Cryptanalysis of the $m-$Permutation Protection Schemes

Hongjun Wu, Feng Bao, Dingfeng Ye, and Robert H. Deng

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singpore 119613
{hongjun, baofeng, dfye, deng}@krdl.org.sg

Abstract. Anderson and Kuhn have proposed the EEPROM modification attack to recover the secret key stored in the EEPROM. At ACISP'98, Fung and Gray proposed an $m-$permutation protection scheme against the EEPROM modification attack. At ACISP'99, Fung and Gray pointed out that in their original scheme, a secret key with too small or too large Hamming weight could be recovered easily. Then they proposed a revised $m-$permutation protection scheme and claimed that their revised scheme does not leak any information of the secret key. In this paper, we break completely both the original and the revised $m-$permutation protection schemes. The original scheme is broken with about $2\log_2 n$ devices from the same batch and about $(3\log_2 n+2)\times m\times n$ probes (n is the length of the secret key and m is the amount of permutations). The revised $m-$permutation protection scheme is more vulnerable than the original one. It could be broken with only one device and about $m\times n^3/3$ probes.

1 Introduction

The design of tamperproof device is an important issue in the applications of cryptographic systems. There are basically two types of attacks against the tamperproof devices. The direct attack is to reverse engineer the device with advanced hardware technology. Another type of attacks is to force the device to produce computational errors. Boneh, DeMillo and Lipton have developed such an attack against tamperproof device [4]. In their attack random errors are introduced into the data on the device. The random errors cause a corresponding erroneous output that can be used to deduce the key. This attack is simple but powerful and is able to break the devices using RSA. A similar attack was reported independently by Bao, Deng et. al. who showed how to attack the RSA, El Gamal and Schnorr Signature schemes [2]. Biham and Shamir later introduced the Differential Fault Analysis or DFA [3]. DFA can be applied to recover a block cipher key from a sealed tamperproof device. To resist these fault-related attacks the device needs to perform fault checking before outputting the encrypted (or decrypted, signed) result.

Anderson and Kuhn introduced the EEPROM modification attack that is quite general and practical. In their attack, an attacker is assumed to be able to write

E. Dawson, A. Clark, and C. Boyd (Eds.): ACISP 2000, LNCS 1841, 97-111, 2000.

arbitrary values to arbitrary locations of the EEPROM, where the secret key is stored, but cannot read a value from the EEPROM. This is because the cost of writing a value to EEPROM is much lower than that of reading a value from EEPROM, i.e., the writing can be done with the low-cost equipment, such as microprobes, while the reading requires much more expensive equipment, such as an electro-optical probe.

To protect the device against the EEPROM modification attack, Fung and Gray proposed a cascaded $m-$permutation scheme that uses an $(m\times n)-$bit encoding for an $n-$bit key [5]. Each batch of devices employs the same permutations (i.e., encoding). The permutation wiring is secret and it is assumed that the attacker has no equipment to reveal the wiring. Fung and Gray claimed that the attack on the $m-$permutation scheme requires $O(n^m)$ probes to compromise the key. In [6], Fung and Gray pointed out that if the Hamming weight of a key is too small or too large, the key could be recovered easily. Then they introduced the revised scheme in which random numbers are introduced to hide the information about the Hamming weight of the secret key.

In this paper, we show that both the original and the revised schemes are not secure. For the original scheme, there exists an attack that could recover the m permutations with about $2\log_2 n$ devices from the same batch and about $(3\log_2 n+2)\times m\times n$ probes. The $m-$permutation scheme achieves only linear growth of complexity with a linear growth of the amount of permutations.

In the revised scheme, m random numbers are introduced into each device to hide the information of the Hamming weight of the secret key. However, these random numbers leak the information about the secret permutations. By modifying these random numbers, we could recover the mappings between those permutations, i.e., the $m-$permutation scheme could be reduced to one permutation scheme. Thus with only one device, we break the revised $m-$permutation scheme with $m\times n^3/3$ probes. Since only one device is needed in this attack, we consider that the revised scheme is more vulnerable than the original scheme. We try to strengthen the revised scheme by eliminating the flaw introduced by the random numbers. However, there still exist an attack that could recover those m permutations with about $m\times n$ devices from the same batch and about $3.5(m\times n)^2$ probes.

A fairly simple and efficient scheme to defeat the EEPROM modification attack is proposed in this paper. By restricting the Hamming weight of the key to be half of n, only one permutation is needed.

This paper is organized as follows. The EEPROM modification attack is introduced in Section 2. Fung and Gray's original and revised $m-$permutation protection schemes are introduced in Section 3. We break the original and the revised $m-$permutation scheme in Section 4 and Section 5, respectively. In Section 6, we break the strengthened version of the revised scheme. Section 7 gives our simple and efficient protection scheme. Section 8 concludes the paper.

