Timing Analysis of Algorithm Substitution Attacks in a Post-Quantum TLS Protocol

Dúnia Marchiori[1], **Alexandre A. Giron**[1,2], **João Pedro A. do Nascimento**[3], **Ricardo Custódio**[1]

[1]Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil

[2]Universidade Tecnológica Federal do Paraná (UTFPR), Toledo, Brazil

[3]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil

{dunia.marchiori,alexandre.giron}@posgrad.ufsc.br,

joao.pedro.nascimento@grad.ufsc.br, ricardo.custodio@ufsc.br

Abstract. *Snowden's revelations about mass surveillance brought to public attention devastating attacks on cryptographic algorithm implementations. One of the most prominent subsets of these attacks is called Algorithm Substitution Attacks (ASA), where a subverted implementation leaks sensitive information. Recently, it has been proposed to modify TLS implementations to use Post-Quantum Cryptography (PQC). In this paper, we propose and analyze ASA in two PQC schemes that can be used in TLS. We attacked the Kyber Key Encapsulation Mechanism (KEM) and Falcon Signature and successfully deployed them in a TLS implementation. Results show that timing analysis can distinguish our Falcon subversion, but it is not enough to detect our attacks deployed in TLS.*

1. Introduction

Back in 1996, Young and Yung started what is now a research area called *Kleptography* [Young and Yung 1996]. Kleptography is characterized by preventing users from evaluating cryptographic implementations, usually embedded in specialized hardware. The authors have managed to create an attack in a scheme where the victim's implementation leaks the cryptographic key information in the ciphertext produced. As a result, the attacker can efficiently recover the key of the victim, which breaks the security of the cryptographic scheme. Since 1996, several cryptographic schemes were studied regarding these threats. Nowadays, the Algorithm Substitution Attacks (ASA) is a consolidated form of this type of attack, which can be applied to all forms of cryptography (symmetric and asymmetric) [Bellare et al. 2015].

In this context, ASA is one of the risks of using cryptographic software and hardware that can not be carefully audited and validated when ready for use. Unfortunately, real cases of these attacks do exist. The Juniper Dual EC incident [Checkoway et al. 2016] and the Snowden revelations [Perlroth et al. 2021] are the most notorious examples. In addition, researchers have already shown how to exploit these vulnerabilities in widely used network protocols, such as the Transport Layer Security (TLS) [Janovsky et al. 2019]. In general, backdoor-ed implementations allow attackers

to violate the privacy of all users of the protocol. This highlights the necessity of proper countermeasures and detection mechanisms against such threats. An example of a detection mechanism is the timing analysis [Teşeleanu 2019].

Before Young and Yung's work, Peter Shor's quantum algorithm was published [Shor 1994]. Differently from the ASA threat, his publication showed a possible threat to cryptography. Grover's algorithm [Grover 1996] followed Shor's work for symmetric schemes. Their algorithms could weaken or even break the current cryptographic schemes if a quantum computer were available for use. Predicting this threat and its consequences, researchers started in advance to develop new schemes of cryptography. These new schemes are called by Post-Quantum Cryptography (PQC). They are designed to protect not only against so-called classic attacks but also quantum computer attacks. It should be noted that PQC algorithms executes in classical computers. At this moment, there is a worldwide effort to propose, evaluate and standardize such algorithms. When PQC is standardized, there will be a long-term transition, where network protocols, hardware modules, operating systems, and other software will implement PQC schemes for the security [Mosca 2018].

However, as in traditional cryptography, PQC is also under the threats of ASA. For instance, NTRU (N-th degree Truncated polynomial Ring Units) is a public-key scheme that already has a kleptographic vulnerability for attackers [Kwant et al. 2018]. The schemes based on the Learning-With-Errors (LWE) problem also suffer from this attack [Yang et al. 2020b]. These examples show that there is still a lot to be researched about detection and countermeasures against ASA, in order to improve the resilience of the PQC schemes.

In this work, we focus on PQC schemes under the threats of ASA. The objective is to perform a timing analysis of such attacks. We deployed the proposed attacks in a Post-Quantum TLS implementation, which uses a Key Exchange such as a Key-Encapsulation Mechanism (KEM), and a Digital Signature Scheme. Although there exists a set of countermeasures against ASA, we consider the scenario where users have no access to audit their cryptographic implementations. This is a realistic scenario, since there are "black-box" devices such as Smart Cards, Trusted Platform Modules (TPMs), and closed hardware designs. Most importantly, the complexity of cryptographic implementations is something that not every developer can audit. In such a scenario, detection mechanisms can be more effective.

1.1. Contributions

In summary, the contributions of this work are listed below:

- We propose a symmetric ASA in Kyber KEM [Avanzi et al. 2020] and an asymmetric ASA in Falcon Signature Scheme [Fouque et al. 2020].
- We use timing analysis aiming to detect such attacks; we show that our attack on Kyber is undetectable in our given scenario.
- We also deployed and measured our attacks in a Post-Quantum TLS implementation, showing the practicability of such attacks.

We expect that these contributions would help for a secure transition process for PQC. By evaluating these attacks and their detection in advance, we contribute to the evolution of the development of such schemes, in addition to the awareness of this threat.

