed by B.Biswas and N.Sendrier, who were members of the SECRET team at the INRIA Lab in 2008 [6]. The HyMES is a variant of the McEliece cryptosystem that provides a smaller key size and faster encryption rate than McEliece for the same security level. The key generation algorithm of HyMES is described below.

In step 1 of Algorithm 3, Goppa polynomial is chosen as an irreducible polynomial. In this case, since the minimum distance of the code satisfies $d_c \leq 2t+1$, a valid user can correct a maximum of t errors. In step 2, the number of supports n is used as the number of all elements of $GF(2^m)$, and the supports are selected randomly from 0 to 2^{m-1} in order. The public-key size of HyMES is $k \times (n-k)$, which is reduced to $k \times k$ relative to the Classical McEliece whose public-key size is $k \times n$. The encryption and decryption algorithms are denoted in Algorithms 4 and 5, respectively.

Algorithm 3 HyMES : Key Generation

Input t, n, m
Output K_{sec}, K_{pub}
1: Generate a random monic polynomial $g(z)$ as the Goppa polynomial $g(z) = z^t + c_{t-1}z^{t-1} + ... + c_1 z + c_0,\ \deg(g(z)) = t,\ c_i \in GF(2^m)$
2: Randomly choose the support $L = \{\alpha_0, ..., \alpha_{n-1}\},\ \alpha_i \in GF(2^m),\ g(\alpha_i) \neq 0$
3: Compute the parity check matrix $\hat{H}$
4: Convert the parity check matrix to the systematic form $\hat{H}_{sys} = \hat{S}\hat{H}\hat{P} = (Q^T \mid I_{n-k})$
5: If the parity check matrix $\hat{H}$ is not converted to the systematic form $(Q^T \mid I_{n-k})$, go back to step 1
6: Replace support L with $L_{sec} = L\hat{P}$ using $\hat{P}$
7: Compute $G_{sys} = (I_k \mid Q) = SGP$
8: return $K_{sec} = (g(z), L_{sec}), K_{pub} = G_{sys} = (I_k \mid Q)$

Algorithm 4 HyMES : Encryption

Input $M, K_{pub} = G_{sys} = (I_k \mid Q)$
Output Ciphertext c
1: Represent the message M as k-bit vector m
2: Generate a random n-bit vector e. $HW(e) \leq t$
3: Return $c = m \times G_{sys} + e$

Algorithm 5 HyMES : Decryption

Input $c, K_{sec} = (g(z), L_{sec})$
Output Message M
1: Compute the syndrome $S_c = (H\hat{P})c^T$
2: Obtain a k-bit $\hat{m} = m \times S \times G \times P = mG_{sys} = m(I_k \mid Q) = (m \mid mQ)$ from S_c using decoding algorithm $D_{Goppa}(c)$
3: Obtain m from the top k-bit of $\hat{m}$.
4: Represent the m as a message M
5: return M

In step 3 of Algorithm 4, it is not necessary to multiply the identity matrix I_k for matrix multiplication with m and G_{sys}, so the encryption speed of HyMES is faster than that of the Classical McEliece.

2.3. Side-Channel Analysis by Exploiting Joint Distributions of Leakages

The side-channel analysis is an attack which exploits any weakness that occurs during the execution of a cryptographic algorithm. The first side-channel analysis using power consumption

was proposed by Kocher et al. in Reference [11]. Later, S. Chari proposed the template attack in Reference [16], and then K.Schramm proposed a collision attack in Reference [17]. Since then, various side-channel attacks have proposed for various algorithms. Regarding code-based cryptography, Heyse et al. proposed SPA on the decoding process of the Classical McEliece in 2010 [9]. The proposed method used the leakage obtained when multiplying a matrix P^{-1} before calculating the syndrome during decryption. However, since HyMES does not multiply P^{-1} before calculating the syndrome in the decryption process, the method proposed in [9] cannot be used. Therefore, we analyzed HyMES by exploiting the joint distributions of leakages. In this section, we describe in detail the analysis exploiting the joint distributions of leakages.

In 2014, Y. Linge proposed a method of determining the secret key of AES by using the joint distributions [13]. They exploited the joint distributions of the input and output of the S-box, which is a nonlinear function of AES. Since this distribution depends on the key value, an attacker can identify the secret key under the assumption that the value of the plaintext pair is unknown. The attack procedure proposed by Y. Linge is described in Algorithm 6. The function g takes k and a as arguments and outputs b. The function ϕ outputs the input's Hamming weight. In Reference [13], the function g is selected as the AES S-box. In this example, the attack is described based on the S-box used in the first round `SubBytes` of AES.

Algorithm 6 Side channel analysis using joint distributions

Input $(a_i, b_i),\ i = 0, ..., M-1$
Output *key*

1: $g : K \times A \rightarrow B,\ N = |K|,\ S(g,k),\ S_d \in \{0, ..., n\} \times \{0, ..., m\}$
2: for $k \in K$ ← computation of theoretical joint distributions
3: $S(g,k) \leftarrow 0$
4: for $a \in A$
5: $S(g,k)(\phi(a), \phi(g(a,k))) \leftarrow S(g,k)(\phi(a), \phi(g(a,k))) + \frac{1}{|A|}$
6: end
7: end
8: $S_d \leftarrow 0$
9: for $i = 0 : M-1$ ← computation of estimated joint distribution
10: $S_d(a_i, b_i) \leftarrow S_d(a_i, b_i) + \frac{1}{M}$
11: end
12: $key \leftarrow 0$
13: for $k \in K$ ← compare theoretical distribution and estimated distribution
14: if $d(S(g,k), S_d) < d(S(g,key), S_d)$
15: $key \leftarrow k$
16: end
17: end
18: return *key*

