Semantically Secure McEliece Public-Key Cryptosystems –Conversions for McEliece PKC–

Kazukuni Kobara and Hideki Imai

Institute of Industrial Science, The University of Tokyo
Roppongi, Minato-ku, Tokyo 106, Japan
TEL : +81-3-3402-6231 Ext 2327
FAX : +81-3-3402-7365
{kobara,imai}@iis.u-tokyo.ac.jp

Abstract. Almost all of the current public-key cryptosystems (PKCs) are based on number theory, such as the integer factoring problem and the discrete logarithm problem (which will be solved in polynomial-time after the emergence of quantum computers). While the McEliece PKC is based on another theory, i.e. coding theory, it is vulnerable against several practical attacks. In this paper, we carefully review currently known attacks to the McEliece PKC, and then point out that, without any decryption oracles or any partial knowledge on the plaintext of the challenge ciphertext, no polynomial-time algorithm is known for inverting the McEliece PKC whose parameters are carefully chosen. Under the assumption that this inverting problem is hard, we propose slightly modified versions of McEliece PKC that can be proven, in the random oracle model, to be semantically secure against adaptive chosen-ciphertext attacks. Our conversions can achieve the reduction of the redundant data down to $1/3 \sim 1/4$ compared with the generic conversions for practical parameters.

1 Introduction

Since the concept of public-key cryptosystem (PKC) was introduced by Diffie and Hellman [5], many researchers have proposed numerous PKCs based on various problems, such as integer factoring, discrete logarithm, decoding a large linear code, knapsack, inverting polynomial equations and so on. While some of them are still alive, most of them were broken by cryptographers due to their intensive cryptanalysis. As a result, almost all of the current (so-called) secure systems employ only a small class of PKCs, such as RSA and elliptic curve cryptosystems, which are all based on either integer factoring problem (IFP) or discrete logarithm problem (DLP). This situation would cause a serious problem after someone discovers one practical algorithm which breaks both IFP and DLP in polynomial-time. No one can say that such an algorithm will never be found. Actually, Shor has already found a (probabilistic) polynomial-time algorithm in [25], even though it requires a quantum computer that is impractical so far. In order to prepare for that unfortunate situation, we need to find another secure scheme relying on neither IFP nor DLP.

The McEliece PKC, proposed by R.J. McEliece in [18], is one of few alternatives[1] for PKCs based on IFP or DLP. It is based on the decoding problem of a large linear code with no visible structure which is conjectured to be an NP-complete problem.[2] While no polynomial-time algorithm has been discovered yet for decoding an arbitrary linear code of large length with no visible structure, a lot of attacks (some of them work in polynomial-time) are known to the McEliece PKC [1,3,4,12,15,28,17,13].

In this paper, we carefully review these attacks in Section 3, and then point out that all the polynomial-time attacks to the McEliece PKC require either decryption oracles or partial knowledge on the corresponding plaintext of the challenge ciphertext. And then without them, no polynomial-time attack is known to invert the McEliece PKC (whose parameters are carefully chosen). Under the assumption that this inverting problem is hard, we convert this problem into semantically secure McEliece PKCs against adaptive chosen-ciphertext attacks (CCA2) by introducing some appropriate conversions. We discuss which conversions are appropriate for the McEliece PKC in Section 4. While some of the generic conversions proposed in [24,9] are also applicable to the McEliece PKC, they have a disadvantage in data redundancy (which is defined by the difference between the ciphertext size and the plaintext size). A large amount of redundant data is needed for the generic conversions since the block size of the McEliece PKC is relatively large. Our conversions in Section 4.4 need less redundant data than the generic ones.

2 McEliece Public-Key Cryptosystem

In this section, we briefly describe the McEliece PKC.

Key generation: Generate the following three matrices G,S,P:

G**:** $k \times n$ generator matrix of a binary Goppa code which can correct up to t errors, and for which an efficient decoding algorithm is known. The parameter t is given by $\lceil \frac{d_{min}-1}{2} \rceil$ where d_{min} denotes the minimum Hamming distance of the code.

S**:** $k \times k$ random binary non-singular matrix

P**:** $n \times n$ random permutation matrix.

Then, compute the $k \times n$ matrix $G' = SGP$.

Secret key: (S, G, P)

Public key: (G', t)

Encryption: The ciphertext c of a message m is calculated as follows:

$$c = mG' \oplus z \qquad (1)$$

[1] Another alternative may be a quantum public-key cryptosystem [21] which will be available after the emergence of quantum computers.

[2] The complete decoding problem of an arbitrary linear code is proven to be NP-complete in [29].

where m is represented in a binary vector of length k, and z denotes a random binary error vector of length n having t 1's.

Decryption: First, calculate cP^{-1}

$$cP^{-1} = (mS)G \oplus zP^{-1} \tag{2}$$

where P^{-1} denotes the inverse of P. Second, apply the decoding algorithm EC for G to cP^{-1}. Since the Hamming weight of zP^{-1} is t, one can obtain mS

$$mS = \mathrm{EC}(cP^{-1}). \tag{3}$$

The plaintext of c is given by

$$m = (mS)S^{-1}. \tag{4}$$

3 Attacks to McEliece PKC

In this section, we review currently known attacks to the McEliece PKC.

While no efficient algorithm has been discovered yet for decomposing G' into (S, G, P) [19], a structural attack has been discovered in [17]. This attack reveals part of structure of a weak G' which is generated from a "binary" Goppa polynomial. However, this attack can be avoided simply by avoiding the use of such weak public keys. (This implies G should not be a BCH code since it is equivalent to a Goppa code whose Goppa polynomial is $1 \cdot x^{2t}$, i.e. "binary". [3]) Next case we have to consider is that an equivalent Goppa code of G' (which is not necessarily G) and whose decoding algorithm is known happens to be found. This probability is estimated in [1][10], and then shown to be negligibly small.