1.2. Text organization

This work is organized as follows. Section 2 gives the necessary background and related works about PQC, TLS and ASA. In Section 3, we present the attacks that we have developed and the timing analysis used in this work. Section 4 shows the results of the evaluations. Lastly, Section 5 gives the conclusions and future work.

2. Preliminaries

We divided this section into three parts. First, we present the concepts related to PQC schemes. The second part shows the characteristics of ASA and an example scenario. The last part presents related works.

2.1. Post-Quantum Cryptography (PQC)

Although quantum computers are not yet commercially available, the study of Post-Quantum Cryptography (PQC) is important. The term PQC refers to the schemes designed under mathematical problems believed to be intractable by non-quantum and quantum attackers. PQC users without quantum computers can expect to be protected against quantum attackers.

Several PQC algorithms have come under scrutiny and are being considered to become standards, which would allow for an earlier transition from traditional encryption schemes (e.g., RSA) to PQC. The best-known standardization process is due to NIST [NIST 2016], which is already in its third round. In this round, the following key encapsulation and authentication algorithms are being evaluated:

- Key Encapsulation Mechanisms (KEMs): Classic McEliece (*code-based cryptography*); Crystals-Kyber, NTRU and Saber (*lattice-based cryptography*).
- Authentication (Digital Signatures): Crystals-Dilithium and Falcon (*lattice-based cryptography*); and Rainbow (*multivariate cryptography*).

PQC KEM mechanisms are Public-key Encryption schemes defined as a triple of Probabilistic Polynomial-Time algorithms (`KeyGen`, `Encaps`, `Decaps`), where `KeyGen` produces the key pair; `Encaps` is an encryption of a session key k, producing a ciphertext; and `Decaps` decrypts the ciphertext and returns k, if successful. Similarly, the PQC algorithms used for authentication can be defined as a triple (`KeyGen`, `Sign`, `Verify`), where a `Sign` algorithm returns a digital signature for a given message; and `Verify` algorithm outputs either "true", if the signature is valid, or "false" otherwise.

2.2. TLS 1.3 Handshake

Version 1.3 of TLS is defined by RFC 8446 [Rescorla 2018], where the main components of the protocol are: a Handshake protocol, where an authenticated key exchange method is executed; a Record protocol for the secure communication; and an Alert protocol for error and alert messages. After a successful handshake, the record protocol protects the traffic between the communicating parties, using the derived (symmetric) cryptographic keys. If an error occurs, the alert protocol is triggered.

The handshake comprises an Authenticated Key Exchange (KEX), where the parties select the cryptographic parameters and establish shared keying material. TLS allows Elliptic Curve Diffie-Hellman Ephemeral (ECDHE), Pre-Shared Key (PSK), or

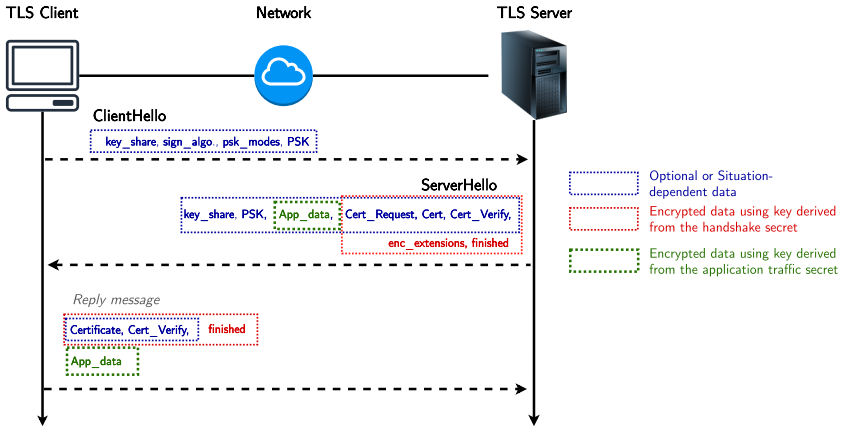

Figure 1. A TLS 1.3 Handshake

both in the handshake. Everything after the KEX is encrypted. The TLS server must be authenticated, and optionally, the client. If PSK is not used, a digital certificate is always used for authentication. The use of PQC algorithms is not defined in RFC 8446 [Rescorla 2018], but there are evaluations of post-quantum TLS implementations [Paquin et al. 2020], in addition to the Google and Cloudfare PQC experiments [Braithwaite 2016, Kwiatkowski et al. 2019].

Figure 1 represents a TLS 1.3 handshake. The (*ClientHello*) message consists of the following data: a random nonce, protocol versions, list of symmetric ciphers and hash pairs supported by the client. In addition, a *key_share* (for ECDHE) and/or *pre_shared_key* (PSK) message extension is sent. At least one of them must be sent (*key_share* or PSK messages). The TLS server then replies with the *ServerHello* message. The server sends its authentication messages: *Certificate*, *CertificateVerify* (a digital signature), *Certificate Request* (if client authentication is required) and *Finished* (a Keyed-Hash Message Authentication Code - HMAC). To reduce round trips, properly encrypted data from the application can be sent at this stage. Additional extensions are encrypted and also sent by the server. The TLS client responds accordingly, allowing the authenticated symmetric key to be used in new data exchanges between client and server.