First, steps 2–7 of Algorithm 6 are the processes involved in making the theoretical joint distributions table, which uses the Hamming weight values of random variables, plaintext a, and S-box output b. This process is performed for key $k' = 0$–255 with a 1-byte size. The $S(g,k)$ is the theoretical joint distribution table of $\phi(a)$ and $\phi(b)$ when the key is k for function g. The size of $S(g,k)$ is 9×9, since the range of possible Hamming weights for a and b is 0-8. Next, steps 9–11 are the processes of constructing the estimated joint distribution table using (a_i, b_i) pairs, which are the estimated Hamming weight values of a and b. Finally, steps 13–18 are the processes for obtaining the similarity between the estimated distribution and the theoretical distribution. The guessed key is the one with the highest similarity to the real key.

In order to construct the estimated joint distribution table, the Hamming weight values of the plaintext a and the output b are estimated from the collected power consumption traces. Although the

Hamming weight values should be accurately estimated for the performance of the attack, it is very difficult to accurately guess the Hamming weight values due to the noise signal. In 2017, an improved method was proposed that use a maximum likelihood method that reflects the influence of noise as a probability [18]. An attack using the joint distributions of leakages is possible because the joint distribution depends on the key value, which occurs because the AES S-box is sufficiently non-linear. The attack using the joint distribution has only been investigated in symmetric key cryptography.

3. Single Trace Analysis Against HyMES

In this section, we propose for the first time a side-channel attack on HyMES exploiting the joint distributions of the leakages. First, we present an outline of the proposed attack. Then, we describe the two steps used to recover the secret key $(g(z), L_{sec})$.

3.1. Outline of the Proposed Attack

Our attack targets the leakage obtained during the computation of $H\hat{P}$. Note that $H\hat{P}$ is required for syndrome operation $S_{\hat{c}} = (H\hat{P})c^T$ in step 1 of Algorithm 5. The implementation of HyMES proposed in Reference [6] includes the process of calculating the column vectors h_i of $H\hat{P}$ in the key generation algorithm. The implementation precomputes the values $h_i,\ i = 0, ..., n-1$ and stores the result as a secret key. Since these values are required to compute the syndrome, they can also be directly calculated in the decryption process if they are not precomputed. Therefore, without loss of generality, we may assume that the h_i is calculated in the same way as described in Algorithm 7.

Algorithm 7 computes the $H\hat{P}$ using the secret key L_{sec} which is substituted with the permutation matrix $\hat{P}$. The h_i with length mt-bit can be expressed as a polynomial of degree $t-1$ degree polynomial over $GF[2^m]$, as shown in Equation (5).

$$h_i = h_i[t-1]z^{t-1} + h_i[t-2]z^{t-2} + ... + h_i[0]. \tag{5}$$

Algorithm 7 HyMES Implementation : Computation of $H\hat{P}$ [6]

Input $g(z) = z^t + g_{t-1}z^{t-1} + ... + g_0,\ L_{sec} = \{\alpha_0, ..., \alpha_{n-1}\},\ n = 2^m$
Output the column vectors $h_i,\ i = 0, ..., n-1$ of $H\hat{P}$

```
for i = 0 : n − 1
    h_i[t − 1] ← 1
    for j = t − 2 : 0
        h_i[j] ← g_{j+1} ⊕ (α_i × h_i[j + 1])
    end
    a ← g_0 ⊕ (α_i × h_i[0])
    for j = 0 : t − 1
        h_i[j] ← h_i[j]/a
    end
end
return h_i, i = 0, ..., n − 1
```

To recover the secret key $(g(z), L_{sec})$, the proposed attack is divided into two stages. The first stage of the attack identifies the Goppa polynomial $g(z)$ based on the joint distributions of leakages. The second stage finds the remaining secret key L_{sec} through the horizontal correlation analysis by using $g(z)$ obtained in the first stage. In Sections 3.2 and 3.3, we explain how to determine $g(z)$ and L_{sec}, respectively.

3.2. Recovering $g(z)$

The multiplication and division involved in Algorithm 7 are computed over the field $GF(2^m)$. In general, the multiplication and division on $GF(2^m)$ are implemented using the exponentiation-table and the log-table for the purpose of speed efficiency. For the root α of the primitive polynomial $p(x)$ that constitutes $GF(2^m)$, the exponentiation-table takes the positive integer i as input and returns α^i. The log-table is constructed to return i by taking the element α^i of $GF(2^m)$ as an input. In Reference [6], the multiplication and division are computed by addition, subtraction, and modulus operations using the pre-computed log-table and exponentiation-table. Note that using the tables is more efficient than directly computing the exponentiations and logarithm values. The implementations of multiplication and division over $GF(2^m)$ are as shown in Algorithms 8 and 9, respectively.