All the other known attacks are for decrypting ciphertexts without breaking public-keys. We categorize them into the following two categories, critical attacks and non-critical attacks, according to whether these attacks can be avoided simply by enlarging the parameter size or not. If avoided, we categorize it in the non-critical attacks. Otherwise, in the critical ones. Interestingly, all the critical attacks require either additional information, such as partial knowledge on the target plaintexts, or an decryption oracle which can decrypt arbitrarily given ciphertexts except the challenge ciphertexts. And then without this additional information and this ability, no efficient algorithm is known to decrypt an arbitrarily given ciphertext of the McEliece PKC.

3.1 Non-critical Attacks

The following two attacks can be avoided simply by enlarging the parameter size or by applying Loidreau's modification [16] without enlarging the parameter size. Thus, not critical.

[3] In [14], a variant of McEliece PKC, where G is a BCH code, was broken. However it is not clear their attack works correctly since further information has failed to appear.

Generalized Information-Set-Decoding Attack. Let G'_k, c_k and z_k denote the k columns picked from G', c and z, respectively. They have the following relationship

$$c_k = mG'_k \oplus z_k. \tag{5}$$

If $z_k = 0$ and G'_k is non-singular, m can be recovered by [1]

$$m = c_k G'^{-1}_k. \tag{6}$$

Even if $z_k \neq 0$, m can be recovered by guessing z_k among small Hamming weights [15] (we call this the generalized information-set-decoding (GISD) attack). The correctness of the recovered plaintext m is verifiable by checking whether the Hamming weight of $c \oplus mG'$ is t or not.

The computational cost of this generalized version (where z_k is guessed) is slightly faster than the original one (where z_k is supposed to be 0), but it is still infeasible for appropriate parameters since its computational cost is asymptotically lower bounded by $C(n, k)/C(n - t, k)$.

Finding-Low-Weight-Codeword Attack. This attack uses an algorithm which finds a low-weight codeword among codewords generated by an arbitrary generator matrix using a database obtained by pre-computation [26,4]. Since the minimum-weight codeword of the following $(k + 1) \times n$ generator matrix

$$\begin{bmatrix} G' \\ c \end{bmatrix} \tag{7}$$

is the error vector z of c where $c = mG' \oplus z$, this algorithm can be used to recover m from a given ciphertext c.

The precise computational cost of this attack is evaluated in [4], and then shown to be infeasible to invert c for appropriate parameters, e.g. $n \geq 2048$ and optimized k and t, even though the original parameters $(n, k, t) = (1024, 524, 50)$ suggested in [18] is feasible with the work factor of $2^{64.2}$. (Under the assumption that each iteration is independent, the expected computational cost of this attack is asymptotically lower bounded by $C(n, k + 1)/C(n - t, k + 1)$ and therefore it is infeasible for appropriate parameters.)

3.2 Critical Attacks

The following attacks cannot be avoided by enlarging the parameter size or by applying Loidreau's modification [16]. Therefore critical.

Known-Partial-Plaintext Attack. The partial knowledge on the target plaintext drastically reduces the computational cost of the attacks to the McEliece PKC [4,13].

For example, let m_l and m_r denote the left k_l bits and the remaining k_r bits in the target plaintext m, i.e. $k = k_l + k_r$ and $m = (m_l||m_r)$. Suppose that an adversary knows m_r. Then the difficulty of recovering unknown plaintext m_l in the McEliece PKC with parameters (n, k) is equivalent to that of recovering the full plaintext in the McEliece PKC with parameters (n, k_l) since

$$\begin{aligned} c &= mG' \oplus z \\ c &= m_lG'_l \oplus m_rG'_r \oplus z \\ c \oplus m_rG'_r &= m_lG'_l \oplus z \\ c' &= m_lG'_l \oplus z, \end{aligned} \tag{8}$$

where G'_l and G'_r are the upper k_l rows and the remaining lower k_r rows in G', respectively.

If k_l is fixed to a small value, the computational cost of recovering the unknown k_l bits from c, m_r and G' is a polynomial of n since even if non-critical attacks are used, it is asymptotically bounded by $k_l^3C(n, k_l)/C(n - t, k_l)$ where k_l is a small constant.

Related-Message Attack. This attack uses the knowledge on the relationship between the target plaintexts [3].

Suppose two messages m_1 and m_2 are encrypted to c_1 and c_2, respectively, where $c_1 = m_1G' \oplus z_1$, $c_2 = m_2G' \oplus z_2$, and $z_1 \neq z_2$. If an adversary knows their linear relation between the plaintexts, e.g. $\delta m = m_1 \oplus m_2$. Then the adversary can efficiently apply the GISD attack to either c_1 or c_2 by choosing k coordinates whose values are 0 in $(\delta mG' \oplus c_1 \oplus c_2)$. Since $z_1 \oplus z_2 = \delta mG' \oplus c_1 \oplus c_2$ and the Hamming weight t of the error vector z is far smaller than $n/2$. Therefore a coordinate being 0 in $(\delta mG' \oplus c_1 \oplus c_2)$ should also be 0 in both z_1 and z_2 with the high probability of

When the same message is encrypted twice (or more) using different error vectors z_1 and z_2, the value $z_1 \oplus z_2$ is simply given by $c_1 \oplus c_2$. This case is referred to as the message-resend attack [3].

Reaction Attack. This attack might be categorized as a chosen-ciphertext attack (CCA), but uses a weaker assumption [12] than the CCA: the adversary observes only the reaction of the receiver who has the private-key, but does not need to receive its decrypted plaintext. (Similar attack is independently proposed in [28]. In this attack, an adversary receives the corresponding plaintexts. Therefore this attack is categorized as a CCA.)

The idea of this attack is the following. The adversary flips one or a few bits of the target ciphertext c. Let c' denote the flipped ciphertext. The adversary transmits c' to the proper receiver and observes his/her reaction. The receiver's reactions can be divided into the following two:

Reaction A: Return a repeat request to the adversary due to uncorrectable error or due to the meaningless plaintext.

Reaction B: Return an acknowledgment or do nothing since the proper plaintext m is decrypted.

If the total weight of the error vector does not exceed t after the flipping, the reaction B is observed. Otherwise the reaction A is observed. Therefore by repeating the above observations polynomial times of n, the adversary can determine the error vector. Once the error vector is determined, the corresponding plaintext is easily decrypted using the GISD attack.

