sciendo

Tatra Mountains Mathematical Publications
DOI: 10.2478/tmmp-2019-0013
Tatra Mt. Math. Publ. **73** (2019), 179–193

PREVENTING POTENTIAL BACKDOORS IN BIKE ALGORITHM

Pavol Zajac — Peter Špaček

Slovak University of Technology in Bratislava, Bratislava, SLOVAKIA

ABSTRACT. BIKE suite of algorithms is one of the candidates in NIST call for public-key post-quantum cryptographic algorithms. It is a key-encapsulation mechanism based on QC-MDPC codes with purely ephemeral keys. The security device implementing such an algorithm therefore needs to generate multiple key pairs in its lifetime very efficiently.

In our paper we explore the situation where BigBrother-type adversary can subtly corrupt the vendor(s) of the security devices (e.g., by altering the standard algorithms). In our model, BigBrother cannot preload the keys or synchronize the key generator by a covert channel, but is able to learn the secrets of security devices by observing the public execution of the KEM protocols. BigBrother typically obtains the secret through the usage of (masked) weak keys. However, we can also imagine other covert channels embedded into the ephemeral public keys by some unknown algorithm.

To prevent these classes of attacks, we propose that the standard should explicitly specify a verifiable algorithm to transform the required key randomness into a set of keys.

1. Introduction

Recent NIST call for public-key post-quantum cryptographic algorithms [4] has motivated many researchers to propose new cryptographic schemes that are believed to be secure against quantum adversaries. One of the proposed candidates for Key Encapsulation is a suite of algorithms named BIKE by Aragon et.al. [1]. The BIKE algorithms are based on quasi-cyclic binary moderate-density parity-check codes (QC-MDPC). BIKE relies on generating one-time ephemeral keys to defeat previous attacks on similar QC-MDPC based encryption schemes [7,8]. We summarize details of the BIKE algorithms in Section 2.

2010 Mathematics Subject Classification: 94A60,68P25.
Keywords: code-based cryptography, BIKE, MDPC, backdoor, kleptography.
This work was supported in part by project Slovak National Grant Agency under grant VEGA 1/0159/17. This research was sponsored in part by the NATO Science for Peace and Security Programme under grant G5448.

NIST call represents a good opportunity to prepare an upgrade of existing public key infrastructure to an era of quantum computing. However, the scientific community must be extremely careful in assessing the proposed standards. We should take care to also address security concerns raised after Snowden's revelations. E.g., according to [14], NSA's Sigint Enabling Project's goal was to

> Influence policies, standards and specification for commercial public key technologies.

There are known cases of standardized suspicious random generators, such as Dual EC [3]. Recently, a vulnerability in RSA key generation in Infineon chips was discovered by N e m e c et.al. [11], that could have been an honest mistake, but could also have been an intentional backdoor.

In this paper, we consider a situation when various interests (from all around the globe, with different motivations) can invest large sums of money to corrupt standards and implementations to weaken the security of publicly available cryptographic tools. As usual, we will denote the adversary of this type with a generic designation of BigBrother. BigBrother will either subvert a standard algorithm, or collude with a manufacturer of the cryptographic device, to weaken the implementation of key generation of BIKE algorithm. BigBrother might also require some degree of plausible deniability, such as efficiency constraints.

In this paper we study a kleptographic mechanism similar to SETUP of Y o u n g and Y u n g [15] for BIKE. BigBrother changes BIKE key generation mechanism in such a way that he can use the generated public key to derive the corresponding internal private key. We discuss the attack model in more details in Section 3. Our concrete attack is then based on masked weak keys similar to ones found for general QC-MDPC based cryptosystem by B a r d e t et.al. [2]. We describe the attack details in Section 4.

Unfortunately, the proposed attack is only one of many potential backdoors that can be hidden in a key generation algorithm, not only for BIKE but for other post-quantum candidates. To prevent this class of attacks, we propose that the keys should be generated in a verifiably (pseudo-)random way from a user provided entropy. We discuss this solution in the context of BIKE algorithm in Section 5.

2. BIKE suite of algorithms

In this section we provide an overview of BIKE algorithms based on the original proposal [1]. Our main focus is on the key generation algorithm, all other algorithms are only provided for the sake of completeness.

Table 1. Suggested security parameters for BIKE candidates [1].