Algorithm 8 HyMES implementation : Multiplication over $GF(2^m)$ [6]

Input $a, b \in GF(2^m)$, $\exp[i] = \alpha^i$, $\log[\alpha^i] = i$, $i = 0, ..., 2^m - 1$ $mod(d) = ((d) \wedge (2^m - 1)) + ((d) >> m)$
Output $a \times b \in GF(2^m)$
1: if a or $b = 0$
2: $\quad a \times b \leftarrow 0$
3: else
4: $\quad a \times b \leftarrow \exp[mod(\log[a]) + \log[b]]$
5: end
6: return $a \times b$

Algorithm 9 HyMES implementation : Division over $GF(2^m)$ [6]

Input $a, b \in GF(2^m)$, $\exp[i] = \alpha^i$, $\log[\alpha^i] = i$, $i = 0, ..., 2^m - 1$ $mod(d) = ((d) \wedge (2^m - 1)) + ((d) >> m)$
Output $a/b \in GF(2^m)$
1: if $a = 0$
2: $\quad a/b \leftarrow 0$
3: else
4: $\quad a/b \leftarrow \exp[mod(\log[a]) - \log[b]]$
5: end
6: return a/b

Now that the basic field operations have been explained, we present the method to recover $g(z)$. To find $g(z)$, we must find the coefficients $g_{t-1}, g_{t-2}, ..., g_0$ of all the orders of $g(z)$. In this section, we only explain how to find g_{t-1}, since the other coefficients are found with the same methods used for g_{t-1}. The full recovery of $g(z)$ using the joint distributions of leakages is described in Algorithm 10.

Function $l(k, a)$ is a function that returns $\log[k \oplus a]$ by taking the elements k and a of $GF(2^m)$, while function $\phi(v)$ takes v as input and returns the Hamming weight of v. The value of a_i and b_i are the estimated Hamming weights of the input a and output b of the function $l(k, a)$, respectively.

When $j = t - 2, t - 3$ in the step 3 loop of Algorithm 7, the operation process can be described as follows for $i = 0 : 2^m - 1$.

$$\begin{aligned}
h_0[t-2] &= g_{t-1} \oplus (\alpha_0 \times h_0[t-1]) \\
h_0[t-3] &= g_{t-2} \oplus (\alpha_0 \times h_0[t-2]) \\
h_1[t-2] &= g_{t-1} \oplus (\alpha_1 \times h_1[t-1]) \\
h_1[t-3] &= g_{t-2} \oplus (\alpha_1 \times h_1[t-2]) \\
&\cdots \\
h_{2^m-1}[t-2] &= g_{t-1} \oplus (\alpha_{2^m-1} \times h_{2^m-1}[t-1]) \\
h_{2^m-1}[t-3] &= g_{t-2} \oplus (\alpha_{2^m-1} \times h_{2^m-1}[t-2]).
\end{aligned} \tag{6}$$

The multiplication operation $\alpha_i \times h_i[t-2]$ over $GF(2^m)$ in Equation (6) is performed by using the pre-computation table as shown in Equation (7).

$$\exp[mod(\log[\alpha_i] + \log[g_{t-1} \oplus (\alpha_i \times 1)])]. \tag{7}$$

Algorithm 10 Analysis for $g(z)$

Input Power consumption trace P for HyMES key generation, A table $\log[]$, $\exp[]$ created with the primitive polynomial $p(x)$

Output $g(z) = z^t + g_{t-1}z^{t-1} + ... + g_0$

$l : K \times A \rightarrow B$, $S(l, g'_i)$, $S_d \in \{0, ..., m\} \times \{0, ..., m\}$
for $j = t-1 : 1$
  for $g'_j = 0 : 2^m - 1$
    $S(l, g'_j) \leftarrow 0$
    for $a = 0 : 2^m - 1$
      $S(l, g'_j)(\phi(a), \phi(l(a, g'_j))) \leftarrow S(l, g'_j)(\phi(a), \phi(g(a, g'_j)) + \frac{1}{2^m}$
    end
  end
  for $i = 0 : 2^m - 1$
    Find POI $\alpha_i \times h_i[j]$, $\log[h_i[j-1]]$
  end
  Estimate data Hamming weight using slice method
  for $i = 0 : 2^m - 1$
    $S_d(a_i, b_i) \leftarrow S_d(a_i, b_i) + \frac{1}{2^m}$
  end
  $g_j \leftarrow 0$
  for $g'_j = 0 : 2^m - 1$
    if $d(S(l, g'_j), S_d) < d(S(l, g_j), S_d)$
      $g_j \leftarrow g'_j$
    end
  end
end
return $g(z) = z^t + g_{t-1}z^{t-1} + ... + g_0$

To find g_{t-1}, we use $\log[g_{t-1} \oplus (\alpha_i \times 1)]$ in the second log table operation of Equation (7). First, we set the Hamming weights of α_i and $\log[g_{t-1} \oplus \alpha_i]$ as random variables, and then construct the theoretical joint distribution tables for 2^m key candidates g'_{t-1}. The range of Hamming weights is 0-m since α_i and $\log[g_{t-1} \oplus \alpha_i]$ are elements in $GF(2^m)$. Therefore, the size of the joint distribution table $S(l, g_{t-1})$ is $(m+1) \times (m+1)$. Next, we find the point at which α_i and $\log[g_{t-1}] \oplus \alpha_i$ are processed from the power consumption trace of Algorithm 7. In this paper, the power consumption model follows the Hamming weight model shown in Equation (8), under the assumption that α and β are the same at all time points.

$$P_v = \alpha HW(v) + \beta + \omega \tag{8}$$

The estimated joint distribution table can be constructed by guessing the Hamming weight of α_i and $\log[g_{t-1} \oplus \alpha_i]$ from the power consumption value P_{α_i} and $P_{\log[g_{t-1} \oplus \alpha_i]}$, respectively. The Hamming weight of α_i and $\log[g_{t-1} \oplus \alpha_i]$ can be estimated using Linge's slice method. This slice method uses the correlation between the Hamming weight and the power consumption value. For example, the number of data points with Hamming weight p is about $\frac{M \times C_N^p}{2^N}$ among N-bit data M with uniform distribution. By sorting the power consumption values for the M data in ascending order, it is possible to guess the Hamming weight of the corresponding data. The performance of the slice method increases when the

values of the actually used data are uniformly distributed, so it is necessary to use a sufficient amount of data. Finally, when the similarity between the theoretical joint distribution and the estimated joint distribution is compared, g'_{t-1} with the highest similarity is selected as the real key g_{t-1}. In this paper, we use the Harmonic Mean Distance, Inner Product Distance, and χ-square Pearson Distance as the comparison method of distribution similarity. The definitions of the three similarity comparison methods are the same as Definition 5–7.