Malleability Attack. This attack allows an adversary to alter any part of the corresponding plaintext of any given ciphertext c without knowing the plaintext m, i.e. the adversary can generate a new ciphertext c' whose plaintext is $m' = m \oplus \delta m$ from any given ciphertext c without knowing m [13,28].

This attack is described as follows. Let $G'[i]$ denote the i-th row of the public matrix G' and $I = \{i_1, i_2, \cdots\}$ denote a set of coordinates i_j whose value is 1 in δm. The ciphertext c' is calculated by

$$c' = c \bigoplus_{i \in I} G'[i] = (m \oplus \delta m)G' \oplus z = m'G' \oplus z. \tag{9}$$

This attack tells us that the McEliece PKC does not satisfy non-malleability[6] even against passive attacks, such as chosen-plaintext attacks. And then under chosen-ciphertext scenario where an adversary can ask an decryption oracle to decrypt a polynomial number of ciphertexts (excluding the challenge ciphertext c), the adversary can decrypt any given ciphertext c by the following way. First the adversary asks the oracle to decrypt c', then the oracle returns $m' = m \oplus \delta m$. Thus he/she can recover the target plaintext of c by $m = m' \oplus \delta m$.

4 Conversions for McEliece PKC

As mentioned in Section 3, without any decryption oracles and any partial knowledge on the corresponding plaintext of the challenge ciphertext, no polynomial-time algorithm is known for inverting the McEliece PKC (whose parameters are carefully chosen). Under the assumption that this inverting problem is hard, this problem can be converted into the hard problem of breaking the indistinguishability of encryption against critical attacks (or more generally against adaptive chosen-ciphertext attacks) by introducing appropriate conversions in the random oracle mode. In this section, we discuss which conversions are appropriate for the McEliece PKC and which are not.

4.1 Notations

We use the following notations in this paper:

$C(n,t)$	: The number of combinations taking t out of n elements.
$Prep(m)$	: Preprocessing to a message m, such as data-compression, data-padding and so on. Its inverse is represented as $Prep^{-1}()$.
$Hash(x)$	: One-way hash function of an arbitrary length binary string x to a fixed length binary string. When the output domain is Z_N where $N = C(n,t)$, we use $Hash_z(x)$ instead of $Hash(x)$.
$Conv(\bar{z})$	: Bijective function which converts an integer $\bar{z} \in Z_N$ where $N = C(n,t)$ into the corresponding error vector z. Its inverse is represented as $Conv^{-1}()$.
$Gen(x)$	: Generator of a cryptographically secure pseudo random sequences of arbitrary length from a fixed length seed x.
$Len(x)$	: Bit-length of x.
$Msb_{x_1}(x_2)$	: The left x_1 bits of x_2.
$Lsb_{x_1}(x_2)$	: The right x_1 bits of x_2.
$Const$	: Predetermined constant used in public.
$Rand$	: Random source which generates a truly random (or computationally indistinguishable pseudo random) sequence.
$\mathcal{E}^{McEliece}(x,z)$	: Encryption of x using the original McEliece PKC with an error vector z.
$\mathcal{D}^{McEliece}(x)$	: Decryption of x using the original McEliece PKC.

4.2 Insufficient Conversions for McEliece PKC

OAEP Conversion. In [2], Bellar and Rogaway proposed a generic conversion called OAEP (Optimal Asymmetric Encryption Padding) which converts a OWTP (One-Way Trapdoor Permutation), such as RSA primitive, into a PKC which is indistinguishable against adaptive chosen-ciphertext attacks (CCA2). The McEliece PKC with this OAEP conversion is given in Fig.1.[4] Unfortunately, this conversion does not work correctly since the reaction attack is still applicable. This does not mean the OAEP conversion has a fault, but the McEliece primitive is not a permutation.

Fujisaki-Okamoto's Simple Conversion. In [8], Fujisaki and Okamoto proposed a generic and simple conversion from a PKC which is indistinguishable against CPA (Chosen-Plaintext Attacks) into a PKC which is indistinguishable against CCA2. The McEliece PKC with this conversion is given in Fig.2.[4] Unfortunately, this conversion does not work correctly since the known-partial-plaintext attack efficiently works unless $Len(r)$ is close to k.

This does not mean Fujisaki-Okamoto's simple conversion has a fault, but the original McEliece PKC is distinguishable even against CPA. Any passive adversary (who do not use decryption oracle) can guess which message of m_0 and m_1 corresponding plaintext of the given ciphertext $\bar{c}$ of the original McEliece PKC by seeing whether the Hamming weight of $m_b G' \oplus \bar{c}$ is t or not, where $b \in \{0,1\}$.

[4] Due to the limitation of pages, we omit the corresponding decryption process.

```
Encryption of m:
    $r, \bar{z} := Rand$
    $\bar{m} := Prep(m)$
    $y_1 := (\bar{m}||Const) \oplus Gen(r)$
    $y_2 := r \oplus Hash(y_1)$
    $z := Conv(\bar{z})$
    $c := \mathcal{E}^{McEliece}((y_1||y_2), z)$
    return $c$
```

Fig. 1. OAEP conversion + McEliece PKC

```
Encryption of m:
    $r := Rand$
    $\bar{m} := Prep(m)$
    $z := Conv(Hash_z(\bar{m}||r))$
    $c := \mathcal{E}^{McEliece}((\bar{m}||r), z)$
    return $c$
```

Fig. 2. Fujisaki-Okamoto's simple conversion + McEliece PKC

```
Encryption of m:
    $r, r' := Rand$
    $\bar{m} := Prep(m)$
    $z := Conv(Hash_z(\bar{m}||r))$
    $y_1 := \mathcal{E}^{McEliece}(r', z)$
    $y_2 := Gen(r') \oplus (\bar{m}||r)$
    $c := y_1||y_2$
    return $c$
```

Fig. 3. Pointcheval's generic conversion

```
Encryption of m:
    $r := Rand$
    $\bar{m} := Prep(m)$
    $z := Conv(Hash_z(\bar{m}||r))$
    $y_1 := \mathcal{E}^{McEliece}(r, z)$
    $y_2 := Gen(r) \oplus \bar{m}$
    $c := y_1||y_2$
    return $c$
```

Fig. 4. Fujisaki-Okamoto's generic conversion

4.3 Generic Conversions Being Applicable to McEliece PKC

Pointcheval's Generic Conversion. In [24], Pointcheval proposed a generic conversion from a PTOWF (Partially Trapdoor One-Way Function) to a PKC which is indistinguishable against CCA2.