	BIKE-1/2			BIKE-3		
Security	r	w	t	r	w	t
Level 1	10163	142	134	11027	134	154
Level 3	19853	206	199	21683	198	226
Level 5	32749	274	264	36131	266	300

BIKE is a suite of three algorithms, BIKE-1 to BIKE-3. These algorithms are based either on McEliece [9] or Niederreiter [12] algorithm, employing quasi-cyclic moderate parity-check matrix codes (QC-MDPC). For a given security level λ, a set of integer parameters (r, w, t) is specified (see Table 1), such that the corresponding problems are hard enough. Each variant of BIKE then works with polynomials in cyclic ring $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$.

Similar to authors of [1], notation h denotes interchangeably both polynomials from $\mathcal{R}$, and a binary vector representing coefficients of h. When we need to state the polynomial coefficients explicitly, we will use notation $h(x)$. Notation $|h|$ denotes Hamming weight of vector/polynomial h.

Each of the three variants of BIKE consists of three procedures:

- *KeyGen*, that generates an ephemeral key pair depending on security level,
- *Encaps*, that creates a cryptogram c and an encapsulated key K using the public key generated by *KeyGen*,
- *Decaps*, that computes the key K from the cryptogram c using the private key generated by *KeyGen*.

In this article we are not interested in the details of individual routines. For the reference, we only quote from [1] the details of the BIKE *KeyGen* routines, and a summary of BIKE algorithms in Table 2.

BIKE *KeyGen* routine works as follows :

Input: Target security level λ,

Output: Private key (h_1, h_2), and public key (f_1, f_2).

(1) Given λ, set parameters r, w according to Table 1.

(2) Generate $h_1, h_2 \overset{\$}{\leftarrow} \mathcal{R}$, with $|h_1| = |h_2| = w/2$.

(3) **BIKE-1** and **BIKE-3** only: Generate $g \overset{\$}{\leftarrow} \mathcal{R}$ of odd weight, so that $|g| \approx r/2$.

(4) Compute public key (depending on the BIKE variant):
- **BIKE-1**: $(f_1, f_2) \leftarrow (gh_2, gh_1)$;
- **BIKE-2**: $(f_1, f_2) \leftarrow (1, h_2h_1^{-1})$;
- **BIKE-3**: $(f_1, f_2) \leftarrow (h_2 + gh_1, g)$.

Table 2. BIKE variants [1].

	BIKE-1	BIKE-2	BIKE-3
SK	(h_1, h_2) with $\lvert h_1\rvert = \lvert h_2\rvert = w/2$		
PK	$(f_1, f_2) \leftarrow (gh_2, gh_1)$	$(f_1, f_2) \leftarrow (1, h_2h_1^{-1})$	$(f_1, f_2) \leftarrow (h_2 + gh_1, g)$
Enc	$(c_1, c_2) \leftarrow (mf_1 + e_1, mf_2 + e_2)$	$c \leftarrow e_1 + e_2f_2$	$(c_1, c_2) \leftarrow (e + e_2f_1, e_1 + e_2f_2)$
	$K \leftarrow KeyDerive(e_1, e_2)$		
Dec	$s \leftarrow c_1h_1 + c_2h_2$ $u \leftarrow 0$	$s \leftarrow ch_1$ $u \leftarrow 0$	$s \leftarrow c_1 + c_2h_1$ $u \leftarrow t/2$
	$(e'_1, e'_2) \leftarrow Decode(s, h_1, h_2, u)$ $K \leftarrow KeyDerive(e'_1, e'_2)$		

Notation $h \xleftarrow{\$} \mathcal{R}$ means that variable h should be sampled uniformly at random from set $\mathcal{R}$. Note that [1] does not specify how this should be accomplished, or how we ensure the additional condition on the weight of the sampled polynomials.

Note that for BIKE-2 a special batch key generation is suggested in [1] (due to the slow modular inversion in public key computation). The only difference is that the batch algorithm should sample a single h_2, and multiple h_1's, which are then stored in a precomputed product of inverses (based on Montgommery's trick). The choice whether batch generation is used or not does not influence the source of h_1's, as they are sampled in the same way from $\mathcal{R}$.

2.1. Analysis of the reference implementation

On the NIST website for proposals, there is a Zip file with a reference implementation of BIKE [13]. In `kem.c` there is a `crypto_kem_keypair` function which handles the key generation. First, the seed is obtained. In case that NIST_RAND is set, seed comes from `randombytes` function by NIST, that uses OpenSSL AES in ECB mode as a random generator. Otherwise, `rand()` is used, and that is not cryptographically secure.

