




How to Backdoor (Classic) McEliece and How to Guard Against Backdoors

Tobias Hemmert¹, Alexander May², Johannes Mittmann¹, and Carl
Richard Theodor Schneider²

¹ Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany,
tobias.hemmert@bsi.bund.de, johannes.mittmann@bsi.bund.de

² Ruhr-University Bochum, Germany, alex.may@rub.de, carl.schneider@rub.de

Keywords: Classic McEliece · Niederreiter · Backdoor · Public-Key Cryptography · SETUP · Post-Quantum Cryptography

Abstract. We show how to backdoor the McEliece cryptosystem such that a backdoored public key is indistinguishable from a usual public key, but allows to efficiently retrieve the underlying secret key.

For good cryptographic reasons, McEliece uses a small random seed δ that generates via some pseudo random generator (PRG) the randomness that determines the secret key. Our backdoor mechanism works by encoding an encryption of δ into the public key. Retrieving δ then allows to efficiently recover the (backdoored) secret key. Interestingly, McEliece can be used itself to encrypt δ , thereby protecting our backdoor mechanism with strong post-quantum security guarantees.

Our construction also works for the current *Classic McEliece* NIST standard proposal for non-compressed secret keys, and therefore opens the door for widespread maliciously backdoored implementations.

Fortunately, our backdoor mechanism can be detected by the owner of the (backdoored) secret key if δ is stored after key generation as specified by the Classic McEliece proposal. Thus, our results provide strong advice for implementers to store δ inside the secret key and use δ to guard against backdoor mechanisms.

1 Introduction

Strong cryptography provides confidentiality to everyone. While this is in general a highly desirable goal, it is a large obstacle for adversarial parties. Thus, there exist strong interests to circumvent cryptographic mechanisms by e.g. installing backdoors in cryptographic protocols. In a nutshell, a backdoored cryptographic scheme is a scheme that provides strong cryptographic properties, unless one possesses a key to a backdoor that allows for easy recovery of some secret value.

The process of establishing backdoors in cryptographic schemes is especially promising during a standardization process. As an example, the Snowden revelations showed that the Dual EC DRBG standard was maliciously backdoored [3].

Since we are now close to standardizing new cryptographic schemes for the era of quantum computers, it is of crucial importance to understand whether

the current candidate schemes allow for backdoor mechanisms. In this work, we address one of the candidates, the McEliece cryptosystem, for which we show how to install backdoors, as well as how to detect them.

Previous Work. The foundational works of Simmons [11,12] describe how digital signatures can be used to secretly communicate information over subliminal channels. Subliminal channels provide a way of adding hidden information to the plain communication by choosing values not at random, but depending on the hidden information. This core idea opened the pathway to a specific type of backdoors, where the hidden information simplifies attacking the backdoored scheme.

Young and Yung [14,15] initiated the area of *kleptography* which considers how an adversary \mathcal{A} can subtly modify a cryptosystem such that it leaks some secret information that only \mathcal{A} is able to recover, and whose output is indistinguishable from a legitimate system for anyone only having black-box access to it. An example for this is captured by their SETUP (Secretly Embedded Trapdoor with Universal Protection) notion. A SETUP mechanism encodes information into the public key of an asymmetric cryptosystem during the key generation process, allowing the adversary \mathcal{A} to later retrieve the underlying secret key using \mathcal{A} 's secret backdoor key. A SETUP requires that given just the generated keys (i.e. without access to source code or to the secret backdoor key), it is impossible to tell whether the keys have been backdoored or not, and that nobody but the owner of the secret backdoor key can retrieve secret keys from public keys even if the implementation eventually gets revealed. This means in particular that a SETUP mechanism has to use asymmetric encryption to embed the secret information in its generated public keys.

RSA backdoors are described by Crépeau and Slakmon [5] who for example encoded half of the bits of the RSA prime p into the public RSA modulus N . However, their backdoor mechanisms do not satisfy all properties of a SETUP since they use a secret permutation that allows anyone with access to the source code to recover the backdoor information. Young and Yung describe a proper kleptographic mechanism for RSA in [16].

For post-quantum secure cryptosystems, very little is known about successful SETUP mechanisms. The work of Kwant, Lange and Thissen [7] describes a backdoor mechanism at the cost of increasing the probability of decryption failures, which might be used to leak information about the secret key. The work of Yang, Xie and Pan [13] however shows that [7] does not fulfill the SETUP notion since the backdoors can be detected efficiently. Moreover, Yang, Xie and Pan [13] introduce SETUP mechanisms for RLWE-based schemes that encode non-quantum secure ECC encryptions of the secret key into the generated public keys.

For code-based cryptosystems and especially McEliece, to the best of our knowledge no SETUP backdoor mechanism is known. Loidreau and Sendrier [8] show that there is a subset of weak Goppa polynomials that makes it feasible to enumerate them when attacking McEliece. This however does not fulfill the SETUP notion, because one can immediately identify from the secret keys that

the resulting scheme has been backdoored. It also does not provide exclusive access. In the context of the NIST standardization process, [4] suggests that maliciously chosen permutation matrices in the key generation algorithm may allow to leak secret values.

For preventing backdoors from a theoretical viewpoint, Bellare, Paterson and Rogaway [2] introduced the watchdog model. However, applying the watchdog model to McEliece does not result in a practical encryption scheme.

Our Contribution. We propose the first SETUP mechanism for McEliece. For didactic reasons, we first address a usual Vanilla Niederreiter version of McEliece, that uses the parity check matrix of a code C as secret key. The randomness for generating C comes from the output of a PRG applied to some secret seed δ .

The public key is a randomized and permuted basis of C . A malicious adversary \mathcal{A} may now backdoor the key generation process of a user \mathcal{U} by encoding an encryption of δ (under \mathcal{A} 's public key $\text{pk}_{\mathcal{A}}$) into \mathcal{U} 's public key $\text{pk}_{\mathcal{U}}$ using a different permutation of C . We show that the resulting backdoored keys are indistinguishable from ordinary McEliece keys under some mild assumption. This indistinguishability even holds when our SETUP mechanism, $\text{pk}_{\mathcal{A}}$, and the secret code C are known. Thus, there is no way to check for the user \mathcal{U} whether the generated secret/public key pair has been backdoored as long as \mathcal{U} only has black-box access to the key generation and cannot inspect the implementation. In the terminology of Young and Yung we therefore provide a *strong* SETUP.

However, if the user \mathcal{U} knows in addition the secret seed δ , then \mathcal{U} can identify backdoored keys. The reason is that the randomness for transforming the secret key $\text{sk}_{\mathcal{U}}$ into the public key $\text{pk}_{\mathcal{U}}$ usually also comes from the PRG output on δ . Thus, δ already fully determines the public key given the secret key. So \mathcal{U} may rerun the secret/public key generation from the verifiable randomness provided by δ to check for the validity of a key pair.

Thus, if the seed δ is included into \mathcal{U} 's secret key $\text{sk}_{\mathcal{U}}$, then our backdoor mechanism is detectable from $\text{sk}_{\mathcal{U}}$. In the terminology of Young and Yung we therefore provide a *weak* SETUP for McEliece when δ is part of the secret key since the backdoor still cannot be detected given only the public key $\text{pk}_{\mathcal{U}}$.

As a main result, we then show that our SETUP backdoor mechanism easily transfers from our Vanilla McEliece scheme to *Classic McEliece* [1], the 4th round NIST standardization candidate. This might at first sight come as a surprise, since our SETUP uses the permutation to embed the backdoor, while Classic McEliece does not permute the entries of C . However, we observe that Classic McEliece inherently includes a permutation that defines the Goppa code, which can be used analogously for our SETUP. Since the proposal [1] requires to store δ as part of the secret key, our construction yields a weak SETUP for Classic McEliece, but turns into a strong SETUP if an implementer chooses to deviate from the specification and deletes δ after key generation. This also emphasizes the importance of making the seed δ a mandatory part of the secret key when specifying any McEliece-based cryptosystem. Our SETUP mechanism applies to the non-compressed key format described in the Classic McEliece submission, not to the compressed formats.