2 The EEPROM Modification Attack

In [1], Anderson and Kuhn proposed the EEPROM modification attack. It is a physical attack in which two microprobing needles are used to set or clear target bits in order to infer them. It is assumed that EEPROM bits cannot be read directly since the equipment required is much more expensive than the microprobing needles. In the EEPROM modification attack, if one bit of the secret key is set correctly, there would be no error in the output of the device; otherwise, error occurs. The secret key can thus be determined bit by bit.

Anderson and Kuhn's attack in [1] is with respect to a DES key. The more general attack described by Fung and Gray [5] is given below:

for $i = 0$ to n-1
    set the i^{th} bit to 1;
    operate the device;
    if the device gives the correct output, then conclude that the bit is 1; otherwise, conclude that the bit is 0 and reset it to 0.

In addition to requiring only low-cost equipment, this attack can be carried out with very few probing actions. In particular, it takes $1.5n$ probes on the average to recover an $n-$bit key.

3 The m – Permutation Protection Schemes

The $m-$ permutation schemes [5,6] provide a physical encoding (m permutations) of keys, along with a logical chip design and hiding the permutation wiring beneath the surface of the chip. The m permutations are considered as the „batch key" which is known only to the manufacturers and to those who are legitimately programming the device. For example, the devices may be manufactured in batches of 10,000 devices all with the same batch key. A single customer purchases a batch of devices and is given the batch key so that he can program secret keys into the cards.

There are several assumptions made. Firstly, the attacker is assumed to be a „clever outsider with moderately sophisticated equipment". Secondly, the encoded key is assumed to be stored in EEPROM and that the attacker cannot read the EEPROM directly. Finally, it is assumed that the attacker is not able to see the exact wiring (i. e., the batch key) of the devices. The following notations are used in the rest of this paper:

K :	The actual key bit vector with length of n bits. It is to be used by the card in encrypting, signing, etc.
P :	The physical key bit vector with length of p bits. It is the actual bit pattern stored in the EEPROM.
π :	A permutation function, $\pi : \{0,1,2,\cdots,n-1\} \rightarrow \{0,1,2,\cdots,n-1\}$

π^{-1}: The inverse function of the permutation π

Element of π: An element of the permutation table $(i, \pi(i))$ ($i \in \{0, 1, \cdots, n-1\}$).

$\pi(K)$: It denotes the permuted result of K under the operation of π, i.e., $(\pi(K))_{\pi(i)} = (K)_i$ for $0 \le i \le n-1$, where $(K)_i$ denotes the value of the i^{th} bit of K.

3.1 One-Permutation Scheme [5]

In this approach, the manufacturer chooses a random permutation of the n-bit key as the batch key and works as follows:

Scheme 1. One-permutation Scheme

1. The device manufacturer chooses randomly a permutation π
2. Set $P = \pi(K)$
3. The wiring implements the inverse of π
4. The device reads K from P since $\pi^{-1}(P) = \pi^{-1}(\pi(K)) = K$.

The attacker can find the value of P as described in Section 2. Then the permutation π can be determined as follows. The attacker first sets P as a vector with Hamming weight one, then operates the device to obtain an output. The value of K can be determined since the Hamming weight of K is also one due to the fact that $K = \pi^{-1}(P)$. Thus one element of π is known. Repeat this process for $n-1$ times, the permutation π can be determined with about $3n$ probes.

3.2 m – Permutation Protection Scheme [5]

In the m – permutation scheme, the manufacturer chooses m random permutations of the n – bit key as the batch key and works as follows:

Scheme 2. m – permutation protection scheme

1. The device manufacturer chooses m permutations: $\pi_0, \pi_1, \cdots, \pi_{m-1}$
   $\pi_i : \{0, 1, 2, \cdots, n-1\} \to \{0, 1, 2, \cdots, n-1\}$ for $0 \le i \le n-1$
2. Let $P = P_0 \,|\, P_1 \,|\, \ldots \,|\, P_{m-1}$, where $|$ denotes concatenation and $P_i = \pi_i(K)$.
3. The wiring implements the inverse of those m permutations.
4. The device reads K_0 from P_0, K_1 from P_1, …, K_{m-1} from P_{m-1}.
   If $K_0 = K_1 = \ldots = K_{m-1}$ is not true, the device gives an error message.

The attack to break the one-permutation scheme could not be applied directly to the $m-$ permutation scheme since without knowing the secret permutations, the modified values of P_i can pass the detection with negligible probability.