In `sampling.c` we can see the implementation for generation of h_0 and h_1. To achieve the wanted hamming weight, the random positions are picked, until we have the right number of ones. To get the random position in each step, OpenSSL AES is incrementally used as the random generator with the seed mentioned above. There is also a check if the same position was not picked twice.

For public key, NTL operation is used according to Table 2. g is again generated with OpenSSL AES as a random generator.

3. Attack model

In our attack scenario we consider the following actors:

- Alice, Bob, Carol: legitimate users that want to establish communication using BIKE KEM;
- Manufacturer: produces security modules (hardware or software components) that implement BIKE KEM;
- BigBrother: an attacker that has an interest in compromising BIKE KEM used by Alice and Bob, either via standards or by corrupting Manufacturer.

We suppose that Alice wants to communicate with Bob (or Bob can ask Alice for opening a secure communication channel for him). Alice first generates an ephemeral key pair for BIKE KEM, and sends the public key to Bob over insecure network (typically, with a signature attached). Bob encapsulates the session key by BIKE KEM, and sends it to Alice. Alice uses the stored ephemeral private key to decapsulate the session key, and then starts a private communication with Bob (protected by the session key). Similarly, if Alice wants to communicate with Carol or some other user, she generates a different ephemeral key pair. Thus, Alice generates many BIKE key pairs, for each communication session.

Let us consider the role of Manufacturer. Typically, Alice and Bob do not create their own software or hardware implementation of cryptographic algorithms. They either use pre-made software libraries in their computers, or buy special hardware, such as smart cards, that implements cryptographic routines and stores cryptographic keys. In general, we will designate any such tool by a single name "security device". A single manufacturer can create multiple types of security devices, and will typically sell them on the market to a large number of users through multiple resellers.

We suppose that BigBrother can observe all communication in the network. Typically, the focus of BigBrother is not only Alice, but to secretly observe the secrets of every user in the network (or as many as he can efficiently do). We suppose that the symmetric cryptography with strong session key that stays secret is enough to thwart BigBrother's oversight. Thus, BigBrother tries to compromise KEM mechanism: if BigBrother can obtain the session key by cracking the key encapsulation, he can decrypt the successive supposedly private communication.

The power of BigBrother type adversary is typically very strong, especially if he can collude with the manufacturer of the security devices. On the other hand, BigBrother can be restricted by legislative, or by public opinion, and wants to hide his activity.

To have a realistic model we exclude some options for BigBrother:

- The security device cannot communicate with BigBrother directly, BigBrother can only eavesdrop on a communication according to a published standard;
- BigBrother cannot directly link a security device to a user (devices are distributed on the market randomly);
- Due to large complexity and large number of keys, analysis of the traffic should be state-less (i.e., based only on the current run of KEM, not the past history).

Additional requirement of BigBrother might be that the corrupted algorithm is innocuous, even if someone sees the source code or reads the standard. However, in many cases this is not required, as most of the cryptographic devices are made as blackboxes.

Y o u n g and Y u n g in [15] defined a SETUP mechanism as an algorithmic modification C' of the publicly known cryptosystem C that uses attacker's public encryption function to hide a secret parameter of the attacked system in public parameters. The security relies on polynomial indistinguishability of C, and C'. Our proposed attack is similar, but instead of relying on generic public key encryption, we propose that the attacker creates a specific weak key and masks the weakness by his secret parameter. A suitable class of weak keys is explored in the next section. We provide this attack only as a proof of concept, thus we do not prove polynomial indistinguishability. Instead, our main focus is on preventing similar kleptographic attacks in general, which is discussed in Section 5.

4. Weak keys in BIKE

A private key of the original QC-MDPC system [10] can be represented by an n-tuple of (randomly generated) sparse polynomials $(h_1, h_2, \ldots, h_n)$ from $R = \mathbb{F}[x]/(x^r - 1)$. Public key is then derived by computing $f_i = h_i/h_n$. This is also the case in BIKE-2 variant, where $n = 2$.

B a r d e t et.al. in [2] described weak keys for the original QC-MDPC. The principal class of weak keys consists of such polynomial tuples, where rational reconstruction problem is easy. Polynomials h_i can be reconstructed from f_i in cases where each $\deg h_i < d < r$, for $i = 1, 2, \ldots, n-1$, and $\deg h_n \leq r - d$. Furthermore, according to [2], this basic weak class can be further extended by using group actions of $(\mathbb{Z}_r, +)$, and $(\mathbb{Z}_r^*, \times)$ acting on exponents of polynomials h_i.