Definition 5. *Harmonic Mean Distance (HMD) is as follows.*

$$HMD(S(g,k),S_d) = \begin{cases} 1 - 2\sum_{i=0}^{i=n}\sum_{j=0}^{j=m}\frac{p_{i,j}f_{i,j}}{p_{i,j}+f_{i,j}}, & p_{i,j}+f_{i,j} \neq 0 \\ 0, & p_{i,j}+f_{i,j} = 0. \end{cases} \tag{9}$$

Definition 6. *Inner Product Distance is as follows.*

$$IPD(S(g,k),S_d) = 1 - \sum_{i=0}^{i=n}\sum_{j=0}^{j=m} p_{i,j}f_{i,j}. \tag{10}$$

Definition 7. *χ-square Pearson Distance is as follows.*

$$\chi^2 PD(S(g,k),S_d) = \begin{cases} \sum_{i=0}^{i=n}\sum_{j=0}^{j=m}\frac{(p_{i,j}-f_{i,j})^2}{f_{i,j}}, & f_{i,j} \neq 0 \\ 0, & f_{i,j} = p_{i,j} \\ \infty, & f_{i,j} = 0 \neq p_{i,j}. \end{cases} \tag{11}$$

3.3. Recovering L_{sec}

In this section, we explain how to find the remaining secret key L_{sec} using $g(z)$ obtained in Section 3.2. The process of finding L_{sec} is described in Algorithm 11. We explain the process of finding L_{sec} by recovering the first support value α_0 as a concrete example. The other support values can be analyzed using the same logic used to find α_0. When $i = 0$ in Algorithm 7, the operation process can be expressed by Equation (12). To find α_0, we perform the horizontal correlation power analysis between the actual power consumption values that occur when values dependent on α_0 are processed and the Hamming weights of the values, which are calculated by guessing the value of α_0 in Equation (12). The number of equations containing α_0 in Equation (12) is $t-1+1+t = 2t$. Therefore, the number of data available for the horizontal correlation power analysis is $2t$. Generally, HyMES with 88-bit security uses the parameter $m = 11,\ t = 32$. Therefore, 64 data points are available for the horizontal correlation power analysis using these parameters, which is a little data for the actual analysis. This can be resolved by using α_0-dependent values. By using all the α_0-dependent values used in the multiplication and division operations in steps 2–21 of Algorithm 11, $8\ t = 256$ data can ultimately be used for the analysis. When the correlation coefficients between actual power consumption values and $8\ t$ power consumption models, which are estimated from 0 to $2^{11}-1$, are obtained, the estimated value with the highest correlation coefficient can be considered as the correct α_0.

$$h_0[t-1] = 1$$

$$h_0[t-2] = g_{t-1} + (\alpha_0 \times h_0[t-1])$$
$$h_0[t-3] = g_{t-2} + (\alpha_0 \times h_0[t-2])$$

$$\cdots$$

$$h_0[0] = g_1 + (\alpha_0 \times h_0[1])$$
$$a = g_0 + (\alpha_0 \times h_0[0])$$

$$h_0[0] = h_0[0]/a$$
$$h_0[1] = h_0[1]/a$$
$$...$$
$$h_0[t-1] = h_0[t-1]/a. \quad (12)$$

Algorithm 11 Analysis for L_{sec}

Input Power consumption values of POI of HyMES $\boldsymbol{P} \in Mat_{1\times 8t}$, A table log[], exp[] created with the primitive polynomial $p(x)$, Goppa polynomial $g(z) = z^t + g_{t-1}z^{t-1} + ... + g_0$

Output $L_{sec} = \{\alpha_0, ..., \alpha_{n-1}\}$

for $i = 0 : n-1$
  for $j = 0 : n-1$
    $pm_1 \leftarrow hw(j)$
    $pm_2 \leftarrow hw(\log[j])$
    $pm_3 \leftarrow hw(\log[j] + \log[h_i[t-1]])$
    $pm_4 \leftarrow hw(mod(\log[j] + \log[h_i[t-1]]))$
    $pm_5 \leftarrow hw(\exp[mod(\log[j] + \log[h_i[t-1]])])$
    for $k = t-2 : 0$
      $pm_{5(t-2-k)+6} \leftarrow hw(h_i[k])$
      $pm_{5(t-2-k)+7} \leftarrow hw(\log[h_i[k]])$
      $pm_{5(t-2-k)+8} \leftarrow hw(\log[j] + \log[h_i[k]])$
      $pm_{5(t-2-k)+9} \leftarrow hw(mod(\log[j] + \log[h_i[k]]))$
      $pm_{5(t-2-k)+9} \leftarrow hw(\exp[mod(\log[j] + \log[h_i[k]])])$
    end
    for $k = 0 : t-1$
      $pm_{3k+5t+1} \leftarrow hw(\log[h_i[k]] - \log[a])$
      $pm_{3k+5t+2} \leftarrow hw(mod(\log[h_i[k]] - \log[a]))$
      $pm_{3k+5t+3} \leftarrow hw(\exp[mod(\log[h_i[k]] - \log[a])])$
    end
    $ct_j \leftarrow \rho(\boldsymbol{PM}, \boldsymbol{T})$
  end
  $\alpha_i \leftarrow argmax_j(\boldsymbol{CT})$
end
return $L_{sec} = \{\alpha_0, ..., \alpha_{n-1}\}$

