A note on CCA2-protected McEliece cryptosystem with a systematic public key

Pavol Zajac[⋆]

UIM FEI STU, Ilkovicova 3, 81219 Bratislava, Slovakia
pavol.zajac@stuba.sk

Abstract. We show that the plaintext of some of the proposed CCA2 conversions of McEliece cryptosystem with a public key in systematic form can be recovered faster than with a general linear decoding. This is due to the fact that an attacker only needs to recover a part of the cleartext to decrypt the relevant plaintext.

1 Introduction

McEliece cryptosystem [8] is one of the oldest public-key systems. In its basic form, its security is based on the hardness of the NP-hard general decoding problem. A classical system is not cryptographically secure (resend attack [3], partially known plaintext attack [5]...). Many of these problems can be avoided by using CCA2-secure conversion of the McEliece cryptosystem [7].

Bernstein et.al. in [1] state that

> If we secure McEliece encryption against chosen-ciphertext attacks then we can use a systematic generator matrix as a public key.

Biswas and Sendrier use systematic public key in their Hybrid McEliece scheme [4], again stating that it has no impact on security. Overbeck and Sendrier in [11] go into more detail, stating that if the plaintext m is uniformly distributed, both versions (systematic and non-systematic generator matrix, respectively) are equally secure. Strenzke in [15] states

> Given such a CCA2-conversion is used, it is also possible to choose the matrix T in such a way that G_p is in systematic form.

Due to its many advantages, the systematic form of public key is used in many published implementations [4,13,16,6], relying on the CCA2 conversions for protecting the message.

In this paper we show that some of the proposed implementations, which use a public key in a systematic form and particular forms of CCA2 conversions can, under a specific choice of parameters, lead to a weaker system than their proposed security level based on the hardness of the general decoding problem. Specifically, we focus on the implementation from Shoufan et.al. [13] with their adaptation of Kabara-Imai scheme, and on the implementation FLEA from Strenzke [14] using the proposed CCA2-scheme from Overbeck [10]. In fact, both of the implementations use the same conversion based on Pointcheval's generic scheme [12].

The presented attack does not work with schemes presented by Kobara and Imai [7] (including their version of Pointcheval's conversion). However, the attack is applicable in general for any MECS conversion where the attacker can verify a part of the cleartext by other means than decoding the whole cleartext (for a relevant range of parameters).

⋆ Support by NATOs Public Diplomacy Division in the framework of "Science for Peace", Project MD.SFPP 984520, is acknowledged.

2 Preliminaries

Let G be a $k \times n$ generating matrix of a code $\mathcal{C}$, for which there is an efficient algorithm $Decode_{\mathcal{C}}$ that can decode any codeword with up to t errors. Let S be a random non-singular $k \times k$ matrix, and let P be a random $n \times n$ permutation matrix. Generic McEliece cryptosystem (MECS) is defined as follows:

Secret key $(Decode_{\mathcal{C}}, S, P)$
Public key $\hat{G} = S \cdot G \cdot P$
Encryption Let m be a k-bit message, and let e be an random n-bit vector with $w_H(e) \leq t$. Then ciphertext is computed as $c = m\hat{G} + e$.
Decryption Decryption is given by the following algorithm:

```
1: $c' \leftarrow cP^{-1}$
2: $m' \leftarrow Decode_{\mathcal{C}}(c')$
3: $m \leftarrow m'S^{-1}$
```

Typically, MECS is implemented by using irreducible binary Goppa codes. Our attack is independent of the code used, it depends only on the choice of parameters (n, k, t), and the parameters of the CCA2 conversion. In the latter text, we will suppose that matrix S is not random, but chosen in such a way that $\hat{G}$ is in a systematic form. I.e., $\hat{G} = (I_k||R)$, where I_k is a $k \times k$ identity matrix, and R is a public $k \times (n-k)$ matrix (but indistinguishable from a random matrix).

We will focus on MECS with CCA2 conversion used in [13,14]. Let us assume that message m is an l bit string. The encryption algorithm works as follows:

```
1: $k_e \leftarrow$ a random $k - l$ bit string
2: $k_i \leftarrow$ a random $l$ bit string
3: $e \leftarrow$ a random $n$-bit string with $w_H(e) = t$
4: $\tilde{m} \leftarrow k_e||hash(m||k_i)$
5: $c \leftarrow \tilde{m} \cdot \hat{G}$
6: $y \leftarrow c \oplus e||m \oplus hash(k_e)||k_i \oplus hash(e)$
```

Here, k_e is a session key used to mask the original message m, and k_i is a key used for protecting the integrity of the ciphertext. We note that the attacker only requires the key k_e, which can be used to decrypt the original message from the second part of the ciphertext.