The results from [2] apply directly to BIKE-2 variant, which is a special case of QC-MDPC with just two polynomials. In this case, private key (h_1, h_2)

is weak, if $\deg h_1 + \deg h_2 < r$. We can find similar classes of weak keys also for BIKE-1 and BIKE-3 variants.

In BIKE-1, the private key (h_1, h_2) is masked by a random private polynomial g to produce public $(f_1, f_2) = (gh_2, gh_1)$. If $\deg g + \deg h_i < r$, the product gh_i in ring $R = \mathbb{F}[x]/(x^r - 1)$ is the same as a direct product in $\mathbb{F}[x]$ (no modulo operation is required). Thus, if both $\deg g + \deg h_1 < r$, and $\deg g + \deg h_1 < r$, we can simply compute

$$g = \gcd(f_1, f_2)/\gcd(h_1, h_2).$$

Here gcd is computed directly in the polynomial ring $\mathbb{F}[x]$. Note that $\gcd(f_1, f_2)$ can be computed from the public key directly. Although $\gcd(h_1, h_2)$ is unknown, it is most likely just a polynomial x^k, where k is the index of the smallest non-zero coefficient of both h_1 or h_2 (as these are random sparse polynomials of potentially large degree). Thus, the potential attacker can recover the weak private key in polynomial time from the corresponding public key.

In BIKE-3, the private key (h_1, h_2) is again masked by a random public polynomial g, but the public key is just $(f_1, f_2) = (h_2 + gh_1, g)$. Now, suppose that $\deg g + \deg h_1 < r$. Again, computing gh_1 in ring R produces the same polynomial as in $\mathbb{F}[x]$. The attacker can recover the weak private key from the public key in polynomial time by simply computing

$$h_2 = f_1 \bmod g, \quad h_1 = (f_1 - h_2)/g.$$

Note that similar observations about group actions of $(\mathbb{Z}_r, +)$, and $(\mathbb{Z}_r^*, \times)$ hold for BIKE-1, and BIKE-3. We will illustrate this with an example. Let us define two weak polynomials $g(x) = \sum_{i=0}^{d-1} g_i x^i$, and $h(x) = \sum_{j=0}^{r-d} h_j x^j$. Clearly, $\deg g + \deg h < r$ for any choice of coefficients of g and h. Let $s \in \mathbb{Z}_r^*$, and let

$$\hat{g}(x) = \sum_{i=0}^{d-1} g_i x^{si} \bmod (x^r - 1) = \sum_{i=0}^{d-1} g_i x^{si \bmod r}.$$

Similarly, we define $\hat{h}(x) = \sum_{j=0}^{d-1} h_j x^{sj} \bmod x^r - 1$. In general, $\deg \hat{g} + \deg \hat{h}$ is not bounded by r. Thus, the lifted product $\hat{f} = \hat{h}\hat{g}$ from R to $\mathbb{F}[x]$ cannot be directly factored to $\hat{h}$ and $\hat{g}$. However, the attacker can compute inverse of s in $\mathbb{Z}_r^*$, and gets polynomial

$$f(x) = \sum_{i=0}^{r-1} \hat{f}_i x^{s^{-1} i \bmod r}.$$

Now, $f = hg$ in both R and $\mathbb{F}[x]$, because group action defined by multiplication of exponents is a ring morphism. That is

$$\varphi(x^i) \cdot \varphi(x^j) = x^{si} \cdot x^{sj} = x^{si+sj} = x^{s(i+j)} = \varphi(x^i \cdot x^j),$$

and similarly,

$$\varphi(x^i + x^j) = \varphi(x^i) + \varphi(x^j).$$

4.1. Using masked weak keys as a backdoor

Weak keys can be easily detected by their joint low degree. Even if we mask these degrees by ring morphisms, it is possible to verify all possible suitable choices of ring morphisms in polynomial time. The attacker can still have some suitable options how to mask the backdoored weak key by some predefined constant (backdoor key). As a proof of work, we have prepared the following algorithm that creates an efficient backdoor for BIKE-1 with enough entropy in backdoored keys to avoid detection.