As pointed out by Fung and Gray, some secret key with small Hamming weight could be recovered easily [5]. To eliminate such weakness, Fung and Gray proposed the revised $m-$ permutation scheme [6] to hide the information about the Hamming weight of the secret key by introducing m random numbers into each device. The revised scheme is given in the next subsection.

3.3 The Revised m – Permutation Protection Scheme

In the revised $m-$ permutation scheme, m random numbers are introduced into each device.

Scheme 3. Revised $m-$ permutation protection scheme

1. Choose m as an odd number.
2. The device manufacturer chooses m permutations $\pi_0, \pi_1, \cdots, \pi_{m-1}$ as the secret information (batch key) for a batch of devices.
   $\pi_i : \{0,1,2,\cdots,n-1\} \to \{0,1,2,\cdots,n-1\}$ for $0 \le i \le n-1$
3. Randomly choose m $n-$ bit words $K_{D_0}, K_{D_1}, \cdots, K_{D_{m-1}}$ for each device.
4. Store in the device $P = P_0 \,|\, P_1 \,|\, \ldots \,|\, P_{m-1}$ where
   $P_i = \pi_i(K \oplus \pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}}) \oplus \pi_{i+2 \bmod m}(K_{D_{i+2 \bmod m}}))$
5. Store in the device $P_D = P_{D_0} \,|\, P_{D_1} \,|\, \ldots \,|\, P_{D_{m-1}}$ where $P_{D_i} = K \oplus K_{D_i}$.
6. The wiring implements the inverse of those m permutations.
7. The device decodes the key as $K = \bigoplus_{i=0}^{m-1} \pi_i^{-1}(P_i)$
8. The device then computes

$$K_{and} = \bigwedge_{i=0}^{m-1} (\pi_i^{-1}(P_i) \oplus \pi_{i+1 \bmod m}(P_{D_{i+1 \bmod m}} \oplus K) \oplus \pi_{i+2 \bmod m}(P_{D_{i+2 \bmod m}} \oplus K))$$

$$K_{or} = \bigvee_{i=0}^{m-1} (\pi_i^{-1}(P_i) \oplus \pi_{i+1 \bmod m}(P_{D_{i+1 \bmod m}} \oplus K) \oplus \pi_{i+2 \bmod m}(P_{D_{i+2 \bmod m}} \oplus K))$$

where $\wedge$ and $\vee$ indicates logical AND and OR respectively.
If $K_{and} = K_{or} = K$, then the device uses K in the crypto application; else return an error message.

In the revised scheme (Scheme 3), the Hamming weight of the secret key is unknown after P and P_D being recovered under the EEPROM modification attack. Fung and Gray claimed that the attacker has only a probability of $2^{-(n-1)}$ to guess the $n-$ bit key

K since the m permutations are unknown to the attacker. However, as we will present in Section 5, there exists an attack that can recover those permutations with only one device and about $m \times n^3 / 3$ probes. Once those permutations are recovered, the whole batch of devices is broken and the secret keys can be determined easily.

4 Cryptanalysis of the Original $m-$Permutation Protection Scheme

Fung and Gray have pointed out a weakness in their original $m-$permutation protection scheme that some keys with small or large Hamming weight could be recovered easily. In this section, we present an attack to break completely the original scheme with about $2\log_2 n$ devices from the same batch and about $3 \times m \times n \times \log_2 n$ probes. Our attack consists of two steps. In the first step, we determine the mappings between those m permutations by analyzing $2\log_2 n$ devices, i.e., we reduce the $m-$ protection scheme to one-permutation scheme. In the second step, we recover the remaining permutation. We start with the first step.

Assume that about $2\log_2 n$ devices from the same batch are available and the values of P $(P = P_0 \mid P_1 \mid ... \mid P_{m-1})$ in these devices are determined already by applying the EEPROM modification attack. The amount of probes needed here is about $3 \times m \times n \times \log_2 n$. We denote P_j^i as the value of P_j in the ith device.