The definition for $f(x, y)$ to be PTOWF is the following:

– For any polynomial-time adversary and for any given $z = f(x, y)$, it is computationally infeasible to get back the partial preimage x,
– With some extra secret information, it is easy to get back the x.

Not only ElGamal[7], Okamoto-Uchiyama[22], Naccache-Stern[20] and Paillier[23] primitives, but also McEliece primitive can be categorized in PTOWF. Therefore Pointcheval's generic conversion is also applicable to the McEliece PKC with the same proof in [24]. The McEliece PKC with this conversion is given in Fig.3.[4]

Fujisaki-Okamoto's Generic Conversion. In [9], Fujisaki and Okamoto proposed a generic conversion from OWE (One-Way Encryption), which includes both OWTP and PTOWF, into a PKC being indistinguishable against ACC2.

```
Encryption of m:
    $r := Rand$
    $\bar{m} := Prep(m)$
    $\bar{z} := Hash_z(r||\bar{m})$
    $(y_1||y_2) := Gen(\bar{z}) \oplus (r||\bar{m})$
    $z := Conv(\bar{z})$
    $c := \mathcal{E}^{McEliece}(y_1, z)||y_2$
    return $c$

Decryption of $c$:
    $y_1 := \mathcal{D}^{McEliece}(Msb_n(c))$
    $z := Msb_n(c) \oplus y_1 G'$
    $\bar{z} := Conv^{-1}(z)$
    $(r||\bar{m}) := Gen(\bar{z}) \oplus (y_1||y_2)$
    If $\bar{z}{=}Hash_z(r||\bar{m})$
        return $Prep^{-1}(\bar{m})$
    Otherwise reject $c$
```

Fig. 5. Conversion α : $Len(y_1) = k$ and $Len(y_2) = Len(r||\bar{m}) - k$. If $Len(r||\bar{m}) = k$, remove y_2.

Needless to say, McEliece primitive can be categorized in the OWE, and therefore their generic conversion is applicable to the McEliece PKC with the same proof in [9]. The McEliece PKC with this conversion[5] is given in Fig.4.[4]

4.4 Our Specific Conversions

While one can design semantically-secure McEliece PKCs by simply employing the above generic conversions, they are not necessarily suited for the McEliece PKC. Since the block size of the McEliece PKC is larger than the well-known PKCs, such as RSA, elliptic curve cryptosystems and so on, the redundancy of data (which is defined by the difference between the bit length of a plaintext and its corresponding ciphertext) becomes large. For example, for $(n, k) = (4096, 2560)$, the generic conversions require more than or equal to 4096 bits for the overhead data. On the other hand, our conversions described in Fig. 5 $\sim$ 7 require less overhead data than the generic ones. For example, for the same settings and $Len(r) = 160$ and $Len(Const) = 160$, our conversion γ requires only 1040 bits. (This might still be large but interestingly this value is smaller than the original McEliece PKC.) The comparison results are summarized in Table 1.

The point of the conversion γ is that not only a plaintext but also an error vector is taken from a part of y_1 (or $(y_2||y_1)$). This reduces the data overhead even than the original McEliece PKC when $Len(r) + Len(Const) < \lfloor \log_2 C(n, t) \rfloor$. The study to reduce the overhead data (and simultaneously to improve the security against related-message attacks) has been performed in [27]. While his conversions do not provide provable security against CCA2 (since either known-partial-plaintext attacks or reaction attacks are applicable at least), his approach to reduce the overhead data should be appreciated.

Indistinguishability of Our Conversions. It is intuitively clear that our conversions resist all the critical attacks in Section 3.2 since it is hard for ad-

[5] They originally proposed to use symmetric encryption (instead of $Gen(r)$). The conversion described here is a variant mentioned in [9].

```
Encryption of $m$:
    $r := Rand$
    $\bar{m} := Prep(m)$
    $y_1 := Gen(r) \oplus \bar{m}$
    $y_2 := r \oplus Hash(y_1)$
    $(y_4||y_3) := (y_2||y_1)$
    $z := Conv(Hash_z(r))$
    $c := y_4||\mathcal{E}^{McEliece}(y_3, z)$
    return $c$
```

```
Decryption of $c$:
    $y_4 := Msb_{Len(c)-n}(c)$
    $y_3 := \mathcal{D}^{McEliece}(Lsb_n(c))$
    $z := Lsb_n(c) \oplus y_3G'$
    $(y_2||y_1) := (y_4||y_3)$
    $r := y_2 \oplus Hash(y_1)$
    $\bar{m} := Gen(r) \oplus y_1$
    If $Conv^{-1}(z) = Hash_z(r)$
        return $Prep^{-1}(\bar{m})$
    Otherwise reject $c$
```

Fig. 6. Conversion β : $Len(y_3) = k$ and $Len(y_4) = Len(r||\bar{m}) - k$ If $Len(r||\bar{m}) = k$, remove y_4.

```
Encryption of $m$:
    $r := Rand$
    $\bar{m} := Prep(m)$
    $y_1 := Gen(r) \oplus (\bar{m}||Const)$
    $y_2 := r \oplus Hash(y_1)$
    $(y_5||y_4||y_3) := (y_2||y_1)$
    $z := Conv(y_4)$
    $c := y_5||\mathcal{E}^{McEliece}(y_3, z)$
    return $c$
```

```
Decryption of $c$:
    $y_5 := Msb_{Len(c)-n}(c)$
    $y_3 := \mathcal{D}^{McEliece}(Lsb_n(c))$
    $z := y_3G' \oplus Lsb_n(c)$
    $\bar{z} := Conv^{-1}(z)$
    $y_4 := Lsb_{\lfloor \log_2 C(n,t) \rfloor}(\bar{z})$
    $(y_2||y_1) := (y_5||y_4||y_3)$
    $r := y_2 \oplus Hash(y_1)$
    $(\bar{m}||Const') := y_1 \oplus Gen(r)$
    If $Const' = Const$
        return $Prep^{-1}(\bar{m})$
    Otherwise reject $c$
```