Last but not least, we show that a backdoor implementer \mathcal{A} may use McEliece itself for encrypting δ , thereby securing our backdoor even in the presence of quantum computers.

Implementer’s Advice. Our results show that inclusion of the secret δ efficiently protects against strong SETUP backdoor mechanisms, though not against weak SETUPS. Thus, our results strongly suggest including δ into the secret key. This enables users to verify the absence of our SETUP mechanism in black-box implementations of McEliece key generation with other (trusted) implementations.

We would like to stress that storing δ is not necessary for McEliece functionality. The original purpose of δ is to provide a small piece of randomness, from which one can efficiently derive the full McEliece secret/public key pairs. To this end, standards usually recommend to store δ . Our work shows another strong benefit of storing δ , since δ serves as a short proof for the correct, non-backdoored, deterministic derivation of the secret/public key pair.

In general, open-source implementations and code reviews are recommended for establishing trust in cryptographic implementations. However, code reviews are not always feasible or sufficient in practice. In these cases, access to δ provides an efficiently verifiable witness of a correct key generation with respect to the specification.

Open Problems. Since we describe the first SETUP backdoor mechanism for code-based cryptography, one might wonder whether our SETUP transfers without much effort to other code-based schemes like BIKE or HQC. However, BIKE/HQC both use cyclic codes, whose structure seems to prevent a direct application of our method. It remains an open problem to derive weak/strong SETUP mechanisms in the cyclic setting.

Paper Organization. In Section 2 we give an introduction to McEliece and the SETUP backdoor mechanism of Young and Yung [15], Section 3 provides the strong SETUP mechanism for Vanilla McEliece (without storing δ), as well as the backdoor identification when δ is provided in the secret key. In Section 4 we provide the necessary modifications to our SETUP for Classic McEliece. Eventually, in Section 5 we show how to use McEliece to hide the encryption of δ in a user’s public key. Appendix A contains a simpler but (instructively) flawed backdoor construction.

2 Background

2.1 McEliece and Binary Goppa Codes

McEliece uses a binary linear $[n, k]$ -code C , i.e., $C \subset \mathbf{F}_2^n$ is a subspace of dimension k . C may be described by a generator matrix $G \in \mathbf{F}_2^{k \times n}$, or equivalently by a so-called parity check matrix $H \in \mathbf{F}_2^{(n-k) \times n}$ whose kernel is C .

Due to efficiency reasons, all modern instantiations of McEliece use a parity check matrix, usually called the Niederreiter version of McEliece. While our

SETUP backdoor mechanism for Vanilla McEliece from Section 3 works for any code, our SETUP mechanism from Section 4 also uses properties of the binary Goppa codes that are used in the *Classic McEliece* scheme [1].

Thus, let us briefly recall the parity check matrix of a binary Goppa code. Let \mathbf{F}_{2^m} be a binary field. Choose $\alpha_1, \dots, \alpha_n$ distinct from \mathbf{F}_{2^m} , and an irreducible Goppa polynomial $g \in \mathbf{F}_{2^m}[x]$ of degree t . This defines a linear length- n code C with minimal distance at least $2t + 1$ and parity check matrix

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & & \ddots & \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix} \begin{pmatrix} g(\alpha_1) & 0 & \cdots & 0 \\ 0 & g(\alpha_2) & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & g(\alpha_n) \end{pmatrix}^{-1}$$

$$= \begin{pmatrix} \frac{1}{g(\alpha_1)} & \frac{1}{g(\alpha_2)} & \cdots & \frac{1}{g(\alpha_n)} \\ \frac{\alpha_1}{g(\alpha_1)} & \frac{\alpha_2}{g(\alpha_2)} & \cdots & \frac{\alpha_n}{g(\alpha_n)} \\ \vdots & & \ddots & \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \frac{\alpha_2^{t-1}}{g(\alpha_2)} & \cdots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}.$$

Notice that $H \in \mathbf{F}_{2^m}^{t \times n}$. If we write the elements of H in an \mathbf{F}_2 -basis, then we end up with an $(mt \times n)$ -matrix, i.e., C is a $k \geq n - mt$ dimensional subspace of \mathbf{F}_2^n .

2.2 SETUP Mechanism

SETUP (Secretly Embedded Trapdoor with Universal Protection) mechanisms were introduced by Young and Yung [14,15]. A SETUP mechanism transforms a cryptosystem Π into a backdoored cryptosystem Π' for a malicious backdoor holder \mathcal{A} with asymmetric key pair $(\text{sk}_{\mathcal{A}}, \text{pk}_{\mathcal{A}})$. This transformation fulfills the following properties.

1. *The input to functions in Π' agrees with the specification of inputs to Π .*
This property ensures the compatibility of Π and Π' .
2. *Π' is still efficient and uses $\text{Enc}_{\text{pk}_{\mathcal{A}}}$ (and possibly other functions as well).*
3. *$\text{Dec}_{\text{sk}_{\mathcal{A}}}$ is not part of Π' and is only known to \mathcal{A} .*
This prevents the use of symmetric schemes and guarantees \mathcal{A} exclusive access to the backdoor, assuming that \mathcal{A} 's used asymmetric scheme is secure. In particular, the legitimate user is not able to decrypt the backdoor information, even with access to the implementation of Π' .
4. *The output of algorithms in Π' is compatible with the specification of outputs of algorithms in Π . At the same time, it contains information efficiently derivable by \mathcal{A} only.*

The output of Π' needs to be compatible to Π in the sense that e.g. a ciphertext created with an encryption function from Π' must be decryptable by the corresponding decryption function in Π . While maintaining this compatibility, output of Π' additionally needs to contain information that only the adversary can derive efficiently.

Moreover, SETUP mechanisms can be grouped into categories of different strength. We focus only on the *weak* and *strong* SETUP from [15].

Weak SETUP. The output of Π and Π' are polynomially indistinguishable, except for \mathcal{A} and the legitimate user \mathcal{U} of the implementation. Thus, in a weak SETUP, \mathcal{U} may identify with the help of the generated secret key $\text{sk}_{\mathcal{U}}$ from Π' the existence of a backdoor. All users knowing only $\text{pk}_{\mathcal{U}}$ and $\text{pk}_{\mathcal{A}}$ cannot identify a backdoor in \mathcal{U} 's key, i.e. all users except \mathcal{U} and \mathcal{A} .

Strong SETUP. The output of Π and Π' are polynomially indistinguishable, except for \mathcal{A} . Thus, a user \mathcal{U} cannot recognize any backdoors, even when \mathcal{U} knows the SETUP mechanism and $\text{pk}_{\mathcal{A}}$.

We will formalize the notions for weak and strong SETUP in Section 3, and especially for proving Theorem 1.

3 Backdooring Vanilla McEliece

Recall that for didactic reasons we first define some generic McEliece system in Niederreiter form, called *Vanilla McEliece*. Our Vanilla McEliece scheme has the advantage that it does not rely on specifics of the underlying code, and as opposed to Classic McEliece explicitly uses a permutation matrix P , in which we embed our strong SETUP mechanism.

Let us start by defining Vanilla McEliece's key generation algorithm.

3.1 Key Generation for Vanilla McEliece

In the key generation process of Vanilla McEliece, see also Figure 1, the secret parity check matrix $H \in \mathbf{F}_2^{(n-k) \times n}$ of a binary linear $[n, k]$ -code C is scrambled by a random invertible linear transformation $S \in \mathbf{F}_2^{(n-k) \times (n-k)}$ and a random permutation matrix $P \in \mathbf{F}_2^{n \times n}$. The resulting public key is $\text{pk} = SHP \in \mathbf{F}_2^{(n-k) \times n}$, and the secret key is $\text{sk} = (C, S, H, P)$. It is important to stress that the randomness for constructing C, S, H, P is chosen from the output of a PRG $G(\cdot)$ applied to a short random seed δ , say of 256 bits. Thus a small seed δ completely determines sk and allows compact storage of the secret key.

