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

Alexander May[1] and Carl Richard Theodor Schneider[2]

[1] Ruhr-University Bochum, Germany, `alex.may@rub.de`
[2] Ruhr-University Bochum, Germany, `research@crtified.me`

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 $\boldsymbol{\delta}$ that generates via some pseudo random number generator (PRNG) the randomness that determines the secret key.

Our backdoor mechanism works by encoding the encryption of $\boldsymbol{\delta}$ into the public key. Retrieving $\boldsymbol{\delta}$ then allows to efficiently recover the (backdoored) secret key. Interestingly, McEliece can be used itself to encrypt $\boldsymbol{\delta}$, 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 $\boldsymbol{\delta}$ after key generation, we show that $\boldsymbol{\delta}$ allows identifying maliciously backdoored keys. Thus, our results provide a strong advice to implementers to store $\boldsymbol{\delta}$ inside the secret key (as the proposal recommends), and use $\boldsymbol{\delta}$ to guard against backdoor mechanisms.

1 Introduction

Strong cryptography provides confidentiality to everyone. While this is in general a highly desirable goal, it also might lead to law enforcement issues. Thus, there exist strong interests of law enforcement agencies to circumvent cryptographic mechanism 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 standardize 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 backdoors.

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) mechanism that is an abstraction for transforming a cryptographic scheme into a backdoored scheme.

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

For post-quantum secure cryptosystem, very little is known about successful SETUP mechanisms. The work of Kwant, Lange and Thissen [KLT17] shows 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 efficiently be detected. 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 view point, 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. We first address the usual (Vanilla) Niederreiter version, that has as secret key the parity check matrix of a (Goppa) code C. The randomness for generating C comes from the output of a PRNG applied to some secret seed $\boldsymbol{\delta}$, which is a minor deviation from the common schoolbook description.

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 $\boldsymbol{\delta}$ (under $\mathcal{A}$'s public key $\mathsf{pk}_{\mathcal{A}}$) into u's public key 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, $\mathsf{pk}_{\mathcal{A}}$, 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 $\boldsymbol{\delta}$, then she can identify backdoored keys. The reason is that the randomness for transforming the secret key sk_u into the public key pk_u usually also comes from the PRNG output on $\boldsymbol{\delta}$. Thus, $\boldsymbol{\delta}$ 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 $\boldsymbol{\delta}$ to check for the validity of its non-backdoored key pair.

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

We also show that our SETUP backdoor mechanism transfers from (Vanilla) McEliece to *Classic McEliece* [MDT$^+$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 McEliece does not permutate the entries of C. However, we show that 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 (Classic) McEliece itself for encrypting δ, thereby securing our backdoor even in the presence of quantum computers.

Implementor's Advice. Our results show that inclusion of the secret $\boldsymbol{\delta}$ efficiently protects against strong SETUP backdoor mechanisms, though not against weak SETUPs. Thus, our results strongly suggest including $\boldsymbol{\delta}$ into the secret key to check for the absence of a SETUP mechanism. We would like to stress that storing $\boldsymbol{\delta}$ is not necessary for McEliece functionality. The original purpose of $\boldsymbol{\delta}$ 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 $\boldsymbol{\delta}$. Our work shows another strong benefit of storing $\boldsymbol{\delta}$, since $\boldsymbol{\delta}$ 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 this setting.

Paper Organization. In section 2 we give some 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 $\boldsymbol{\delta}$), as well as the backdoor identification when $\boldsymbol{\delta}$ 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 Classic McEliece to hide the encryption of $\boldsymbol{\delta}$ in a user's public key.

2 Background

2.1 McEliece and Binary Goppa Code

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 $H \in \mathbf{F}_2^{(n-k)\times n}$ that generates C's kernel.

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 Goppa code that is used in the *Classic McEliece* scheme [MDT+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 $\alpha_1, \ldots, \alpha_n$ 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 $2t+1$ 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 an $k = n - mt$ 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 Π into a backdoored cryptosystem Π' for a malicious backdoor holder $\mathcal{A}$ with asymmetric key pair $(\mathsf{sk}_{\mathcal{A}}, \mathsf{pk}_{\mathcal{A}})$. 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 to Π'.