Fig. 7. Conversion γ : $Len(y_3) = k$, $Len(y_4) = \lfloor \log_2 C(n,t) \rfloor$, $Len(y_5) = Len(\bar{m}) + Len(Const) + Len(r) - Len(y_4) - k$. If $Len(\bar{m}) + Len(Const) + Len(r) = Len(y_4) + k$, remove y_5.

versaries to abuse decryption oracles because of the difficulty of generating an appropriate ciphertext without knowing its plaintext, and to guess the input to the original McEliece PKC in our conversions even if they know the plaintext to our conversions.

More formally, the following theorem is true for our conversions in the random oracle model (where Gen, $Hash$ and $Hash_z$ are assumed to be ideal).

Theorem 1 *To break the indistinguishability of encryption of our specific conversions in an adaptive-chosen-ciphertext scenario is polynomial equivalent to decrypt the whole plaintext of an arbitrarily given ciphertext of the original McEliece PKC without any help of decryption oracles and any knowledge on the target plaintext.*

Table 1. Comparison between Data Redundancy and Conversions

Conversion Scheme	Conversion Type	Complexity[*2]	$\geq 2^{56.3}$	$\geq 2^{101.9}$	$\geq 2^{186.2}$
		Data Redundancy[*1] = Ciphertext Size - Plaintext Size			
		(n,k)	$(1024, 644)$	$(2048, 1289)$	$(4096, 2560)$
		t	38	69	128
Pointcheval's [24]	Generic	$n + Len(r)$	1184	2208	4256
Fujisaki -Okamoto's [9]	Generic	n	1024	2048	4096
Our proposal α and β	Specific	$n - k + Len(r)$	540	919	1696
Our proposal γ	Specific	$n - k + Len(r)$ $+Len(Const)$ $-\lfloor \log_2 C(n,t) \rfloor$	470	648	1040
Original McEliece	None	n-k	380	759	1536

*1: The numerical results are obtained under the setting that $Len(r) = 160$ and $Len(Const) = 160$.
*2: The asymptotic lower bound of the expected number of iterations to invert an arbitrary ciphertext of the original McEliece PKC using the finding-low-weight-codeword attack. The exact complexity is estimated in [4].

Note that, as mentioned in Section 3, it is still infeasible to decrypt the whole plaintext of an arbitrarily given ciphertext of the original McEliece PKC with appropriate parameters (without any help of decryption oracles and any knowledge on the target plaintext).

This theorem can be proven, in the random oracle model, by showing how to construct an algorithm which decrypts an arbitrary ciphertext of the original McEliece PKC using an algorithm which distinguishes a ciphertext of our converted versions in the adaptive-chosen-ciphertext scenario. (It is obvious that an algorithm, which can decrypt the original McEliece PKC, can also distinguish a ciphertext of our converted versions.) Details are described in Appendix A.

5 Conclusion

We carefully reviewed the currently known attacks to the McEliece PKC, and then confirmed that, without any decryption oracles and any partial knowledge on the corresponding plaintext of the challenge ciphertext, no polynomial-time algorithm is known for inverting the McEliece PKC whose parameters are carefully chosen. Under the assumption that this inverting problem is hard, we investigated, in the random oracle mode, how to convert this hard problem into the hard problem of breaking the indistinguishability of encryption with CCA2. While some of the generic conversions are applicable to the McEliece PKC, they have a disadvantage in data redundancy. A large amount of redundant data is needed for the generic conversions since the block size of the McEliece PKC is relatively large. Our specific conversions can achieve the reduction of the redundant data down to $1/3 \sim 1/4$ compared with the generic conversions for practical parameters. This means about 3K bits can be saved for $n = 4096$, with providing semantic security against CCA2.

Acknowledgments

The authors would like to thank Hung-Min Sun, Pierre Loidreau, Kwangjo Kim and Yuliang Zheng for useful discussions and comments.

References

1. C. M. Adams and H. Meijer. “Security-Related Comments Regarding McEliece's Public-Key Cryptosystem”. In *Proc. of CRYPTO '87, LNCS 293*, pages 224–228. Springer–Verlag, 1988.
2. M. Bellare and P. Rogaway. “Optimal Asymmetric Encryption”. In *Proc. of EUROCRYPT '94, LNCS 950*, pages 92–111, 1995.
3. T. Berson. “Failure of the McEliece Public-Key Cryptosystem Under Message-Resend and Related-Message Attack”. In *Proc. of CRYPTO '97, LNCS 1294*, pages 213–220. Springer–Verlag, 1997.
4. A. Canteaut and N. Sendrier. “Cryptoanalysis of the Original McEliece Cryptosystem”. In *Proc. of ASIACRYPT '98*, pages 187–199, 1998.
5. W. Diffie and M. Hellman. “New directions in cryptography”. *IEEE Trans. IT*, 22(6):644–654, 1976.
6. D. Dolve, C. Dwork, and M. Naor. “Non-Malleable Cryptography”. In *Proc. of the 23rd STOC.* ACM Press, 1991.
7. T. ElGamal. “A public-key cryptosystem and a signature scheme bsed on discrete logarithms”. In *Proc. of CRYPTO '84*, pages 10–18, 1985.
8. E. Fujisaki and T. Okamoto. “How to Enhance the Security of Public-Key Encryption at Minimum Cost”. In *Proc. of PKC'99, LNCS 1560*, pages 53–68, 1999.
9. E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In *Proc. of CRYPTO '99, LNCS 1666*, pages 535–554, 1999.
10. J. K. Gibson. “Equivalent Goppa Codes and Trapdoors to McEliece's Public Key Cryptosystem”. In *Proc. of EUROCRYPT '91, LNCS 547*, pages 517–521. Springer–Verlag, 1991.
11. S. Goldwasser and S. Micali. “Probabilistic encryption”. *Journal of Computer and System Sciences*, pages 270–299, 1984.
12. C. Hall, I. Goldberg, and B. Schneier. “Reaction Attacks Against Several Public-Key Cryptosystems”. In *Proc. of the 2nd International Conference on Information and Communications Security (ICICS'99), LNCS 1726*, pages 2–12, 1999.
13. K. Kobara and H. Imai. “Countermeasure against Reaction Attacks (in Japanese)”. In *The 2000 Symposium on Cryptography and Information Security : A12*, January 2000.
14. V.I. Korzhik and A.I. Turkin. “Cryptanalysis of McEliece's Public-Key Cryptosystem”. In *Proc. of EUROCRYPT '91, LNCS 547*, pages 68–70. Springer–Verlag, 1991.

