

# How to backdoor LWE-like cryptosystems

Tobias Hemmert

Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany,  
tobias.hemmert@bsi.bund.de

**Keywords:** Backdoor · Public-Key Cryptography · SETUP · Elliptic-Curve Cryptography · Post-Quantum Cryptography

**Abstract.** We present a rather generic backdoor mechanism that can be applied to many LWE-like public-key cryptosystems. Our construction manipulates the key generation algorithm of such schemes in a way that allows a malicious adversary in possession of secret backdoor information to recover generated secret keys from corresponding public keys. To any user of the cryptosystem however, the output of our backdoored key generation is indistinguishable from output of the legitimate key generation algorithm. Our construction relies on elliptic-curve cryptography and draws on existing work on encoding of elliptic curve points as bit strings.

Our backdoor mechanism can be applied to public-key cryptosystems where the secret key is generated from a secret seed and the public key includes a public seed of the same length. This holds - though not exclusively - for many cryptosystems based on LWE. In particular, we point out that our construction can be applied to backdoor HQC, FrodoKEM, Kyber and Dilithium.

We also suggest a countermeasure that makes our backdoor detectable by users of the cryptosystem. To this end, we modify the key generation such that the public and secret key are pseudorandomly generated from a single seed which is included in the generated secret key. This allows any user of the key generation algorithm to regenerate keys using an independent implementation, making our backdooring attempt detectable.

## 1 Introduction

A central objective of cryptography is to provide confidential communication in the presence of adversarial behaviour. To this end, cryptographic algorithms and protocols are designed to withstand a variety of different attacks, ranging from passive eavesdropping to active attacks. Actual cryptographic systems must also be resistant to side-channel attacks which exploit imperfections in implementations.

In the 90s, Young and Yung [8] initiated the area of kleptography which studies in how far cryptographic schemes are amenable to the insertion of backdoors by a malicious adversary. In the usual scenario for these kinds of attacks,

the adversary is able to manipulate the cryptographic scheme and the legitimate user only has black box access to it. The adversary could for example be a malicious vendor providing cryptographic modules. A kleptographic mechanism manipulates a scheme in such a way that its output is indistinguishable from the output of the non-backdoored scheme, but at the same time the adversary is exclusively able to recover secret information (such as the user’s secret keys or some plaintext) from public output using some secret backdoor information.

Currently, there are many activities concerning the migration to post-quantum algorithms. After three rounds of evaluation, NIST has selected a number of schemes for standardisation and continues to evaluate others in a fourth round. It is interesting to understand in how far these schemes could be backdoored by a malicious implementor. In this paper, we present a rather generic backdoor that can be applied to the NIST PQC candidates HQC, FrodoKEM, Kyber and Dilithium, and suggest a modification of these schemes that provides a countermeasure.

*Idea of the backdoor mechanism.* For our backdoor mechanism, we target key generation algorithms of public-key schemes where the public and secret key are pseudorandomly generated by independently chosen seeds  $\delta_{\text{pub}}, \delta_{\text{priv}} \in \{0, 1\}^\ell$ . This typically holds for many schemes based on LWE for example. The idea of our backdoor is to choose  $\delta_{\text{priv}}$  in a way that it looks random, but it can actually be recovered from  $\delta_{\text{pub}}$  by the adversary.

We briefly sketch the rough idea how to achieve this using elliptic-curve cryptography. Let  $E$  be an elliptic curve over a prime-order field  $\mathbb{F}_q$  such that the group  $E(\mathbb{F}_q)$  has order  $p$  and is generated by some point  $Q$ . Suppose for the moment that we have an injective map  $f: \{0, 1\}^\ell \rightarrow E(\mathbb{F}_q)$  which is ‘almost bijective’, meaning that only a negligible number of elements of  $E(\mathbb{F}_q)$  is not contained in the image of  $f$ . Suppose further  $f$  can be efficiently inverted on its image. Now suppose an adversary chooses her secret and public key as  $\text{sk}_A := x \stackrel{\$}{\leftarrow} \{0, 1, \dots, p-1\}$  and  $\text{pk}_A := xQ$ . The backdoored key generation may then choose  $\delta_{\text{pub}} := f^{-1}(yQ)$  for  $y \stackrel{\$}{\leftarrow} \{0, 1, \dots, p-1\}$ , and  $\delta_{\text{priv}} := f^{-1}(y \cdot \text{pk}_A)$  since  $yQ$  and  $y \cdot \text{pk}_A$  are in the image of  $f$  with overwhelming probability. Note that then  $\text{pk}_A, yQ, y \cdot \text{pk}_A$  form a Diffie-Hellman triple. This allows the adversary to recover  $\delta_{\text{priv}}$  (and thus the secret key) from  $\delta_{\text{pub}}$  using  $\text{sk}_A$ .

Now if the DDH assumption holds in  $E(\mathbb{F}_q)$ , then the seeds thus chosen cannot be distinguished from independently and randomly chosen seeds, so there is no way for the user of the cryptosystem to detect the backdoor given only black box access. Unfortunately, it is currently unknown how to construct an almost bijective embedding  $f$  for curves in which the DDH assumption is conjectured to hold [5]. Such maps are well-known for certain super-singular elliptic curves, but these are not suitable for ECC. However, [3] shows how to construct embeddings into Edwards curves containing roughly half the points in their image. Note that the DDH assumption does not hold in Edwards curves either since they have a subgroup of small order. Nevertheless, we show in this paper how to use these embeddings to obtain a strong kleptographic mechanism.