We know that $P_j^i = \pi_j \circ \pi_0^{-1}(P_0^i)$ since $P_j^i = \pi_j(K^i)$ and $P_0^i = \pi_0(K^i)$ (K^i is the secret key in the ith device). The permutation $\pi_j \circ \pi_0^{-1}$ is determined as follows. Consider two $(2\log_2 n) \times n$ binary matrices M and N with the ith row $M^i = P_j^i$ and $N^i = P_0^i$. We note that $M^i = \pi_j \circ \pi_0^{-1}(N^i)$, i.e., M is obtained by exchanging the columns of N under the permutation $\pi_j \circ \pi_0^{-1}$. Clearly, if all the columns of N are different, the permutation $\pi_j \circ \pi_0^{-1}$ can be determined uniquely. Assume that all the keys are randomly generated, then the columns of the matrix N are n random elements in a set with n^2 elements. From the birthday paradox, the probability that all these elements are different is about 0.61 (for almost all the key length $40 \le n \le 4096$). Thus the permutations $\pi_j \circ \pi_0^{-1}$ $(0 < j \le m-1)$ could be uniquely determined with about $2\log_2 n$ devices with probability about 0.61. If a few elements of the permutations $\pi_j \circ \pi_0^{-1}$ $(0 < j \le m-1)$ could not be recovered (i.e., some columns of the matrix N are with the same value), some key bits would be unknown. However, those key bits can be determined easily by exhaustive search. In the rest of this section, we simply assume that $\pi_j \circ \pi_0^{-1}$ $(0 < j \le m-1)$ are uniquely determined already. Then there is only one unknown permutation left, i.e., if we can find π_0, we will know all the π_i since $\pi_i = (\pi_i \circ \pi_0^{-1}) \circ \pi_0$. We give below the details to recover π_0.

To recover π_0, we need to write a key with Hamming weight one into the device correctly so that $\pi_0^{-1}(P_0)=\pi_1^{-1}(P_1)=\cdots=\pi_{m-1}^{-1}(P_{m-1})$. If P_0 is set as an $n-$bit word with Hamming weight one, then P_i could be determined easily since $P_i=\pi_i\circ\pi_0^{-1}(P_0)$ and $\pi_i\circ\pi_0^{-1}$ is known already. Thus we are able to write any key with Hamming weight one into the device. Once knowing the device output, the value of the key with Hamming weight one could be determined. Since $P_0=\pi_0(K)$, one element of π_0 is determined. Set the bit with value 1 at different positions in P_0 and repeat the attack, we could recover π_0 with about $2\times m\times n$ probes.

From π_0 and $\pi_i\circ\pi_0^{-1}$ $(i=1,\cdots,m-1)$, we know all the permutations and thus can break the $m-$protection scheme. About $2\log_2 n$ devices are needed in this attack and the total amount of probes needed is about $3m\times n\times\log_2 n+2m\times n$ $=(3\log_2 n+2)\times m\times n$.

The attack in this section needs about $2\log_2 n$ devices from the same batch. In case devices from a number of batches are well mixed, a simple method could be used to group those devices. We write the P of one device into all the devices, then those devices that operate properly belong to the same batch.

5 Cryptanalysis of the Revised $m-$Permutation Protection Scheme

In the revised scheme, m random numbers are introduced into each device to hide the information of the Hamming weight of the secret key. However, the revised scheme is in fact more vulnerable than the original one since those random numbers leak the information about the permutations. With only one device, those permutations could be recovered with about $m\times n^3/3$ probes. Similar to the attack in Section 4, the attack in this section consists of two steps. The first step of the attack is to reduce the $m-$protection scheme to one-permutation scheme by modifying the random numbers. The second step is to recover the remaining permutation. We start with the first step.

Assume that the values of P and P_D in a device are determined already by applying the EEPROM modification attack. We note that those m numbers K_{D_i} $(i=0,1,\cdots,m-1)$ in the revised scheme are randomly chosen. Obviously, if we replace any K_{D_i} with another random number, the value of the secret key K will not be affected and the device will operate properly. Suppose we want to modify the jth bit of a particular random number K_{D_i}. This bit is denoted as $K_{D_{i,j}}$. We know that this bit appears only in P_{D_i}, $P_{i-1 \bmod m}$ and $P_{i-2 \bmod m}$ since

$$P_{D_i}=K\oplus K_{D_i}, \tag{1}$$

$$P_{i-1 \bmod m} = \pi_{i-1 \bmod m}(K \oplus \pi_i(K_{D_i}) \oplus \pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}})) \quad (2)$$

$$P_{i-2 \bmod m} = \pi_{i-2 \bmod m}(K \oplus \pi_{i-1 \bmod m}(K_{D_{i-1 \bmod m}}) \oplus \pi_i(K_{D_i})) \quad (3)$$

Modifying the bit $K_{D_{i,j}}$ in P_{D_i} is trivial since we only need to invert the value of $P_{D_{i,j}}$. Modifying $K_{D_{i,j}}$ in $P_{i-1 \bmod m}$ and $P_{i-2 \bmod m}$ without knowing the permutation $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ requires about $n^2/2$ trials on average. Thus with about n^2 probes, we could modify the value of $K_{D_{i,j}}$ successfully (If it is not modified correctly, the device gives error message). If the values of $P_{D_{i,j}}$, $P_{i-1 \bmod, j'}$ and $P_{i-2 \bmod, j''}$ are modified and the device operates properly, we know from (1), (2) and (3) that

$$\pi_{i-1 \bmod m} \circ \pi_i(j) = j', \quad \pi_{i-2 \bmod m} \circ \pi_i(j) = j''$$