The invertible matrix S does not affect the code C . The matrix P permutes the coordinates of C , resulting in a code equivalent to C . From a security perspective, the transformations S, P are supposed to completely hide the structure of the underlying C . The security of McEliece is based on pk behaving like a random parity check matrix, for which the *syndrome decoding problem* is hard.

3.2 Vanilla McEliece Strong SETUP

Our SETUP mechanism for Vanilla McEliece manipulates the key generation in such a way that the keys are indistinguishable from legitimate keys, but

$\text{KGen}_V(1^n)$

```

1 :  $\delta \leftarrow_{\$} \{0, 1\}^s$ 
2 :  $r := G(\delta)$  //  $G$  is a PRG
3 : Generate  $C$  with parity check matrix  $H$  from  $r$ .
4 : Compute random  $S, P$  from  $r$ .
5 : return  $\text{sk} := (C, S, H, P)$ ,  $\text{pk} := SHP$ 

```

Fig. 1. Vanilla McEliece Key Generation

knowledge of a secret backdoor allows an adversary to recover the secret key from the corresponding public key. This is achieved by encrypting the random seed δ using a public-key encryption scheme Π_A of the adversary's choice with the public key pk_A to obtain a ciphertext $c \leftarrow_{\$} \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$. Then c is embedded in the random permutation P such that it can be recovered just from the public key.

Encoding via permutation. Let us denote by $P^{(n)} \subset \mathbf{F}_2^{n \times n}$ the set of n -dimensional permutation matrices, so $|P^{(n)}| = n!$. We write a permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ as $\pi = (\pi_1, \dots, \pi_n)$ with $\pi_i = \pi(i)$. Let e_i be the i -th unit vector, written in column form. Then we define the permutation matrix $P_\pi \in P^{(n)}$ corresponding to π as

$$P_\pi = (e_{\pi(1)} \dots e_{\pi(n)}).$$

We use an efficiently computable bijection from Kreher and Stinson [6] (Algorithms 2.15 and 2.16, see also Figure 2),

$$f_n : \{0, 1, \dots, n! - 1\} \rightarrow P^{(n)},$$

that maps numbers to permutation matrices, and vice versa.

We use f_n to encode $c \leftarrow_{\$} \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$ as a permutation. Notice that the algorithms from Figure 2 efficiently compute f_n and f_n^{-1} both in time $\mathcal{O}(n^2)$.

Idea of our Vanilla McEliece Backdoor. Our backdoored key generation $\widetilde{\text{KGen}}_V$ is described in Figure 3.

The parity check matrix H and the invertible matrix S are generated from the random seed δ as in the non-backdoored key generation KGen_V from Figure 1.

We assume w.l.o.g. that the columns of H are pairwise distinct, otherwise the code C defined by H has minimal distance at most 2 and is not suitable for McEliece. Therefore, SH also has pairwise distinct columns. Thus, we can unambiguously sort the columns of SH in lexicographic order $<_{\text{lex}}$. Let $P' \in P^{(n)}$ be the permutation that realizes this sorting.

Using standard rejection sampling, we expand $c \leftarrow_{\$} \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$ to a bit representation of a number $a \in \{0, 1, \dots, n! - 1\}$. Notice that $\widetilde{\text{KGen}}_V$ requires

$$\ell \leq \log_2(n!), \tag{1}$$

| $f_n(a)$ | $f_n^{-1}(P)$ |
|--|--|
| <pre> 1: $\pi_n := 1$ 2: for $j = 1..n - 1$ do 3: $d := (a \bmod (j + 1)!)/j!$ 4: $a := a - d \cdot j!$ 5: $\pi_{n-j} := d + 1$ 6: for $i = n - j + 1..n$ do 7: if $\pi_i > d$ then 8: $\pi_i := \pi_i + 1$ 9: fi 10: endfor 11: endfor 12: return $P_{(\pi_1, \dots, \pi_n)}$ </pre> | <pre> 1: Let $\pi = (\pi_1, \dots, \pi_n)$ with $P = P_\pi$. 2: $a := 0$ 3: for $j = 1..n$ do 4: $a := a + (\pi_j - 1)(n - j)!$ 5: for $i = j + 1..n$ 6: if $\pi_i > \pi_j$ then 7: $\pi_i := \pi_i - 1$ 8: fi 9: endfor 10: endfor 11: return a </pre> |

Fig. 2. Algorithms for computing f_n and its inverse f_n^{-1}

| $\widetilde{\text{KGen}}_V(1^n, \text{pk}_A)$ |
|--|
| <pre> 1: $\delta \leftarrow \{0, 1\}^s$ 2: $r := G(\delta)$ 3: Generate C with parity check matrix H from r. 4: Compute random S from r. 5: Find permutation P' with $\text{Col}_1(SHP') <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_n(SHP')$. 6: repeat // Rejection sampling of $a \in \{0, 1, \dots, n! - 1\}$ 7: $c \leftarrow \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$ 8: $s \leftarrow \{0, 1\}^{\lceil \log_2 n! \rceil - \ell}$ 9: $c' := c s$ and $a := \sum_{i=1}^{\lceil \log_2 n! \rceil} c'_i 2^{i-1}$ 10: until $a < n!$ 11: $P := f_n(a)$ and $\tilde{P} := P'P$ 12: return $\tilde{\text{sk}} := (C, S, H, \tilde{P}), \tilde{\text{pk}} := SH\tilde{P}$ </pre> |

Fig. 3. Backdoored Vanilla McEliece Key Generation.

which is easily met for many encryptions Enc . See Section 5, where we realize Enc with McEliece encryption itself.

If we further assume that c is uniformly distributed in \mathbf{F}_2^ℓ , then a is uniform in $\{0, 1, \dots, n! - 1\}$ and hence $P = f_n(a)$ is uniform in $P^{(n)}$. This implies that $\tilde{P} = P'P$ is uniform in $P^{(n)}$.

Secret Key Recovery. Notice that our backdoored key generation algorithm $\widetilde{\text{KGen}}_{\mathbb{V}}$ from Figure 3 generates public keys of the form $\widetilde{\text{pk}} := SH\widetilde{P} = SHP'P$, where SHP' has lexicographically sorted columns.

This enables *anybody* to efficiently compute P . We then derive $a = f_n^{-1}(P)$, from which we can recover $\mathbf{c} = \text{Enc}_{\text{pk}_{\mathcal{A}}}(\boldsymbol{\delta})$. This enables the adversary \mathcal{A} to recover $\boldsymbol{\delta}$ using $\text{Dec}_{\text{sk}_{\mathcal{A}}}$. One then basically reruns $\widetilde{\text{KGen}}_{\mathbb{V}}$ to recover the backdoored secret key $\widetilde{\text{sk}}$. The details are given in Figure 4.

$\text{RECOVER}_{\mathbb{V}}(\text{sk}_{\mathcal{A}}, \widetilde{\text{pk}})$

- 1 : Find permutation P with $\text{Col}_1(\widetilde{\text{pk}}P^{-1}) <_{\text{lex}} \cdots <_{\text{lex}} \text{Col}_n(\widetilde{\text{pk}}P^{-1})$.
- 2 : $a := f_n^{-1}(P)$. Write $a = \sum_{i=1}^{\lceil \log_2 n \rceil} c'_i 2^{i-1}$.
- 3 : $\mathbf{c} := c'_1 \dots c'_\ell$
- 4 : $\boldsymbol{\delta} := \text{Dec}_{\text{sk}_{\mathcal{A}}}(\mathbf{c})$
- 5 : $\mathbf{r} := G(\boldsymbol{\delta})$
- 6 : Generate C with parity check H from \mathbf{r} .
- 7 : Compute random S from \mathbf{r} .
- 8 : Find permutation P' with $\text{Col}_1(SHP') <_{\text{lex}} \cdots <_{\text{lex}} \text{Col}_n(SHP')$.
- 9 : $\widetilde{P} := P'P$
- 10 : **return** $\widetilde{\text{sk}} = (C, S, H, \widetilde{P})$

Fig. 4. Vanilla McEliece Secret Key Recovery

Proof of strong SETUP. Let us first check that our backdoor mechanism for McEliece indeed satisfies the SETUP definition of [15] from Subsection 2.2.