*Related work.* [7] uses a similar approach to backdooring LWE-like cryptosystems. Using the above notation, their backdoored key generation basically takes  $\delta_{\text{pub}}$  and  $\delta_{\text{priv}}$  to be the x-coordinate of  $yQ$  respectively  $y \cdot \text{pk}_A$  for a randomly chosen  $y \xleftarrow{\$} \{0, 1, \dots, p-1\}$ . However, contrary to the assertion in [7], the generated backdoored keys are clearly not indistinguishable from non-backdoored ones:  $\delta_{\text{pub}}$  is always the x-coordinate of a point on the curve (so this yields only about half of all  $\ell$ -bit strings), whereas  $\delta_{\text{pub}}$  is chosen uniformly at random among all  $\ell$ -bit strings for the non-backdoored scheme.

Young and Yung [9] describe a kleptographic mechanism for RSA key generation also via ECC using a curve-or-twist approach. We point out here that their idea can directly be applied to yield a kleptographic mechanism for the kind of LWE-like schemes that we consider in this paper. However, their scheme requires the adversary to pick  $\ell$  ECC public/secret key pairs on an elliptic curve and  $\ell$  pairs on its non-trivial quadratic twist, and requires  $\ell$  computations of DH triples in the backdoored key generation algorithm, which may slow it down considerably. It is less efficient and may therefore be more easily detectable in practice than the scheme we describe in this paper.

Our construction can be regarded as an application of the work of [5] and [3] on encoding elliptic curve points as bit strings. In fact, [3] mentions the usefulness of their work for kleptography.

*Our contribution.* We describe an efficient kleptographic mechanism for LWE-like public-key schemes. Our backdoor manipulates the key generation such that an adversary is exclusively able to recover generated secret keys from corresponding public keys. At the same time, the generated keys are indistinguishable from non-backdoored keys to everyone except the adversary. In the terminology of Young and Yung, we provide a strong SETUP mechanism for LWE-like public-key schemes. We describe how to modify key generation in a way that prevents application of our strong SETUP. We explain how our backdoor can be applied to HQC, FrodoKEM, Kyber and Dilithium.

*Countermeasure.* We propose a modification of the LWE key generation algorithm that makes our backdoor detectable by the user of the cryptosystem. Instead of generating the secret and public seed independently, the idea is first to pseudorandomly generate them from a single seed and second to include it in the generated secret key. This allows the user of the cryptosystem to regenerate the keys from this seed using an independent implementation, making our backdooring attempt detectable.

*Structure of the paper.* In section 2, we explain some background. In section 3, we define an LWE key generation algorithm that we use to explain our construction. We describe our backdoored key generation and prove that it provides a strong SETUP. Finally, section 4 describes our proposed countermeasure and section 5 points out how our backdoor applies to HQC, FrodoKEM, Kyber and Dilithium.

*Acknowledgements.* I would like to thank Stephan Ehlen, Manfred Lochter and Andreas Wiemers for interesting discussions.

## 2 Background

We give an introduction to the SETUP formalism introduced by Young and Yung [8]. Then we review a result on encoding elliptic curve points as bit strings which is central to our backdoor construction. Finally, we prove a lemma on the distribution of what we call 'altered' DH-triples that we need for the indistinguishability proof of our backdoor construction.

### 2.1 Kleptographic schemes

Kleptography studies how amenable cryptosystems are to manipulation by a malicious adversary such that given only black box access, the manipulated cryptosystem is indistinguishable from a legitimate one, but at the same time it allows the adversary to exclusively recover secret information from its output.

More formally, suppose the adversary has chosen a public/secret key pair  $(pk_A, sk_A)$ . For example, this could be an RSA or DH public/secret key pair. It will be used to embed secret information in the output of a cryptosystem that will only be recoverable by the adversary. Let  $C$  be some cryptosystem. Following [8], a Secretly Embedded Trapdoor with Universal Protection (SETUP) of  $C$  is a modified cryptosystem  $C'$  that satisfies the following properties:

1. The input of  $C'$  agrees with the public specifications of the input of  $C$ .
2.  $C'$  is still efficient and uses the adversary's public key  $pk_A$ .
3. The adversary's secret key  $sk_A$  is only known to the adversary; in particular, it is not contained within  $C'$ .
4. The output of  $C'$  agrees with the public specifications of the output of  $C$ . At the same time, it contains published bits that allow the adversary to derive some secret information (such as the user's secret key) using  $sk_A$ .
5. Even if the specification of  $C'$  and the presence of the SETUP algorithm is fully known, this still does not allow anyone (except the adversary) to recover the secret information derivable from the output of  $C'$ .

In addition, SETUPS must satisfy an indistinguishability property. We say that  $C'$  is a *weak SETUP* if the output of  $C$  and  $C'$  are polynomially indistinguishable to everyone except the adversary and the user of the cryptosystem. We say that  $C'$  is a *strong SETUP* if the output of  $C$  and  $C'$  are polynomially indistinguishable to everyone except the adversary. Thus in a strong SETUP, even the output that the user of the cryptosystem has access to (including generated secret keys for example) must be indistinguishable from the output of the non-backdoored system.

## 2.2 Encoding elliptic curve points as bit strings

The following theorem is proved in [3], based on ideas of [5].

**Theorem 1.** *Suppose:*

- $q$  is prime with  $q \equiv 3 \pmod{4}$ .
- $d = -(2+s^2)^2/(2-s^2)^2 \in \mathbb{F}_q$  for some  $s \in \mathbb{F}_q^*$  such that  $(s^2-2)(s^2+2) \neq 0$ .

*Note that the above implies that  $d$  is not a square in  $F_q$ . We consider the complete Edwards curve  $E: x^2 + y^2 = 1 + dx^2y^2$  over  $\mathbb{F}_q$ . Then there exists an efficiently computable injective map*

$$\iota: \{0, 1, 2, \dots, (q-1)/2\} \rightarrow E(\mathbb{F}_q)$$

*such that for any  $P \in E(\mathbb{F}_q)$ , we can efficiently determine whether  $P \in \text{im}(\iota)$ , and if so, can efficiently compute its preimage.*

*Proof.* This follows from Theorem 4 and section 3.5 of [3]. □

We omit the details of how  $\iota$  and its inverse are computed since we will only need the properties stated in Theorem 1. We give an example of a curve that satisfies the conditions of Theorem 1 and that is also suitable for our backdoor construction.