2. Π' remains efficient and uses $\mathsf{Enc}_{\mathsf{pk}_\mathcal{A}}$, and optionally other additional functions.
   Requiring efficiency is obviously needed for practicability of a backdoor. The use of $\mathsf{Enc}_{\mathsf{pk}_\mathcal{A}}$ restricts the attacker to efficient encryption schemes.
3. $\mathsf{Dec}_{\mathsf{sk}_\mathcal{A}}$ is not part of Π.
   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 Π' contains information efficiently derivable for $\mathcal{A}$, but remains compatible with the output of algorithms in Π.
   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 the adversary can derive efficiently.
   This property allows formalization of a backdoor in the sense that there exists an efficient algorithm RECOVER to extract some information of the output of Π'.

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 Π and Π' 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 sk_u from Π' the existence of a backdoor.

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

In the following section we introduce a strong SETUP mechanism for (Vanilla) McEliece, which we later also adapt to Classic McEliece.

3 Backdooring (Vanilla) McEliece

Our SETUP mechanism backdoors the McEliece key generation routine, where we encode information into the public key that allows to extract the corresponding secret key.

In (Vanilla) McEliece's key generation process, 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 $P \in \mathbf{F}^{n\times n}$. I.e., the public key is $\mathsf{pk} = SHP \in \mathbf{F}_2^{(n-k)\times n}$, and the secret key is $\mathsf{sk} = (C, S, H, P)$. It is important to stress that the randomness for constructing C, S, H, P is chosen from the output of an PRNG applied to a short random seed $\boldsymbol{\delta}$, say of 256 bit. Thus, a small seed $\boldsymbol{\delta}$ completely determines sk and allows compact storage of the secret key. While the source of randomness is not covered in schoolbook descriptions of McEliece, we decide to include it at this point as it will naturally lead to the possible backdoor in Classic McEliece.

The invertible matrix S just defines a basis change of H, which 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. McEliece's security is based on pk behaving like a random parity check matrix, for which the *syndrome decoding* problem is hard.

3.1 (Vanilla) McEliece Strong SETUP

Our high-level SETUP idea is that a malicious $\mathcal{A}$ hides the small random seed $\boldsymbol{\delta}$ in the public key pk. To this end, we encrypt our $\boldsymbol{\delta}$ under the adversary's public key $\mathsf{pk}_{\mathcal{A}}$, resulting in an ℓ-bit ciphertext $\boldsymbol{c} \leftarrow\$ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$ for some semantically secure encryption Enc with *random ciphertexts*.

Instead of applying a random permutation P, in our backdoored key generation we choose a permutation $\widetilde{P}$ that permutes pk such that pk's first row starts with the ciphertext $\boldsymbol{c} \in \mathbf{F}_2^{\ell}$. Since usually $\ell \ll n$, there exists with overwhelming probability such a permutation $\widetilde{P}$.

From pk, the adversary decrypts $\boldsymbol{\delta} = \mathsf{Dec}_{\mathsf{sk}_{\mathcal{A}}}(\boldsymbol{c})$, runs the PRNG on $\boldsymbol{\delta}$, and deterministically reconstructs sk. However, by the semantic security of Enc, the first row of pk does not provide useful information to anybody not knowing $\mathsf{sk}_{\mathcal{A}}$. Moreover, we show that the backdoored parity check matrix $\widetilde{\mathsf{pk}}$ is indistinguishable from a pk that is produced with the original Vanilla McEliece scheme.

Let us more formally capture the notion of *random ciphertexts*.

Definition 1. *A public key encryption schemes* Enc *provides* random ciphertexts *if* $\boldsymbol{c} \leftarrow\$ \mathsf{Enc}_{\mathsf{pk}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$ *for some random message* $\boldsymbol{\delta}$ *from the message space is polynomially indistinguishable from a uniformly random bitstring of length* ℓ.

Notice that any public key encryption scheme Enc can be transformed into one that provides random ciphertexts by applying a PRF to its ciphertexts. However, we show in section 5 that McEliece encryption itself already provides random ciphertexts under a mild assumption (see Definition 2).

Backdooring Key Generation. The original and backdoored key generation is described in Figure 1. Notice that we define in the backdoored key generation $\mathsf{KGen}_{\mathsf{V}}$ the permutation matrix as the combination of a purely random P and a permutation P' that sends the bits of $\boldsymbol{c}$ to the desired coordinates. Thus, $\widetilde{P} = PP'$ is chosen randomly among all permutations that sends the bits of $\boldsymbol{c}$ to the correct positions.