1. *The input to functions in backdoored McEliece agrees with the specification of inputs to McEliece.*

All domains remain unchanged.

2. *Backdoored McEliece is still efficient and uses $\text{Enc}_{\text{pk}_{\mathcal{A}}}$ (and possibly other functions as well).*

Our $\widetilde{\text{KGen}}_{\mathbb{V}}(1^n)$ applies $\text{Enc}_{\text{pk}_{\mathcal{A}}}$, which we assume to be efficient. Since \mathbf{c}' is uniformly distributed in $\{0, 1\}^{\lceil \log_2 n \rceil}$, a is uniform in $\{0, \dots, 2^{\lceil \log_2 n \rceil} - 1\}$, so $\Pr[a < n!] = n!/2^{\lceil \log_2 n \rceil} \geq n!/2^{\log_2 n + 1} = 1/2$. Therefore, the expected number of samples is at most 2, making the rejection sampling efficient. Since $f_n(a)$ and P' are also efficiently computable, our modification remains efficient.

3. *$\text{Dec}_{\text{sk}_{\mathcal{A}}}$ is not part of backdoored McEliece and is only known by \mathcal{A} .*

We solely use $\text{Dec}_{\text{sk}_{\mathcal{A}}}$ in $\text{RECOVER}_{\mathbb{V}}(\text{sk}_{\mathcal{A}}, \widetilde{\text{pk}})$.

4. *The output of algorithms in backdoored McEliece is compatible with the specification of outputs of algorithms in McEliece. At the same time, it contains information efficiently derivable for \mathcal{A} only.*

The output of our backdoored McEliece scheme is fully compatible with the original McEliece scheme, in particular the original decryption function works on the backdoored key pairs $(\widetilde{\text{pk}}, \widetilde{\text{sk}})$. Moreover, our $\widetilde{\text{pk}}$ allows to recover the full secret key $\widetilde{\text{sk}}$ using $\text{RECOVER}_V(\text{sk}_A, \widetilde{\text{pk}})$.

It remains to show that our backdoor mechanism provides a *strong* SETUP for Vanilla McEliece. As the schemes only differ with regard to their key generation, it suffices to show that secret and public keys (sk, pk) and $(\widetilde{\text{sk}}, \widetilde{\text{pk}})$ output by their respective key generation algorithms are polynomially indistinguishable for anyone who knows pk_A — but not sk_A .

Recall that we used the randomness of $\mathbf{c} \leftarrow_{\$} \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$ to derive a random P . In the high-level idea, we showed that uniformly distributed \mathbf{c} lead to uniformly distributed P . Therefore, we want our ciphertexts \mathbf{c} to be indistinguishable from random bit strings even given the adversary's public key pk_A .

This is captured more formally by the following definition.

Definition 1. *Let $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme with ciphertexts $\mathbf{c} \in \mathbf{F}_2^\ell$. For any algorithm $A^\mathcal{O}$ with oracle access to \mathcal{O} , define its advantage $\text{Adv}_{\Pi}^{\text{IND}\$-\text{CPA}}(A)$ to be*

$$\left| \Pr \left[A^{\text{Enc}_{\text{pk}(\cdot)}}(\text{pk}) = 1 \mid (\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen} \right] - \Pr \left[A^{\mathcal{S}(\cdot)}(\text{pk}) = 1 \mid (\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen} \right] \right|.$$

Here the oracle $\text{Enc}_{\text{pk}(\cdot)}$ on input \mathbf{m} returns $\mathbf{c} \leftarrow_{\$} \text{Enc}_{\text{pk}}(\mathbf{m})$, and $\mathcal{S}(\cdot)$ returns a uniformly random $\mathbf{c} \leftarrow_{\$} \{0, 1\}^\ell$ on any input. Π provides indistinguishability from random bits under a chosen plaintext attack, in short IND $\$$ -CPA, if for any ppt adversary A with access to an oracle, $\text{Adv}_{\Pi}^{\text{IND}\$-\text{CPA}}(A)$ is negligible.

It is not hard to see that IND $\$$ -CPA implies IND-CPA. The IND $\$$ -CPA notion has been considered in [10] in the context of symmetric encryption. In Section 5 we show that a variant of the McEliece cryptosystem provides IND $\$$ -CPA under reasonable assumptions.

Theorem 1. *Assume that the adversary's public-key encryption scheme Enc_{pk_A} is IND $\$$ -CPA and pk_A is publicly known. Then original keys $(\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen}(1^n)$ and backdoored keys $(\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow_{\$} \widetilde{\text{KGen}}(1^n, \text{pk}_A)$ are polynomially indistinguishable. Therefore, our algorithms $\widetilde{\text{KGen}}_V(1^n, \text{pk}_A)$ and $\text{RECOVER}_V(\text{sk}_A, \widetilde{\text{pk}})$ define a strong SETUP mechanism for Vanilla McEliece.*

Proof. For any ppt distinguisher D , we define D 's advantage $\text{Adv}_{\text{KGen}, \text{KGen}}^{\text{KeyDistinguish}}(D)$ for distinguishing original from backdoored keys as

$$\left| \Pr \left[D(\widetilde{\text{sk}}, \widetilde{\text{pk}}, \text{pk}_A) = 1 \mid (\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow_{\$} \widetilde{\text{KGen}}_V(1^n, \text{pk}_A) \right] - \Pr \left[D(\text{sk}, \text{pk}, \text{pk}_A) = 1 \mid (\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen}_V(1^n) \right] \right|.$$

Now consider the IND $\$$ -CPA game described in Definition 1 where the oracle $\mathcal{O}_{\text{IND}\$-\text{CPA}}$ is either $\text{Enc}_{\text{pk}_{\mathcal{A}}}(\cdot)$ or $\$(\cdot)$ and we have to decide which one. In Figure 5, we use D to construct an adversary $A_D^{\mathcal{O}_{\text{IND}\$-\text{CPA}}}$ against the IND $\$$ -CPA game. Here, the algorithm $\widetilde{\text{KGen}}_{\mathbf{V}}(1^n, \mathcal{O}_{\text{IND}\$-\text{CPA}})$ is the same as $\widetilde{\text{KGen}}_{\mathbf{V}}(1^n)$ from Figure 3, with the only difference that $\mathbf{c} \leftarrow_{\$} \mathcal{O}_{\text{IND}\$-\text{CPA}}(\boldsymbol{\delta})$ is sampled from the given oracle on input $\boldsymbol{\delta}$ in step 7.