15. P. J. Lee and E. F. Brickell. "An Observation on the Security of McEliece's Public-Key Cryptosystem". In *Proc. of EUROCRYPT '88, LNCS 330*, pages 275–280. Springer–Verlag, 1988.
16. P. Loidreau. "Strengthening McEliece Cryptosystem". In *Proc. of ASIACRYPT 2000*. Springer–Verlag, 2000.
17. P. Loidreau and N. Sendrier. "Some weak keys in McEliece public-key cryptosystem". In *Proc. of IEEE International Symposium on Information Theory, ISIT '98*, page 382, 1998.
18. R. J. McEliece. "A Public-Key Cryptosystem Based on Algebraic Coding Theory". In *Deep Space Network Progress Report*, 1978.
19. A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. "McEliece public-key encryption". In *"Handbook of Applied Cryptography"*, page 299. CRC Press, 1997.
20. D. Naccache and J. Stern. "A New Cryptosystem based on Higher Residues". In *Proc. of the 5th CCS*, pages 59–66. ACM Press, 1998.
21. T. Okamoto, K. Tanaka, and S. Uchiyama. "Quantum Public-Key Cryptosystems". In *Proc. of CRYPTO 2000, LNCS 1880*, pages 147–165. Springer–Verlag, 2000.
22. T. Okamoto and S. Uchiyama. "A New Public Key Cryptosystem as Secure as Factoring". In *Proc. of EUROCRYPT '98, LNCS 1403*, pages 129–146, 1999.
23. P. Paillier. "Public-Key Cryptosystems Based on Discrete Logarithms Residues". In *Proc. of EUROCRYPT '99, LNCS 1592*, pages 223–238. Springer–Verlag, 1999.
24. D. Pointcheval. "Chosen-Ciphertext Security for Any One-Way Cryptosystem". In *Proc. of PKC 2000, LNCS 1751*, pages 129–146. Springer–Verlag, 2000.
25. P.W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing*, 26:1484–1509, 1997.
26. J. Stern. "A method for finding codewords of small weight". In *Proc. of Coding Theory and Applications , LNCS 388*, pages 106–113. Springer–Verlag, 1989.
27. H. M. Sun. "Improving the Security of the McEliece Public-Key Cryptosystem". In *Proc. of ASIACRYPT '98*, pages 200–213, 1998.
28. H. M. Sun. "Further Cryptanalysis of the McEliece Public-Key Cryptosystem". *IEEE Trans. on communication letters*, 4:18–19, 2000.
29. A. Vardy. "The Intractability of Computing the Minimum Distance of a Code". *IEEE Trans. on IT*, 43:1757–1766, 1997.

A Proof of Theorem 1

A.1 Indistinguishability of Encryption

Recall a security notion called indistinguishability of encryption [11]. In this notion, an adversary $\mathcal{A}$ selects two distinct plaintexts m_0 and m_1 of the same length in the find stage, and then, in the guess stage, $\mathcal{A}$ is given c which is the encryption of m_b where b is either 0 or 1 with the probability of 1/2. Then $\mathcal{A}$ tries to guess b. The advantage of $\mathcal{A}$ is defined by $2Pr(\text{Win}) - 1$ where $Pr(\text{Win})$ denotes the expected probability of $\mathcal{A}$ guessing b correctly. If $\mathcal{A}$ has a decryption oracle D (which decrypts any other ciphertexts than the target ciphertext c), it is called that this experiment is in the adaptive-chosen-ciphertext scenario. Otherwise, if $\mathcal{A}$ does not have it, it is called that this experiment is in the adaptive-chosen-plaintext scenario.

A.2 Random Oracle

A random oracle is an ideal hash or an ideal generator which returns truly random numbers distributed uniformly over the output region for a new query, but it returns the same value for the same query. On such random oracles, the following lemma is true.

Lemma 1 *Suppose that f is a random oracle. Then it is impossible to get any significant information on $f(x)$ without asking x to the oracle, even if one knows all the other outputs of f except one corresponding to x.*

It is obvious that Lemma 1 is true since the output value of f is truly random.

A.3 Adaptive-Chosen-Ciphertext Security

The proof of Theorem 1 for the conversion α is given by showing Lemma 3 is true. Before we show it, we prove Lemma 2 first.

Lemma 2 (Adaptive-Chosen-Plaintext Security) *Suppose that there exists, for any $Hash_z$ and any Gen, an algorithm $\mathcal{A}$ which accepts m_0, m_1 and c of conversion α where c is the ciphertext of m_b and $b \in \{0,1\}$, asks at most q_G queries to Gen, asks at most q_H queries to $Hash_z$, runs in at most t steps and guesses b with advantage of ϵ. Then one can design an algorithm $\mathcal{B}$ which accepts a ciphertext $\bar{c}$ of the original McEliece PKC, runs in t' steps and decrypts it with probability ϵ' where*

$$\epsilon' \geq \epsilon - \frac{q_H}{2^{Len(r)+1}},$$
$$t' = t + Poly(n, q_G, q_H)$$

and $Poly(n, q_G, q_H)$ denotes a polynomial of n, q_G and q_H.