4. Experiment

In this section, we describe the experimental results of the proposed attack. The experiments were conducted by using simulated power traces under the assumption that the power consumption models follow Equation (8). We assume that the α and β in Equation (8) are equal to 1 and 0 at all points. We use the parameters $m = 11$ and $t = 32$ [6] for the analysis. We present the analysis results of the coefficients g_{31} and α_0, where g_{31} is the 32nd term of the secret key $g(z)$ and α_0 is the first element of the support L_{sec}. Although we only present experimental results for when g_{31} is 1499 and α_0 is 999, similar results can be confirmed for other secret key values.

4.1. Find $g(z)$ Using Harmonic Mean Distance

To find g_{31}, we conducted an analysis by exploiting the joint distributions of two random variables $A = \alpha_i \times h_i[t-1]$ and $B = \log[h_i[t-2]] = \log[g_{t-1} \oplus \alpha_i]$. The similarity between the theoretical distribution according to the guessing key g'_{31} and the distribution estimated from the simulation trace were obtained using three similarity comparison methods. The power consumption model is set to

$\alpha = 1,\ \beta = 0$, and $\sigma = 0.1$, where σ is the noise standard deviation. Figure 1 shows the similarity between the theoretical distribution and the estimated distribution using HMD. Note that the smaller distance indicates high similarity between the two distributions.

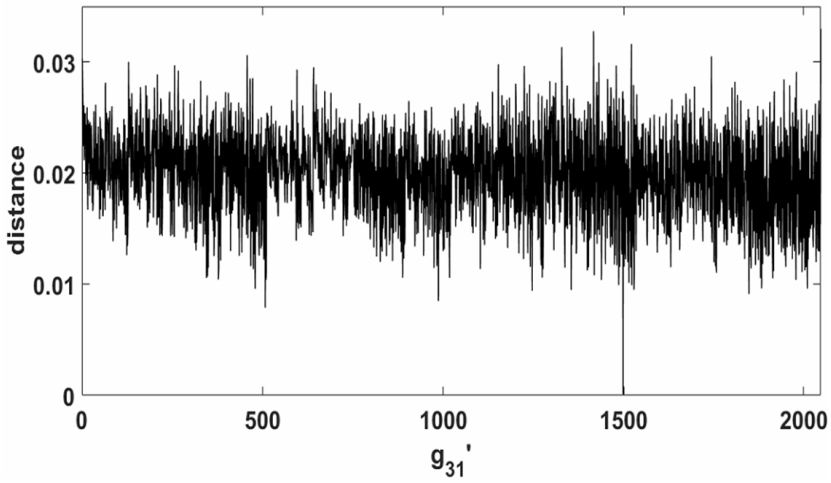

Figure 1. Harmonic Mean Distance (HMD) using two random variables A, B.

When $g'_{31} = 1499$, the highest similarity was found at 0.0001016 and the second highest was fount at 0.007898 when $g'_{31} = 508$. This indicates that the difference between the distance of the correct key and the distance of the wrong key is not very large, so that when the noise value becomes large enough, the analysis failure rate rises. To solve this problem, two random variables $C = mod(\log[\alpha_i] + \log[h_i[t-2]])$ and $D = \exp[C]$ can be respectively added to A and B to form a joint distribution using four random variables in total. The use of four random variables magnifies the difference between the correct key and wrong key, which allows the correct key to be guessed with a higher success rate.

Figure 2 shows the result of HMD between the theoretical joint distribution and the estimated distribution constructed by using four random variables. The experimental result shows that the distance at $g'_{31} = 1499$ is 0.01776 and that the distance at $g'_{31} = 927$ is 0.3911. We found that the difference between the distance to the correct guessing key and the wrong key is larger when using four random variables than when using two random variables. As mentioned earlier, many random variables elaborates the joint distributions of the leakages.

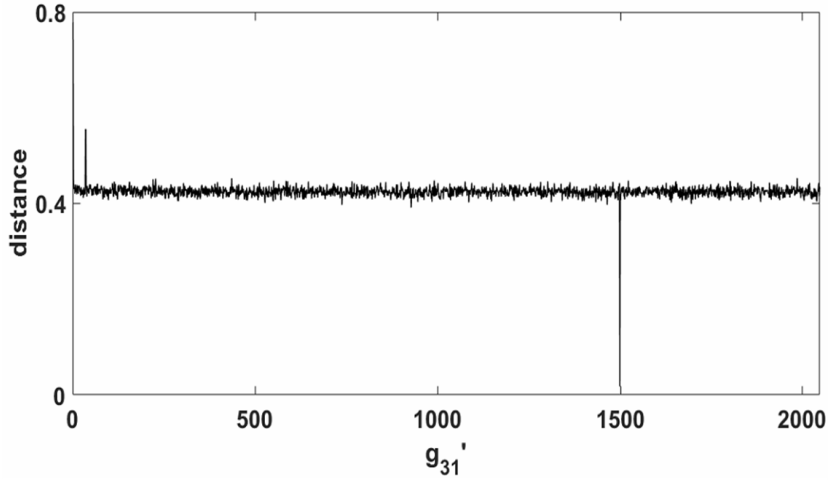

Figure 2. Harmonic Mean Distance using four random variables A, B, C, D.