$A_D^{\mathcal{O}_{\text{IND}\$-\text{CPA}}}(1^n, \text{pk}_{\mathcal{A}})$

1 : $(\widetilde{\text{sk}}, \widetilde{\text{pk}}), \leftarrow \widetilde{\text{KGen}}_{\mathbf{V}}(1^n, \mathcal{O}_{\text{IND}\$-\text{CPA}})$, where we compute $\mathbf{c} \leftarrow_{\$} \mathcal{O}_{\text{IND}\$-\text{CPA}}(\boldsymbol{\delta}) \in \mathbf{F}_2^\ell$
2 : **return** $b \leftarrow_{\$} D(\widetilde{\text{sk}}, \widetilde{\text{pk}}, \text{pk}_{\mathcal{A}})$

Fig. 5. Adversary against IND $\$$ -CPA game constructed from D

In case $\mathcal{O}_{\text{IND}\$-\text{CPA}} = \text{Enc}_{\text{pk}_{\mathcal{A}}}$, we perfectly simulate $\widetilde{\text{KGen}}_{\mathbf{V}}(1^n, \text{pk}_{\mathcal{A}})$. Hence

$$\Pr\left[A_D^{\text{Enc}_{\text{pk}(\cdot)}}(1^n, \text{pk}_{\mathcal{A}}) = 1\right] = \Pr\left[D(\widetilde{\text{sk}}, \widetilde{\text{pk}}, \text{pk}_{\mathcal{A}}) = 1 \mid (\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow_{\$} \widetilde{\text{KGen}}_{\mathbf{V}}(1^n, \text{pk}_{\mathcal{A}})\right]$$

Now suppose $\mathcal{O}_{\text{IND}\$-\text{CPA}} = \(\cdot) . In this case, $\widetilde{\text{KGen}}_{\mathbf{V}}(1^n, \mathcal{O}_{\text{IND}\$-\text{CPA}})$ computes a uniformly distributed \mathbf{c} and therefore also a uniform permutation. This differs from the output distribution of $\text{KGen}_{\mathbf{V}}$ only in the fact that $\text{KGen}_{\mathbf{V}}$ generates P from $\mathbf{r} = G(\boldsymbol{\delta})$. Since G is a PRG, we obtain

$$\left| \Pr\left[A_D^{\$(\cdot)}(1^n, \text{pk}_{\mathcal{A}}) = 1\right] - \Pr\left[D(\text{sk}, \text{pk}, \text{pk}_{\mathcal{A}}) = 1 \mid (\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen}_{\mathbf{V}}(1^n)\right] \right| \leq \text{negl}(n).$$

Putting all this together and using that $\text{Adv}_{\Pi_{\mathcal{A}}}^{\text{IND}\$-\text{CPA}}\left(A_D^{\mathcal{O}_{\text{IND}\$-\text{CPA}}}\right) \leq \text{negl}(n)$ by assumption, we deduce that $\text{Adv}_{\widetilde{\text{KGen}}, \text{KGen}}^{\text{KeyDistinguish}}(D) \leq \text{negl}(n)$. \square

3.3 From Strong to Weak SETUP

Recall that in the original McEliece key generation, we derive all randomness from a random seed $\boldsymbol{\delta}$ and hence a key pair (sk, pk) is solely determined by $\boldsymbol{\delta}$. Thus, inclusion of $\boldsymbol{\delta}$ into the secret key allows for a simple verification check of the validity of a key pair, thereby preventing our strong SETUP mechanism.

Denote by $\widetilde{\text{KGen}}_{\mathbf{V}}^{\boldsymbol{\delta}}$ the same algorithm as $\widetilde{\text{KGen}}_{\mathbf{V}}$ with the only difference that the random seed $\boldsymbol{\delta}$ is also included in $\widetilde{\text{sk}}$.

Theorem 2. *Algorithms $\widetilde{\text{KGen}}_{\mathbf{V}}^{\boldsymbol{\delta}}(1^n, \text{pk}_{\mathcal{A}})$ and $\text{RECOVER}_{\mathbf{V}}(\text{sk}_{\mathcal{A}}, \widetilde{\text{pk}})$ define a weak SETUP mechanism for Vanilla McEliece.*

Proof. Let $(\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \widetilde{\text{KGen}}_{\mathbb{V}}^{\delta}(1^n, \text{pk}_{\mathcal{A}})$ with $\tilde{\text{sk}} = (C, S, H, \tilde{P}, \delta)$. Run $\widetilde{\text{KGen}}_{\mathbb{V}}$ with randomness $\mathbf{r} := G(\delta)$, let the output be $\text{sk} = (C, S, H, P)$. We conclude that $\tilde{\text{sk}}$ is backdoored if and only if $P \neq \tilde{P}$.

Thus, we can decide via our secret key $\tilde{\text{sk}}$ whether our scheme has been backdoored. Since $\widetilde{\text{KGen}}_{\mathbb{V}}^{\delta}$ and $\widetilde{\text{KGen}}_{\mathbb{V}}$ differ only by the format of $\tilde{\text{sk}}$, by Theorem 1 our scheme still provides a weak SETUP mechanism. \square

Remark 1. Vanilla/Classic McEliece uses pseudorandomness from a PRG output to construct its secret key. One might think that constructing the secret key from true randomness only makes the scheme more secure. However, our results show that the reproducibility feature of pseudorandomness provides an effective way for detecting backdoors, a feature that cannot be realized by true randomness.

4 How to Backdoor Classic McEliece

In this section, we show that the strategy of embedding a backdoor in the secret permutation P from Section 3 also transfers to *Classic McEliece*.

Changes from Vanilla to Classic McEliece. A simplified version $\text{KGen}_{\mathbb{C}}^{\delta}$ of the Classic McEliece key generation is outlined in Figure 6. The main differences to the Classic McEliece specification are that we do not explicitly describe in detail how the Goppa code is derived from the seed, and that we leave out components of sk that are not relevant to our construction. As in Vanilla McEliece one also uses a seed δ to compute the randomness \mathbf{r} for the Goppa code C and its parity check matrix H . However as opposed to Vanilla McEliece, Classic McEliece does not involve a random invertible S , and further completely omits the use of a permutation matrix P . Instead, let S be the *deterministic* Gaussian elimination matrix that sends H to the unique reduced row-echelon form

$$SH = [I_{n-k} \| T].$$

To this end, we assume that the first $n-k$ columns of H define a full rank matrix. Cases with a slightly relaxed condition can also be handled by some parameter sets of Classic McEliece, but our backdoor remains applicable as these parameter sets perform additional swaps on the α_i of the code to create the same structure $[I_{n-k} \| T]$.

The reason for choosing S as above is that the public key $\text{pk} = T$ is a matrix in $\mathbb{F}_2^{(n-k) \times k}$, thus saving $n-k$ columns in comparison to Vanilla McEliece for efficiency reasons. $\text{KGen}_{\mathbb{C}}^{\delta}$ outputs C and δ as secret key. The Classic McEliece NIST submission also includes in sk a random string used for implicit rejection in case decapsulation fails and some additional data that is only relevant for compression purposes; otherwise the key formats are identical, and so for simplicity we refer to $\text{KGen}_{\mathbb{C}}^{\delta}$ as the original Classic McEliece key generation. For key compression, Classic McEliece also defines multiple truncation levels of sk , where access to the seed δ is required to regenerate the removed elements. Our

construction will yield a weak SETUP only for the uncompressed representation of \mathbf{sk} . Finally, we also consider a modified version KGen_C of the Classic McEliece key generation that does not include the seed in the secret key for which we obtain a strong SETUP.

At first sight, it seems that the absence of P prevents the direct applicability of our SETUP technique from Section 3. Moreover, the deterministic S does not allow for backdoor manipulations either. However, we show in the following that the definition of the Goppa code C already implicitly introduces a permutation P , to which we apply a backdoor mechanism analogous to Section 3.

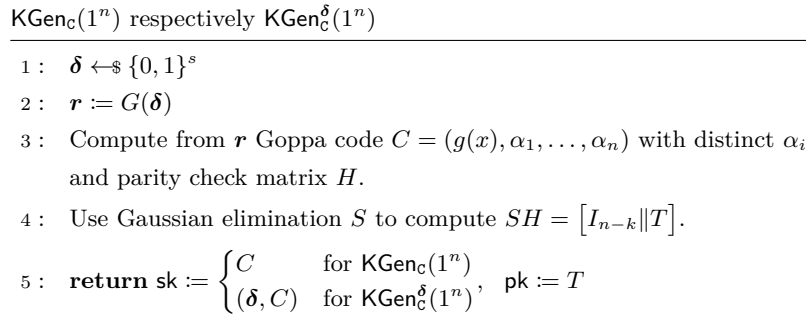


Fig. 6. Simplified representation of the Classic McEliece key generation KGen_C^δ and a modification KGen_C that does not include the seed in the secret key.