Notice that $\widetilde{\mathsf{KGen}_{\mathsf{V}}}$ works provided that

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

Fig. 1. Original and Backdoored Vanilla McEliece Key Generation

```
KGen_V(1^n)
1 : $\boldsymbol{\delta} \leftarrow\!\$ \{0,1\}^s$
2 : $\boldsymbol{r} \coloneqq G(\boldsymbol{\delta})$
3 : Generate $C$ with parity check $H$ from $\boldsymbol{r}$.
4 : Compute random $S, P$ from $\boldsymbol{r}$.
5 : return $\mathsf{sk} \coloneqq (C, S, H, P)$, $\mathsf{pk} \coloneqq (SHP)$
```

```
$\widetilde{\mathsf{KGen}}_{\mathsf{V}}(1^n, \mathsf{pk}_{\mathcal{A}})$
1 : $\boldsymbol{\delta} \leftarrow\!\$ \{0,1\}^s$
2 : $\boldsymbol{c} \leftarrow\!\$ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$ with Enc from Def. 1.
3 : $\boldsymbol{r} \coloneqq G(\boldsymbol{\delta})$
4 : Generate $C$ with parity check $H$ from $\boldsymbol{r}$.
5 : Compute random $S, P$ from $\boldsymbol{r}$.
6 : Find $P'$ with $\mathsf{Row}_1(SHPP') \in \boldsymbol{c} \times \mathbf{F}_2^{n-\ell}$.
7 : $\widetilde{P} \coloneqq PP'$
8 : return $\widetilde{\mathsf{sk}} \coloneqq (C, S, H, \widetilde{P}), \widetilde{\mathsf{pk}} \coloneqq SH\widetilde{P}$
```

We show both statements in the following lemmata.

Lemma 1. *For McEliece parameters, with overwhelming probability $\boldsymbol{c} \in \mathbf{F}_2^{\ell}$ can be encoded in pk_u.*

Proof. The row $\boldsymbol{v} = \mathsf{Row}_1(SHP)$ has n entries, and is randomly distributed by the random choice of S. Let X be a random variable for the Hamming weight $\mathsf{wt}(\boldsymbol{v})$. Notice that we can encode any $\boldsymbol{c} \in \mathbf{F}_2^{\ell}$ as long as $X \in [\ell, n-\ell]$. Since X is binomially distributed with $\mathbf{E}[X] = \frac{n}{2}$, a simple Chernoff bound shows that the Hamming weight X does not lie in the interval $[\frac{1}{4}n, \frac{3}{4}n]$ with exponentially small probability $\Pr[|X - \mathbf{E}[X]| \geq \frac{1}{4}n] \leq 2e^{\frac{n}{12}}$.

If we use McEliece parameters for encrypting $\boldsymbol{c}$, we obtain $\ell \approx n - k \approx \frac{1}{4}n$, which guarantees that we can encode $\boldsymbol{c}$ with overwhelming probability.

See section 5 for more details and numerical instantiations of Lemma 1.

Lemma 2. *P' is efficiently computable.*

Proof. Let us assume by the proof of Lemma 1 that $\boldsymbol{v} = \mathsf{Row}_1(SHP)$ has its Hamming weight in $[\ell, n-\ell]$, and let $\boldsymbol{c} = c_1 \ldots c_{\ell}$. Then we can reshuffle the first entries in an insertion-sort fashion to $\boldsymbol{c}$, as follows.

– Let $i \in \{1, \ldots, n\}$ be minimal such that $c_i \neq v_i$, and let $v_j, j > i$, be minimal with $c_i = v_j$. Then exchange columns i and j.

This process ends after a maximum of n column exchanges with the desired reshuffle result that the first row starts with $\boldsymbol{c}$.

Secret Key Recovery. In Figure 2, we detail the secret key recovery.

