A Lightweight Identification Protocol Based on Lattices

Samed Düzlü[1], Juliane Krämer[1], Thomas Pöppelmann[2], and Patrick Struck[1]

[1] Universität Regensburg, Germany
{samed.duzlu,juliane.kraemer,patrick.struck}@ur.de
[2] Infineon Technologies AG, Germany
thomas.poeppelmann@infineon.com

Abstract. In this work we present a lightweight lattice-based identification protocol based on the CPA-secured public key encryption scheme Kyber. It is designed as a replacement for existing classical ECC- or RSA-based identification protocols in IoT, smart card applications, or for device authentication. The proposed protocol is simple, efficient, and implementations are supposed to be easy to harden against side-channel attacks. Compared to standard constructions for identification protocols based on lattice-based KEMs, our construction achieves this by avoiding the Fujisaki-Okamoto transform and its impact on implementation security.

Moreover, contrary to prior lattice-based identification protocols or standard constructions using signatures, our work does not require rejection sampling and can use more efficient parameters than signature schemes. We provide a generic construction from CPA-secured public key encryption schemes to identification protocols and give a security proof of the protocol in the ROM. Moreover, we instantiate the generic construction with Kyber, for which we use the proposed parameter sets for NIST security levels I, III, and V. To show that the protocol is suitable for constrained devices, we implemented one selected parameter set on an ARM Cortex-M4 microcontroller. As the protocol is based on existing algorithms for Kyber, we make use of existing SW components (e.g., fast NTT implementations) for our implementation.

Keywords: Lattice-Based Cryptography, Identification Protocol, Post-Quantum Cryptography, LWE

1 Introduction

It is currently expected that large-scale quantum computers will be able to break classical cryptographic hardness assumptions in the future. This expectation has led to a standardization process by the National Institute of Standards and Technology (NIST). NIST aims to standardize digital signature schemes as well as key encapsulation mechanisms (KEMs) and public key encryption (PKE) schemes that are supposed to be secured against attacks by quantum computers. From the pool for third round candidates, NIST recently selected three signature schemes (Dilithium, Falcon, SPHINCS$^+$) and one KEM (Kyber) for standardization.

While KEMs/PKEs and digital signature schemes are fundamental constructions that are in focus of the NIST process, further post-quantum cryptographic schemes will also be required in the future. For instance, instead of more advanced functionality, in some applications it may be sufficient to just verify that a communicating party is indeed the claimed identity, a property that is known as *authenticity*. This can be achieved by using an identification protocol that allows one party to prove its identity to another party. Such protocols enable one party (the prover) to convince another party (the verifier) that it knows some secret without revealing it.

Identification protocols can be based on a specific underlying problem, e.g., the hardness of lattice problems as shown in [37]. Another approach is to construct them from KEMs [3] or digital signature schemes. However, the resulting protocols may carry the overhead of the inherent security requirements of those schemes. In particular, simple constructions like [3] often require a KEM secured against chosen ciphertext attacks (CCA), which entails overhead from the Fujisaki-Okamoto (FO) transform. In such protocols, which we will refer to as generic 2-pass protocols, the verifier sends a ciphertext as a challenge and the prover authenticates by sending back the underlying message. In these protocols, the prover acts as a decryption oracle, which results in the necessity of CCA security for the underlying encryption scheme. However, it could be beneficial to remove the need for CCA security of the KEM by modifying the protocol for better efficiency or implementation security [4, 7, 43]. This may be required when establishing the authenticity of devices in a cost-effective manner on very constrained devices. Practical examples are standards like USB Type-C Authentication [49] or Qi 1.3 for wireless device charging [15] that now specify or even require ECDSA-based device authentication to test that parts are manufactured by trusted vendors and according to the necessary safety standardization. In the long term, such standards will have to be moved to schemes that offer sufficient quantum resistance.

1.1 Contribution

In this work we present a novel 4-pass identification protocol based on lattices that is efficient, lightweight, and easy to be securely implemented. The protocol is based on Kyber, however, by moving from a 2-pass protocol to a 4-pass protocol, we only need the CPA-secured variant of Kyber rather than the more costly CCA-secured variant. As a disadvantage, one might argue that we can only prove security assuming random oracles, whereas the 2-pass protocol does not require random oracles. We note, however, that achieving CCA-secured encryption via the FO transform also requires random oracles. The 2-pass protocol therefore inherits this assumption from the underlying encryption scheme. While our protocol requires more communication, it avoids the costly FO transform *without* adding costs in form of additional assumptions. In light of several works which indicate that side-channel security for the FO transform is a delicate matter [4, 7, 43], our new protocol provides an interesting alternative, despite the slightly extra cost in communication. Our idea avoids the need for CCA security and therefore the FO transform, by separating the challenge and a challenge verification, which is implicit in the CCA case. This separation allows the verification to be independent of the secret key including all data that is derived from it, so that the verification step on the prover side does not need to be secured against side-channel leakage. In particular for lattice-based cryptography, hardening the challenge verification against side-channels is expensive due to sampling procedures.