Idea of Backdoored Key Generation. Recall from Subsection 2.1 that the parity check matrix of a binary Goppa code is of the form

$$H = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \frac{g(\alpha_1)}{\alpha_1} & \frac{g(\alpha_2)}{\alpha_2} & \dots & \frac{g(\alpha_n)}{\alpha_n} \\ \frac{g(\alpha_1)}{g(\alpha_1)} & \frac{g(\alpha_2)}{g(\alpha_2)} & \dots & \frac{g(\alpha_n)}{g(\alpha_n)} \\ \vdots & & \ddots & \\ \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \frac{\alpha_2^{t-1}}{g(\alpha_2)} & \dots & \frac{\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix}.$$

Here, the $\alpha_i \in \mathbf{F}_{2^m}$ are random, distinct field elements. Note that the i -th column \mathbf{h}_i of H depends on α_i only. Thus, a random choice of the α_i already induces a random permutation of the columns. Moreover, we may use our SETUP mechanism from Section 3 that modifies the permutation to encode the encryption of δ in the public key by modifying the order of the α_i . Due to manipulation of the order of the α_i , this SETUP mechanism is limited to the fully expanded representation of \mathbf{sk} and does not apply to the compressed key formats mentioned above.

In our backdoored key generation algorithm from Figure 7 we permute only the last k columns of the parity check matrix which constitute the (backdoored) public key.

$\widetilde{\text{KGen}}_{\mathcal{C}}(1^n, \text{pk}_{\mathcal{A}})$ respectively $\widetilde{\text{KGen}}_{\mathcal{C}}^{\delta}(1^n, \text{pk}_{\mathcal{A}})$

```

1:  $\delta \leftarrow_{\$} \{0, 1\}^s$ 
2:  $r := G(\delta)$ 
3: Compute from  $r$  Goppa code  $C = (g(x), \alpha_1, \dots, \alpha_n)$  with distinct  $\alpha_i$ 
   and parity check matrix  $H$ .
4: Use Gaussian elimination  $S$  to compute  $SH = [I_{n-k} \| T]$ .
5: Find permutation  $P'$  with  $\text{Col}_1(TP') <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_k(TP')$ .
6: repeat // Rejection sampling of  $a \in \{0, 1, \dots, k! - 1\}$ 
7:    $c \leftarrow_{\$} \text{Enc}_{\text{pk}_{\mathcal{A}}}(\delta) \in \mathbf{F}_2^{\ell}$ 
8:    $s \leftarrow_{\$} \{0, 1\}^{\lceil \log_2 k! \rceil - \ell}$ 
9:    $c' := c \| s$  and  $a := \sum_{i=1}^{\lceil \log_2 k! \rceil} c'_i 2^{i-1}$ 
10: until  $a < k!$ 
11: Set  $P := f_k(a)$ 
12: Set  $\tilde{P} := \begin{pmatrix} I_{n-k} & \mathbf{0} \\ \mathbf{0} & P'P \end{pmatrix}$ .
13: Compute  $\tilde{C} := (g(x), (\alpha_1, \dots, \alpha_n) \cdot \tilde{P})$ .
14: return  $\tilde{\text{sk}} := \begin{cases} \tilde{C} & \text{for } \widetilde{\text{KGen}}_{\mathcal{C}}(1^n, \text{pk}_{\mathcal{A}}) \\ (\delta, \tilde{C}) & \text{for } \widetilde{\text{KGen}}_{\mathcal{C}}^{\delta}(1^n, \text{pk}_{\mathcal{A}}) \end{cases}, \tilde{\text{pk}} := TP'P$ 

```

Fig. 7. Backdoored Classic McEliece Key Generation

Classic McEliece Secret Key Recovery. In Figure 8, we detail the secret key recovery.

The correctness of our $\text{RECOVER}_C(\text{sk}_A, \widetilde{\text{pk}})$ follows analogously to the discussion in Subsection 3.2.

| | |
|--|--|
| $\text{RECOVER}_C(\text{sk}_A, \widetilde{\text{pk}})$ | <ol style="list-style-type: none"> 1 : Find permutation P with $\text{Col}_1(\widetilde{\text{pk}}P^{-1}) <_{\text{lex}} \cdots <_{\text{lex}} \text{Col}_k(\widetilde{\text{pk}}P^{-1})$. 2 : $a := f_k^{-1}(P)$. Write $a = \sum_{i=1}^{\lceil \log_2 k \rceil} c'_i 2^{i-1}$. 3 : $\mathbf{c} := c'_1 \dots c'_\ell$ 4 : $\boldsymbol{\delta} := \text{Dec}_{\text{sk}_A}(\mathbf{c})$ 5 : $\mathbf{r} := G(\boldsymbol{\delta})$ 6 : Compute from \mathbf{r} Goppa code $C = (g(x), \alpha_1, \dots, \alpha_n)$ with distinct α_i and parity check matrix H. 7 : Use Gaussian elimination S to compute $SH = [I_{n-k} \ T]$. 8 : Find permutation P' with $\text{Col}_1(TP') <_{\text{lex}} \cdots <_{\text{lex}} \text{Col}_k(TP')$. 9 : Set $\widetilde{P} := \begin{pmatrix} I_{n-k} & \mathbf{0} \\ \mathbf{0} & P'P \end{pmatrix}$. 10 : Compute $\widetilde{C} := (g(x), (\alpha_1, \dots, \alpha_n) \cdot \widetilde{P})$. 11 : return $\widetilde{\text{sk}} := \widetilde{C}$ |
|--|--|

Fig. 8. Classic McEliece Secret Key Recovery

Analogously to Theorem 1 and Theorem 2 we obtain a weak/strong SETUP for Classic McEliece, depending on whether we include $\boldsymbol{\delta}$ into sk or not.

Theorem 3. *Assume that the adversary's public-key encryption scheme Enc_{pk_A} is IND\$-CPA and pk_A is publicly known. Then original keys $(\text{sk}, \text{pk}) \leftarrow \text{KGen}_C(1^n)$ and backdoored keys $(\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow \text{KGen}_C^\delta(1^n, \text{pk}_A)$ are polynomially indistinguishable. Therefore, our algorithm $\widetilde{\text{KGen}}_C^\delta$ in combination with RECOVER_C defines a weak SETUP mechanism for Classic McEliece, and $\widetilde{\text{KGen}}_C$ defines a strong SETUP mechanism for a modification of Classic McEliece that does not include the PRG-seed $\boldsymbol{\delta}$ in the user's secret key sk.*

5 How to Use McEliece Encryption Against Classic McEliece

We propose to use a variant of the McEliece cryptosystem for the adversary's encryption algorithm Enc . Our scheme can be used to backdoor Classic McEliece for all parameter sets proposed in the NIST submission.

IND\\$-CPA McEliece Encryption. As adversary \mathcal{A} 's **Enc** we use the *Randomized Niederreiter Cryptosystem* from [9]. Randomized Niederreiter public keys are scrambled $(n - k) \times n$ parity check matrices of some binary Goppa codes with minimal distance at least $2t + 1$, just as in our Vanilla McEliece scheme. Let n_1, n_2 with $n_1 + n_2 = n$ and define $t_i = \lfloor \frac{n_i t}{n} \rfloor$ for $i = 1, 2$. We take messages from $\mathcal{M}_{\text{RN}} = \{\mathbf{m} \in \mathbf{F}_2^{n_2} \mid \text{wt}(\mathbf{m}) = t_2\}$, and pad them by a randomly chosen bitstring from $\mathcal{P}_{\text{RN}} = \{\mathbf{r} \in \mathbf{F}_2^{n_1} \mid \text{wt}(\mathbf{r}) = t_1\}$. The padded message $\mathbf{e} = (\mathbf{m} \parallel \mathbf{r}) \in \mathbf{F}_2^n$ is an error vector of weight at most t , for which we compute the so-called syndrome $\mathbf{e} \cdot \mathbf{pk}^T$.

