




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

Tobias Hemmert<sup>1</sup>, Alexander May<sup>2</sup>, Johannes Mittmann<sup>1</sup>, and Carl  
Richard Theodor Schneider<sup>2</sup>

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

<sup>2</sup> 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  $\delta$  that generates via some pseudo random generator (PRG) the randomness that determines the secret key.

Our backdoor mechanism works by encoding an encryption of  $\delta$  into the public key. Retrieving  $\delta$  then allows to efficiently recover the (backdoored) secret key. Interestingly, McEliece can be used itself to encrypt  $\delta$ , thereby protecting our backdoor mechanism with strong post-quantum security guarantees.

Our backdoor mechanism also works for the current *Classic McEliece* NIST standard proposal, and therefore opens the door for widespread maliciously backdoored implementations.

Fortunately, there is a simple fix to guard (Classic) McEliece against backdoors. While it is not strictly necessary to store  $\delta$  after key generation, we show that  $\delta$  allows identifying maliciously backdoored keys. Thus, our results provide a strong advice to implementers to store  $\delta$  inside the secret key (as the proposal recommends), and use  $\delta$  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 backdoor that allows for easy recovery of the scheme's secret key.

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 [BLN16].

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 most prominent candidates, the McEliece cryptosystem, for which we show how to install backdoors, as well as how to detect them.

*Previous work.* Backdoors were introduced into modern cryptography in the foundational works of Simmons on subliminal channels [Sim83,Sim85]. The notion of backdoors was more formally captured by Young and Yung [YY96,YY97], also denoted kleptography. In this work, we will use their SETUP (Secretly Embedded Trapdoor with Universal Protection) formalism that is an abstraction for transforming a cryptographic scheme into a backdoored scheme.

A SETUP mechanism encodes information into the public key during the key generation process of a public key cryptosystem, allowing  $\mathcal{A}$  to later retrieve the underlying secret key. RSA instantiations of such SETUP mechanisms were given by Crépeau and Slakmon [CS03] who for example encoded half of the bits of the RSA prime  $p$  into the public RSA modulus  $N$ .

For post-quantum secure cryptosystems, very little is known about successful SETUP mechanisms. The work of Kwant, Lange and Thissen [KLT17] 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 [YXP20] however shows that [KLT17] does not fulfill the SETUP notion since the backdoors can be detected efficiently. Moreover, Yang, Xie and Pan [YXP20] introduce SETUP mechanisms for RLWE-based schemes that encode non-quantum secure ECC encryptions of plaintexts into the ciphertexts.

For code-based cryptosystems and especially McEliece, to the best of our knowledge no SETUP backdoor mechanism is known. Loidreau and Sendrier [LS01] propose to use weak Goppa polynomials inside 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.

For preventing backdoors from a theoretical viewpoint, Bellare, Paterson and Rogaway [BPR14] 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  $\delta$ .

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  $u$  by encoding an encryption of  $\delta$  (under  $\mathcal{A}$ 's public key  $\mathbf{pk}_{\mathcal{A}}$ ) into  $u$ 's public key  $\mathbf{pk}_u$  using a different permutation of  $C$ . We show that the resulting backdoored keys are

indistinguishable from ordinary McEliece 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 user  $u$  whether her secret/public key pair has been backdoored. In the terminology of Young and Yung we therefore provide a *strong* SETUP.

However, if user  $u$  knows in addition the secret seed  $\delta$ , then she can identify backdoored keys. The reason is that the randomness for transforming the secret key  $\text{sk}_u$  into the public key  $\text{pk}_u$  usually also comes from the PRG output on  $\delta$ . Thus,  $\delta$  already fully determines the public key from the secret key. This makes it impossible for  $\mathcal{A}$  to embed backdoors. Moreover,  $u$  may rerun the secret/public key generation from the verifiable randomness provided by  $\delta$  to check for the validity of its non-backdoored key pair.

Thus, if the seed  $\delta$  is included into  $u$ 's secret key  $\text{sk}_u$ , then our backdoor mechanism is detectable from  $\text{sk}_u$ . In the terminology of Young and Yung we therefore provide a *weak* SETUP for McEliece when  $\delta$  is part of the secret key.

As a main result, we then show that our SETUP backdoor mechanism easily transfers from our Vanilla McEliece scheme to *Classic McEliece* [MDT<sup>+</sup>20], the 3rd 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.