2.3. Algorithm Substitution Attacks (ASA)

Figure 2 presents a simplified ASA example scenario. The starting point of this attack requires a cryptographic key k and an algorithm A'. Both k and A' are inserted in the user's device or software. In fact, A' replaces a cryptographic algorithm A, which is implemented by the user. This replacement (i.e., substitution) embeds a backdoor in the outputs of the user. The backdoor is encrypted in a way that it is only accessible by those who possess k. An additional requirement is that the attacker has to capture the outputs of the user (e.g., a public key, ciphertexts, or digital signatures). The attack is considered successful when the attacker is able to recover the secret key information from the encrypted data without being noticed.

This attack relies first on the secrecy of the substitution. Note that, in Figure 2, one user is attacked with the subverted version of the cryptographic algorithm (A'). However, if the attacker subverts a reference software implementation, every user of this software is compromised. For example, in the Dual EC incident [Checkoway et al. 2016],

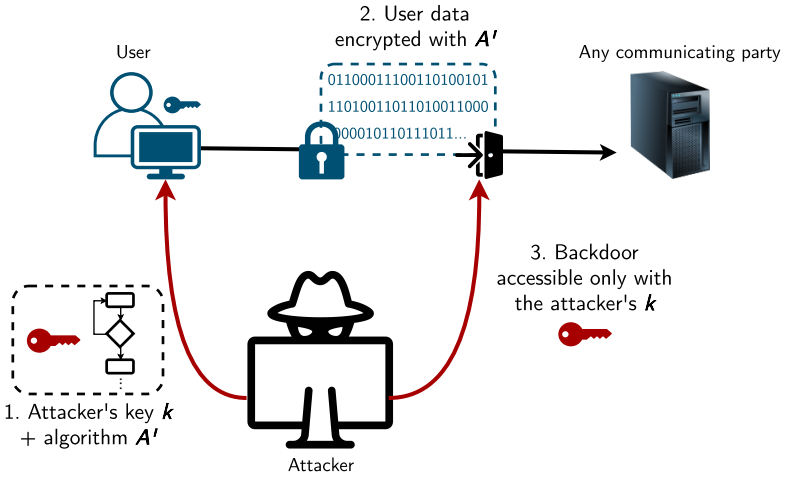

Figure 2. Algorithm Substitution Attack (ASA) example scenario.

a VPN software implementation was subverted, meaning that an attacker could efficiently decrypt every VPN connection created with that software. Another example, considered by some to be a flaw, but by others an ASA, was found in Infineon smart cards and TPMs allowing an attacker to recover private keys from public keys. The key-generation algorithm sometimes creates public keys that are vulnerable to Coppersmith's attack [Nemec et al. 2017].

In general, the attack can be symmetric or asymmetric, regarding the type of cryptographic key embedded by the attacker. The asymmetric attack means that the attacker generates a key pair (public, private) and embeds the public key. In the asymmetric setting, the attacker has the advantage that the backdoor can only be accessible by his private key [Baek et al. 2019]. However, there are scenarios where the symmetric setting can be sufficient. When the attack is performed in closed-hardware products, it can be difficult to retrieve the symmetric key. On the other hand, normally, asymmetric attacks are stateful, meaning that A' has to keep track of the information leaked in the outputs. In the stateful setting, the disadvantage is that the attack could be detected with a *state reset* (see Section 2.4). Therefore, from the attacker's perspective, stateless attacks are preferable to avoid detection.

2.4. Related Work

This paper aims to evaluate if we can detect ASA in a Post-Quantum TLS version. We focused our search on PQC schemes that are finalists in the NIST Round 3 standardization process [NIST 2016]. Therefore, we start this section with the ASA attacks already demonstrated in PQC schemes. Secondly, we look into the general detection mechanisms for these attacks. We were not able to find any publication addressing the ASA threat in a Post-Quantum TLS implementation.

Regarding lattice-based schemes, researchers found ASA in NTRU [Kwant et al. 2018] and in LWE-based schemes, such as NewHope [Yang et al. 2020a]. The main difference is that the authors claim that the attack in NTRU can be efficiently detectable, but that is not the case of LWE-based cryptosystems [Yang et al. 2020b]. Furthermore, the NTRU attacks are based on a substitution of NTRU encryption, whereas

LWE has attacks based on both key generation and encryption. Regarding other PQC schemes, we could not find ASA in code-based cryptography (such as Classic McEliece) nor in multivariate-based cryptography (such as Rainbow) in the literature.

In the literature, there are general and specific detection mechanisms for ASA. In addition, there are also countermeasures to prevents the ASA, such as a Cryptographic Reverse Firewall [Mironov and Stephens-Davidowitz 2015]. However, the countermeasures often require access to the implementation or rely on a trusted third party. In this paper, we focus on detection mechanisms because we consider the scenario where the user has no access or knowledge about the internals of his device or cryptographic implementation. Otherwise, a detection mechanism could be by software inspection.

A stateful ASA in a randomized scheme can be detected through a **state reset**. A state reset is a mechanism that returns the modified algorithm to its initial state. A simple reboot or a clone of the virtual machine are examples. Since the attack uses a deterministic subversion key, an analyst can compare two outputs after state resets; if the outputs are equal, it means that the scheme is subverted [Baek et al. 2019].