3 Attack description and complexity

If the McEliece public key is in a systematic form $\hat{G} = (I_k||R)$, we can rewrite the CCA2-protected ciphertext y as follows:

$$\begin{array}{ccccc} y_1|| & y_2|| & y_3|| & y_4|| & y_5 = \\ k_e \oplus e_1|| & hash(m||k_i) \oplus e_2|| & \tilde{m} \cdot R \oplus e_3|| & m \oplus hash(k_e)|| & k_i \oplus hash(e) \end{array}$$

The legitimate recipient will use his private key to decode the $y_1||y_2||y_3$, extract $\tilde{m}$, and e, decrypt message m from $\tilde{m}$ and y_4, decrypt integrity key k_i, and check the integrity using $hash(m||k_i)$. The attacker can compromise the message, if he can decode $y_1||y_2||y_3$ without the private key of the MECS. Typically, the security level is given by the complexity of the generalized information set decoding, which is supposed to be the most efficient attack the attacker has for a properly implemented system.

However, the attacker is really only interested in a $(k-l)$-bit string $y_1 = k_e \oplus e_1$. If he can correct errors in e_1 part, he can recover k_e, and from $y_4 = m \oplus hash(k_e)$ compute the hidden message. The attacker can guess e_1, and compute $m = y_4 \oplus hash(y_1 \oplus e_1)$. If the guess was incorrect, he gets a random string (property of the *hash* function). Otherwise he gets the original message, which we can assume is distinguishable from a random string. Even if the message is just a random session key for some subsequent symmetric encryption, attacker can always try to decrypt the symmetric message as well (or verify MAC, etc.).

The attack can thus be summarized as follows: Instead of decoding whole $\tilde{m}$ with t errors in n bits, the attacker only needs to locate errors in the first $(k-l)$-bits to compute k_e. He can verify the correctness of the decoded k_e and recover m by computing $m = y_4 \oplus hash(y_1 \oplus e_1)$.

We do not know whether generalized information set decoding can be used to recover just a part of the error vector more efficiently than the whole error vector. However, even a simple brute-force attack can be more efficient than the prescribed security level, if the parameters are chosen incorrectly. Suppose that security level is s. Choose the encryption key of size $k - l = s$ (if it is shorter, the system is trivially broken just by guessing $hash(k_e)$). The attacker needs only to enumerate s-bit vectors within a certain Hamming distance (approximately $s \cdot \lceil t/n \rceil$) of y_1 to uncover k_e, instead of enumerating all 2^s vectors. Thus the attack is more efficient than the prescribed security level s.

If the attacker needs to recover $k - l$ bits of the cleartext, the complexity of the attack is approximately

$$\sum_{i=0}^{\lceil t/n \rceil \cdot (k-l)} \binom{k-l}{i} \leq 2^{H(t/n)(k-l)}.$$

The security of the system thus depends not only on the system parameters (n, k, t), but also on the choice of l. Usually, the selection of l is based on the typical length of the hash function output. However, larger values of l decrease a relative overhead of the analysed encryption scheme, thus it may seem more attractive for implementers to use longer messages and larger hash functions. Thus, we get a very counter-intuitive property of the system: Using a hash function with longer output decreases the overall security instead of increasing it.

Overbeck in [10] introduces a specific conversion for small messages $(k - 3l \geq l)$, where

$\tilde{m} \leftarrow k_e || m \oplus hash(k_e) || k_i \oplus hash(e) || hash(m || k_i)$

In this case, the ciphertext is just encoded $\tilde{m}$ with the added errors.

Similarly to the previous attack, if the public key is in a systematic form, attacker tries to correct errors in the k_e part of size $k - 3l$, and verify them using $m \oplus hash(k_e)$ part. As the size of k_e is much smaller than in the previous case, it is easier to recover k_e. However, the verification part is now also corrupted by approximately $l \cdot \lceil t/n \rceil$ errors. The success rate thus depends on the entropy of the message m. If m is a random session key used for further encryption, the attacker must correct errors in both parts. In this case the complexity of the attack is based on decoding errors in $k - 2l$ bits (out of n). However, the implementers of the system should always suppose that the worst case scenario, when the message m can be easily distinguished even in the presence of errors, and the attacker only needs to correctly recover $k - 3l$ bits of k_e.

We summarize the complexity of the attack for some of the recommended parameter choices from [8,1,2,9] in Table 1, along with the limits to parameter l based on security level. Maximum

Table 1. Complexity of the attack for selected parameter choices of the MECS. Max. l denotes the maximum value of parameter l, for which the estimated attack cost is approximately equal to the expected security level. Choices of l larger than this maximum lead to insecure systems.

Sec. Level	(n, k, t)	$H(t/n)$	$l = 2s$	$l = 6s$	Max. l
50 [8]	(1024,524,50)	0.281	119	63	346
80 [1]	(2048,1751,27)	0.101	161	129	961
80 [9]	(1702,1219,45)	0.176	187	130	765
128 [2]	(3178,2384,68)	0.149	318	241	1526
256 [2]	(6944,5208,136)	0.139	653	511	3368

size l should be divided by 3 if one uses a conversion from [10] for small messages. If l is chosen to be double the security level (preventing collision attacks in hash function), the system can resist the attack in all analysed cases. However, we should caution that the complexity estimate is based on the estimated complexity of a simple brute force attack, and may be improved in the future by more advanced attacks.