*Example 1.* Let  $q = 2^{257} - 2^6 - 2^4 - 2^3 - 2^2 - 1$  and  $d = 1088 \in \mathbb{F}_q$  and consider the curve

$$E: x^2 + y^2 = 1 + 1088x^2y^2$$

over  $\mathbb{F}_q$ . Then  $q$  and  $d$  satisfy the conditions of Theorem 1. Furthermore,  $\#E(F_q) = 4p$  for a prime  $p$ . See Appendix A for a Sage script and output to verify these properties.

## 2.3 Distribution of altered DH triples

One issue with the Edwards curves in the previous section is that the DDH assumption trivially does not hold in them since they have a nontrivial subgroup of order 4. However, the DDH assumption will be crucial for our backdoor construction to obtain a strong SETUP. This will be resolved by using the following simple lemma and the fact that - to the best of our knowledge - the DDH assumption does typically hold for sufficiently large prime-order subgroups of elliptic curves.

**Lemma 1.** *Suppose  $G$  is a cyclic group of order  $np$  where  $p$  is prime such that  $p \nmid n$ . Let  $g$  be a generator of  $G$ . Let  $K$  be the (unique) subgroup of order  $n$  and  $H$  be the (unique) subgroup of order  $p$ . We consider two distributions on  $G^3$ :*

- Distribution  $\mathcal{R}_1: (g^a, g^b, g^c)$  where  $a, b, c$  are randomly and independently chosen from  $\mathbb{Z}_{np}$ .

- Distribution  $\mathcal{R}_2$ :  $(g^a, g^b, k \cdot g^{ab})$  where  $a, b$  are randomly and independently chosen from  $\mathbb{Z}_{np}$ , and  $k$  is randomly and independently chosen from  $K$ .

If the DDH assumption holds in  $H$ , then these two distributions are indistinguishable by ppt algorithms.

*Proof.* As  $\gcd(n, p) = 1$ , there is an efficiently computable isomorphism  $G \cong K \times H$ . Thus we may identify  $G$  with  $K \times H$  and we write  $g = (g_K, g_H) \in K \times H$ . By the Chinese Remainder Theorem, choosing an element  $a \xrightarrow{\S} \mathbb{Z}_{np}$  is equivalent to independently choosing  $a_1 \xrightarrow{\S} \mathbb{Z}_n$  and  $a_2 \xrightarrow{\S} \mathbb{Z}_p$ . Thus we may describe  $\mathcal{R}_1$  as:

$$(g_K^{a_1}, g_H^{a_2}), (g_K^{b_1}, g_H^{b_2}), (g_K^{c_1}, g_H^{c_2}) \text{ where } a_1, b_1, c_1 \xrightarrow{\S} \mathbb{Z}_n \text{ and } a_2, b_2, c_2 \xrightarrow{\S} \mathbb{Z}_p \text{ are chosen randomly and independently.}$$

$\mathcal{R}_2$  may be described as:

$$(g_K^{a_1}, g_H^{a_2}), (g_K^{b_1}, g_H^{b_2}), (k \cdot g_K^{a_1 b_1}, g_H^{a_2 b_2}) \text{ where } a_1, b_1 \xrightarrow{\S} \mathbb{Z}_n \text{ and } a_2, b_2 \xrightarrow{\S} \mathbb{Z}_p \text{ and } k \xrightarrow{\S} K \text{ are chosen randomly and independently.}$$

Notice that as  $k$  is chosen randomly and independently,  $k \cdot g_K^{a_1 b_1}$  is a uniformly random element of  $K$  independently chosen from all other components. Thus  $\mathcal{R}_2$  is exactly equal to the distribution

$$(g_K^{a_1}, g_H^{a_2}), (g_K^{b_1}, g_H^{b_2}), (g_K^{c_1}, g_H^{a_2 b_2}) \text{ where } a_1, b_1, c_1 \xrightarrow{\S} \mathbb{Z}_n \text{ and } a_2, b_2 \xrightarrow{\S} \mathbb{Z}_p \text{ are chosen randomly and independently.}$$

It is now clear that this distribution is indistinguishable from  $\mathcal{R}_1$  if the DDH assumption holds in  $H$ .  $\square$

We refer to triples drawn from  $\mathcal{R}_2$  as *altered DH triples*. We also need a slightly more general version of this lemma where we consider  $m$  triples while keeping one component static:

**Lemma 2.** *Suppose  $G$  is a cyclic group of order  $np$  where  $p$  is prime such that  $p \nmid n$ . Let  $g$  be a generator of  $G$ . Let  $K$  be the (unique) subgroup of order  $n$  and  $H$  be the (unique) subgroup of order  $p$ . We consider two distributions on  $G^{2m+1}$ :*

- Distribution  $\mathcal{R}'_1$ :  $(g^a, g^{b_1}, \dots, g^{b_m}, g^{c_1}, \dots, g^{c_m})$  where  $a, b_i, c_i$  are randomly and independently chosen from  $\mathbb{Z}_{np}$ .
- Distribution  $\mathcal{R}'_2$ :  $(g^a, g^{b_1}, \dots, g^{b_m}, k_1 g^{ab_1}, \dots, k_m g^{ab_m})$  where  $a, b_i$  are randomly and independently chosen from  $\mathbb{Z}_{np}$ , and the  $k_i$  are randomly and independently chosen from  $K$ .