Notice that $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$ recovers the backdoored secret key $\widetilde{\mathsf{sk}} \coloneqq (C, S, H, \widetilde{P})$ that includes the backdoored permutation $\widetilde{P}$. It is important to

Fig. 2. Vanilla McEliece Secret Key Recovery

```
$\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$
1 : Parse $\mathsf{Row}_1(\widetilde{\mathsf{pk}}) = \boldsymbol{c} \| \{0,1\}^{n-\ell}$
2 : $\boldsymbol{\delta} := \mathsf{Dec}_{\mathsf{sk}_{\mathcal{A}}}(\boldsymbol{c})$
3 : $\boldsymbol{r} := G(\boldsymbol{\delta})$
4 : Generate $C$ with parity check $H$ from $\boldsymbol{r}$.
5 : Compute random $S, P$ from $\boldsymbol{r}$.
6 : Solve $\widetilde{\mathsf{pk}} = (SHP)P'$ for the unknown $P' \in \mathbf{F}_2^{n \times n}$. Set $\widetilde{P} = PP'$.
7 : return $\widetilde{\mathsf{sk}} = (C, S, H, \widetilde{P})$
```

recover the full secret key, since C, S, and $\widetilde{P}$ with $\widetilde{\mathsf{pk}} = SH\widetilde{P}$ are used in McEliece's decryption process. In Figure 3, we show how messages that are embedded into a vector $\boldsymbol{m}$ of $\mathsf{wt}(\boldsymbol{m}) = t$ are encrypted and decrypted. The correctness of encryption/decryption follows immediately for both backdoor-free keys and our backdoored key pair $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$.

Fig. 3. McEliece encryption/decryption

```
$\mathsf{Enc}_{\mathsf{V}}(\mathsf{pk}, \boldsymbol{m})$
1 : return $\boldsymbol{c} = \mathsf{pk} \cdot \boldsymbol{m} = HSP\boldsymbol{m}$
```

```
$\mathsf{Dec}_{\mathsf{V}}(\mathsf{sk}, \boldsymbol{c})$
1 : Compute $S^{-1}\boldsymbol{c} = HP\boldsymbol{m}$
2 : Use efficient decoding in $C$ to recover $P\boldsymbol{m}$
3 : return $m = P^{-1}(P\boldsymbol{m})$
```

It remains to be shown how we realize the computation of P' in line 6 of algorithm $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$. Notice that we have two matrices $\widetilde{\mathsf{pk}}, SHP \in \mathbf{F}_2^{(n-k)\times n}$ that differ only by a column permutation P'. We need that P' is uniquely determined in order to guarantee correct decryption. It is easy to see that P' is uniquely determined iff SHP has pairwise different columns. Let us assume for a moment that the columns of SHP are indeed pairwise different. Then it is easy to match every column of SHP with the corresponding column in $\widetilde{\mathsf{pk}}$, therefore efficiently recovering the permutation P'.

To apply this method, we need to show that SHP has pairwise different columns. This is equivalent to pairwise different columns in SH. Let $H = (\boldsymbol{h}_1 \| \boldsymbol{h}_2 \| \ldots, \boldsymbol{h}_n) \in \mathbf{F}_2^{(n-k)\times n}$, then $SH = (S\boldsymbol{h}_1 \| S\boldsymbol{h}_2 \| \ldots, S\boldsymbol{h}_n) \in \mathbf{F}_2^{(n-k)\times n}$ for some bijective S. Therefore, pairwise different columns in SH are equivalent to pairwise different columns in H.

It is easy to see that any uniquely decodable linear code C has a parity check matrix with pairwise different columns. Assume for contradiction that $\boldsymbol{h}_i = \boldsymbol{h}_j$. Let $\boldsymbol{e}$ be an error vector with $e_i \neq e_j$. Define $\boldsymbol{e'}$ as $\boldsymbol{e}$ with e_i and e_j exchanged. Then $\boldsymbol{s} = H\boldsymbol{e} = H\boldsymbol{e'}$, and therefore the syndrome $\boldsymbol{s}$ has non-unique decoding.

In the special case of Goppa codes, it is also easy to see that H has pairwise different columns. Recall from subsection 2.1 that the i-th column $\boldsymbol{h}_i$ of H is of the form

$$\boldsymbol{h}_i = \frac{1}{g(\alpha_i)} \left(1, \alpha_i, \alpha_i^2, \ldots \alpha_i^{t-1}\right).$$

Assume that $\boldsymbol{h}_i = \boldsymbol{h}_j$ for some $i \neq j$. By the first coordinate we have $g(a_i) = g(a_j)$, which immediately implies $\alpha_i = \alpha_j$ by the second coordinate. This contradicts the definition of a Goppa code, for which all $\alpha_1, \ldots, \alpha_n$ are pairwise different.