Figure 3 shows the success rate of the attack according to the noise standard deviation. Recall that the attack is based on the joint distributions of leakages constructed by using two random variables.

The success rate of the attack was measured by increasing the noise standard deviation from 0 to 2.5 in increments of 0.1. We analyzed 10,000 case according to the noise standard deviation. The experimental result using IPD shows that even if the noise standard deviation is zero, the success rate is still zero. The result when χ^2PD is used shows that the success rate of the attack is 1 until the noise standard deviation reaches 0.3, and the success rate of the attack starts to decrease rapidly after 0.4. The result when HMD is used is the best among the results of the three methods. When the noise standard deviation is 0.4, the success rate of the attack begins to decrease to 0.9814.

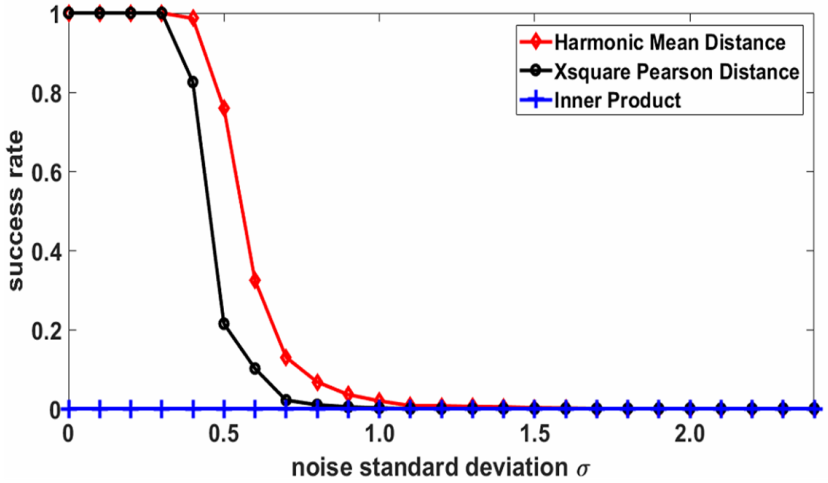

Figure 3. Success rate according to noise standard deviation (using two random variables).

Figure 4 shows the experimental results using four random variables. The other experimental conditions are the same as those shown in Figure 3. The success rate of the attack according to the noise standard deviation is highest when HMD is used. The success rate of the attack remains at 1 up to the noise standard deviation of 0.6, and then starts to decrease from 0.7 to 0.9823. Figure 4 shows that using four random variables improves the performance of the attack. However, in practice, environments with a noise standard deviation of 0.6 are extremely rare and can be difficult to analyze. The success rate of the attack is not high even in an environment where the noise standard deviation is small because of the errors between the estimated Hamming weight and the actual value used in constructing the estimated joint distribution.

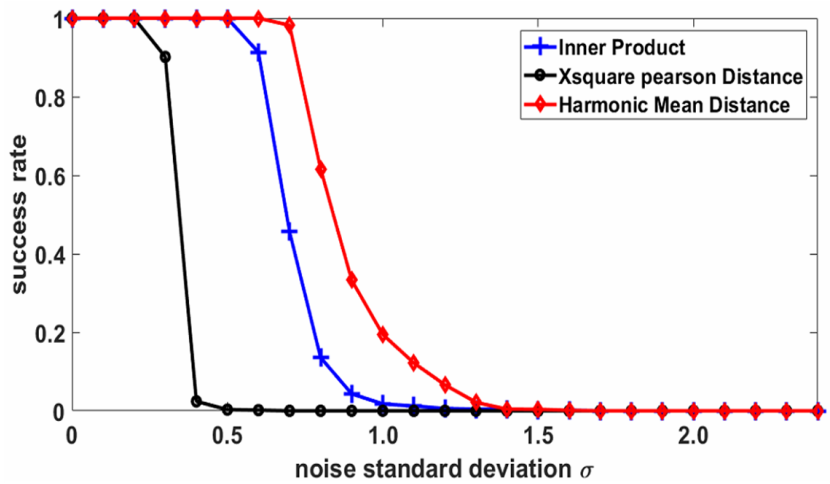

Figure 4. Success rate according to noise standard deviation (using 4 random variables).

Table 1 shows the success rate of the slice method for the data with a maximum Hamming weight of 11. The success rate was calculated based on the noise standard deviation for the case of $\alpha = 1$ and

the case of $\alpha = 2$. The data used is a random value according to the uniform distribution. The numbers of data point used were 2048, which is the minimum number for applying the slice method to 11-bit data, and 20,480, which is 10 times larger than the minimum number. The experimental result presented above shows that the more data points used and the larger the value of α, the better the performance of the slice method.

Table 1. Performance of slice method.

$\alpha = 1$			$\alpha = 2$		
N	σ	Correct Rate	N	σ	Correct Rate
2048	0.1	0.9581	2048	0.1	0.9542
	0.5	0.6975		0.5	0.9335
	1	0.4293		1	0.6971
	2	0.2790		2	0.4280
	3	0.2349		3	0.3268
	4	0.2149		4	0.2772
	5	0.2024		5	0.2503
20480	0.1	0.9846	20480	0.1	0.9856
	0.5	0.7008		0.5	0.9532
	1	0.4288		1	0.7012
	2	0.2783		2	0.4292
	3	0.2337		3	0.3272
	4	0.2144		4	0.2789
	5	0.2035		5	0.2509

4.2. Find $g(z)$ Using Maximum Likelihood

If the Hamming weight of POI (Point of Interesting) is estimated with only one trace, it is very difficult to correct the error due to the noise. However, conducting an analysis using the maximum likelihood method [18] can improve the performance of the attack. This can be attributed to the fact that the maximum likelihood reflects the influence of the noise with probability. The results of the analysis of the maximum likelihood method are shown in Figure 5.