Proof.

The algorithm $\mathcal{B}$ can be constructed as follows. First the algorithm $\mathcal{B}$ simulates both Gen and $Hash_z$ referred by the algorithm $\mathcal{A}$. From the assumption of $\mathcal{A}$ in Lemma 2, $\mathcal{A}$ must be able to distinguish b with the advantage of ϵ for any Gen and any $Hash_z$ as long as the algorithm $\mathcal{B}$ simulates them correctly. Then, $\mathcal{B}$ chooses b, r and y_2 at random, and then defines both $Hash_z$ and Gen so that the ciphertext of $(r||m_b)$ should be $(\bar{c}||y_2)$ where $\bar{c}$ is a ciphertext of the original McEliece PKC which $\mathcal{B}$ wants to decrypt. That is,

$$Gen(\bar{z}) \stackrel{\text{def}}{=} (r||m_b) \oplus (y_1||y_2) \tag{10}$$

$$Hash_z(r||m_b) \stackrel{\text{def}}{=} Conv^{-1}(z) = \bar{z}. \tag{11}$$

For the other queries than $\bar{z}$ to Gen and $(r||m_b)$ to $Hash_z$, they return random values, respectively. Even for these Gen and $Hash_z$, $\mathcal{A}$ must be able to distinguish b with the advantage of ϵ from the assumption in Lemma 2 as long as $\mathcal{B}$ simulates them correctly.[6]

[6] If $\mathcal{A}$ distinguishes b only for certain combinations of $Hash_z$ and Gen, then the fault must be in either Gen or $Hash_z$ or in both used in the combinations, and therefore

Then, can $\mathcal{B}$ simulate them correctly for any queries? The answer is no since $\mathcal{B}$ does not know $\bar{z}$, and therefore $\mathcal{B}$ cannot simulate $Hash_z$ correctly when $(r||m_b)$ is asked to $Hash_z$. We define the following two events AskG and AskH.

Definition 1 *Let AskG denote the event that $\bar{z}$ is asked to Gen among the q_G queries to Gen and this query is performed before $(r||m_b)$ is asked to $Hash_z$. Let AskH denote the event that $(r||m_b)$ is asked to $Hash_z$ among the q_H queries to $Hash_z$ and this query is performed before $\bar{z}$ is asked to Gen.*

Since $Pr(\text{AskG} \wedge \text{AskH}) = 0$ in this definition, the following holds

$$Pr(\text{AskG} \vee \text{AskH}) = Pr(\text{AskG}) + Pr(\text{AskH}). \tag{12}$$

Next, we estimate the upper-limit of $Pr(\text{Win})$, the probability of $\mathcal{A}$ guessing b correctly. From Lemma 1, without asking either $\bar{z}$ to *Gen* or asking $(r||m_b)$ to $hash_z$, one cannot get any information on the connectivity between $(z, y_1||y_2)$ and $(r||m_b)$, and therefore cannot guess b with a significant probability after the event $(\neg\text{AskG}\wedge\neg\text{AskH})$. After the other event, i.e. after the event $(\text{AskG}\vee\text{AskH})$, $\mathcal{A}$ might guess b with more significant probability. By assuming this probability to be 1, the upper-limit of $Pr(\text{Win})$ is obtained as follows:

$$\begin{aligned} Pr(\text{Win}) &\le Pr(\text{AskG} \vee \text{AskH}) + \frac{(1 - Pr(\text{AskG} \vee \text{AskH}))}{2} \\ &\le \frac{Pr(\text{AskG} \vee \text{AskH}) + 1}{2}. \end{aligned} \tag{13}$$

From the definition of advantage, i.e. $Pr(\text{Win}) = (\epsilon + 1)/2$, the following relationship holds

$$Pr(\text{AskG} \vee \text{AskH}) \ge \epsilon. \tag{14}$$

Since both r and b are chosen at random by $\mathcal{B}$, $\mathcal{A}$ cannot know them without asking $\bar{z}$ to *Gen* or asking $(r||m_b)$ to $Hash_z$. Thus the probability of one query to $Hash_z$ accidentally being $(r||m_b)$ is $1/2^{Len(r)+1}$, and then that of at most q_H queries is given by

$$Pr(\text{AskH}) \le 1 - \left(1 - \frac{1}{2^{Len(r)+1}}\right)^{q_H} \le \frac{q_H}{2^{Len(r)+1}}. \tag{15}$$

The algorithm $\mathcal{B}$ can simulate both *Gen* and $Hash_z$ correctly unless the event AskH happens. And then, after the event AskG, $\mathcal{B}$ can recover the whole plaintext of the target ciphertext $\bar{c}$ of the original McEliece PKC using $\bar{z}$ asked to $\mathcal{B}$. Thus, after the event $(\text{AskG} \wedge \neg\text{AskH})$, $\mathcal{B}$ can recover it. The lower-limit of this probability is given by

this fault can be easily removed just avoiding using the combinations. Otherwise, i.e. if $\mathcal{A}$ distinguishes b for any combinations of $Hash_z$ and *Gen*, the fault must be in the conversion structure.

$$\begin{aligned} Pr(\text{AskG} \wedge \neg\text{AskH}) &= Pr(\text{AskG}) \\ &= Pr(\text{AskG} \vee \text{AskH}) - Pr(\text{AskH}) \\ &\geq \epsilon - \frac{q_H}{2^{Len(r)+1}} \end{aligned} \tag{16}$$

from (12), (14) and (15).

The number of steps of $\mathcal{B}$ is at most $t + (T_{Dec} + T_G) \cdot q_G + T_H \cdot q_H$ where T_{Dec} is the number of steps for decrypting the original McEliece PKC using a new query to Gen as $\bar{z}$, T_G is both for checking whether a query to Gen is new or not and for returning the corresponding value, and T_H is both for checking whether a query to $Hash_z$ is new or not and for returning the corresponding value. Since these parameters, T_{Dec}, T_G and T_H can be written in a polynomial of n, q_G and q_H, the total number of steps of $\mathcal{B}$ is also written in a a polynomial of them.