Furthermore, we provide a set of parameters following Kyber and an implementation targeting ARM Cortex-M4 based microcontrollers.

Our 4-pass identification protocol is, in fact, generic, using only a CPA-secured PKE. This allows to easily instantiate the generic version with different PKEs. In particular, the generic approach enables crypto-agile implementations using post-quantum assumptions other than lattice problems.

1.2 Related Work

In [3, 22] it is shown that KEMs can be used to construct identification schemes. The constructions require a OW-CCA and OW-ftCCA secured encryption scheme, respectively, whereas our construction only requires IND-CPA security. Additionally, the instantiations in [3, 22] are based on the Diffie-Hellman assumption and, hence, do not achieve post-quantum security. Similarly, using digital signature schemes allows to construct lattice-based identification protocols. Conversely, signature schemes like Dilithium [39] make use of the Fiat-Shamir transform with aborts [38]. However, the usage of rejection sampling results in a scheme that is less efficient than lattice-based KEM constructions and challenging to be secured against implementation attacks [41].

Note that in contrast to works like [1, 37, 38] our scheme has 4 steps of communication instead of 2. However, all three schemes have built-in aborts with significantly lower success ratio. Moreover, the additional communication step in the present case allows the honest generation of challenges, which is not the case in [1].

The authors of [10, 11] also proposed identification protocols specifically for smart cards and embedded devices. They measure the performance of GLP signatures [26] and BLISS signatures [19] in an ID-scheme setting and also evaluate a commitment protocol proposed in [18]. These identification protocols did not receive much attention and in contrast to our work, they do not consider side-channel attacks explicitly.

Aside from lattice-based cryptography, there are other assumptions on which quantum-secured cryptography can be based. Among those, there have been attempts to construct identification protocols based on multivariate polynomials [44], codes [47], and isogenies [16, 24, 29], where [29] is based on the SIDH problem and may be vulnerable due to the recent attack [14].

The symmetric counterpart of identification schemes is the notion of authentication schemes, where the prover and verifier have a shared secret. An instantiation using lattices has been developed in [28, 34] based on the learning parity with noise (LPN) problem.

1.3 Outline

In Section 2, we give a brief discussion on the required technical background for the presentation of the identification protocol. In Section 3, we give a description of the identification protocol and provide a formal security reduction. In Section 4 we present an instantiation with the lattice-based PKE scheme Kyber and give design rationales of our construction. We further provide a choice of parameters for the instantiation and give details on the implementation.

2 Background

In this section, we explain the notation and basic concepts that are required for the description and analysis of the identification protocol.

2.1 Cyptographic Primitives

We will make use of cryptographic hash functions. The hash functions will be denoted F, H, and G_i, for i in some index set I. The family $(G_i)_{i\in I}$ of hash functions is denoted $\mathcal{G}$ for short. The hash functions are separated according to their use in the identification protocol, i.e., F is used to generate a random message (challenge), while $\mathcal{G}$ is used to generate the internal randomness of the underlying encryption algorithm, and H is used for the commitment computation.

In the instantiation, the distinct hash functions will be implemented from a single hash function using domain separation [5]. In the security proof, we will model the hash functions as random oracles [6].

IND-CPA *security of PKEs.* As the identification protocol is based on IND-CPA secured public-key encryption scheme, we briefly recall its definition, cf. [33].

The security experiment for the IND-CPA security of a PKE scheme $\mathsf{PKE} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ is given as

1. $\mathtt{KGen}$ is run with output $(\mathsf{pk}, \mathsf{sk})$,
2. The adversary $\mathcal{A}$ receives the public key pk and outputs two messages m_0 and m_1 of the same length
3. A random bit $b \in \{0, 1\}$ is chosen and $\mathcal{A}$ receives $c_b \coloneqq \mathtt{Enc}_{\mathsf{pk}}(m_b)$,
4. $\mathcal{A}$ outputs $b' \in \{0, 1\}$,
5. Finally, $\mathcal{A}$ wins, if $b' = b$ and loses otherwise.