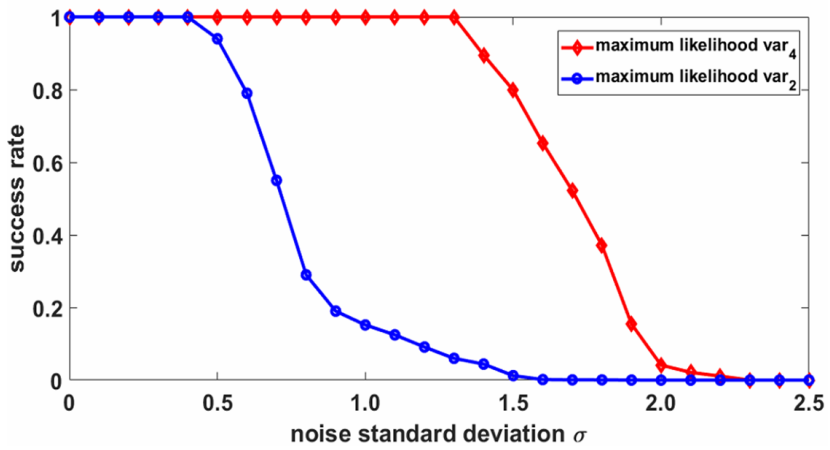

Figure 5. Success rate according to noise standard deviation (using maximum likelihood).

In the case of using two variables, the success rate of the attack remains at 1 up to a noise standard deviation of 0.4, which results in a smaller but better result than the result of the method shown in Figure 3. In the case of using four variables, the success rate of the attack remain at 1 up to a noise standard deviation of 1.3, which is about twice as high as the analysis result obtained using the best result HMD performance.

4.3. Find α_0

Figure 6 shows the success rate of the attack according to the noise standard deviation when the horizontal correlation power analysis is conducted with 256 partial traces to identify α_0. The success rate of the attack was measured by increasing the noise standard deviation from 0 to 2.5 in increments of 0.1. The horizontal correlation power analysis was performed 10,000 times according to the noise standard deviation. The success rate of the attack remains at 1 up to a noise standard deviation of 2.7, and then starts to decrease from 2.8.

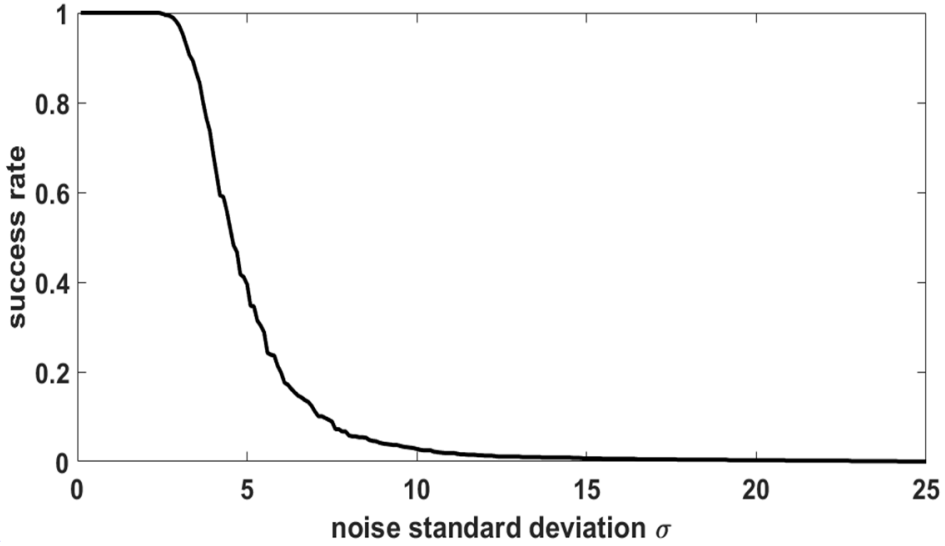

Figure 6. Success rate according to noise standard deviation ($\alpha_0 = 999$).

4.4. Experiments Using Multiple Traces

In the previous experiments, only one trace was used for analysis. The experimental results obtained using the maximum likelihood yielded an analysis success rate of 1 up to a noise standard deviation of 1.3. However, the noise standard deviation of 1.3 is a small value in a typical actual analysis environment, and it may be difficult to analyze according to the trace collection environment. Therefore, we experimentally confirmed that the performance of the attack can be improved by using additional traces. For each experiment with one trace, we accumulate the scores from the most probable candidate key to the least probable candidate key. More concretely, 5–1 points were accumulated from the first candidate key to the fifth candidate key.

Figure 7 shows the experimental result obtained using 10 traces. The noise standard deviation was increased by 0.1. The analysis was conducted 10000 times for each noise standard deviation. The analysis performed for each trace is the maximum likelihood when using four variables. We performed a maximum likelihood analysis for each trace and used four random variables. The attack success rate remains at 1 up to a noise standard deviation of 2.1, and then begins to decrease from 2.2. In this paper, we experimented with 10 traces, but we can obtain better experimental results by using more traces.