We thus determined one element of $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$. Repeat this attack for the rest bits of K_{D_i}, $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ are recovered with about $\sum_{i=2}^{n} i^2$ probes. Similar attack can be applied to recover the permutations $\pi_{i-1 \bmod m} \circ \pi_i$ and $\pi_{i-2 \bmod m} \circ \pi_i$ $(i = 0,1,\cdots,m-1)$. From these permutations, $\pi_i \circ \pi_0^{-1}$ $(i = 1,\cdots,m-1)$ are obtained as follows:

$$\begin{aligned}
\pi_i \circ \pi_0^{-1} &= (\pi_i \circ \pi_{i-1}^{-1}) \circ (\pi_{i-1} \circ \pi_{i-2}^{-1}) \cdots (\pi_1 \circ \pi_0^{-1}) \\
&= (\pi_i \circ \pi_{i+1 \bmod m} \circ \pi_{i+1 \bmod m}^{-1} \circ \pi_{i-1}^{-1}) \circ (\pi_{i-1} \circ \pi_i \circ \pi_i^{-1} \circ \pi_{i-2}^{-1}) \\
&\quad \cdots (\pi_1 \circ \pi_2 \circ \pi_2^{-1} \circ \pi_0^{-1}) \\
&= (\pi_i \circ \pi_{i+1 \bmod m}) \circ (\pi_{i-1} \circ \pi_{i+1 \bmod m})^{-1} \circ (\pi_{i-1} \circ \pi_i) \circ (\pi_{i-2} \circ \pi_i)^{-1} \\
&\quad \cdots (\pi_1 \circ \pi_2) \circ (\pi_0 \circ \pi_2)^{-1}
\end{aligned} \quad (4)$$

After $\pi_i \circ \pi_0^{-1}$ $(i = 1,\cdots,m-1)$ being recovered, only π_0 remains to be recovered. To recover π_0, we need to write a key with Hamming weight one into the device correctly, i.e., the values of K and $K_{D_i}(i = 0,1,\cdots,m-1)$ should be set correctly in P and P_D. We deal first with K. We choose K as an $n-$bit word with Hamming weight one. It appears in $P_i(i = 0,1,\cdots,m-1)$ and $P_{D_i}(i = 0,1,\cdots,m-1)$. The value of K in P_{D_i} is K itself since $P_{D_i} = K \oplus K_{D_i}(i = 0,1,\cdots,m-1)$. But K_0, the value of K in P_0, is unknown since $K_0 = \pi_0(K)$ and π_0 is unknown. We randomly set K_0 as an $n-$bit word with Hamming weight one. The probability that $K_0 = \pi_0(K)$ is n^{-1}. After setting the value of K in P_0, the value of K in P_i could be determined since $K_i = \pi_i(K) = \pi_i \circ \pi_0^{-1}(K_0)$ and $\pi_i \circ \pi_0^{-1}$ is known from (4). Thus the values of K in P and P_D are determined with probability n^{-1}. We then deal with the random

numbers $K_{D_i}(i=0,1,\cdots,m-1)$. The simplest way is to set their values as zero. Then their values in P and P_D are zero. Now, we are able to write a key with Hamming weight one into a device with success rate n^{-1}. Once the device operates properly, we know that the key is written successfully into the device. If that happens and the bit K^j and $K_0^{j'}$ are with value one, then $j'=\pi_0(j)$, i.e., one element of π_0 is recovered. The amount of probes needed is about $m\times n$. Repeat this attack, we could finally recover π_0 with about $\sum_{i=2}^{n} m\times i$ probes.

After recovering π_0 and $\pi_i\circ\pi_0^{-1}(i=1,\cdots,m-1)$, we break the revised $m-$ protection scheme completely. Only one device is needed in this attack and the total amount of probes needed is about $\sum_{i=2}^{n} m\times i^2+\sum_{i=2}^{n} m\times i\approx m\times n^3/3$.

In the next section, we will discuss whether the revised protection scheme could be strengthened or not. Our analysis gives negative result.