The *advantage of* $\mathcal{A}$ *against the* IND-CPA *security of* PKE is then defined as

$$\mathbf{Adv}_{\mathsf{PKE}}^{\text{IND-CPA}}(\mathcal{A}) = \mathbb{P}\Big(\mathcal{A} \text{ wins game IND-CPA}\Big).$$

2.2 Protocol Security

There are three distinct security notions for identification protocols, namely *passive* and *active* (*attack*) security, and the security against *man-in-the-middle* attacks. We will focus on active attack security, which is described in terms of three phases, the *setup phase*, the *probing phase*, and the *impersonation phase*.

In the setup phase, the keys are generated and the adversary receives the public key. In the probing phase, the adversary takes the role of the verifier and can interact with an honest prover. The adversary is allowed to invoke the honest prover multiple times. After the probing phase, the adversary proceeds to the impersonation phase. Here, the adversary takes the role of the prover, interacting with an honest verifier. The adversary wins, i.e., breaks the identification protocol, if the honest verifier accepts at the end.

Setup Phase: A key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathtt{KGen}()$ is generated. The adversary $\mathcal{A}$ receives pk and the probing phase starts.

Probing Phase: In this phase, the adversary $\mathcal{A}$ can interact with an honest prover, knowing the secret key sk. At the end of the phase, the impersonation phase starts.

Impersonation Phase: In this phase, the adversary interacts with an honest verifier, knowing the public key pk. The adversary wins if the prover accepts, i.e., outputs 1 at the end.

For an adversary $\mathcal{A}$ against the active security AS, we denote by $\mathbf{Adv}^{\mathsf{AS}}(\mathcal{A})$ the probability that the honest verifier outputs 1 at the end of the security experiment above, i.e., that the adversary successfully impersonates an honest prover.

3 The Identification Protocol

In this section, we present an identification protocol (Fig. 1) based on an IND-CPA secured PKE scheme and prove its active security. The security of the identification protocol is independent of the underlying security assumptions on the PKE scheme. In Section 4, we provide an instantiation with the lattice-based Kyber PKE scheme, a collection of parameters, and a comparison of the implementation with other identification protocol constructions.

3.1 Description of the Identification Protocol

Before giving details, we want to briefly describe the identification on a high-level. See also Section 4.1 for more details on design rationales.

The protocol is executed between a verifier $\mathtt{V}$ who knows the public key and a prover $\mathtt{P}$ who knows the corresponding secret key. The protocol starts with a *challenge computation*, where the underlying encryption algorithm is made deterministic by generating message and random coins from a seed. The resulting ciphertext is send to the prover as the challenge. In the following *response computation*, the prover decrypts the ciphertext to receive a message. This is the only step, where the secret key is used. The resulting message is hashed together with a random value and the hash is send to the verifier. The verifier sends the seed used to generate the message and random coins to the prover. During the *challenge verification*, the prover can use this seed to check that the challenge was honestly generated. In this step, the prover does not make use of its secret (indeed, not even results of any computations using the secret key are required in this step). Only then, the prover sends its random value chosen in the response computation. Finally, in the *response verification*, the verifier checks that the hash value it received is compatible with the hash of the random response it received after the challenge verification with the original message it generated in the challenge computation.

The underlying PKE scheme and public parameters. Let $\mathsf{PKE} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$ be a PKE scheme. Further, let F, G_i, H be hash functions, with $\mathcal{G} = (G_i)_{i \in I}$ is a (finite) family I of hash functions. We assume that PKE is instantiated with the hash functions $\mathcal{G}$. Lastly, α is a security parameter.

The identification protocol will be denoted Π_{PKE} and is depicted in Fig. 1. In what follows, we give a description of the steps.

Key Generation. The public and secret keys of Π_{PKE} are the same as the key pairs of PKE. Thus, the key generation is done by running the $\mathtt{KGen}$ algorithm, resulting in a key pair $(\mathsf{pk}, \mathsf{sk})$.

Identification. Given the key pair $(\mathsf{pk}, \mathsf{sk})$ as above, the 4-pass identification procedure of Π_{PKE} is as follows.