A more general detection mechanism is by side-channel analysis, for instance, a **timing analysis**. In this analysis, the execution time of both subverted and non-subverted algorithms are compared. In this scenario, kleptographic attacks can be detected with timing analysis [Teşeleanu 2019], but that is not the case if the attacker is able to mask execution times. None of the attacks in the PQC schemes mentioned went through timing analysis, to the best of our knowledge. Janovsky et al. [Janovsky et al. 2019] used timing analysis aiming the detection attacks in TLS 1.2 and 1.3, but considering only ECDH and RSA algorithms. Similarly, Berndt et al. [Berndt et al. 2020] explored ASA in TLS, but also in Signal and Wireguard protocols. Their works show that timing analysis is not always successful in detecting a subverted protocol.

3. Methodology

In this paper, we perform a timing analysis of ASA in PQC. We compare the timings of a subverted implementation against a non-subverted one. Two attacks are proposed: the first exploits Kyber in the key-generation process and the second exploits Falcon signatures. We also evaluate the scenario where we integrate the attacks in TLS. We use PQC implementations provided by the Open Quantum Safe (OQS) project [Stebila and Mosca 2016].

The choice of the attacked algorithms is based on the following: Kyber and Falcon are finalists in the NIST Standardization process with competitive performance [Avanzi et al. 2020, Fouque et al. 2020]; we were able to find ASA attacks on both algorithms; and the fact that a PQC TLS requires a Key Exchange mechanism, where Kyber can be used, and a Signature Scheme, such as Falcon. To the best of our knowledge, these ASA on Kyber and Falcon were not published before.

We compared the statistics computed for the subverted versions and the non-subverted versions. The metrics evaluated in this paper are grouped as follows.

- Cryptographic primitives: we measured the execution times (in microseconds) and CPU cycles. We sampled $1,000$ independent executions and computed the following statistics: minimum, first quartile, median, third quartile, and maximum. We also computed the average time of one million executions;

- TLS Protocol: we focused on the TLS handshake completion time. We compute the average handshake time of $1,000$ connections. Mininet [Lantz and O'Connor 2015] was used to simulate network conditions. We varied the network latency (in milliseconds), based on the methodology of Paquin et al. [Paquin et al. 2020].

All of the experiments were performed in a Linux machine equipped with an Intel i5-8250U 1.60GHz CPU. We do not employ AVX extensions or other optimizations in the PQC algorithms. Furthermore, we disabled network services (except for the TLS evaluation), hyperthreading, and turbo boost technologies to improve timing accuracy. The source code of our proposed attacks is available for download[1].

3.1. Proposed Attack on Kyber

We propose a symmetric ASA in the KYBER_KEM.KEYGEN algorithm. The attack injects information about the private key in the public key. As a result, the attacker can recover the private key. It is a stateless attack since only one public key is needed to recover private information.

Algorithm 1 shows the original Kyber KEM key generation algorithm (KYBER_KEM.KEYGEN()), and the subverted version (KYBER_KEM.KEYGEN′()) is presented in Algorithm 2. There is only one modification in Algorithm 2. In the original algorithm, the byte array z receives a random value (line 2), but on the subverted version, z receives the 32-byte hash of ρ. The value ρ is available in the public key pk. The 32-byte hash function is simplified as H. This change was made so the private key is created in a deterministic approach, but the pseudo-randomness of z is maintained.

Algorithm 1: KYBER_KEM.KEYGEN()

1 : $(pk, sk') \leftarrow$ KYBER_PKE.KEYGEN()
2 : $z \leftarrow\$ \{0,1\}^{256}$
3 : $sk \leftarrow (sk'||pk||\mathrm{H}(pk)||z)$
4 : **return** (pk, sk)

Algorithm 2: KYBER_KEM.KEYGEN′()

1 : $(pk, sk') \leftarrow$ KYBER_PKE.KEYGEN′()
2 : $z \leftarrow \mathrm{H}(\rho)$
3 : $sk \leftarrow (sk'||pk||\mathrm{H}(pk)||z)$
4 : **return** (pk, sk)

The Kyber KEM key generation algorithm evokes the Kyber PKE key generation algorithm, which the original version is detailed in Algorithm 3 (KYBER_PKE.KEYGEN()), whereas the subverted version (KYBER_PKE.KEYGEN′()) is presented in Algorithm 4. Lines 1–2 in Algorithm 3 are modified. Originally in these lines, d received a random 32-byte value, that is then applied to a hash function that returns 64-bytes, forming two values, ρ and σ, each of length of 32 bytes.

The subverted algorithm transforms the procedures in lines 1–2 in the original algorithm to lines 1–4 in Algorithm 4. These lines were modified so the value ρ appended to the public key pk in line 22 reveals information to the attacker. The information is the value d, used to compute σ, which, in turn, is used to compute the private key sk.

The modifications in Algorithm 4 are the following. First, the length of the byte array d is changed to 8 bytes. Then, the 64-byte hash of d is computed by $\mathrm{G}(d)$, resulting

[1] https://github.com/DuniaMarchiori/liboqs

in (x, σ), that are 32-byte each. The byte array σ is used in the construction of the private key. Having access to the value of d infers that σ can be computed and, consequently, that the private key can be recovered. Therefore, d is the information that is leaked in this attack.