□

Lemma 3 (Adaptive-Chosen-Ciphertext Security) *Suppose that there exists, for any $Hash_z$ and Gen, an algorithm $\mathcal{A}$ which accepts m_0, m_1 and c of conversion α where c is the ciphertext of m_b and $b \in \{0,1\}$, asks at most q_G queries to Gen, asks at most q_H queries to $Hash_z$, asks at most q_D queries to a decryption oracle D, runs in at most t steps and guesses b with advantage of ϵ. Then one can design an algorithm $\mathcal{B}$ which accepts a ciphertext $\bar{c}$ of the original McEliece PKC, runs in t' steps and decrypts it with probability ϵ' where*

$$\begin{aligned} \epsilon' &\geq \epsilon - \frac{q_H}{2^{Len(r)+1}} - \frac{q_D}{C(n,t)}, \\ t' &= t + Poly(n, q_G, q_H, q_D) \end{aligned}$$

and $Poly(n, q_G, q_H, q_D)$ denotes a polynomial of n, q_G, q_H and q_D.

Proof.

The algorithm $\mathcal{B}$ can be constructed as follows. First, the algorithm $\mathcal{B}$ simulates random oracles Gen, $Hash_z$ and the decryption oracle D referred by $\mathcal{A}$. As long as $\mathcal{B}$ simulates them correctly, $\mathcal{A}$ must be able to distinguish the given ciphertext with advantage of ϵ. How to simulate both Gen and $Hash_z$ is the same as in the proof of Lemma 2. The decryption oracle D can be simulated using the following plaintext-extractor [2]. The plaintext-extractor accepts a ciphertext, e.g. $(\bar{c}'||y_2')$ where $\bar{c}'$ denotes a ciphertext of the original McEliece PKC, and then outputs either the corresponding plaintext of $(\bar{c}'||y_2')$, or reject it as an inappropriate ciphertext.

Let g_i and G_i denote the i-th pair of query and its answer for Gen. And then let h_j and H_j denote the j-th pair of query and its answer for $Hash_z$. From the queries and the answers obtained while simulating Gen and $Hash_z$, the plaintext-extractor finds a pair of (g_i, G_i) and (h_j, H_j) satisfying $Conv(g_i) = z'$, $Conv(H_j) = z'$ and $G_i \oplus (y_1'||y_2') = h_j$ where y_1' and z' denote the plaintext and the error vector of $\bar{c}'$, respectively. If found, $\mathcal{B}$ outputs $Lsb_{Len(m')}(h_i)$ as the plaintext of $(\bar{c}'||y_2')$ where $Len(m') = Len(\bar{c}') + Len(y_2') - n + k - Len(r')$. Otherwise $\mathcal{B}$ rejects it as an inappropriate ciphertext.

The plaintext-extractor can simulate D unless $\mathcal{A}$ asks an appropriate ciphertext to D without asking both $\bar{z}'$ and $G_i \oplus (y_1'||y_2')$ to Gen and $Hash_z$, respectively. In this case, the plaintext-extractor rejects the appropriate ciphertext, and therefore does not simulate D correctly. However it is a small chance that $\mathcal{A}$ could generate an appropriate ciphertext without asking them. Since the definition of appropriate is to satisfy

$$Hash_z(Gen(\bar{z}') \oplus (y_1'||y_2')) = \bar{z}', \tag{17}$$

and it is impossible for $\mathcal{A}$ to know whether (17) is true or not without asking $\bar{z}'$ to Gen and asking $Gen(\bar{z}') \oplus (y_1'||y_2')$ to $Hash_z$, respectively, from Lemma 1.

We define AskD as the following event that at least one query out of at most q_D queries to D accidentally becomes an appropriate ciphertext before the queries used in (17) are asked. Since the probability of one query to D being accidentally an appropriate ciphertext is $1/C(n,t)$, the upper-limit of $Pr(\text{AskD})$ is given by

$$Pr(\text{AskD}) \leq 1 - \left(1 - \frac{1}{C(n,t)}\right)^{q_D} \leq \frac{q_D}{C(n,t)}. \tag{18}$$

Unless either AskD or AskH happens, $\mathcal{B}$ can correctly simulate the oracles referred by $\mathcal{A}$. In addition, when AskG happens, $\mathcal{B}$ can recover the whole plaintext of $\bar{c}$, the ciphertext of the original McEliece PKC. The lower-limit of this probability $Pr(\text{AskG} \wedge \neg\text{AskD} \wedge \neg\text{AskH})$ is given by

$$\begin{aligned} & Pr(\text{AskG} \wedge \neg\text{AskH} \wedge \neg\text{AskD}) \\ & = Pr(\text{AskG} \wedge \neg\text{AskH}) - Pr(\text{AskG} \wedge \neg\text{AskH} \wedge \text{AskD}) \\ & \geq Pr(\text{AskG} \wedge \neg\text{AskH}) - Pr(\text{AskD}) \\ & \geq \epsilon - \frac{q_H}{2^{Len(r)+1}} - \frac{q_D}{C(n,t)}. \end{aligned} \tag{19}$$

The number of steps of $\mathcal{B}$ is at most $t + (T_{Dec} + T_G) \cdot q_G + T_H \cdot q_H + T_D \cdot q_D$ where T_{Dec}, T_G and T_H are the same as the parameters in the proof of Lemma 2. The number of steps T_D is for the knowledge-extractor to verify whether (17) holds and then to return the result. Since these parameters, T_{Dec}, T_G, T_H and T_D can be written in a polynomial of n, q_G, q_H and q_D, the total number of steps of $\mathcal{B}$ is also written in a polynomial of them.

$\square$

Using the similar discussion to the conversion α, the lower limit of ϵ's for conversions β and γ are given by

$$\epsilon' \geq \epsilon - \frac{(q_G + q_{H_z} + q_D)}{2^{Len(r)}} \tag{20}$$

and

$$\epsilon' \geq \epsilon - \frac{q_G}{2^{Len(r)}} - \frac{q_D}{2^{Len(Const)}}, \tag{21}$$

respectively.