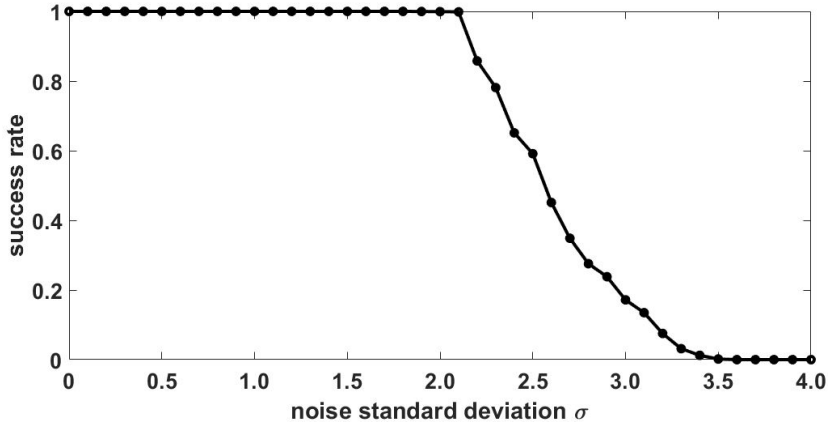

Figure 7. Success rate according to noise standard deviation (using 10 power traces).

5. Conclusions

In this paper, we analyzed HyMES by exploiting the joint distributions of leakages. We exploited the nonlinearity of the precomputation table used in calculating the column vector h_i of the parity check matrix H in the decryption process. Previously, the side-channel analysis by exploiting the joint distributions of leakages was only investigated for the symmetric key cryptography. To the best of our knowledge, this is the first to analyze the public-key cryptosystem exploiting the joint distributions of leakages. The experimental results using a single simulated power trace show a success rate of 100% up to a noise standard deviation 1.3. We demonstrated that the proposed attack can be applied to all code-based cryptosystems that calculate h_i in the manner proposed in Reference [6].

Author Contributions: All authors have contributed to this work. B.P. and S.K. analyzed the algorithm and drafted and revised the manuscript. B.P. and S.K. performed the experiment and analyzed the result. B.P. and S.H. devised a methodology of additional experiment. H.K.K. and S.C.S. verified the analytical methods and supervised this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2019R1A2C2088960).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
2. Nguyen, H.B. An Overview on the Ntru Cryptographic System. Ph.D. Thesis, San Diego State University, San Diego, CA, USA, 2014.
3. Moody, D. The NIST Post-Quantum Crypto "Competition". Available online: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/asiacrypt-2017-moody-pqc.pdf (accessed on 12 December 2017).
4. Mceliece, R.J. A public-key cryptosystem based on algebraic. *Coding Theory* **1978**, *4244*, 114–116.
5. Hudde, H.C. Development and Evaluation of a Code-Based Cryptography Library for Constrained Devices. Ph.D. Thesis, Ruhr Universitat Bochum, Bochum, Germany, 2013.
6. Biswas, B.; Sendrier, N. McEliece Cryptosystem Implementation: Theory and Practice. In *International Workshop on Post-Quantum Cryptography*; Springer: Berlin, Germany, 2008; pp. 47–62.
7. Strenzke, F.; Tews, E.; Molter, H.G.; Overbeck, R.; Shoufan, A. Side channels in the McEliece PKC. In *International Workshop on Post-Quantum Cryptography*; Springer: Berlin, Germany, 2008; pp. 216–229.
8. Strenzke, F. A timing attack against the secret permutation in the McEliece PKC. In *Post-Quantum Cryptography*; Springer: Berlin, Germany, 2010; pp. 95–107.
9. Heyse, S.; Moradi, A.; Paar, C. Practical power analysis attacks on software implementations of McEliece. In *International Workshop on Post-Quantum Cryptography*; Springer: Berlin, Germany, 2010; pp. 108–125.

10. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO '96*; Springer: Berlin, Germany, 1996; pp. 104–113.
11. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology—CRYPTO' 99*; Springer: Berlin, Germany, 1999; pp. 388–397.
12. Bernstein, D.J.; Chou, T.; Schwabe, P. McBits: Fast constant-time code-based cryptography. In *Cryptographic Hardware and Embedded Systems—CHES 2013*; Springer: Berlin, Germany, 2013; pp. 250–272.
13. Linge, Y.; Dumas, C.; Lambert-Lacroix, S. Using the joint distributions of a cryptographic function in side channel analysis. In *Constructive Side-Channel Analysis and Secure Design*; Springer: Cham, Switzerland, 2014; pp. 199–213.
14. Sidelnikov, V.M.; Shestakov, S.O. On insecurity of cryptosystemsbased on generalized Reed-Solomon codes. In *Discrete Mathematics and Applications*; Walter de Gruyter: Berlin, Germany, 1992; pp. 439–444.
15. Marcus, M. White Paper on McEliece with Binary Goppa Codes. Available online: https://www.hyperelliptic.org/tanja/students/m_marcus/whitepaper.pdf (accessed in 28 February 2019).
16. Chari, S.; Rao, J.R.; Rohatgi, P. Template attacks. In *Cryptographic Hardware and Embedded Systems—CHES 2002*; Springer: Berlin, Germany, 2003; pp. 13–28.
17. Schramm, K.; Leander, G.; Felke, P.; Paar, C. A Collision-Attack on AES Combining Side Channel and Differential-Attack. In *Cryptographic Hardware and Embedded Systems—CHES 2004*; Springer: Berlin, Germany, 2004; pp. 163–175.
18. Clavier, C.; Reynaud, L. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In *Cryptographic Hardware and Embedded Systems—CHES 2017*; Springer: Cham, Switzerland, 2017; pp. 24–44.