The key generation and encryption algorithm are detailed in Figure 9.

| $\text{KGen}_{\text{RN}}(1^n)$ | $\text{Enc}_{\text{RN}}(1^n, \mathbf{pk}, \mathbf{m})$ |
|--|--|
| 1 : Generate random Goppa code C with parity check matrix $H \in \mathbf{F}_2^{(n-k) \times n}$ | 1 : $\mathbf{r} \leftarrow \mathcal{P}_{\text{RN}}$ |
| 2 : Generate random invertible $S \in \mathbf{F}_2^{(n-k) \times (n-k)}$ and permutation matrix $P \in \mathbf{F}_2^{n \times n}$ | 2 : $\mathbf{e} := \mathbf{m} \parallel \mathbf{r}$ |
| 3 : return $\mathbf{sk} := (C, S, H, P)$, $\mathbf{pk} := SHP$ | 3 : return $\mathbf{c} := \mathbf{e} \cdot \mathbf{pk}^T$ |

Fig. 9. Randomized Niederreiter key generation and encryption for messages $\mathbf{m} \in \mathcal{M}_{\text{RN}}$

Clearly, in order to achieve IND-CPA security, the syndrome decoding problem for the code with $(n - k) \times n_1$ parity check matrix that encodes the randomness \mathbf{r} needs to be hard. Note that this code has dimension at least $k_1 := n_1 - (n - k)$. Proposition 2 of [9] shows that under the standard assumptions that public keys \mathbf{pk} are indistinguishable from random matrices, and that syndrome decoding of random linear $[n_1, k_1, t_1]$ -codes is hard, Randomized Niederreiter provides IND-CPA.

Actually, the authors of [9] prove an even stronger property, called *admissibility* (see Definition 5 in [9]). It is easily seen that admissibility does not only imply IND-CPA, but even IND\\$-CPA from Definition 1. Thus, according to Theorems 1 and 3, Randomized Niederreiter yields a strong SETUP mechanism if δ is not part of the secret, and a weak SETUP mechanism otherwise.

Application to Classic McEliece. For our concrete instantiation of Randomized Niederreiter, we propose to use the Goppa codes from the highest category 5 of the Classic McEliece submission, for which $n = 8192$, $k = 6528$ and $t = 128$. We need to pick n_2 large enough such that $|\mathcal{M}_{\text{RN}}| = \binom{n_2}{t_2} \geq 2^{256}$ so that we are able to encrypt all possible 256-bit strings δ using some suitable encoding $\{0, 1\}^{256} \rightarrow \mathcal{M}_{\text{RN}}$. It is easily checked that the choice $n_2 = 2250$ and hence $t_2 = \lfloor \frac{n_2 t}{n} \rfloor = 35$ suffices. The ciphertext size is $\ell = n - k = 1664$. Table 1 shows that this is significantly smaller than $\log_2(k!)$ for all Classic McEliece parameter sets of given code dimension k , thus satisfying our necessary condition from Equation 1.

Table 1. Parameters for Classic McEliece and the number of bits $\lceil \log_2(k!) \rceil$ for encoding a random permutation P

| Target instance | Category | n | k | $\lceil \log_2(k!) \rceil$ |
|---------------------|----------|------|------|----------------------------|
| kem/mceliece348864 | 1 | 3488 | 2720 | 27117 |
| kem/mceliece460896 | 3 | 4608 | 3360 | 34520 |
| kem/mceliece6688128 | 5 | 6688 | 5024 | 54528 |
| kem/mceliece6960119 | 5 | 6960 | 5413 | 59332 |
| kem/mceliece8192128 | 5 | 8192 | 6528 | 73316 |

References

- Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., Maurich, I.v., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece: Conservative code-based cryptography. <https://classic.mceliece.org/nist/mceliece-20201010.pdf> (Oct 2020)
- Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_1
- Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: A Standardized Back Door. In: Ryan, P.Y.A., Naccache, D., Quisquater, J.J. (eds.) The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday, pp. 256–281. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49301-4_17
- Classic McEliece Comparison Task Force: Classic McEliece vs. NTS-KEM (Jun 2018), <https://classic.mceliece.org/nist/vsntskem-20180629.pdf>
- Crépeau, C., Slakmon, A.: Simple backdoors for RSA key generation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 403–416. Springer, Heidelberg (Apr 2003). https://doi.org/10.1007/3-540-36563-X_28
- Kreher, D.L., Stinson, D.R.: Combinatorial Algorithms: Generation, Enumeration, and Search. CRC Press (1999)
- Kwant, R., Lange, T., Thissen, K.: Lattice klepto - turning post-quantum crypto against itself. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 336–354. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-72565-9_17
- Loidreau, P., Sendrier, N.: Weak keys in the McEliece public-key cryptosystem. IEEE Transactions on Information Theory **47**(3), 1207–1211 (Mar 2001). <https://doi.org/10.1109/18.915687>
- Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the McEliece cryptosystem without random oracles. Des. Codes Cryptography **49**, 289–305 (12 2008). <https://doi.org/10.1007/s10623-008-9175-9>
- Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy, B., Meier, W. (eds.) Fast Software Encryption. Lecture Notes in Computer Science, vol. 3017, pp. 348–358. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25937-4_22
- Simmons, G.J.: The prisoners’ problem and the subliminal channel. In: Chaum, D. (ed.) CRYPTO’83. pp. 51–67. Plenum Press, New York, USA (1983)

12. Simmons, G.J.: The subliminal channel and digital signature. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT’84. LNCS, vol. 209, pp. 364–378. Springer, Heidelberg (Apr 1985). https://doi.org/10.1007/3-540-39757-4_25
13. Yang, Z., Xie, T., Pan, Y.: Lattice klepto revisited. In: Sun, H.M., Shieh, S.P., Gu, G., Ateniese, G. (eds.) ASIACCS 20. pp. 867–873. ACM Press (Oct 2020). <https://doi.org/10.1145/3320269.3384768>
14. Young, A., Yung, M.: The dark side of “black-box” cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO’96. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_8
15. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT’97. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_6
16. Young, A., Yung, M.: Kleptography from standard assumptions and applications. In: Garay, J.A., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 271–290. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_18

A Appendix: A Simpler (but Flawed) SETUP Mechanism

We consider the Vanilla McEliece key generation and describe a simpler attempt at constructing a backdoor. This construction does not even yield a weak SETUP because the backdoor can be efficiently detected by just considering the public keys. The distinguisher may be interesting in its own right and is also described below.

A.1 A Flawed SETUP

A description of the original and our simpler (but flawed) backdoored key generation $\widetilde{\text{KGen}}_{\mathbb{F}}$ can be found in Figure 10.

The matrices S and H are generated exactly as in the non-backdoored scheme. The key difference is that instead of applying a random permutation P , we choose a permutation \tilde{P} that permutes the columns of pk such that pk ’s first row contains the ciphertext $\mathbf{c} \in \mathbb{F}_2^\ell$. This is done by choosing the permutation matrix as the combination of a purely random P and a permutation P' that sends the bits of \mathbf{c} to the desired coordinates.

Notice that $\widetilde{\text{KGen}}_{\mathbb{F}}$ works provided that

1. $\mathbf{c} \in \mathbb{F}_2^\ell$ can be encoded in the first row $\mathbf{v} = \text{Row}_1(SHP)$ of the public key, and
2. P' is efficiently computable.