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

*Implementor's Advice.* Our results show that inclusion of the secret  $\delta$  efficiently protects against strong SETUP backdoor mechanisms, though not against weak SETUPS. Thus, our results strongly suggest including  $\delta$  into the secret key to check for the absence of a SETUP mechanism. We would like to stress that storing  $\delta$  is not necessary for McEliece functionality. The original purpose of  $\delta$  is to provide a small piece of randomness, from which one can efficiently derive the (quite large) McEliece secret/public key pairs. To this end, standards usually recommend to store  $\delta$ . Our work shows another strong benefit of storing  $\delta$ , since  $\delta$  serves as a short proof for the correct, non-backdoored, deterministic derivation of the secret/public key pair.

*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 [YY97], Section 3 provides the

strong SETUP mechanism for Vanilla McEliece (without storing  $\delta$ ), as well as the backdoor identification when  $\delta$  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  $\delta$  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 [MDT<sup>+</sup>20].

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

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

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  $k = n - mt$  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 [YY96,YY97]. A SETUP mechanism transforms a cryptosystem  $\Pi$  into a backdoored cryptosystem  $\Pi'$  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  $\Pi'$  agrees with the specification of inputs to  $\Pi$ .*  
This property ensures the compatibility of  $\Pi$  and  $\Pi'$ .
2.  *$\Pi'$  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  $\Pi'$  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.
4. *The output of algorithms in  $\Pi'$  is compatible with the specification of outputs of algorithms in  $\Pi$ . At the same time, it contains information efficiently derivable by  $\mathcal{A}$  only.*  
The output of  $\Pi'$  needs to be compatible to  $\Pi$  in the sense that e.g. a ciphertext created with an encryption function from  $\Pi'$  must be decryptable by the corresponding decryption function in  $\Pi$ . While maintaining this compatibility, output of  $\Pi'$  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 [YY97].

*Weak SETUP.* The output of  $\Pi$  and  $\Pi'$  are polynomially indistinguishable, except for  $\mathcal{A}$  and the legitimate user  $u$  of the implementation. Thus, in a weak SETUP,  $u$  may identify with the help of her secret key  $\text{sk}_u$  from  $\Pi'$  the existence of a backdoor. All users, except  $u$  and  $\mathcal{A}$ , that only know  $\text{pk}_u$  and  $\text{pk}_{\mathcal{A}}$  cannot identify a backdoor in  $u$ 's key.