Being a symmetric attack, the leaked information is encrypted using a symmetric key $symk_a$ that only the attacker has access to. Then, using the Advanced Encryption Standard (AES) algorithm with the Galois Counter Mode (GCM) mode, d is encrypted, resulting in a ciphertext, an authentication tag and an initialization vector (IV). AES-GCM is a good choice here because it produces a ciphertext that fits the required size. The length of both the tag and IV is 12 bytes each, and the length of the ciphertext is 8 bytes. These values are concatenated to create the value of the 32-byte array ρ, that is appended to the public key.

The remaining steps in Algorithm 4 are the same as the original algorithm. Additional details of the Kyber scheme can be found in the official specification [Avanzi et al. 2020], including the description of the functions NTT and ENCODE_{12}. In addition, the functions for seed expansion to polynomials following a uniform and centered binomial distribution were simplified and called SAMPLEUNIFORM and SAMPLECBD.

The output in Algorithm 2 is the subverted public key pk and the private key sk. Assuming the subverted public key is published, the subverted value ρ appended to pk can be accessed. Decrypting the ciphertext using the symmetric key $symk_a$, and the IV and tag contained in ρ, the attacker can recover the random value d, which can be used to create the private key as detailed in Algorithms 2 and 4.

Algorithm 3: KYBER_PKE.KEYGEN()

$d \leftarrow\$ \{0,1\}^{256}$
$(\rho, \sigma) \leftarrow \text{G}(d)$
$\hat{\text{A}} \in \mathcal{R}_q^{k \times k} \leftarrow \text{SAMPLEUNIFORM}(\rho)$
$s, e \in \mathcal{R}_q^k \leftarrow \text{SAMPLECBD}(\sigma)$
$\hat{s} \leftarrow \text{NTT}(s)$
$\hat{e} \leftarrow \text{NTT}(e)$
$\hat{t} \leftarrow \hat{\text{A}} \circ \hat{s} + \hat{e}$
$pk \leftarrow (\text{ENCODE}_{12}(\hat{t} \bmod {}^+q)||\rho)$
$sk \leftarrow \text{ENCODE}_{12}(\hat{s} \bmod {}^+q)$
return (pk, sk)

Algorithm 4: KYBER_PKE.KEYGEN′()

$d \leftarrow\$ \{0,1\}^{64}$
$(x, \sigma) \leftarrow \text{G}(d)$
$(\text{ciphertext}, \text{IV}, \text{tag}) \leftarrow \text{AES256\_GCM}(d, symk_a)$
$\rho \leftarrow (\text{ciphertext}||\text{IV}||\text{tag})$
$\hat{\text{A}} \in \mathcal{R}_q^{k \times k} \leftarrow \text{SAMPLEUNIFORM}(\rho)$
$s, e \in \mathcal{R}_q^k \leftarrow \text{SAMPLECBD}(\sigma)$
$\hat{s} \leftarrow \text{NTT}(s)$
$\hat{e} \leftarrow \text{NTT}(e)$
$\hat{t} \leftarrow \hat{\text{A}} \circ \hat{s} + \hat{e}$
$pk \leftarrow (\text{ENCODE}_{12}(\hat{t} \bmod {}^+q)||\rho)$
$sk \leftarrow \text{ENCODE}_{12}(\hat{s} \bmod {}^+q)$
return (pk, sk)

3.2. Proposed Attack on Falcon

We propose an ASA in the FALCON.SIGN algorithm. The attack leaks the private key of the user in the resulting signatures. The attacker needs two consecutive signatures to recover the private key of the victim; therefore it is a stateful attack.

Algorithm 5 shows the details of the original Falcon signing algorithm (FALCON.SIGN), whereas the subverted version (FALCON.SIGN′) is presented in Algorithm 6. The modifications are presented in lines 1-7. First, the subverted version takes as input the user's private key seed sk_u, with 32-bytes length, which is used to generate the original $sk = (B, T)$; the message m to be signed; and the attacker's public key pk_a. This pk_a is instantiated here as an Elliptical Curve (EC) public key. Secondly, the subversion exploits the random parameter r of the signature. Instead of a uniformly-random nonce, r is now computed as an encryption of sk_u using the attacker's pk_a. The encryption is an Elliptical Curve Integrated Encryption Scheme (ECIES) [Martínez et al. 2015]. When the hard-coded $state$ is 0, sk_u is encrypted. The parts of the ECIES ciphertext are assigned in r, depending on the value of $state$. The rest of the SIGN algorithm is the same as the original. We do not change the Falcon operations and functions such as Hash-to-point computation, Fast Fourier Transformations (simplified here as FFTS), Sampling and Compression function. Besides, we do not change the public parameters q and n. Additional details can be found in the Falcon specification [Fouque et al. 2020].