SETUP Mechanism for (Vanilla) McEliece. Let us now check that our backdoor mechanism for McEliece indeed follows the SETUP definition of Yung, Young [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 remains efficient and uses $\mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}$, and optionally other additional functions.
   Our $\mathsf{KGen}_{\mathsf{V}}(1^n)$ simply applies $\mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}$, which we assume to be efficient. Since P' is also efficiently computable, our modification remains efficient.
3. $\mathsf{Dec}_{\mathsf{sk}_{\mathcal{A}}}$ is not part of C.
   We solely use $\mathsf{Dec}_{\mathsf{sk}_{\mathcal{A}}}$ in $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$.
4. The output of algorithms in C' contains information efficiently derivable for the adversary, but remains compatible with the output of algorithms in C.
   Our $\widetilde{\mathsf{pk}}$ allows to recover the full secret key $\widetilde{\mathsf{sk}}$ using $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$. Moreover, the output of our backdoored McEliece scheme is fully compatible with the original McEliece scheme, especially the original decryption function works on the backdoored key pairs $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$.

Theorem 1. *Algorithms $\widetilde{\mathsf{KGen}}_{\mathsf{V}}(1^n, \mathsf{pk}_{\mathcal{A}})$ and $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$ define a* strong *SETUP mechanism for (Vanilla) McEliece, when the PRNG-seed $\boldsymbol{\delta}$ is* not *part of a user's secret key sk_u.*

Proof. We show that the output of McEliece and its backdoored version are polynomially indistinguishable, except for the adversary. First notice that C, S, H are identically distributed in $\mathsf{KGen}_{\mathsf{V}}(1^n)$ and $\widetilde{\mathsf{KGen}}_{\mathsf{V}}(1^n, \mathsf{pk}_{\mathcal{A}})$. Therefore, one can only distinguish via $\widetilde{P}$ or $\widetilde{\mathsf{pk}}$.

Since $\widetilde{P} = P \cdot P'$ for some uniform random permutation P, our permutation $\widetilde{P}$ is also chosen uniformly at random. Thus, P and $\widetilde{P}$ are identically distributed as well.

It remains to show that an one cannot distinguish via pk and $\widetilde{\mathsf{pk}}$, which differ by a permutation that sends the encryption $\boldsymbol{c} \leftarrow\!\$\ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$ for some random $\boldsymbol{\delta}$ to the first ℓ coordinates in the first row. Since Enc provides random ciphertexts (Definition 1), $\boldsymbol{c}$ is indistinguishable from a random length-ℓ bitstring. Thus, pk and $\widetilde{\mathsf{pk}}$ only differ by a random column permutation that is determined by $\boldsymbol{c}$.

Even if the adversaries public key $\mathsf{pk}_{\mathcal{A}}$ is leaked, the semantic security of $\mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}$ guarantees the security of $\boldsymbol{\delta}$, therefore McEliece key pairs $(\mathsf{pk}_u, \mathsf{sk}_u)$ remain indistinguishable from backdoored pairs $(\widetilde{\mathsf{pk}_u}, \widetilde{\mathsf{sk}_u})$.

Therefore, our backdoor for the Vanilla McEliece key generation fulfills the definition of a strong SETUP.

3.2 From Strong to Weak SETUP

Recall that our backdoor mechanism works by encrypting the seed $\boldsymbol{\delta}$ into the backdoored $\widetilde{\mathsf{pk}}$. From $\boldsymbol{\delta}$ we derive all randomness $\boldsymbol{r}$ that was used to construct the original secret key sk. The algorithm that transfers sk into $\widetilde{\mathsf{sk}}$ is deterministic given $\boldsymbol{c} \leftarrow\!\$\ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$.

In other words, an original McEliece key pair $(\mathsf{sk}, \mathsf{pk})$ is solely determined by the seed $\boldsymbol{\delta}$. Thus, inclusion of $\boldsymbol{\delta}$ into the secret key allows for a simple verification check of the validity of a key pair, thereby preventing any strong SETUP mechanism.