*Strong SETUP.* The output of  $\Pi$  and  $\Pi'$  are polynomially indistinguishable, except for  $\mathcal{A}$ . Thus, a user  $u$  cannot recognize any backdoors, even when she 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  $\delta$ , say of 256 bits. Thus a small seed  $\delta$  completely determines  $\text{sk}$  and allows compact storage of the secret key.

```

KGenv(1n)
-----
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

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  $\text{pk}$  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 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  $\delta$  using a public-key encryption scheme  $\Pi_{\mathcal{A}}$  of the adversary's choice with her public key  $\text{pk}_{\mathcal{A}}$  to obtain a ciphertext  $c \leftarrow_{\$} \text{Enc}_{\text{pk}_{\mathcal{A}}}(\delta) \in \mathbf{F}_2^\ell$ . 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  $\pi = (p_1, \dots, p_n)$  with  $p_i = \pi(i)$ . 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  $\pi$  as

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

We use an efficiently computable bijection from Kreher and Stinson [KS99] (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.

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

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

We use  $f_n$  to encode  $\mathbf{c} \leftarrow \text{Enc}_{\text{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^\ell$  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  $\widetilde{\text{KGen}}_{\mathbf{V}}$  is described in Figure 3.

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

**Fig. 3.** Backdoored Vanilla McEliece Key Generation.

The parity check matrix  $H$  and the invertible matrix  $S$  are generated from the random seed  $\delta$  as in the non-backdoored key generation  $\text{KGen}_V$  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  $\mathbf{c} \leftarrow \text{Enc}_{\text{pk}_A}(\delta) \in \mathbf{F}_2^\ell$  to a bit representation of a number  $a \in \{0, 1, \dots, n! - 1\}$ . If we assume that  $\mathbf{c}$  is uniformly distributed in  $\mathbf{F}_2^\ell$ , 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  $P^{(n)}$ .

Notice that  $\widetilde{\text{KGen}}_V$  requires

$$\ell \leq \log_2(n!), \quad (1)$$

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

*Secret Key Recovery.* Notice that our backdoored key generation algorithm  $\widetilde{\text{KGen}}_V$  from Figure 3 generates public keys of the form  $\tilde{\text{pk}} := SH\tilde{P} = SHP'P$ , 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  $\mathbf{c} = \text{Enc}_{\text{pk}_A}(\delta)$ . This enables the adversary  $\mathcal{A}$  to recover  $\delta$  using  $\text{Dec}_{\text{sk}_A}$ . One then basically reruns  $\widetilde{\text{KGen}}_V$  to recover the backdoored secret key  $\tilde{\text{sk}}$ . The details are given in Figure 4.

---

$\text{RECOVER}_V(\text{sk}_A, \tilde{\text{pk}})$

- 1 : Find permutation  $P$  with  $\text{Col}_1(\tilde{\text{pk}}P^{-1}) <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_n(\tilde{\text{pk}}P^{-1})$ .
- 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 :  $\delta := \text{Dec}_{\text{sk}_A}(\mathbf{c})$
- 5 :  $\mathbf{r} := G(\delta)$
- 6 : Generate  $C$  with parity check  $H$  from  $\mathbf{r}$ .
- 7 : Compute random  $S$  from  $\mathbf{r}$ .
- 8 : Find permutation  $P'$  with  $\text{Col}_1(SHP') <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_n(SHP')$ .
- 9 :  $\tilde{P} := P'P$
- 10 : **return**  $\tilde{\text{sk}} = (C, S, H, \tilde{P})$

**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 [YY97] 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  $\widetilde{\text{KGen}}_{\mathcal{V}}(1^n)$  applies  $\text{Enc}_{\text{pk}_{\mathcal{A}}}$ , which we assume to be 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  $\text{RECOVER}_{\mathcal{V}}(\text{sk}_{\mathcal{A}}, \widetilde{\text{pk}})$ .

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  $(\text{pk}, \widetilde{\text{sk}})$ . Moreover, our  $\widetilde{\text{pk}}$  allows to recover the full secret key  $\widetilde{\text{sk}}$  using  $\text{RECOVER}_{\mathcal{V}}(\text{sk}_{\mathcal{A}}, \widetilde{\text{pk}})$ .

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  $(\text{sk}, \text{pk})$  and  $(\widetilde{\text{sk}}, \widetilde{\text{pk}})$  output by their respective key generation algorithms are polynomially indistinguishable for anyone who knows  $\text{pk}_{\mathcal{A}}$  — but not  $\text{sk}_{\mathcal{A}}$ .