6 Is It Possible to Strengthen the Revised Scheme

The attack in Section 5 is based essentially on the fact that each random number appears at only three locations in the EEPROM. The mappings between the permutation tables could be determined by modifying one by one the bit of those random numbers. To resist the attack in Section 5, each random number should appear at far more than three locations in the EEPROM. For example, the P_i in the revised scheme can be modified as

$$P_i=\pi_i(K\oplus\pi_{i+1 \bmod m}(K_{D_{i+1 \bmod m}})\oplus\pi_{i+2 \bmod m}(K_{D_{i+2 \bmod m}})\cdots\oplus\pi_{i+8 \bmod m}(K_{D_{i+8 \bmod m}}),$$

then each random number appears at eight positions in the EEPROM. However, the revised scheme strengthened in this way is still not secure. We can recover those permutation tables if about $m\times n$ devices from the same batch are available. In the rest of this section, we present a new attack to break only Scheme 3, but the same attack can be applied to the scheme where each random number appears at a number of locations in the EEPROM.

Assume that about $m\times n$ devices from the same batch are available and the values of P $(P=P_0\,|\,P_1\,|\ldots|\,P_{m-1})$ and P_D $(P_D=P_{D_0}\,|\,P_{D_1}\,|\ldots|\,P_{D_{m-1}})$ in these devices are determined already by applying the EEPROM modification attack. The amount of probes needed here is about $3\times(mn)^2$. We denote P^i and P_D^i as the values of P and P_D in the ith device, respectively.

Our aim is to write a P' with Hamming weight one into the device. Fung and Gray have considered this kind of attack and concluded that it is impossible to apply it to break their revised scheme [6]. They consider that if a P' with Hamming weight one is written into the EEPROM and the value of P_D' is randomly set, then the probability

that the device could operate properly (no error message) is negligibly small (about $2^{-m\times n}$). However, with about $m\times n$ devices from the same batch, it is possible to construct a right pair (P' , P'_D) (a right pair means that with which the device operates properly and gives no error message). The method to construct such a pair is given below.

Algorithm 1. This algorithm is to construct a pair (P', P'_D) for any given P'. It needs $m\times n$ devices from the same batch.

1. Form two $mn\times mn$ binary matrices M and N with the ith column $M_i = (P^i)^T$ and $N_i = (P_D^i)^T$.
2. Solve the linear equations $M \cdot x^T = P'^T$. Let $P'_D = (N \cdot x^T)^T$.
3. The pair (P', P'_D) is the one we need.

Then we need to show: 1) the equation $M \cdot x^T = P'^T$ could be solved, i.e., the matrix M is invertible with large probability, 2) with the pair (P', P'_D) generated in Algorithm 1, the device operates properly.

To show that the Matrix M is invertible with large probability, we start with the following theorem.

Theorem 1. In Scheme 3 (the revised protection scheme), assume that all the keys and random numbers in the devices are generated independently and randomly. Choose $m\times n$ devices from the same batch. Form an $mn\times mn$ binary matrix M, with the ith column $M_i = (P^i)^T$. Then the matrix M is a random matrix.

The proof of Theorem 1 is given in the Appendix. In theorem 1, we deal only with Scheme3. But the same result could be obtained if each random number appears at more than three locations in the EEPROM.

From Theorem 1, we know matrix M is randomly generated. So it is invertible with probability about 0.29. With slightly more than $m\times n$ devices, an invertible matrix M could be formed. So the pair (P', P'_D) in Algorithm 1 can be obtained.

Then we show below that with the pair (P', P'_D) generated in Algorithm 1, the device operates properly.

Theorem 2. In Scheme 3, choose any n devices from the same batch. Let $P' = \bigoplus_{i=0}^{n-1} P^i$ and $P'_D = \bigoplus_{i=0}^{n-1} P_o^i$. If P' and P'_D are written into the device, the device will operate properly (no error message is given).

Proof. Since $P' = \bigoplus_{i=0}^{n-1} P^i$ and $P'_D = \bigoplus_{i=0}^{n-1} P^i_{D_i}$, it is equivalent to encode a key $K' = \bigoplus_{i=0}^{n-1} K^i$ with m random numbers $K'_{D_j} = \bigoplus_{i=0}^{n-1} K^i_{D_j}$ $(j = 0,1,\cdots,m-1)$. Thus, the key K' will be decoded correctly and the device will operate properly.

From the discussion above, we know that from slightly more than $m \times n$ devices, a right pair (P', P'_D) in which the Hamming weight of P' is one could be obtained easily. Once we obtained such a pair, we could recover one element of those permutation tables as follows. Suppose only the bit $P'_{i,j}$ in P' is with value one. From Scheme 3, we know that the key K' is decoded as $K' = \bigoplus_{l=0}^{m-1} \pi_l^{-1}(P'_l) = \pi_i^{-1}(P'_i)$. So the Hamming weight of K' is only one. By analyzing the output of the device, the value of K' can be determined. Suppose the bit with value one in K' is $K'_{j'}$, then $j' = \pi_i^{-1}(j)$, i.e., one element of π_i is recovered. Set the non-zero bit at other positions in P' and repeat this attack, we can finally recover all the permutation tables. The amount of probes needed is about $mn \times (0.5mn) = 0.5 \times (m \times n)^2$.
The attack in this section thus break the revised protection scheme even if the scheme allows each random number appearing at more than three locations. It needs slightly more than $m \times n$ devices from the same batch. The total amount of probes needed is about $3 \times (mn)^2 + 0.5 \times (mn)^2 = 3.5 \times (mn)^2$.