The attacker generates a random polynomial $p \in R$. He stores this value on the compromised security device (or within the obfuscated key generation code). Polynomial p acts as the attacker's backdoor key. During the key generation, sparse polynomials h are generated according to formula

$$\hat{h}(x) = \sum_{j=0}^{r-d} h_j x^{sj+a} \bmod (x^r - 1),$$

where d corresponds to the chosen security level. Parameters s, a define ring morphism for additional masking. These parameters are either randomly generated (more work for the attacker, but better masking), or stored constants, or can be omitted entirely (using $s = 1, a = 0$).

The compromised algorithm then generates truly random polynomial q of degree below d, and computes:

$$\hat{q}(x) = \sum_{i=0}^{d-1} q_i x^{si} \bmod (x^r - 1).$$

We do not need offset a, as $\hat{q}$ is hidden. Multiplication by s in the exponent is however necessary.

Finally, the key generation algorithm compromised by the attacker computes $\hat{g} = p\hat{q}$, and presents $\hat{g}, \hat{h}_1, \hat{h}_2$ as the private part of the ephemeral key. Public key is computed normally as $(F_1, F_2) = (\hat{g}\hat{h}_2, \hat{g}\hat{h}_1)$.

Once the BigBrother captures compromised ephemeral public key (F_1, F_2) used in BIKE-1 KEM, he computes unmasked public key

$$(\hat{f}_1, \hat{f}_2) = (p^{-1} f_1, p^{-1} f_2).$$

BigBrother knows that

$$\hat{f}(x) = \hat{q}(x)h(x) = \left(\sum_{i=0}^{d-1} q_i x^{si}\right) \cdot \left(\sum_{j=0}^{r-d} h_j x^{sj+a}\right) \bmod (x^r - 1).$$

Coefficients of $\hat{f}$ have been computed from terms in the form

$$q_i x^{si} h_j x^{sj+a} = q_i h_j x^{s(i+j)+a \bmod r}.$$

BigBrother either knows (s, a), or tests all possible pairs, and computes:

$$f(x) = \sum_{i=0}^{r} \hat{f}_i x^{s^{-1}i - s^{-1}a \bmod r}.$$

Coefficients of f are now simply those obtained from terms $q_i h_j x^{(i+j)}$. Furthermore, $i + j < r$, thus polynomials f are simple products of q and corresponding h in $\mathbb{F}[x]$.

BigBrother now computes $\gcd(f_1, f_2)$, and gets $g'(x) = x^k g(x)$. He can then compute

$$h'_1 = f_2/g', \quad \text{and} \quad h'_2 = f_1/g'.$$

Thus, up to a rotation (x^k term), he recovered all parts necessary to reconstruct the private key of the cipher.

While this backdoor provides enough entropy (in selection of q) to avoid black-box detection, it is not as strong as SETUP mechanism from [15]. If polynomial p is reverse engineered or leaked, anyone can unmask the public key (F_1, F_2) and check, whether the key is weak or not (and potentially break the encryption). It is an open question, how can we embed asymmetric backdoor in BIKE-1, or how can we extend the weak key masking to other BIKE variants. However, our goal is not to provide backdoor algorithms (we believe that BigBrother has enough resources to do this better than academia), but to try to respecify the key generation algorithm in a way that can prevent backdoored implementations in general.

5. Generic backdoor prevention

In Section 4, we have presented a specific type of backdoor based on generating and masking weak keys. This specific type of backdoor represents only one potential way in which the attacker can leak sufficient information about the private key into the public key. In general, we can suspect that the attacker will try to leak some critical information about private key into a public key with a hidden algorithm.

(Public) Keys in quantum-resistant primitives are typically much longer than the key required for symmetric cipher of a corresponding security level. For example, in BIKE-1 with Level-1 security, corresponding to AES-128, the public key has $n = 20326$ bits. To construct the key pair directly from a true randomness, we need to collect at least

$$q = r + 2 \left\lceil \log_2 \binom{r}{w/2} \right\rceil \quad \text{bits of entropy}$$

(r bits are used to construct g, and the rest to construct h_1, and h_2, respectively).

In QC-MDPC case, w is very small related to r, thus $n > r \gg q - r$. In the case of Level-1 security this means 11377 bits, most of which are required to construct g, and we only need 1214 bits to construct both h_1, and h_2. From an information theoretic point of view, the public key represents a large enough channel to transfer extra information that will allow to identify private keys (e.g., by finding a suitable algorithm that will choose a special non-random g, that still retains enough entropy corresponding to a given security level).