Recall that we used the randomness of  $\mathbf{c} \leftarrow_{\$} \text{Enc}_{\text{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^\ell$  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  $\text{pk}_{\mathcal{A}}$ .

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  $\mathcal{S}(\cdot)$  on any input returns a uniformly random  $\mathbf{c} \leftarrow_{\$} \{0, 1\}^\ell$ .  $\Pi$  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 [Rog04] 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  $\text{Enc}_{\text{pk}_A}$  is IND $\mathcal{S}$ -CPA and  $\text{pk}_A$  is publicly known. Then original keys  $(\text{sk}, \text{pk}) \leftarrow \mathcal{K}\text{Gen}(1^n)$  and backdoored keys  $(\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \mathcal{K}\widetilde{\text{Gen}}(1^n, \text{pk}_A)$  are polynomially indistinguishable. Therefore, our algorithms  $\mathcal{K}\widetilde{\text{Gen}}_V(1^n, \text{pk}_A)$  and  $\text{RECOVER}_V(\text{sk}_A, \tilde{\text{pk}})$  define a strong SETUP mechanism for Vanilla McEliece.

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

$$\left| \Pr \left[ D(\tilde{\text{sk}}, \tilde{\text{pk}}, \text{pk}_A) = 1 \mid (\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \mathcal{K}\widetilde{\text{Gen}}_V(1^n, \text{pk}_A) \right] - \Pr \left[ D(\text{sk}, \text{pk}, \text{pk}_A) = 1 \mid (\text{sk}, \text{pk}) \leftarrow \mathcal{K}\text{Gen}_V(1^n) \right] \right|.$$

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

---

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

- 1 :  $(\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \mathcal{K}\widetilde{\text{Gen}}_V(1^n, \mathcal{O}_{\text{IND}\mathcal{S}\text{-CPA}})$ , where we compute  $\mathbf{c} \leftarrow \mathcal{O}_{\text{IND}\mathcal{S}\text{-CPA}}(\delta) \in \mathbf{F}_2^{\ell}$
- 2 : **return**  $b \leftarrow D(\tilde{\text{sk}}, \tilde{\text{pk}}, \text{pk}_A)$

**Fig. 5.** Adversary against IND $\mathcal{S}$ -CPA game constructed from  $D$

In case  $\mathcal{O}_{\text{IND}\mathcal{S}\text{-CPA}} = \text{Enc}_{\text{pk}_A}$ , we perfectly simulate  $\mathcal{K}\widetilde{\text{Gen}}_V(1^n, \text{pk}_A)$ . Hence

$$\Pr \left[ A_D^{\text{Enc}_{\text{pk}_A}(\cdot)}(1^n, \text{pk}_A) = 1 \right] = \Pr \left[ D(\tilde{\text{sk}}, \tilde{\text{pk}}, \text{pk}_A) = 1 \mid (\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \mathcal{K}\widetilde{\text{Gen}}_V(1^n, \text{pk}_A) \right]$$

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

$$\left| \Pr \left[ A_D^{\mathcal{S}(\cdot)}(1^n, \text{pk}_A) = 1 \right] - \Pr \left[ D(\text{sk}, \text{pk}, \text{pk}_A) = 1 \mid (\text{sk}, \text{pk}) \leftarrow \mathcal{K}\text{Gen}_V(1^n) \right] \right| \leq \text{negl}(n).$$

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

### 3.3 From Strong to Weak SETUP

Recall that in the original McEliece key generation, we derive all randomness from a random seed  $\delta$  and hence a key pair  $(\text{sk}, \text{pk})$  is solely determined by  $\delta$ . Thus, inclusion of  $\delta$  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  $\widetilde{\text{KGen}}_{\mathbb{V}}^{\delta}$  the same algorithm as  $\widetilde{\text{KGen}}_{\mathbb{V}}$  with the only exception that the random seed  $\delta$  is also included in  $\widetilde{\text{sk}}$ .

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

*Proof.* Let  $(\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow \widetilde{\text{KGen}}_{\mathbb{V}}^{\delta}(1^n, \text{pk}_{\mathcal{A}})$  with  $\widetilde{\text{sk}} = (C, S, H, \widetilde{P}, \delta)$ . 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  $\widetilde{\text{sk}}$  is backdoored if and only if  $P \neq \widetilde{P}$ .

Thus, we can decide via our secret key  $\widetilde{\text{sk}}$ , whether our scheme is backdoored. Since  $\widetilde{\text{KGen}}_{\mathbb{V}}^{\delta}$  and  $\widetilde{\text{KGen}}_{\mathbb{V}}$  differ only by the format of  $\widetilde{\text{sk}}$ , 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.* The Classic McEliece key generation is outlined in Figure 6. As in Vanilla McEliece one also uses a seed  $\delta$  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. The general case can also be handled in Classic McEliece, but the details are irrelevant for the application of our backdoor SETUP mechanism. The reason for choosing  $S$  as above is that the public key  $\text{pk} = T$  is a matrix in  $\mathbf{F}_2^{(n-k) \times k}$ , thus saving  $n-k$  columns in comparison to Vanilla McEliece for efficiency reasons.

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.