If the DDH assumption holds in  $H$ , then these two distributions are indistinguishable by ppt algorithms.

*Proof.* The proof works analogously to the proof of Lemma 1 except that in the last step of the proof, the indistinguishability follows from the DDH assumption on  $H$  and [4], Exercise 10.11.  $\square$

### 3 A strong SETUP for LWE key generation

#### 3.1 Textbook LWE key generation

Algorithm 1 describes the key generation algorithm of a textbook public-key cryptosystem based on LWE. For concreteness, it serves as a model key generation algorithm that we use to explain our backdoor construction. However, it will become clear that our backdoor is not limited to schemes based on LWE. We omit the description of an associated encryption/decryption or signature/verification algorithm since only the key generation is relevant to our backdoor construction. KGen uses the following notation and parameters:

- $n, m$  are positive integers.
- $\ell$  is the seed length in number of bits (e.g. 256).
- $G$  is a pseudorandom function (e.g. SHAKE256).
- $\chi$  is a distribution on  $\mathbb{Z}$  (e.g. a discretized approximate Gaussian distribution).

---

**Algorithm 1** Textbook LWE key generation algorithm KGen

---

- 1:  $\delta_{\text{pub}} \xleftarrow{\$} \{0, 1\}^\ell$
  - 2:  $\delta_{\text{priv}} \xleftarrow{\$} \{0, 1\}^\ell$
  - 3: Pseudorandomly generate  $A \xleftarrow{\$} \mathbb{Z}_m^{n \times n}$  from  $G(\delta_{\text{pub}})$ .
  - 4: Pseudorandomly generate  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_m^n$  and  $e \leftarrow \chi^n$  from  $G(\delta_{\text{priv}})$ .
  - 5:  $\mathbf{b} \leftarrow A\mathbf{s} + e \pmod{m}$
  - 6: **return**  $\text{pk} \leftarrow (\delta_{\text{pub}}, \mathbf{b})$ ,  $\text{sk} \leftarrow \delta_{\text{priv}}$
- 

In many schemes based on LWE, the secret key comprises the vector  $\mathbf{s}$  instead of  $\delta_{\text{priv}}$ . Note that  $\mathbf{s}$  can be computed from  $\delta_{\text{priv}}$ , so the scheme as described in Algorithm 1 is at least as hard to backdoor since we give a legitimate user possibly more information to recognize backdoored keys. The backdoor we describe below applies equally well to the scheme where only  $\mathbf{s}$  is returned as secret key.

#### 3.2 Definition of backdoored key generation

We assume that  $\ell \geq 256$ . Choose a prime  $q < 2^{\ell+1}$  with  $q \equiv 3 \pmod{4}$  such that  $2^{\ell+1} - q$  is small compared to  $2^\ell$ ; more precisely, we demand that  $1 - q/2^{\ell+1}$  be negligible. Choose an Edwards curve  $E$  over  $\mathbb{F}_q$  such that:

- $E$  satisfies the conditions of Theorem 1.
- $\#E(F_q) = 4p$  for some prime  $p$ .
- The DDH assumption holds in the subgroup of order  $p$  (to the best of our knowledge).

Note that the DDH assumption does not hold in the whole group  $E(\mathbb{F}_q)$  since it has a non-trivial subgroup of small order. For  $\ell = 256$ , we gave an example of a suitable prime  $q$  and curve  $E$  in Example 1.

Under the conditions above,  $E(\mathbb{F}_q)$  is a cyclic group. We fix a generator  $Q \in E(\mathbb{F}_q)$  of it and a generator  $R \in E(\mathbb{F}_q)$  of the unique subgroup of order 4. By Theorem 1, there is now an efficiently computable and efficiently invertible injective map

$$\iota_E: \{0, 1, 2, \dots, (q-1)/2\} \rightarrow E(\mathbb{F}_q)$$

Let  $\text{INTTOBITSTR}: \{0, 1, 2, \dots, 2^\ell - 1\} \rightarrow \{0, 1\}^\ell$  be a bijection encoding integers as bit strings. Then  $\text{INTTOBINSTR} \circ \iota_E^{-1}$  defines an encoding of roughly half the points in  $E(\mathbb{F}_q)$  as  $\ell$ -bit strings.

The adversary now randomly picks  $\widetilde{\text{sk}}_{\mathcal{A}} \xleftarrow{\$} \mathbb{Z}_{4p}$  and sets  $\text{pk}_{\mathcal{A}} := \text{sk}_{\mathcal{A}} \cdot Q \in E(\mathbb{F}_q)$ . Our backdoored key generation  $\widetilde{\text{KGen}}$  is described in Algorithm 2. It generates an altered DH triple  $\text{pk}_{\mathcal{A}}, bQ, rR + b \cdot \text{pk}_{\mathcal{A}}$  in the sense of Lemma 1, using rejection sampling to make sure that the latter two elements are contained in  $\text{im}(\iota_E)$ . Then the seeds  $\delta_{\text{pub}}$  and  $\delta_{\text{priv}}$  are computed as the bit strings corresponding to these elements under  $\iota_E$ . From there, the generation of the public and secret keys proceeds as in  $\text{KGen}$ .