7 How to Prevent the EEPROM Modification Attack

We now know that all the $m-$ permutation schemes are not secure. The flaw in those schemes is that those m permutations could be reduced to one permutation. We note that all the attacks in this paper have one common step: a key with Hamming weight one is written into the EEPROM to recover the permutation table element by element. To hide the permutation, we believe that the most essential way is to disallow a key with too small (or too large) Hamming weight being written into the device. Based on this observation, we give below a fairly simple and efficient scheme to resist the EEPROM modification attack.

Scheme 4. This scheme protects an $n-$bit secret key against the EEPROM modification attack with the use of only one $n-$bit permutation. It is the strengthened version of the one-permutation scheme given in Subsection 3.1.

1. Choose a permutation π as the batch key.
2. Choose the secret key K with Hamming weight $n/2$.
3. Let $P = \pi(K)$ and write P into the EEPROM.
4. The wiring implements the inverse of π.

5. The device reads K from P. If the Hamming weight of K is $n/2$, K is accepted; otherwise, the device gives error message.

The value of P could be recovered by applying the EEPROM modification attack. However the permutation π could not be recovered. The reason is that the output of π (the value of P) is known, but the input of π (the value of K) is unknown. By applying the EEPROM modification attack, the secret key could only be recovered by exhaustive search and the complexity is $0.5\times\binom{n}{n/2}$. For $n=128$, the complexity is about $2^{122.17}$ and it is sufficient to defeat the exhaustive key search. We thus believe that Scheme 4 is sufficient to resist the EEPROM modification attack. However, it should be noted that any compromise of the secret key degrades the security of the devices of the same batch.

As pointed out by the anonymous referee, some public key cryptosystems such as RSA do not allow the Hamming weight to be controlled. Scheme 4 could not be used to protect the private keys in these cryptosystems. To resist the EEPROM modification attack, we recommend the use of a hash function. The one-permutation scheme in Subsection 3.1 is used to protect the key together with its hashed value. The device hashes the key and compares the result with the hashed value stored in the EEPROM. If these two values are equal, the key is used in the crypto applications; otherwise, the device gives error message.

8 Conclusion

In this paper, we showed that Fung and Gray's original and revised $m-$ permutation schemes are not secure. We then proposed a very simple and efficient scheme to resist the EEPROM modification attack by allowing only the key with Hamming weight $n/2$ being written into the device.

Acknowledgement

We would like to thank the anonymous referees of ACISP 2000 for their helpful comments.

References

1. R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices", in *Security protocols: International Workshop'97*, LNCS 1361, Springer-Verlag, pp.125-136, 1997.

2. F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimhaly and T. Ngair, "Breaking Public Key Crystosystems on Tamper Resistant Devices in the Presence of Transient Faults", in *Security Protocols: International Workshop'97*, LNCS 1361, Springer-Verlag, 1997.
3. E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", in *Advances in Cryptology - Crypto'97*, LNCS 1294, Springer-Verlag, pp. 513-525, 1997.
4. D. Boneh, R.A. Demillo and RJ Lipton, "On the Importance of Checking Cryptographic Protocols for Faults", in *Advances in Cryptology - Eurocrypt'97*, LNCS 1233, Springer-Verlag, pp. 37-51, 1997.
5. W.W. Fung and J.W. Gray, "Protection Against EEPROM Modification Attacks", in *Information Security and Privacy - Proc. of ACISP'98*, LNCS 1438, Springer-Verlag, pp. 250-260, 1998.
6. W.W. Fung and J.W. Gray, "On m-permutation Protection Scheme Against Modification Attack", in *Information Security and Privacy - Proc. of ACISP'99*, LNCS 1587, Springer-Verlag, pp. 77-87, 1999.

Appendix. Proof of Theorem 1

Lemma 1. Consider the operation over $GF(2)$. Let N_1 and N_2 be two $m \times n$ $(m \le n)$ binary matrices. If N_1 is with rank m and each element of N_2 is generated independently and randomly, then $N_1 \cdot (N_2)^T$ is randomly generated.