Generating a lot of truly random bits is difficult in most of the security devices. On the other hand, for Level-1 security we only need $\lambda = 128$ true random bits, and try to derive another required key bits by using an approved PRNG, or some custom algorithm that can be faster and better adapted to the special structure required in the private key. The custom algorithm approach however can help the attacker to construct and hide a possible exploit more easily. A recent example is a vulnerability in RSA key generation in Infineon chips that was discovered by N e m e c et.al. [11]. In this case, RSA key is generated from a small seed by a custom algorithm that can find required secret primes faster than a standard generate and verify approach. Unfortunately, the custom algorithm leaks too much information about the secret keys, and they can be reconstructed from the RSA's public key. It is not possible to decide whether the flawed key generation was a security design mistake, or a covert channel artificially designed to compromise the users security. We thus propose that a verifiable key generation from a provided random string is specified directly in the standard (for all post-quantum standards where applicable).

5.1. Verifiable key generation

Under the term verifiable key generation we understand a deterministic algorithm KD that derives key material for public key system from a fixed truly random bit-string for which the following security definitions hold:

(1) Verifiability. Let k_A be a secret parameter known only to legitimate user (audit key, of length λ). Given K, k, and k_A, a legitimate user can verify that $K = KD(k)$.

(2) Pseudo-randomness. Let k_0, k_1 be two bitstrings of length λ chosen by the attacker. Challenger computes $K = KD(k_b)$, where b is a random bit, and reveals K to the attacker. Attacker wins if he can guess b. Verifiable key generation is secure if advantage of the attacker in this security game is negligible (with respect to λ).

The first property says that a legitimate user owns a specific (audit) key, that allows him to verify whether the key K was correctly derived from a provided pre-key k. On the other hand, pseudo-randomness means that the attacker cannot prepare a pre-key k for some desired key K. It is easy to see, that if he could do this, he would trivially win the proposed security game. Moreover, the attacker cannot predict any bit of K given a choice of k. If he could do this, he would be able to win the game (the prediction algorithm becomes a distinguisher in the security game).

In practice, function KD can be instantiated by using a standardized XOF function SHAKE of appropriate security level. XOF (an extendable-output function) is a function on bit strings in which the output can be extended to any desired length [5]. The proposed key derivation for BIKE-1 works as follows:

(1) Alice chooses secret audit key k_A (of length λ) in the security device initialisation phase (e.g., card personification).

(2) When generating a key, the security device generates a random pre-key k (of length λ).

(3) $q = \varphi_1\Big(SHAKE\big(k|k_A|c_1, r\big)\Big)$.

(4) $h_1 = \varphi_2\Big(SHAKE\big(k|k_A|c_2, \lceil\log_2\binom{r}{w/2}\rceil\big)\Big)$.

(5) $h_2 = \varphi_2\Big(SHAKE\big(k|k_A|c_3, \lceil\log_2\binom{r}{w/2}\rceil\big)\Big)$.

(6) Secret key is a tuple: (q, h_1, h_2). Pre-key k is presented to Alice along with the secret key, or stored in a separate audit compartment (write-once) of the security device.

Algorithm φ_1 just ensures that q has an odd Hamming weight (in a deterministic way, e.g., by flipping the last bit if the Hamming weight is even). Algorithm φ_2 is a bijective transformation from bitstring of length $\lceil\log_2\binom{r}{w/2}\rceil$ to a bit string of length r and weight $w/2$. It is also possible to use faster deterministic algorithm to construct a constant-weight bit-string from a given source of randomness (see, e.g. [6]). In that case we need to increase the number of bits generated by SHAKE, accordingly.

Constants c_1, c_2, c_3 are chosen to provide domain separation between q, h_1, h_2 (so we can consider them as independently generated). To strengthen the system, we can generate three separate pre-keys instead. In practice, we can just derive all len bits required to generate q, h_1, h_2 with a simple call to $SHAKE(k|k_A, len)$, but we must then ensure proper parsing of the resulting bitstring. In case where SHAKE algorithm is not implemented on the security device, it can be replaced by multiple calls to a standard hash function with additional counter input.

We can see that on request, Alice can verify in an independent device (with an independent software), that (q, h_1, h_2) are indeed derived from k. The attacker cannot provide another k' with this property, as this would mean that he can break one-wayness of the SHAKE algorithm.