---

**Algorithm 2** Backdoored key generation algorithm  $\widetilde{\text{KGen}}(\text{pk}_{\mathcal{A}})$

---

- 1: **repeat**
  - 2:    $b \xleftarrow{\$} \mathbb{Z}_{4p}$
  - 3:    $r \xleftarrow{\$} \{0, 1, 2, 3\}$
  - 4:   **until**  $bQ \in \text{im}(\iota_E)$  and  $rR + b \cdot \text{pk}_{\mathcal{A}} \in \text{im}(\iota_E)$
  - 5:    $\delta_{\text{pub}} \leftarrow \text{INTTOBITSTR}(\iota_E^{-1}(bQ))$
  - 6:    $\delta_{\text{priv}} \leftarrow \text{INTTOBITSTR}(\iota_E^{-1}(rR + b \cdot \text{pk}_{\mathcal{A}}))$
  - 7:   Pseudorandomly generate  $A \xleftarrow{\$} \mathbb{Z}_m^{n \times n}$  from  $G(\delta_{\text{pub}})$ .
  - 8:   Pseudorandomly generate  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_m^n$  and  $e \leftarrow \chi^n$  from  $G(\delta_{\text{priv}})$ .
  - 9:    $\mathbf{b} \leftarrow A\mathbf{s} + e \pmod{m}$
  - 10: **return**  $\widetilde{\text{pk}} \leftarrow (\delta_{\text{pub}}, \mathbf{b}), \widetilde{\text{sk}} \leftarrow \delta_{\text{priv}}$
- 

To show that Algorithm 2 terminates and is efficient, we show that the loop in lines 1-4 terminates after a few repetitions with high probability:

**Lemma 3.** *The loop in lines 1-4 in Algorithm 2 terminates after an expected number of about four repetitions.*

*Proof.* Note that if we randomly and independently sampled  $c \xleftarrow{\$} \mathbb{Z}_{4p}$  and checked whether  $cQ \in \text{im}(\iota_E)$  instead of the second condition  $rR + b \cdot \text{pk}_{\mathcal{A}} \in \text{im}(\iota_E)$ , we would expect to require about four repetitions because about half the elements in  $E(\mathbb{F}_q)$  are contained in  $\text{im}(\iota_E)$ . The same must then hold for the actual loop in lines 1-4 since if it deviated in a significant manner, this would

yield an efficient distinguisher for the distributions  $\mathcal{R}'_1$  and  $\mathcal{R}'_2$  from Lemma 2, contradicting the DDH assumption in the subgroup of  $E(\mathbb{F}_q)$  of order  $p$ .  $\square$

### 3.3 Recovery of secret keys

The adversary's algorithm that, using  $\text{sk}_A$ , recovers secret keys from corresponding public keys generated by  $\widetilde{\text{KGen}}$  is described in Algorithm 3. It is straightforward to check that key recovery actually works.

---

#### Algorithm 3 Secret key recovery algorithm

---

**Input:**  $\text{sk}_A$ ; LWE public key  $\widetilde{\text{pk}} = (\delta_{\text{pub}}, \mathbf{b})$  generated by  $\widetilde{\text{KGen}}(\text{pk}_A)$   
**Output:**  $\text{sk}$  corresponding to  $\widetilde{\text{pk}}$

- 1: Pseudorandomly generate  $A \xleftarrow{\$} \mathbb{Z}_m^{n \times n}$  from  $G(\delta_{\text{pub}})$ .
- 2:  $B := \iota_E(\text{INTTOBITSTR}^{-1}(\delta_{\text{pub}}))$
- 3:  $C := \text{sk}_A \cdot B \in E(\mathbb{F}_q)$
- 4: **for**  $r = 0, 1, 2, 3$  **do**
- 5:     **if**  $rR + C \in \text{im}(\iota_E)$  **then**
- 6:          $\delta_{\text{priv}} := \text{INTTOBITSTR}(\iota_E^{-1}(rR + C))$
- 7:         Pseudorandomly generate  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_m^n$  and  $\mathbf{e} \leftarrow \chi^n$  from  $G(\delta_{\text{priv}})$ .
- 8:         **if**  $\mathbf{b} = A\mathbf{s} + \mathbf{e} \pmod{m}$  **then**
- 9:             **return**  $\widetilde{\text{sk}} := \delta_{\text{priv}}$
- 10:         **end if**
- 11:     **end if**
- 12: **end for**

---

### 3.4 Proof of strong SETUP

Since we only change the key generation algorithm, it is sufficient for us to check that  $\widetilde{\text{KGen}}$  satisfies the properties of a SETUP listed in subsection 2.1.

1. *The input of  $\widetilde{\text{KGen}}$  agrees with the public specifications of the input of  $\text{KGen}$ . The interface of the key generation algorithm remains unchanged.*
2.  *$\widetilde{\text{KGen}}$  is still efficient and uses the adversary's public key  $\text{pk}_A$ . This follows from Lemma 3.*
3. *The adversary's secret key  $\text{sk}_A$  is only known to the adversary; in particular, it is not contained within  $\widetilde{\text{KGen}}$ . This is clear from the description of  $\widetilde{\text{KGen}}$ .*
4. *The output of  $\widetilde{\text{KGen}}$  agrees with the public specifications of the output of  $\text{KGen}$ . At the same time, it contains published bits that allow the adversary to derive some secret information (such as the user's secret key) using  $\text{sk}_A$ . The output interface of  $\text{KGen}$  and  $\widetilde{\text{KGen}}$  are the same. The adversary is able to recover generated secret keys from corresponding public keys as described in Algorithm 3.*

5. Even if the specification of  $\widetilde{\text{KGen}}$  and the presence of the *SETUP* algorithm is fully known, this still does not allow anyone (except the adversary) to recover the secret information from the output of  $\widetilde{\text{KGen}}$ .

This is a consequence of the indistinguishability result we prove in Theorem 2.

We now prove our main result, namely that keys output by  $\text{KGen}$  and  $\widetilde{\text{KGen}}$  are indistinguishable. This is the only missing property to show that our backdoor provides a strong *SETUP* for *LWE* key generation.