The conversions presented by Kobara and Imai [7] require the attacker to recover the whole cleartext $\tilde{m}$ to retrieve original message m. The complexity of the brute-force attack in this case is approximately $2^{H(t/n)k}$. However, if the attacker can repair all errors in k bits of encoded code-word, he has already succeeded in the information-set decoding attack, even if the public key matrix is not in a systematic form. Thus the attack is not relevant in this case, as it should be already thwarted by a proper choice of the parameters of the MECS.

4 Conclusions

Some of the proposed CCA2 conversions of the McEliece cryptosystem [13,14,10] use only a part of the cleartext to embed a session key k_e for the payload message. An additional parameter l controls the size of the payload message, and inversely the size of the session key (in the studied systems $k - l$, or $k - 3l$, respectively). If the public key for the system is in a systematic form, the session key as a part of cleartext is a specific part of the ciphertext, masked only by a small number of error bits. If the size of k_e is not large enough (e.g., due to the implementers trying to reduce the data redundancy, or as a possible backdoor), the attacker can recover the k_e and the message by trying to guess the error bit positions.

We provide the estimated complexities of the attack and the upper bound on l for [13,14] in Table 1. These estimates are based on a simple brute force search of error positions, and do not take into account a possibility of more advanced attacks.

References

1. Daniel J. Bernstein, Tanja Lange, and Christiane Peters, *Attacking and defending the McEliece cryptosystem.*, PQCrypto (Johannes Buchmann and Jintai Ding, eds.), Lecture Notes in Computer Science, vol. 5299, Springer, 2008, pp. 31–46.
2. Daniel J. Bernstein, Tanja Lange, and Christiane Peters, *Smaller decoding exponents: Ball-collision decoding*, CRYPTO (Phillip Rogaway, ed.), Lecture Notes in Computer Science, vol. 6841, Springer, 2011, pp. 743–760.
3. Thomas A Berson, *Failure of the mceliece public-key cryptosystem under message-resend and related-message attack*, Advances in CryptologyCRYPTO'97, Springer, 1997, pp. 213–220.
4. Bhaskar Biswas and Nicolas Sendrier, *McEliece Cryptosystem Implementation : Theory and Practice*, Proceedings of the 2nd International Workshop on Post-Quantum Cryptography (Berlin, Heidelberg), PQCrypto '08, Springer-Verlag, 2008, pp. 47–62.

5. Anne Canteaut and Nicolas Sendrier, *Cryptanalysis of the original McEliece cryptosystem*, Advances in Cryptology—ASIACRYPT98, Springer, 1998, pp. 187–199.
6. Thomas Eisenbarth, Tim Güneysu, Stefan Heyse, and Christof Paar, *MicroEliece : McEliece for Embedded Devices*, Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (Berlin, Heidelberg), CHES '09, Springer-Verlag, 2009, pp. 49–64.
7. Kazukuni Kobara and Hideki Imai, *Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC*, Public Key Cryptography (Kwangjo Kim, ed.), LNCS, vol. 1992, Springer, 2001, pp. 19–35.
8. R. J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, DSN progress report **42** (1978), no. 44, 114–116.
9. Robert Niebuhr, Mohammed Meziani, Stanislav Bulygin, and Johannes Buchmann, *Selecting parameters for secure mceliece-based cryptosystems*, Int. J. Inf. Sec. **11** (2012), no. 3, 137–147.
10. R Overbeck, *An analysis of side channels in the mceliece pkc (2008)*, `https://www.cosic.esat.kuleuven.be/nato_arw/slides_participants/Overbeck_slides_nato08.pdf`.
11. Raphael Overbeck and Nicolas Sendrier, *Code-based cryptography*, Post-quantum cryptography, Springer, 2009, pp. 95–145.
12. David Pointcheval, *Chosen-ciphertext security for any one-way cryptosystem*, Public Key Cryptography, Springer, 2000, pp. 129–146.
13. Abdulhadi Shoufan, Thorsten Wink, H. Gregor Molter, Sorin A. Huss, and Eike Kohnert, *A novel cryptoprocessor architecture for the McEliece public-key cryptosystem*, IEEE Trans. Computers **59** (2010), no. 11, 1533–1546.
14. Falko Strenzke, *A side-channel secure and flexible platform-independent implementation of the McEliece PKC–flea version 0.1. 1–*, `http://www.cryptosource.de/flea_doc.pdf`.
15. Falko Strenzke, *How to implement the public key operations in code-based cryptography on memory-constrained devices*, IACR Cryptology ePrint Archive **2010** (2010), 465.
16. Falko Strenzke, *A smart card implementation of the McEliece PKC*, The 4th IFIP WG 11.2, Proceedings (Berlin, Heidelberg), WISTP'10, Springer-Verlag, 2010, pp. 47–59.