5.2. Efficiency of the verifiable key generation

We implemented verifiable key generation into BIKE reference implementation using C language. Here we provide some hints and details of our implementation.

Details of $\varphi_2(x)$:

```
(1) Create bitstring $h = (11\dots100\dots0)$, with $|h| = w/2$.
(2) For $i = 1$ to $w/2$:
    (a) $P = ExtractBits(x, \lceil\log_2 r\rceil)$,
    (b) $p = Int(P) \bmod r$,
    (c) $Swap(h, i, p)$.
```

ExtractBits function extracts a specified number of bits from a bitstring x. Function *Int* converts a bitstring to an integer. $Swap(x, i, j)$ exchanges bits in x on positions i and j.

We used OpenSSL EVP shake256 as an expandable output function (XOF). The experiments were done under Oracle VM Virtualbox platform, with 64 Bit OS Ubuntu 18.04.1 LTS, 1.9 GiB of memory and Intel Core i7-3630QM CPU @ 2.40 GHz x 2.

Table 3. Performance (in million of cycles) of BIKE with security level of 128 bits.

	Variable key generation			Original key generation		
	KeyGen	Enc	Dec	KeyGen	Enc	Dec
BIKE1	1.32	1.14	28.52	0.9	0.96	27.61
BIKE2	14.14	0.51	26.97	14.06	0.55	27.59
BIKE3	0.8	1.26	32.53	0.76	1.29	30.54

Table 4. Performance (in million of cycles) of BIKE with security level of 96 bits.

	Variable key generation			Original key generation		
	KeyGen	Enc	Dec	KeyGen	Enc	Dec
BIKE1	0.57	0.60	12.37	0.57	0.68	12.25
BIKE2	7.39	0.33	13.48	7.69	0.35	12.13
BIKE3	0.44	0.68	12.24	0.4	0.78	13.75

Table 5. Performance (in million of cycles) of BIKE with security level of 64 bits.

	Variable key generation			Original key generation		
	KeyGen	Enc	Dec	KeyGen	Enc	Dec
BIKE1	0.25	0.27	4.62	0.23	0.26	4.39
BIKE2	4.47	0.15	4.48	4.63	0.15	4.66
BIKE3	0.19	0.27	5.34	0.18	0.28	5.36

As we can see in tables 3, 4 and 5, results are very similar with both approaches for key generation.

6. Conclusions and open questions

In this paper we have shown that the attacker with non-interactive access to key generation algorithm of the BIKE algorithm can create and hide weak keys. The attacker can then compute each ephemeral key generated by the compromised security device, while the generated keys retain enough entropy to avoid detection. Furthermore, if the attacker's backdoor mask p remains secret, the ephemeral keys should be secure against third parties. It is an open question, whether it is possible to construct a similar BIKE backdoor with asymetric hidden backdoor key (similar to Young's SETUP mechanism for RSA) that only the attacker can exploit even if the secret parameter in the compromised security device is found out.

We have not studied another post-quantum candidates, but we believe there will be many other weaknesses against kleptographic attacks (i.e., backdoored implementations, typically in the key generation phase). These attacks can target not only keys, but also some nonces or prescribed random values. For example, in McEliece cryptosystem, each encryption requires a random error vector to protect secrets. This error vector also requires significantly more entropy to generate than necessary (in respect to security level). If the attacker can control how the error vector is generated for each encryption, he might be able to create instances that are easy to solve (instead of a general hard version of the problem with a random choice of the error vector). This problem can be solved on a protocol level by taking control of the error vector from the sender, as can be seen in our proposal [16].

Similarly, to avoid backdoors in key generation, we should standardize a pseudorandom key generation that computes the actual key using a one-way pseudorandom algorithm seeded by a truly random bit string of the minimal possible size prescribed by the security level. Thus the security device user can either provide its own seed from the independent device, or at least verify that the attacker could not intentionally create a key from some class of weak keys (due to one-way property of the key derivation algorithm). If the key derivation is fast, this can even save speed in the key generation phase (because entropy extraction from the physical sources of randomness is typically a slow process). In this case, a security evaluation should extend to the whole algorithm including the key derivation from a random bitstring of a fixed size.