Algorithm 5: FALCON.SIGN(sk, m)

1 : $r \leftarrow\$ \{0,1\}^{320}$
2 : $c \leftarrow \text{HASH}(r||m, q, n)$
3 : $t \leftarrow \text{FFTS}(c)$
4 : **do**
5 : **do**
6 : $z \leftarrow \text{FFSAMPLING}_n(t, T)$
7 : $s = (t - z)\hat{B}$
8 : **while** $||s||^2 > \lfloor\beta\rfloor^2$
9 : $(s_1, s_2) \leftarrow \text{INVFFT}(s)$
10 : $s \leftarrow \text{COMPRESS}(s_2, 8 \cdot s_{len} - 328)$
11 : **while** $(s = \perp)$
12 : **return** $\sigma = (r, s)$

Algorithm 6: FALCON.SIGN′(sk_u, m, pk_a)

1 : **if** $state = 0$ **then**
2 : $buffer \leftarrow \text{ECIES}(\mathsf{pk}_a, \mathsf{sk}_u)$
3 : $r \leftarrow buffer[0..39]$
4 : $state \leftarrow state + 1$
5 : **else**
6 : $r \leftarrow buffer[40..79]$
7 : $state \leftarrow 0$
8 : $c \leftarrow \text{HASH}(r||m, q, n)$
9 : $t \leftarrow \text{FFTS}(c)$
10 : **do**
11 : **do**
12 : $z \leftarrow \text{FFSAMPLING}_n(t, T)$
13 : $s = (t - z)\hat{B}$
14 : **while** $||s||^2 > \lfloor\beta\rfloor^2$
15 : $(s_1, s_2) \leftarrow \text{INVFFT}(s)$
16 : $s \leftarrow \text{COMPRESS}(s_2, 8 \cdot s_{len} - 328)$
17 : **while** $(s = \perp)$
18 : **return** $\sigma = (r, s)$

The output given by Algorithm 6 is a subverted signature σ. Assuming that the user transmits at least two consecutive signatures after the subversion, the attacker can recover the user's private key. Having the consecutive signatures, the attacker reconstructs r from σ_i, σ_{i+1} and computes $sk_u = \text{ECIES}(sk_a, r)$, recovering the seed to generate the private key. Since our modification is strict in r, and the modified r is still pseudo-random, FALCON.SIGN' produces valid signatures. Therefore, the signature verification process is

the same as the non-subverted version. ECIES is a good choice for the attacker, due to the following: it allows an asymmetric attack (differently of the attack on Kyber), in which no other than the attacker can retrieve the user's private key, because the attacker's public key is the key embedded in the subverted algorithm; and ECIES generates an ephemeral public key (in the victim) for each signature pairs. This design turns our attack difficult to detect by means of state resets.

4. Results

In this section, we discuss the results of the two attacks proposed in Section 3. First, we analyze the execution time of the original and the subverted versions in Section 4.1. Then, in Section 4.2 we show the results of the subverted primitives in a TLS implementation.

4.1. Timing Analysis

Figure 3 presents the average execution time of million executions of KYBER.KEYGEN and FALCON.SIGN. All available security levels were tested (512, 768, 1024). Regarding Kyber, there is no significant difference between the attacked and non-attacked versions. Our symmetric attack uses AES-NI specialized hardware instructions, which is common in modern processors. The fast performance can explain why one can not detect the proposed attack with timing analysis. On the other hand, the slowdown caused by our asymmetric ASA in Falcon is evident.

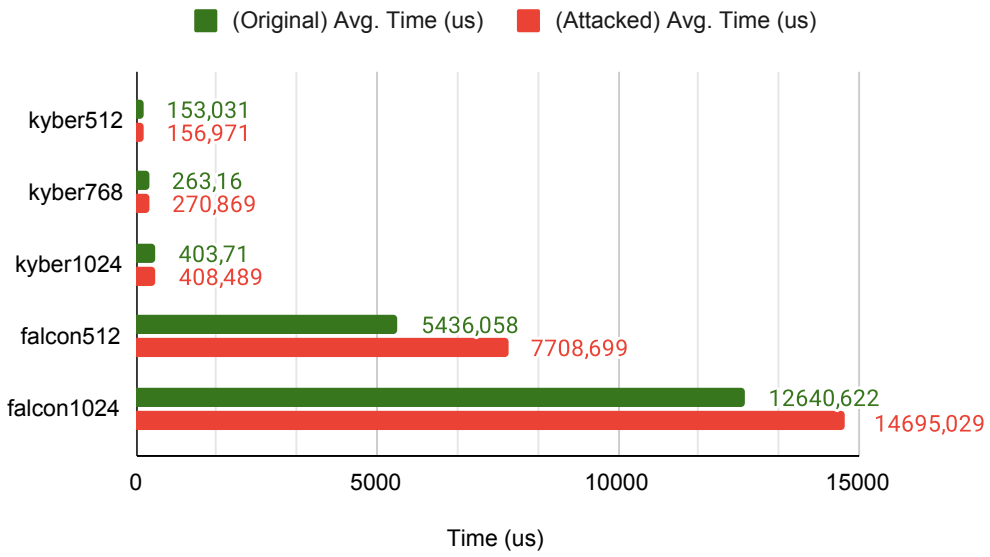

Figure 3. Average execution time of subverted and non-subverted version.