Proof. Select m linearly independent columns from N_1 and form an $m \times m$ matrix N_1'. The remaining columns form a matrix N_1''. Select any m columns from N_2 and form an $m \times m$ matrix N_2'. The remaining columns form a matrix N_2''. Clearly, the matrix $N_1' \cdot (N_2')^T$ is randomly generated since N_1' is an invertible matrix and N_2' is a random matrix. The matrix $N_1 \cdot (N_2)^T = N_1' \cdot (N_2')^T + N_1'' \cdot (N_2'')^T$, where $N_1' \cdot (N_2')^T$ and $N_1'' \cdot (N_2'')^T$ are two independent matrices since N_2' and N_2'' are independent from each other. So the matrix $N_1 \cdot (N_2)^T$ is randomly generated.

Lemma 2. An $m \times (m+1)$ binary matrix M, if $M_{i,i} = M_{i,i+1} = 1$ for $0 \le i \le m-2$, $M_{m-1,m} = 1$ and all the other elements are with value 0, i.e., M is in the following form:

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

Then the rank of M is m.

Proof. Consider the last m columns of M. They form an $m \times m$ triangular matrix M'. $\det(M') = \prod_{i=0}^{m-1} M'_{i,i} = 1$, i.e., the rank of M' is m. So the rank of M is m.

Lemma 3. For an $m \times m$ binary matrix M, if $M_{i,j} = 1$ for $\{(i, j) \mid i = j$, or $j = m$ or $i = m\}$ and the other elements are with value 0, i.e., M is in the following form:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 1 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \tag{A.2}$$

Then the rank of M is m if and only if m is an odd number.

Proof. Denote an $i \times i$ matrix M in the form of (A.2) as $M_{i \times i}$. The following relationship holds:

$$\det(M_{(i+1)\times(i+1)}) = 1 + \det(M_{i \times i}).$$

Since $\det(M_{1\times 1}) = 1$, we know that $\det(M) = m \bmod 2$. So if and only if m is an odd number, $\det(M) = 1$, i.e., the rank of M is m.

Theorem 1. In Scheme 3 (the revised protection scheme), assume that all the keys and random numbers in the devices are generated independently and randomly. Choose $m \times n$ devices from the same batch. Form an $mn \times mn$ binary matrix M, with the ith column $M_i = (P^i)^T$. The matrix M is randomly generated.

Proof. Let r_j^i $(0 \le i \le m \times n - 1, 0 \le j \le m)$ be $n-$bit binary numbers, $r_j^i = \pi_{j+1 \bmod m}(K^i_{D_{j+1 \bmod m}})$ for $j \le m-1$ and $r_m^i = K^i$ (where $K^i_{D_{j+1 \bmod m}}$ is the $j+1$th random

number in the ith device and K^i is the key in the ith device. Assume $K^i_{D_{j+1 \bmod m}}$ and K^i are generated independently and randomly, so r^i_j $(0 \le i \le m \times n-1, 0 \le j \le m)$ are generated independently and randomly.

Let s^i_j $(0 \le i \le m \times n-1, 0 \le j \le n-1)$ be $(m+1)-$bit binary numbers, the kth bit of s^i_j is determined from $s^i_{j,k} = r^i_{(j\times(m+1)+k-(j\times(m+1)+k \bmod n))/n, j\times(m+1)+k \bmod n}$, i.e., s^i_j $(0 \le i \le m \times n-1, 0 \le j < n)$ are the permuted result from r^i_j $(0 \le i \le m \times n-1,$ $0 \le j \le m)$. Clearly, the elements $s^i_{j,k}$ are generated independently and randomly.

Form an $(m+1)n \times mn$ matrix S with $S_{i,j} = s^j_{(i-i \bmod m+1)/m+1, i \bmod m+1}$. Since every element of S is generated independently and randomly, the probability that S^T with rank mn is about

Form an $mn \times (m+1)n$ matrix T and an $mn \times mn$ matrix P:

$$T = \begin{bmatrix} T_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & T_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & T_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & T_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & T_1 \end{bmatrix} \qquad P = \begin{bmatrix} P_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & P_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & P_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & P_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & P_1 \end{bmatrix}$$

where T_1 is an $m \times (m+1)$ binary matrix in the form of (A.1) and P_1 is an $m \times m$ matrix in the form of (A.2). From Lemma 2 and 3, it is easy to see that both P and T are with rank mn.

Define an $mn \times mn$ matrix M' as $M' = P \cdot T \cdot S$. From Lemma 1, we know that M' is randomly generated since P and T are with rank mn and S is randomly generated.

The matrix M formed in Scheme 3 could be considered as being formed by exchanging the columns of M'. So the matrix M is randomly generated.