REFERENCES

[1] ARAGON, N.—BARRETO, P.—BETTAIEB, S.—BIDOUX, L.—BLAZY, O. DENEUVILLE, J.-C.—GABORIT, P.—GUERON, S.—GUNEYSU, T.—MELCHOR, C. A. et al.: *BIKE: Bit Flipping Key Encapsulation*, (2017).
`https://hal.archives-ouvertes.fr/hal-01671903/document`

[2] BARDET, M.—DRAGOI, V.—LUQUE, J.-G.—OTMANI, A.: *MDPC public key encryption scheme.* In: 8th International Conference on Cryptology in Africa, Fes, Moroco April 13–15, 2016. *Progress in Cryptology—AFRICACRYPT 2016* (D. Pointcheval, A. Nitaj, T. Rachidi). *Lecture Notes in Comput. Sci. Vol. 9646,* Springer-Verlag, Berlin, 2016 pp. 346–367.

[3] BERNSTEIN, D. J.—LANGE, T .—NIEDERHAGEN, R.: *Dual EC: A standardized back door.* In: *The New Codebreakers* (Peter Y. A. Ryan, David Naccache, Jean-Jacques Quisquater, eds.). Lecture Notes in Comput. Sci. Vol. 9100, Springer-Verlag, Berlin, 2016. pp. 256–281.

[4] CHEN, L.—CHEN, L.—JORDAN, S.—LIU, Y.-K.—MOODY, D.—PERALTA, R.–PERLNER, R.—SMITH-TONE, D.: *Report on Post-Quantum Cryptography.* National Institute of Standards and Technology (NIST), US Department of Commerce, USA, 2016.
`https://doi.org/10.6028/NIST.IR.8105`

[5] DWORKIN, M. J.: *SHA-3 Standard: Permutation-based Hash and Extendable-output Functions.* Federal Inf. Process. Stds. (NIST FIPS) - Technical report no. 202, 2015. https://doi.org/10.6028/NIST.FIPS.202

[6] FABŠIČ, T.—GROŠEK, O.—NEMOGA, K.—ZAJAC, P.: *On generating invertible circulant binary matrices with a prescribed number of ones*, Cryptogr. Commun. **10** (2018), 159–175.

[7] FABŠIČ, T.—HROMADA, V.—STANKOVSKI, P.—ZAJAC, P.—GUO, Q.—JOHANSSON, T.: *A reaction attack on the QC-LDPC McEliece cryptosystem.* In: *International Workshop on Post-Quantum Cryptography*, *Lecture Notes in Comput. Sci. Vol. 10346*, Springer, Cham, 2017. pp. 51–68.

[8] GUO, Q.—JOHANSSON, T.—STANKOVSKI, P.: *A key recovery attack on MDPC with CCA security using decoding errors.* In: The 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4–8, 2016. (Jung Hee Cheon, ed. et al). *Proceedings, Part I, Advances in Cryptology–ASIACRYPT 2016.* Springer-Verlag, Berlin, 2016. pp. 789–815.

[9] MCELIECE, R. J.: *A public-key cryptosystem based on algebraic coding theory*, DSN Progress Report **42** (1978), 114–116.

[10] MISOCZKI, R.—TILLICH, J.-P.—SENDRIER, N.—BARRETO, P. S.: *MDPC-McEliece: New McEliece variants from moderate density parity-check codes.* In: *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, IEEE, 2013, pp. 2069–2073.

[11] NEMEC, M.—SYS, M.—SVENDA, P.—KLINEC, D.—MATYAS, V.: *The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli.* In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017, pp. 1631–1648.

[12] NIEDERREITER, H.: *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory **15** (1986), 159–166.

[13] NIST: . *Post-Quantum Cryptography. Round 1 Submissions*, Project, 2018. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`.

[14] THE NEW YORK TIMES: . *Secret Documents Reveal N.S.A. Campaign Against Encryption*, 2013. `http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal--campaign-against-encryption.html`.

[15] YOUNG, A.—YUNG, M.: *The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?* In: *Advances in cryptology—CRYPTO '96* (Santa Barbara, CA), *Lecture Notes in Comput. Sci. Vol. 1109*, Springer-Verlag, Berlin 1996, pp. 89–103.

[16] ZAJAC, P.: *Hybrid encryption from McEliece cryptosystem with pseudo-random error vector.* presented at CECC17, 2018. Preprint,

Received September 18, 2018

Institute of Computer Science and Mathematics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Ilkovičova 3
SK–812-19 Bratislava
SLOVAKIA

E-mail: pavol.zajac@stuba.sk
peter.spacek@stuba.sk