Table 1 gives more insights about the timing analysis of the cryptographic implementations. The results are a compilation of a thousand KYBER.KEYGEN and FALCON.SIGN executions. Each pair of lines allows comparing the subverted version against the original implementation. The subverted versions are marked with an apostrophe (’). Again, Kyber versions are indistinguishable under timing analysis. For instance, the quartile intervals (Q1-Q3) overlap in all Kyber pairs. However, Falcon implementations are

Table 1. Timing Analysis of KYBER.KEYGEN and FALCON.SIGN algorithms.

Algorithm	Min (μs)	Q1 (μs)	Q3 (μs)	Max (μs)
kyber512	152.643	156.039	157.208	239.651
kyber'512	156.743	157.106	160.585	229.148
kyber768	258.199	264.040	267.651	316.825
kyber'768	260.625	261.193	265.665	432.644
kyber1024	389.906	390.936	400.812	591.818
kyber'1024	393.417	394.423	401.295	590.241
falcon512	5305.085	5357.700	5401.045	6625.238
falcon'512	7338.234	7383.622	7463.104	8611.342
falcon1024	11 560.391	11 664.326	11 691.744	14 222.915
falcon'1024	13 542.865	13 605.375	13 639.466	16 428.984

distinguishable since the results do not overlap between the quartile intervals. Our asymmetric attack using ECIES is costly because it requires an ephemeral public key generation, shared secret computation and AES encryption. This computational cost explains the differences in the execution times.

4.2. Attack on Post-Quantum TLS

We managed to deploy our attack in OpenSSL based on the OQS project [Stebila and Mosca 2016]. The flow of the messages is explained as follows: the TLS client executes the subverted KYBER.KEYGEN and send its fresh (and subverted) public key in the *ClientHello* message; upon reception, the TLS Server executes KYBER.ENCAPS and the subverted FALCON.SIGN, replying with a subverted signature; the TLS Client executes KYBER.DECAPS and FALCON.VERIFY, establishing the connection. The attacker watches the whole communication. The attacker can break TLS confidentiality by recovering the Kyber private key of the client, allowing to decrypt the ciphertext sent by the server and derive the symmetric keys used in their connection. The attacker can also break authentication by capturing two consecutive signatures and assuming that the TLS implementation stores the Falcon private key seed as mentioned in the specification. In this case, the attacker is able to impersonate the TLS server (which breaks HTTPS security).

Figure 4 shows the results of a thousand TLS handshakes. We evaluate our attacks in the same security parameter (λ) for Kyber and Falcon. When increasing the network latency in the simulation, we can not distinguish the attacked implementations from the original ones. This indicates that, under realistic conditions, the attack is imperceptible using this metric.

5. Conclusions

In this work, we propose two Algorithm Substitution Attacks. One is a symmetric attack in the Kyber KEM and the other one is a asymmetric attack in the Falcon signature scheme. In our experiments, the proposed attack in Kyber is undetectable, while the one in Falcon can be detected with a timing analysis. As seen in Table 1, the execution time of

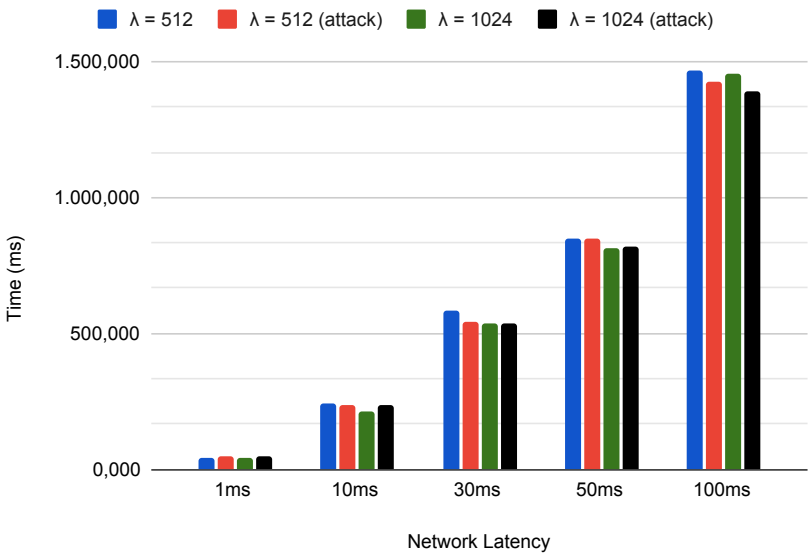

Figure 4. Handshake Time of the TLS implementations

the subverted versions of Falcon is slower than the original algorithms. Whereas the execution time of the subverted versions of Kyber is around the same values as the original versions.

The slower execution time of the subverted versions of the Falcon signature schemes were due to the use of ECIES for encryption. However, with the use of ECIES, the attack is resistant to state resets because a new ephemeral key is created for every encryption. On the other hand, since the subverted versions of Kyber used AES for encryption, our attack achieved fast performance because it used AES-NI specialized instructions. Therefore, the attack is difficult to detect with timing analysis.

We also analyzed the behavior of our subverted algorithms in a Post-Quantum TLS implementation. Our experiments showed that higher latency values conceal the execution time variations of the subverted algorithms, resulting in an unnoticeable attack in these conditions. The impact showed in our scenario is the capability of breaking both confidentiality and authentication of the TLS connections.