```

KGenc(1n)
-----
1 :  $\delta \leftarrow_{\$} \{0, 1\}^s$ 
2 :  $\mathbf{r} := G(\delta)$ 
3 : Compute from  $\mathbf{r}$  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 : return  $\text{sk} := C, \text{pk} := T$ 

```

**Fig. 6.** Original Classic McEliece Key Generation

*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  $\alpha_i$  only. Thus, a random choice of the  $\alpha_i$  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  $\delta$  in the public key by modifying the order of the  $\alpha_i$ .

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.

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

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

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

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

---

```

1 :  $\delta \leftarrow \mathcal{S}\{0, 1\}^s$ 
2 :  $\mathbf{r} := G(\delta)$ 
3 : Compute from  $\mathbf{r}$  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 :    $\mathbf{c} \leftarrow \mathcal{S} \text{Enc}_{\text{pk}_{\mathcal{A}}}(\delta) \in \mathbf{F}_2^\ell$ 
8 :    $\mathbf{s} \leftarrow \mathcal{S}\{0, 1\}^{\lceil \log_2 k! \rceil - \ell}$ 
9 :    $\mathbf{c}' := \mathbf{c} \|\mathbf{s}$  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  $\tilde{\text{sk}} := \begin{cases} \tilde{C} & \text{for } \widetilde{\text{KGen}}_{\text{C}}(1^n, \text{pk}_{\mathcal{A}}) \\ (\delta, \tilde{C}) & \text{for } \widetilde{\text{KGen}}_{\text{C}}^{\delta}(1^n, \text{pk}_{\mathcal{A}}) \end{cases}, \tilde{\text{pk}} := TP'P$ 

```

**Fig. 7.** Backdoored Classic McEliece Key Generation

$\text{RECOVER}_{\text{C}}(\text{sk}_{\mathcal{A}}, \tilde{\text{pk}})$

---

```

1 : Find permutation  $P$  with  $\text{Col}_1(\tilde{\text{pk}}P^{-1}) <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_k(\tilde{\text{pk}}P^{-1})$ .
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 :  $\delta := \text{Dec}_{\text{sk}_{\mathcal{A}}}(\mathbf{c})$ 
5 :  $\mathbf{r} := G(\delta)$ 
6 : Compute from  $\mathbf{r}$  Goppa code  $C = (g(x), \alpha_1, \dots, \alpha_n)$  with distinct  $\alpha_i$ 
   and parity check matrix  $H$ .
7 : Use Gaussian elimination  $S$  to compute  $SH = [I_{n-k} \| T]$ .
8 : Find permutation  $P'$  with  $\text{Col}_1(TP') <_{\text{lex}} \dots <_{\text{lex}} \text{Col}_k(TP')$ .
9 : Set  $\tilde{P} := \begin{pmatrix} I_{n-k} & \mathbf{0} \\ \mathbf{0} & P'P \end{pmatrix}$ .
10 : Compute  $\tilde{C} := (g(x), (\alpha_1, \dots, \alpha_n) \cdot \tilde{P})$ .
11 : return  $\tilde{\text{sk}} := \tilde{C}$ 

```

**Fig. 8.** Classic McEliece Secret Key Recovery

**Theorem 3.** Assume that the adversary’s public-key encryption scheme  $\text{Enc}_{\text{pk}_A}$  is IND $\mathcal{S}$ -CPA and  $\text{pk}_A$  is publicly known. Then original keys  $(\text{sk}, \text{pk}) \leftarrow \mathcal{K}\text{Gen}_C(1^n)$  and backdoored keys  $(\widetilde{\text{sk}}, \widetilde{\text{pk}}) \leftarrow \mathcal{S}\widetilde{\mathcal{K}}\text{Gen}_C(1^n, \text{pk}_A)$  are polynomially indistinguishable. Therefore, our algorithm  $\widetilde{\mathcal{K}}\text{Gen}_C$  (respectively  $\widetilde{\mathcal{K}}\text{Gen}_C^\delta$ ) in combination with  $\text{RECOVER}_C$  defines a strong SETUP (respectively weak SETUP) mechanism for Classic McEliece, when the PRG-seed  $\delta$  is not part (respectively is part) of a user’s secret key  $\text{sk}_u$ .

## 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  $\text{Enc}$ . Our scheme can be used to backdoor Classic McEliece for all parameter sets proposed in the NIST submission.

*IND $\mathcal{S}$ -CPA McEliece Encryption.* As adversary  $\mathcal{A}$ ’s  $\text{Enc}$  we use the *Randomized Niederreiter Cryptosystem* from [NIKM08]. 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  $\mathbf{e} \cdot \text{pk}^T$ .

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

$\mathcal{K}\text{Gen}_{\text{RN}}(1^n)$	$\text{Enc}_{\text{RN}}(1^n, \text{pk}, \mathbf{m})$
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 : <b>return</b> $\text{sk} := (C, S, H, P)$ , $\text{pk} := SHP$	3 : <b>return</b> $\mathbf{c} := \mathbf{e} \cdot \text{pk}^T$

**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 [NIKM08] shows that under the standard assumptions that public keys  $\text{pk}$  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 [NIKM08] prove an even stronger property, called *admissibility* (see Definition 5 in [NIKM08]). 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  $\delta$  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  $\delta$  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

- BLN16. Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. Dual EC: A Standardized Back Door. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, Lecture Notes in Computer Science, pages 256–281. Springer, Berlin, Heidelberg, 2016.
- BPR14. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, August 2014.