**Theorem 2.** *Suppose the DDH assumption holds for the subgroup of order  $p$  in  $E(\mathbb{F}_q)$  and  $\text{pk}_{\mathcal{A}} = aQ$  where  $a \xleftarrow{\$} \mathbb{Z}_{4p}$ . Suppose  $D$  is a ppt distinguisher with binary output and taking  $\text{pk}_{\mathcal{A}}$  and an *LWE* key pair  $(\text{pk}, \text{sk})$  as input. Then its advantage  $\text{Adv}_{\text{KGen}, \widetilde{\text{KGen}}}^{\text{KeyDistinguish}}(D)$  defined by*

$$\begin{aligned} & |\Pr[D(\text{pk}_{\mathcal{A}}, \text{pk}, \text{sk}) = 0 \mid (\text{pk}, \text{sk}) \leftarrow \text{KGen}] \\ & - \Pr[D(\text{pk}_{\mathcal{A}}, \widetilde{\text{pk}}, \widetilde{\text{sk}}) = 0 \mid (\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \widetilde{\text{KGen}}(\text{pk}_{\mathcal{A}})]| \end{aligned}$$

is negligible. Thus  $\widetilde{\text{KGen}}$  is a strong *SETUP* for  $\text{KGen}$ .

*Remark 1.* For simplicity, we state the result for the case where the distinguisher is given access to only one generated key pair. However, using Lemma 2, the proof below is easily adapted to the case where the distinguisher is given access to an arbitrary number of generated key pairs.

*Proof.* Suppose we are given a distinguisher  $D$  as described above. In Algorithm 4, we now use  $D$  to construct a distinguisher  $D'$  that distinguishes between the two distributions  $\mathcal{R}_1$  and  $\mathcal{R}_2$  described in Lemma 1 where we set  $G = E(\mathbb{F}_q)$ ,  $K$  is the subgroup of order 4 and  $H$  is the subgroup of order  $p$ .  $D'$  basically simulates  $\widetilde{\text{KGen}}$ , using  $x$  as the adversary's public key and  $y$  and  $z$  as the points on  $E(\mathbb{F}_q)$  from which the public and secret seeds are derived.

Let  $\Omega$  be the event that  $y \in \text{im}(\iota_E)$  and  $z \in \text{im}(\iota_E)$ . We have that

$$\Pr[\Omega \mid (x, y, z) \leftarrow \mathcal{R}_2] \approx \Pr[\Omega \mid (x, y, z) \leftarrow \mathcal{R}_1] = \left( \frac{\#\text{im}(\iota_E)}{\#E(\mathbb{F}_q)} \right)^2 \approx \frac{1}{4}$$

where in the last step, we use that  $\#\text{im}(\iota_E) = (q+1)/2$  and  $\#E(\mathbb{F}_q) \approx q$  by Hasse's theorem.

Assuming that  $\Omega$  occurs, we have:

- If  $(x, y, z) \leftarrow \mathcal{R}_1$ , then the generation of  $\text{pk}$  and  $\text{sk}$  in  $D'$  almost perfectly simulates  $\text{KGen}$ , with only one difference:  $\delta_{\text{pub}}$  and  $\delta_{\text{priv}}$  are chosen uniformly at random from  $\left\{ \mathbf{b} \in \{0, 1\}^\ell \mid \text{INTTOBITSTR}^{-1}(\mathbf{b}) \leq (q-1)/2 \right\}$  instead of  $\{0, 1\}^\ell$ . Since  $1 - q/2^{\ell+1}$  is negligible by the choice of  $q$ , we may neglect this difference. Hence

$$\begin{aligned} & \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_1, \Omega] \\ & \approx \Pr[D(\text{pk}_{\mathcal{A}}, \text{pk}, \text{sk}) = 0 \mid a \leftarrow \mathbb{Z}_{4p}, \text{pk}_{\mathcal{A}} := aQ, (\text{pk}, \text{sk}) \leftarrow \text{KGen}] \end{aligned}$$

---

**Algorithm 4** Distinguisher  $D'$ 

---

**Input:** Triple  $(x, y, z) \in E(\mathbb{F}_q)^3$

**Output:** A bit  $w \in \{0, 1\}$  indicating a guess whether  $(x, y, z) \leftarrow \mathcal{R}_1$  or  $(x, y, z) \leftarrow \mathcal{R}_2$

```
1: if  $y \notin \text{im}(\iota_E)$  or  $z \notin \text{im}(\iota_E)$  then
2:    $w \stackrel{\$}{\leftarrow} \{0, 1\}$ 
3:   return  $w$ 
4: end if
5:  $\delta_{\text{pub}} \leftarrow \text{INTTOBITSTR}(\iota_E^{-1}(y))$ 
6:  $\delta_{\text{priv}} \leftarrow \text{INTTOBITSTR}(\iota_E^{-1}(z))$ 
7: Pseudorandomly generate  $A \stackrel{\$}{\leftarrow} \mathbb{Z}_m^{n \times n}$  from  $G(\delta_{\text{pub}})$ .
8: Pseudorandomly generate  $\mathbf{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_m^n$  and  $\mathbf{e} \leftarrow \chi^n$  from  $G(\delta_{\text{priv}})$ .
9:  $\mathbf{b} \leftarrow A\mathbf{s} + \mathbf{e} \pmod{m}$ 
10:  $\text{pk} := (\delta_{\text{pub}}, \mathbf{b})$ ,  $\text{sk} := \delta_{\text{priv}}$ 
11:  $w \leftarrow D(x, \text{pk}, \text{sk})$ 
12: return  $w$ 
```

---

– If  $(x, y, z) \leftarrow \mathcal{R}_2$ , then the generation of  $\text{pk}$  and  $\text{sk}$  in  $D'$  perfectly simulates  $\widetilde{\text{KGen}}(x)$ . Hence

$$\begin{aligned} & \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_2, \Omega] \\ &= \Pr[D(\text{pk}_{\mathcal{A}}, \widetilde{\text{pk}}, \widetilde{\text{sk}}) = 0 \mid a \leftarrow \mathbb{Z}_{4p}, \text{pk}_{\mathcal{A}} := aQ, (\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \widetilde{\text{KGen}}(\text{pk}_{\mathcal{A}})] \end{aligned}$$