Fig. 4. Weak SETUP McEliece Key Generation

```
$\widetilde{\mathsf{KGen}_{\mathsf{V}}^{\boldsymbol{\delta}}}(1^n, \mathsf{pk}_{\mathcal{A}})$
1 : $\boldsymbol{\delta} \leftarrow\!\$\ \{0,1\}^s$
2 : $\boldsymbol{c} \leftarrow\!\$\ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\boldsymbol{\delta}) \in \mathbf{F}_2^{\ell}$ with $\mathsf{Enc}$ from Definition 1
3 : $\boldsymbol{r} := G(\boldsymbol{\delta})$
4 : Generate $C$ with parity check $H$ from $\boldsymbol{r}$
5 : Compute random $S, P$ from $\boldsymbol{r}$
6 : Find $P'$ with $\mathsf{Row}_1(SHPP') \in \boldsymbol{c} \times \mathbf{F}_2^{n-\ell}$
7 : $\widetilde{P} := PP'$
8 : return $\widetilde{\mathsf{sk}} := (\boldsymbol{\delta}, C, S, H, \widetilde{P}), \widetilde{\mathsf{pk}} := SH\widetilde{P}$
```

The verification process that checks for backdoor-freeness now simply reruns McEliece key generation, and checks whether it obtains C, S, H, P, pk. This means that the strong SETUP mechanism from subsection 3.1 now becomes a only a weak SETUP mechanism, because user u can detect it via its secret key. This is summarized in the following theorem.

Theorem 2. *Algorithms* $\widetilde{\mathsf{KGen}}^{\boldsymbol{\delta}}_{\mathsf{V}}(1^n, \mathsf{pk}_{\mathcal{A}})$ *and* $\textsc{Recover}_{\mathsf{V}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$ *define a* weak *SETUP mechanism for (Vanilla) McEliece, when the PRNG-seed* $\boldsymbol{\delta}$ *is part of a user's secret key* sk_u.

Our backdoor utilizes the dependency on only $\boldsymbol{\delta}$ as a randomness source. A possible alternative that comes to mind is choosing C, S, H, P by utilizing a true randomness source. This prevents rerunning the key generation due to the obvious lack of a seed $\boldsymbol{\delta}$, allowing an adversarial implementation to embed other helpful intermediate values (e.g. $g(x)$) without any possibility of detection.

4 How to Backdoor Classic McEliece

In this section, we show that the general ideas from section 3 also transfer to *Classic McEliece*. However some care has to be taken, since (mostly for efficiency reasons) the details of *Classic McEliece* slightly differ.

Changes from Vanilla to Classic McEliece. The Classic McEliece key generation is shown in Figure 5. As in Vanilla McEliece one also uses a seed $\boldsymbol{\delta}$ to compute the randomness $\boldsymbol{r}$ 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 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

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

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 $\mathsf{pk} = T$ is a matrix in $\mathbf{F}_2^{(n-k)\times k}$, thus saving $n-k$ columns in comparison to Vanilla McEliece.

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 also does not allow for backdoor manipulations. However, we show in the following that the definition of the Goppa code C already implicitly introduces a permutation P, for which we apply our backdoor mechanism analogous to section 3.

Idea of Backdoored Key Generation. Recall from subsection 2.1 that the parity check matrix of Goppa is of the form

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

Here, the $\alpha_i \in \mathbf{F}_q$ are random, distinct field elements. Note that the i-th column $\boldsymbol{h}_i$ of H depends on α_i only. Thus, a random choice of the α_i already induces a

$\mathsf{KGen}_{\mathsf{C}}(1^n)$

```
1: $\delta \leftarrow\$ \{0,1\}^s$
2: $\boldsymbol{r} \coloneqq G(\delta)$
3: Compute Goppa code $C = (g(x), \alpha_1, \ldots, \alpha_n)$ with distinct $\alpha_i$ and parity check $H$ from $\boldsymbol{r}$.
4: Use Gaussian elimination $S$ to compute $SH = \left[I_{n-k} \| T\right]$.
5: return $\mathsf{sk} \coloneqq (C, S, H)$, $\mathsf{pk} \coloneqq T$
```

Fig. 5. Original Classic McEliece Key Generation

random permutation of the columns. Moreover, we may use our SETUP mechanism from section 3 that modifies the permutation to encode the encryption of $\boldsymbol{\delta}$ in the public key by modifying the order of the α_i.