- CS03. Claude Crépeau and Alain Slakmon. Simple backdoors for RSA key generation. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 403–416. Springer, Heidelberg, April 2003.

- KLT17. Robin Kwant, Tanja Lange, and Kimberley Thissen. Lattice klepto - turning post-quantum crypto against itself. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017*, volume 10719 of *LNCS*, pages 336–354. Springer, Heidelberg, August 2017.
- KS99. Donald L. Kreher and Douglas R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*. CRC Press, 1999.
- LS01. P. Loidreau and N. Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1211, March 2001.
- MDT<sup>+</sup>20. Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece: Conservative code-based cryptography, October 2020.
- NIKM08. Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Des. Codes Cryptography*, 49:289–305, 12 2008.
- Rog04. Phillip Rogaway. Nonce-based symmetric encryption. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, pages 348–358, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- Sim83. Gustavus J. Simmons. The prisoners’ problem and the subliminal channel. In David Chaum, editor, *CRYPTO’83*, pages 51–67. Plenum Press, New York, USA, 1983.
- Sim85. Gustavus J. Simmons. The subliminal channel and digital signature. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *EUROCRYPT’84*, volume 209 of *LNCS*, pages 364–378. Springer, Heidelberg, April 1985.
- YXP20. Zhaomin Yang, Tianyuan Xie, and Yanbin Pan. Lattice klepto revisited. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 867–873. ACM Press, October 2020.
- YY96. Adam Young and Moti Yung. The dark side of “black-box” cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *CRYPTO’96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, August 1996.
- YY97. Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *EUROCRYPT’97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.

## 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  $\widetilde{\text{KGen}}_V^F$  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  $\text{pk}$  such that  $\text{pk}$ '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.

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

**Fig. 10.** Original and Backdoored Vanilla McEliece Key Generation

Notice that  $\widetilde{\text{KGen}}_V^F$  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.

Regarding the second statement, we can compute  $P'$  in an insertion-sort fashion: Iterating through the first  $\ell$  entries of the first row of  $SHP$  from left to right, if an entry differs from the corresponding one in  $\mathbf{c}$ , 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  $\mathbf{pk}$  sampled from  $\text{KGen}$  or  $\widetilde{\text{KGen}}^F$ , we consider the random variables

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

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

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

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

Now suppose  $(\text{sk}, \mathbf{pk}) \leftarrow \text{KGen}_V$  where  $\text{sk} = (C, S, H, P)$ . Observe that  $\mathbf{pk}$  is obtained from  $SH$  by randomly permuting its columns. This means that the first  $\ell$  entries of  $\text{Row}_1(\mathbf{pk})$  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  $\text{wt}(\text{Row}_1(\mathbf{pk})) = \text{wt}(\text{Row}_1(SH))$ . 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.

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  $\mathbf{pk}$ , considers the Hamming weight of the first  $\ell$  bits of its first row. Depending on whether this  $\ell$ -bit string has a higher probability of occurrence assuming  $\text{Binom}(\ell, \frac{1}{2})$  or

```

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


---


1:  $n :=$  number of columns of  $\text{pk}$ 
2:  $\mathbf{r} := \text{Row}_1(\text{pk})$ 
3:  $\mathbf{c} := r_1 \dots r_\ell$ 
4: if  $p_{\ell, \frac{1}{2}}^{\text{Binom}}(\text{wt}(\mathbf{c})) < p_{n, \text{wt}(\mathbf{r}), \ell}^{\text{Hypergeom}}(\text{wt}(\mathbf{c}))$  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.

$\text{Hypergeom}(n, \text{wt}(\text{Row}_1(\text{pk})), \ell)$  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 distance<sup>3</sup> 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  $\ell$  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  $\ell$  bits of the first row of  $\text{pk}$  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  $\text{pk}$  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  $\ell$  bits. This is in contrast with the backdoored scheme for which the first  $\ell$  bits of the first row of the resulting  $\text{pk}$  are always uniformly distributed: For they are completely determined by the ciphertext  $\mathbf{c}$  which is indistinguishable from a random bitstring by assumption.

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