Assuming that  $\Omega$  does not occur (i.e. its complement  $\overline{\Omega}$  occurs),  $D'$  outputs a uniformly random guess. Hence

$$\begin{aligned} & \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_1, \overline{\Omega}] \\ &= \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_2, \overline{\Omega}] = \frac{1}{2} \end{aligned}$$

Thus we obtain

$$\begin{aligned} & \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_1] \\ &\approx \frac{1}{4} \cdot \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_1, \Omega] \\ &\quad + \frac{3}{4} \cdot \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_1, \overline{\Omega}] \\ &\approx \frac{3}{8} + \frac{1}{4} \cdot \Pr[D(\text{pk}_{\mathcal{A}}, \text{pk}, \text{sk}) = 0 \mid a \leftarrow \mathbb{Z}_{4p}, \text{pk}_{\mathcal{A}} := aQ, (\text{pk}, \text{sk}) \leftarrow \text{KGen}] \end{aligned}$$

Similarly, we obtain

$$\begin{aligned}
& \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_2] \\
& \approx \frac{1}{4} \cdot \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_2, \Omega] \\
& \quad + \frac{3}{4} \cdot \Pr[D'(x, y, z) = 0 \mid (x, y, z) \leftarrow \mathcal{R}_2, \bar{\Omega}] \\
& = \frac{3}{8} + \frac{1}{4} \cdot \Pr[D(\text{pk}_{\mathcal{A}}, \widetilde{\text{pk}}, \widetilde{\text{sk}}) = 0 \mid a \leftarrow \mathbb{Z}_{4p}, \text{pk}_{\mathcal{A}} := aQ, (\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \widetilde{\text{KGen}}(\text{pk}_{\mathcal{A}})]
\end{aligned}$$

Hence we deduce

$$\text{Adv}_{\mathcal{R}_1, \mathcal{R}_2}^{\text{Distinguish}}(D') \approx \frac{1}{4} \cdot \text{Adv}_{\text{KGen}, \widetilde{\text{KGen}}}^{\text{KeyDistinguish}}(D)$$

By the DDH assumption on  $H$  and Lemma 1,  $\text{Adv}_{\mathcal{R}_1, \mathcal{R}_2}^{\text{Distinguish}}(D')$  is negligible and so  $\text{Adv}_{\text{KGen}, \widetilde{\text{KGen}}}^{\text{KeyDistinguish}}(D)$  is also negligible. This proves the claim.  $\square$

## 4 Proposed countermeasure

Algorithm 5 describes a modified LWE key generation algorithm that is resistant to our strong SETUP mechanism described above. It uses a hash function  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell \times \{0, 1\}^\ell$  (which we assume to behave like a random oracle) to derive the public and secret seed from a single seed  $\delta$ . Importantly,  $\delta$  is then included in  $\text{sk}$ , enabling the user to re-run the key generation using a trusted implementation to check it was executed correctly. This prevents application of our strong SETUP mechanism. However, our backdoor construction can still be applied to yield a weak SETUP.

---

### Algorithm 5 Key generation algorithm resistant to strong SETUP

---

- 1:  $\delta \xleftarrow{\$} \{0, 1\}^\ell$
  - 2:  $(\delta_{\text{pub}}, \delta_{\text{priv}}) := H(\delta) \in \{0, 1\}^\ell \times \{0, 1\}^\ell$
  - 3: Pseudorandomly generate  $A \xleftarrow{\$} \mathbb{Z}_m^{n \times n}$  from  $G(\delta_{\text{pub}})$ .
  - 4: Pseudorandomly generate  $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_m^n$  and  $\mathbf{e} \leftarrow \chi^n$  from  $G(\delta_{\text{priv}})$ .
  - 5:  $\mathbf{b} \leftarrow A\mathbf{s} + \mathbf{e} \pmod{m}$
  - 6: **return**  $\text{pk} \leftarrow (\delta_{\text{pub}}, \mathbf{b}), \text{sk} \leftarrow \delta$
- 

## 5 Application to NIST PQC candidates

Note that our backdoor construction does not use any specifics about LWE. It uses the following features of  $\text{KGen}$ :

- The secret key is generated from a single seed  $\delta_{\text{priv}}$ .
- The public key includes a public seed  $\delta_{\text{pub}}$  that has the same length as  $\delta_{\text{priv}}$  but is otherwise chosen independently of  $\delta_{\text{priv}}$ .

Thus it is easily seen that it can be applied to more generic key generation algorithms such as  $\text{KGen}'$  described in Algorithm 6. Here,  $\text{SKGen}$  and  $\text{PKGen}$  are (deterministic) functions that generate the secret and public keys from the private and public seeds. Note that the LWE key generation algorithm  $\text{KGen}$  is a special case of  $\text{KGen}'$ .

---

**Algorithm 6** Generic key generation algorithm  $\text{KGen}'$

---

- 1:  $\delta_{\text{pub}} \xleftarrow{\$} \{0, 1\}^\ell$
  - 2:  $\delta_{\text{priv}} \xleftarrow{\$} \{0, 1\}^\ell$
  - 3:  $\text{sk} := \text{SKGen}(\delta_{\text{priv}})$
  - 4:  $\text{pk}' := \text{PKGen}(\delta_{\text{pub}}, \delta_{\text{priv}})$
  - 5:  $\text{pk} := (\text{pk}', \delta_{\text{pub}})$
  - 6: **return**  $\text{pk}, \text{sk}$
- 

So in the following, we briefly indicate how our backdoor mechanism applies to some of the candidate schemes in the NIST post-quantum standardisation process. Our proposed countermeasure in section 4 can be adapted to each of these schemes to prevent our strong  $\text{SETUP}$  mechanism.