In our backdoored key generation algorithm from Figure 6 we permute only the last k columns of the parity check matrix which constitute the (backdoored) public key.

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

```
1: $\delta \leftarrow\$ \{0,1\}^s$
2: $\boldsymbol{c} \leftarrow\$ \mathsf{Enc}_{\mathsf{pk}_{\mathcal{A}}}(\delta) \in \mathbf{F}_2^{\ell}$ with Enc from Definition 1
3: $\boldsymbol{r} \coloneqq G(\delta)$
4: Compute from $\boldsymbol{r}$ Goppa code $C = (g(x), \alpha_1, \ldots, \alpha_n)$ with distinct $\alpha_i$ and parity check $H$.
5: Use Gaussian elimination $S$ to compute $SH = \left[I_{n-k} \| T\right]$.
6: Find permutation $\widetilde{P} \in \mathbf{F}_2^{k \times k}$ with $\mathsf{Row}_1(T\widetilde{P}) \in \boldsymbol{c} \times \mathbf{F}_2^{k-\ell}$ . Set $P = \begin{pmatrix} I_{n-k} & \mathbf{0} \\ \mathbf{0} & \widetilde{P} \end{pmatrix}$.
7: Compute $\widetilde{C} \coloneqq \big(g(x), (\alpha_1, \ldots, \alpha_n) \cdot P\big)$.
8: Compute $\widetilde{H} = H \cdot P$.
9: return $\widetilde{\mathsf{sk}} \coloneqq \begin{cases} (\widetilde{C}, S, \widetilde{H}) & \text{for } \widetilde{\mathsf{KGen}_{\mathsf{C}}}(1^n, \mathsf{pk}_{\mathcal{A}}) \\ (\boldsymbol{\delta}, \widetilde{C}, S, \widetilde{H}) & \text{for } \overset{\delta}{\widetilde{\mathsf{KGen}_{\mathsf{C}}}}(1^n, \mathsf{pk}_{\mathcal{A}}) \end{cases}$, $\widetilde{\mathsf{pk}} \coloneqq T\widetilde{P}$
```

Fig. 6. Backdoored Classic McEliece Key Generation

Classic McEliece Secret Key Recovery. In Figure 7, we detail the secret key recovery.

The correctness of our $\textsc{Recover}_{\mathsf{C}}(\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}})$ follows analogous to the discussion in subsection 3.1.

```
RECOVER_C($\mathsf{sk}_{\mathcal{A}}, \widetilde{\mathsf{pk}}$)
1 : Parse $\mathsf{Row}_1(\widetilde{\mathsf{pk}}) = \boldsymbol{c} \| \{0,1\}^{k-\ell}$
2 : $\boldsymbol{\delta} \coloneqq \mathsf{Dec}_{\mathsf{sk}_{\mathcal{A}}}(\boldsymbol{c})$
3 : $\boldsymbol{r} \coloneqq G(\boldsymbol{\delta})$
4 : Compute from $\boldsymbol{r}$ Goppa code $C = (g(x), \alpha_1, \ldots, \alpha_n)$ with parity check matrix $H$.
5 : Use Gaussian elimination $S$ to compute $SH = \left[I_{n-k} \| T\right]$.
6 : Solve $\widetilde{\mathsf{pk}} \coloneqq T\widetilde{P}$ for the unknown $\widetilde{P}$. Set $P = \begin{pmatrix} I_{n-k} & \mathbf{0} \\ \mathbf{0} & \widetilde{P} \end{pmatrix}$.
7 : Compute $\widetilde{C} \coloneqq \big(g(x), (\alpha_1, \ldots, \alpha_n) \cdot P\big)$.
8 : Compute $\widetilde{H} = H \cdot P$.
9 : return $\widetilde{\mathsf{sk}} \coloneqq (\widetilde{C}, S, \widetilde{H})$
```

Fig. 7. Classic McEliece Secret Key Recovery

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

Theorem 3. *Algorithm* $\widetilde{\mathsf{KGen}_\mathsf{C}}$ *(respectively* $\widetilde{\mathsf{KGen}}_\mathsf{C}^{\boldsymbol{\delta}}$*) in combination with* RECOVER$_\mathsf{C}$ *define a* strong *SETUP (respectively* weak *SETUP) mechanism for Classic McEliece, when the PRNG-seed* $\boldsymbol{\delta}$ *is not part (respectively is part) of a user's secret key* sk_u.