We briefly sketch why these statements hold. Regarding the first statement, notice that $\mathbf{c} \in \mathbb{F}_2^\ell$ can be encoded in the first row of the public key if the Hamming weight of \mathbf{v} lies in the interval $[\ell, n - \ell]$. A simple Chernoff bound shows that under reasonable assumptions (such as $\ell \leq \frac{1}{4}n$), the probability that this holds is exponentially close to 1.

| |
|--|
| $\text{KGen}_V(1^n)$ |
| 1 : $\delta \leftarrow_{\$} \{0, 1\}^s$ 2 : $r := G(\delta)$ 3 : Generate C with parity check matrix H from r . 4 : Compute random S, P from r . 5 : return $\text{sk} := (C, S, H, P), \text{pk} := (SHP)$ |
| $\widetilde{\text{KGen}}_V^F(1^n, \text{pk}_{\mathcal{A}})$ |
| 1 : $\delta \leftarrow_{\$} \{0, 1\}^s$ 2 : $c \leftarrow_{\$} \text{Enc}_{\text{pk}_{\mathcal{A}}}(\delta) \in \mathbf{F}_2^\ell$ 3 : $r := G(\delta)$ 4 : Generate C with parity check matrix H from r . 5 : Compute random S, P from r . 6 : Find P' with $\text{Row}_1(SHP P') \in \{c\} \times \mathbf{F}_2^{n-\ell}$. 7 : $\tilde{P} := PP'$ 8 : return $\tilde{\text{sk}} := (C, S, H, \tilde{P}), \tilde{\text{pk}} := SH\tilde{P}$ |

Fig. 10. Original and Backdoored Vanilla McEliece Key Generation

Regarding the second statement, we can compute P' in an insertion-sort fashion: Iterating through the first ℓ entries of the first row of SHP from left to right, if an entry differs from the corresponding one in c , we swap this column with the first column to the right with the same entry in the first row.

A.2 The distinguisher

In order for the described backdoored keys to be indistinguishable from non-backdoored ones, it is clearly necessary that the ciphertexts of the adversary's encryption scheme look like random bitstrings. So let us assume that the adversary's scheme provides indistinguishability from random bits under a chosen plaintext attack (see Definition 1). Under this condition, does the described backdoored scheme provide a SETUP mechanism? Perhaps surprisingly, it turns out that it does not even provide a weak SETUP. To see this, for a public key pk sampled from KGen or $\widetilde{\text{KGen}}_V^F$, we consider the random variables

$$X := \text{wt}(v_1 \dots v_\ell), \quad Y := \text{wt}(\mathbf{v}),$$

where $\mathbf{v} = v_1 \dots v_n := \text{Row}_1(\text{pk})$, and we make the following observation:

Lemma 1. *If $(\text{sk}, \text{pk}) \leftarrow_{\$} \widetilde{\text{KGen}}_V^F$, then $X \mid Y = w \sim \text{Binom}(\ell, \frac{1}{2})$.
If $(\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen}_V$, then $X \mid Y = w \sim \text{Hypergeom}(n, w, \ell)$.*

Proof. First suppose $(\text{sk}, \text{pk}) \leftarrow_{\S} \widetilde{\text{KGen}}_{\mathbb{V}}^{\mathbb{F}}$. Then the first ℓ entries of $\text{Row}_1(\text{pk})$ are given by an encryption $c \leftarrow_{\S} \text{Enc}_{\text{pk}_A}(\delta)$ of a random seed δ . Since Enc_{pk_A} provides random ciphertexts, c is uniformly distributed among all ℓ -bit strings (or at least computationally indistinguishable from it). Hence $X = \text{wt}(c)$ is binomially distributed as required, independent of the Hamming weight of $\text{Row}_1(\text{pk})$.

Now suppose $(\text{sk}, \text{pk}) \leftarrow_{\S} \text{KGen}_{\mathbb{V}}$ where $\text{sk} = (C, S, H, P)$. Observe that pk is obtained from SH by randomly permuting its columns. This means that the first ℓ entries of $\text{Row}_1(\text{pk})$ are obtained by randomly sampling without replacement from the entries in the first row of SH . Hence $X \mid \text{wt}(\text{Row}_1(SH)) = w \sim \text{Hypergeom}(n, w, \ell)$. As permuting the columns of SH does not change the Hamming weight of its first row, we have $\text{wt}(\text{Row}_1(\text{pk})) = \text{wt}(\text{Row}_1(SH))$. This implies the claim. \square

Hence the conditional distributions of $X \mid Y = w$ differ noticeably in the backdoored and non-backdoored case. A maximum-likelihood distinguisher can thus be used to distinguish backdoored from non-backdoored keys with non-negligible advantage.

```

 $\mathcal{D}(\text{pk}, \ell)$ 


---


1:  $n :=$  number of columns of  $\text{pk}$ 
2:  $r := \text{Row}_1(\text{pk})$ 
3:  $c := r_1 \dots r_{\ell}$ 
4: if  $p_{\ell, \frac{1}{2}}^{\text{Binom}}(\text{wt}(c)) < p_{n, \text{wt}(r), \ell}^{\text{Hypergeom}}(\text{wt}(c))$  then
5:   return NON-BACKDOORED
6: else
7:   return BACKDOORED
8: fi

```

Fig. 11. Distinguishing backdoored and non-backdoored public keys. $p_{\ell, \frac{1}{2}}^{\text{Binom}}$ and $p_{n, w, \ell}^{\text{Hypergeom}}$ denote the probability mass functions of the binomial respectively hypergeometric distribution.

This observation can be used to construct a distinguisher. Our distinguisher \mathcal{D} described in Figure 11 is inspired by Lemma 1 and requires only the public key and the ciphertext length of the adversary's encryption scheme. It is basically a maximum-likelihood distinguisher that, given a public key pk , considers the Hamming weight of the first ℓ bits of its first row. Depending on whether this ℓ -bit string has a higher probability of occurrence assuming $\text{Binom}(\ell, \frac{1}{2})$ or $\text{Hypergeom}(n, \text{wt}(\text{Row}_1(\text{pk})), \ell)$ as the underlying distribution, the distinguisher outputs that the public key is backdoored or, respectively, non-backdoored.

Lemma 1 implies that the distinguishing advantage of \mathcal{D} is given by the statistical distance³ between the distributions $\text{Hypergeom}(n, \text{wt}(\text{Row}_1(\text{pk})), \ell)$ and $\text{Binom}(\ell, \frac{1}{2})$. Notice that it depends on $\text{wt}(\text{Row}_1(\text{pk}))$. It is minimal for $\text{wt}(\text{Row}_1(\text{pk})) = \frac{n}{2}$, however even in this case it is far from negligible for reasonable n and ℓ occurring for practical McEliece parameter sets. For example, applying the Randomized Niederreiter scheme described in Section 5 to the highest Classic McEliece Category 5 parameter set (see Table 1), even in the favourable case that half the entries in the first row of the public key equal one respectively zero, the distinguishing advantage is about 0.071. It thus clearly does not even provide a weak SETUP because we can distinguish backdoored and non-backdoored keys from just the public keys.

Intuitively speaking, the problem with this attempt at a backdoor construction is the following: In the non-backdoored scheme, the distribution of the first ℓ bits of the first row of pk is in fact dependent on the Hamming weight of the entire row. For example, if there happen to be in total more ones than zeros in the first row of pk or equivalently of the associated SH , then applying a random permutation to the columns of SH also results in a bias towards more ones than zeros in the first ℓ bits. This is in contrast with the backdoored scheme for which the first ℓ bits of the first row of the resulting pk are always uniformly distributed since they are completely determined by the ciphertext \mathbf{c} — which is indistinguishable from a random bitstring by assumption.

³ The *statistical distance* between two discrete distributions with probability mass functions p and q defined over the same set \mathcal{X} is given by $d(p, q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |p(x) - q(x)|$.