### 5.1 HQC

HQC is a code-based key encapsulation mechanism that is currently a candidate in the fourth round of the NIST PQC standardisation process. According to the third-round specification [6], on a high level the key generation algorithm can be described by  $\text{KGen}'$  with  $\ell = 320$ . The public and private seed are obtained from the output of a SHAKE256-based PRNG, but the seed that serves as the PRNG input is not included in the generated secret key. Thus our backdoor construction yields a strong  $\text{SETUP}$  for HQC key generation.

### 5.2 FrodoKEM

FrodoKEM is a lattice-based key encapsulation mechanism that was an alternate candidate in the third round of the NIST PQC standardisation process. According to the third-round specification [1], the key generation of FrodoKEM conforms to  $\text{KGen}'$ , with two differences:

- In addition to information deterministically derived from  $\delta_{\text{priv}}$ , the generated secret key also contains a seed  $\mathbf{s}$  chosen uniformly at random.  $\mathbf{s}$  is used for implicit rejection. Thus it is only used in case decapsulation fails, and so it does not hinder application of our backdoor and still allows the adversary to recover those parts of the secret key required for decryption.

- In FrodoKEM,  $\delta_{\text{priv}}$  is 256 bits long for all proposed parameter sets.  $\delta_{\text{pub}}$  however is 128 bits long for proposed Level 1 parameters, 192 bits long for proposed Level 3 parameters and 256 bits long for proposed Level 5 parameters. Thus for Level 1 and 3 parameters, the public seed is too short for direct application of our backdoor construction.

Hence our backdoor construction can be directly applied to FrodoKEM Level 5 parameters and yields a strong SETUP.

For Level 1 and Level 3 parameters, since the FrodoKEM key generation does not include the secret seed in the secret key output, one could internally derive the secret key from a seed of length 128 or 192 bits instead of 256 bits. However, this may make our backdoor construction detectable (at least in theory): Firstly, because the elliptic curves used for the backdoor may not offer a sufficient security level any more, and secondly because collisions among generated secret keys are more likely for shorter secret seed length.

### 5.3 Kyber and Dilithium

Kyber is a lattice-based key encapsulation mechanism and Dilithium is a lattice-based signature scheme. Both have been selected for standardisation by NIST. The subsequent comments refer to Kyber, but they also apply to Dilithium. According to its third-round specification [2], Kyber’s key generation conforms to  $\text{KGen}'$  with two main differences:

- Every generated secret key additionally contains the corresponding public key and an additional seed used for rejection sampling (similar to FrodoKEM).
- Kyber’s key generation partially implements our countermeasure described in section 4 by deriving the public and private seed from a single seed  $\mathbf{d}$  of length 256 bits.

The first point does not hinder application of our backdoor mechanism. Regarding the second point, similar to HQC, note that the specification does not require to include  $\mathbf{d}$  in the generated secret key. Thus our backdoor construction can in fact be applied to Kyber by internally choosing  $\delta_{\text{pub}}$  and  $\delta_{\text{priv}}$  independently as described in  $\widetilde{\text{KGen}}$ . As explained in section 4, our backdoor could be detected if  $\mathbf{d}$  were included in the generated secret key.

## References

1. Alkim, E., Bos, J.W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM Learning With Errors Key Encapsulation. NIST PQC Round 3 (04 Jun, 2021)
2. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-Kyber. NIST PQC Round 3 (04 Aug 2021)
3. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: Elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. pp. 967–980. ACM (2013), <https://doi.org/10.1145/2508859.2516734>
4. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020), <https://toc.cryptobook.us/>
5. Fouque, P., Joux, A., Tibouchi, M.: Injective Encodings to Elliptic Curves. In: Boyd, C., Simpson, L. (eds.) Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7959, pp. 203–218. Springer (2013), [https://doi.org/10.1007/978-3-642-39059-3\\_14](https://doi.org/10.1007/978-3-642-39059-3_14)
6. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic (HQC). NIST PQC Round 3 (06 Jun 2021)
7. Yang, Z., Xie, T., Pan, Y.: Lattice Klepto Revisited. In: Sun, H., Shieh, S., Gu, G., Ateniese, G. (eds.) ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020. pp. 867–873. ACM (2020), <https://doi.org/10.1145/3320269.3384768>
8. Young, A.L., Yung, M.: Kleptography: Using Cryptography Against Cryptography. In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer (1997), [https://doi.org/10.1007/3-540-69053-0\\_6](https://doi.org/10.1007/3-540-69053-0_6)
9. Young, A.L., Yung, M.: Kleptography from Standard Assumptions and Applications. In: Garay, J.A., Prisco, R.D. (eds.) Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6280, pp. 271–290. Springer (2010). [https://doi.org/10.1007/978-3-642-15317-4\\_18](https://doi.org/10.1007/978-3-642-15317-4_18), [https://doi.org/10.1007/978-3-642-15317-4\\_18](https://doi.org/10.1007/978-3-642-15317-4_18)

## A Sage code for verification of properties of the curve in Example 1

```
sage: q=2^257-2^6-2^4-2^3-2^2-1
sage: d=1088
sage: s=146973528516760745529631214696625590505311252126814220187
44685276304061464803
sage: is_prime(q)
True
sage: mod(q,4)
3
sage: mod(-(2+s^2)^2/(2-s^2)^2,q)
1088
sage: kronecker(1-d,q)
-1
sage: E_twisted=EllipticCurve(GF(q),[0,4/(1-d)-2,0,1,0])
sage: E=E_twisted.quadratic_twist()
sage: is_prime(int(E.count_points()/4))
True
```