5 How to use Classic McEliece Encryption Against McEliece

We propose to use the Classic McEliece standard as the encryption algorithm Enc of $\mathcal{A}$ in our SETUP mechanism. First, we show that the short ciphertext length of Classic McEliece is beneficial for embedding an encryption of $\boldsymbol{\delta}$. Second, one should use a post-quantum resistant Enc for not weakening our SETUP against quantum adversaries. And third, we show that under a mild assumption on the hardness of decoding Classic McEliece encryption indeed provides random ciphertexts, thereby satisfying Definition 1.

How to encrypt $\boldsymbol{\delta}$*.* Classic McEliece chooses seed length $s = 256$. We embed the random $\boldsymbol{\delta} \in \mathbf{F}_2^{256}$ into a message $\boldsymbol{m} \in \mathbf{F}_2^n$ with weight $\mathsf{wt}(\boldsymbol{m}) = t$ using an injective function. Such an injective embedding exists, and is also efficiently realizable, provided that $2^s < \binom{n}{t}$.

Let us encrypt message $\boldsymbol{m}$ under $\mathsf{pk}_{\mathcal{A}}$, using the algorithm in Figure 3, i.e. $\boldsymbol{c} = \mathsf{pk}_{\mathcal{A}} \cdot \boldsymbol{m} \in \mathbf{F}_2^{n-k}$. In addition, Classic McEliece appends the 256-bit SHA3-hash of $\boldsymbol{c}$ to achieve CCA security. The length of the resulting ciphertext is $\ell = n - k + 256$ bits.

Both SETUPs from section 3 and section 4 require encoding ciphertexts in the first row of the public parity check matrix. While the SETUP from section 3 gives n coordinates for this task, the SETUP from section 4 has to work with only k coordinates.

We provide concrete parameters for Classic McEliece instances in Table 1. For all instances, the ciphertext size $\ell = n - k + 256$ is significantly smaller than k. This even holds if a user u takes the Category-1 instance and $\mathcal{A}$ chooses from Category 5.

Target instance	Category	n	k	ℓ	t	$2^s < \binom{n}{t}$
`kem/mceliece348864`	1	3488	2720	1024	64	✓
`kem/mceliece460896`	3	4608	3360	1504	96	✓
`kem/mceliece6688128`	5	6688	5024	1920	128	✓
`kem/mceliece6960119`	5	6960	5413	1803	119	✓
`kem/mceliece8192128`	5	8192	6528	1920	128	✓

Table 1. Parameters for Classic McEliece and the resulting size for $\boldsymbol{c}$

Ciphertext Randomness of Classic McEliece. A Classic McEliece ciphertext consists of a random weight-t linear combination of the columns of $\mathsf{pk}_{\mathcal{A}}$, called $\boldsymbol{c}$, appended by a SHA-3 hash of $\boldsymbol{c} \in \mathbf{F}_2^{n-k}$. We have to show that $\boldsymbol{c}$ is polynomially indistinguishable from a random point $\boldsymbol{p} \in \mathbf{F}_2^{n-k}$. This is what we call the *Decodable Point Assumption*.

Definition 2 (Decodable Point Assumption). *Let $H' \in \mathbf{F}_2^{(n-k)\times n}$ be the parity check matrix of a code C with distance $2t+1$ for which decoding is hard (e.g. H' is a McEliece* pk*). Then one cannot distinguish ciphertexts $\boldsymbol{c} \in \mathbf{F}_2^{n-k}$ in distance t from C from random points $\boldsymbol{p} \in \mathbf{F}_2^{n-k}$.*

In other words, the *Decodable Point Assumption* states that one cannot distinguish points within the unique decoding radius of C from random points. Since McEliece relies on the hardness of *syndrome decoding*, i.e., the hardness of inverting the function $\boldsymbol{m} \mapsto H'\boldsymbol{m}$, our *Decodable Point Assumption* can be seen as a distinguishing version of the computational *syndrome decoding* problem. From our assumption, we trivially obtain the following corollary.

Corollary 1. *Under the* Decodable Point Assumption*, Classic McEliece encryption provides random ciphertexts.*

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.

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$^+$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.

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.