5.1. Open problems

We discussed attacks in only 2 of the 7 finalists in the NIST Round 3 standardization process in this work. Therefore, these analyses can be done to other post-quantum algorithms. Additional metrics can be included in the analysis, such as power consumption measurements. Besides, the asymmetric ASA on Falcon proposed here does not provide a quantum-resistant encryption process to the attacker. There is little work on ASA that offers post-quantum security for the attacker. In addition, we proposed a symmetric ASA on Kyber, but an asymmetric one could also be designed.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Author 2 thanks the support of the

Federal University of Technology (UTFPR).

References

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2020). CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.0). Submission to the NIST's post-quantum cryptography standardization process.

Baek, J., Susilo, W., Kim, J., and Chow, Y.-W. (2019). Subversion in practice: How to efficiently undermine signatures. *IEEE Access*, 7:68799–68811.

Bellare, M., Jaeger, J., and Kane, D. (2015). Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1431–1440, New York, NY, USA. Association for Computing Machinery.

Berndt, S., Wichelmann, J., Pott, C., Traving, T.-H., and Eisenbarth, T. (2020). ASAP: Algorithm Substitution Attacks on Cryptographic Protocols. Cryptology ePrint Archive, Report 2020/1452. Available at: `https://eprint.iacr.org/2020/1452`. Accessed on 2021-05-02.

Braithwaite, M. (2016). Experimenting with post-quantum cryptography. Available at: `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`. Accessed on 2021-02-25.

Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.-P., Rescorla, E., and Shacham, H. (2016). A systematic analysis of the juniper dual EC incident. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 468–479, New York, NY, USA. Association for Computing Machinery.

Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. (2020). Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU documentation (version 1.2). Submission to the NIST's post-quantum cryptography standardization process.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, Philadelphia Pennsylvania USA. ACM.

Janovsky, A., Krhovjak, J., and Matyas, V. (2019). Bringing kleptography to real-world TLS. In Blazy, O. and Yeun, C. Y., editors, *Information Security Theory and Practice*, pages 15–27, Cham. Springer International Publishing.

Kwant, R., Lange, T., and Thissen, K. (2018). Lattice klepto. In Adams, C. and Camenisch, J., editors, *Selected Areas in Cryptography – SAC 2017*, pages 336–354, Cham. Springer International Publishing.

Kwiatkowski, K., Langley, A., Sullivan, N., Levin, D., Mislove, A., and Valenta, L. (2019). Measuring TLS key exchange with post-quantum KEM. In *Workshop Record of the Second PQC Standardization Conference.*

Lantz, B. and O'Connor, B. (2015). A mininet-based virtual testbed for distributed sdn development. *ACM SIGCOMM Computer Communication Review*, 45(4):365–366.

Martínez, V. G., Encinas, L. H., and Dios, A. Q. (2015). Security and practical considerations when implementing the elliptic curve integrated encryption scheme. *Cryptologia*, 39(3):244–269.

Mironov, I. and Stephens-Davidowitz, N. (2015). Cryptographic reverse firewalls. In Oswald, E. and Fischlin, M., editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 657–686, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mosca, M. (2018). Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security Privacy*, 16(5):38–41.

Nemec, M., Sys, M., Svenda, P., Klinec, D., and Matyas, V. (2017). 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*, CCS '17, page 1631–1648, New York, NY, USA. Association for Computing Machinery.

NIST (2016). Post-quantum cryptography. Available at: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`. Accessed on 2020-06-26.

Paquin, C., Stebila, D., and Tamvada, G. (2020). Benchmarking post-quantum cryptography in TLS. In Ding, J. and Tillich, J.-P., editors, *Post-Quantum Cryptography*, pages 72–91, Cham. Springer International Publishing.

Perlroth, N., Larson, J., and Shane, S. (2021). N.S.A. able to foil basic safeguards of privacy on web. Available at: `https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html`. Accessed on 2021-04-21.

Rescorla, E. (2018). The transport layer security (TLS) protocol version 1.3. RFC 8446, RFC Editor. Available at: `http://www.rfc-editor.org/rfc/rfc8446.txt`. Accessed on 2021-05-02.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134, Santa Fe, NM, USA. IEEE, IEEE.

Stebila, D. and Mosca, M. (2016). Post-quantum key exchange for the internet and the open quantum safe project. In *International Conference on Selected Areas in Cryptography*, pages 14–37. Springer.

Teşeleanu, G. (2019). Threshold kleptographic attacks on discrete logarithm based signatures. In Lange, T. and Dunkelman, O., editors, *Progress in Cryptology – LATINCRYPT 2017*, pages 401–414, Cham. Springer International Publishing.

Yang, Z., Chen, R., Li, C., Qu, L., and Yang, G. (2020a). On the security of LWE cryptosystem against subversion attacks. *The Computer Journal*, 63(4):495–507.

Yang, Z., Xie, T., and Pan, Y. (2020b). Lattice klepto revisited. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '20, page 867–873, New York, NY, USA. Association for Computing Machinery.

Young, A. and Yung, M. (1996). The dark side of "black-box" cryptography or: Should we trust capstone? In Koblitz, N., editor, *Advances in Cryptology — CRYPTO '96*, pages 89–103, Berlin, Heidelberg. Springer Berlin Heidelberg.