`Verifier: Challenge Computation.` In the first step, the verifier `V` picks a random value $\lambda \in \{0,1\}^\alpha$. Then, λ is used to compute a challenge message $m \leftarrow F(\lambda)$. We let $\mathsf{coins} = \mathcal{G}(\lambda)$ be the random coins used during the encryption and set $c = \mathtt{Enc}(\mathsf{pk}, m, \mathsf{coins})$.

`1st Transmission: V`$\rightarrow$`P.` The verifier sends c to the prover `P`.

`Prover: Response Computation.` The prover decrypts c to get a message $\tilde{m} = \mathtt{Dec}(\mathsf{sk}, c)$. This is the only step where the secret key sk is used. Then, the prover samples a random value $r \leftarrow_{\$} \{0,1\}^\alpha$ and computes $h \coloneqq H(r, \tilde{m})$. Note that r is independent of the challenge c and the secret s.

`2nd Transmission: P`$\rightarrow$`V.` The prover sends the hash digest h of $(r, \tilde{m})$ to the verifier. With h, the prover commits on its random value r.

`Verifier.` The verifier stores h.

`3rd Transmission: V`$\rightarrow$`P.` The verifier sends λ to the prover.

`Prover: Challenge Verification.` The prover uses λ to re-generate $m' \leftarrow F(\lambda)$ and the random values $\mathsf{coins}' = \mathcal{G}(\lambda)$, with which it computes $c' = \mathtt{Enc}(\mathsf{pk}, m', \mathsf{coins}')$. Then, the prover checks whether $c = c'$ and aborts if not. Note that the prover uses m' instead of $\tilde{m}$ to re-compute the ciphertext, making this step independent of its secret s.

Verifier $\mathtt{V}$		*Prover* $\mathtt{P}$
pk		sk
	.... Protocol	
Challenge Computation $\lambda \leftarrow_\$ \{0,1\}^\alpha$ $m \leftarrow F(\lambda)$ $\mathsf{coins} \leftarrow \mathcal{G}(\lambda)$ $c \leftarrow \mathtt{Enc}(\mathsf{pk}, m; \mathsf{coins})$		
	$\xrightarrow{\quad c \quad}$	**Response Computation** store c $\tilde{m} \leftarrow \mathtt{Dec}(\mathsf{sk}, c)$ $r \leftarrow_\$ \{0,1\}^\alpha$ $h \leftarrow H(r, \tilde{m})$
store h	$\xleftarrow{\quad h \quad}$	
	$\xrightarrow{\quad \lambda \quad}$	**Challenge Verification** $m' \leftarrow F(\lambda)$ $\mathsf{coins}' \leftarrow \mathcal{G}(\lambda)$ $c' \leftarrow \mathtt{Enc}(\mathsf{pk}, m'; \mathsf{coins}')$ **if** $c' \neq c$ **return** $\perp$
Response Verification $h' \leftarrow H(r, m)$ **if** $h' \neq h$ **return** 0 **return** 1	$\xleftarrow{\quad r \quad}$	

Fig. 1. The generalized identification protocol Π with hash functions F and H, and a family of hash functions $\mathcal{G} = (G_i)$ depending on the size of coins for the given PKE.

`4th Transmission: P`$\rightarrow$`V.` The prover sends its commitment r to the verifier.

`Verifier: Response Verification.` The verifier checks, if $h = H(r, m)$ and outputs 1 if it holds. Otherwise, the verifier outputs 0.

3.2 Security Analysis of the Identification Protocol

We proceed with the security analysis of the identification protocol described above. We show that its active attack security AS reduces to the IND-CPA security of the underlying PKE scheme. As a consequence, the identification protocol instantiated with $\mathsf{Kyber.CPAPKE}$ is secured, see Corollary 4.1. The security reduction is proved in the random oracle model and an extension to the quantum random oracle model is discussed in Section 3.3.

Theorem 3.1. *Let Π be the identification protocol described in Fig. 1 based on a PKE scheme* $\mathsf{PKE} = (\mathtt{KGen}, \mathtt{Enc}, \mathtt{Dec})$*. Then, in the random oracle model, for any adversary $\mathcal{A}$ against Π, making q queries to $\mathcal{G}$, there exists an adversary $\mathcal{B}$ against* PKE *such that*

$$\mathbf{Adv}_{\Pi}^{\mathsf{AS}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{PKE}}^{\text{IND-CPA}}(\mathcal{B}) \ + \frac{q}{2^{\alpha}} ,$$

where the hash functions are modeled as random oracles.

Proof. Let $\mathcal{A}$ be an adversary against Π instantiated with PKE. We construct an adversary $\mathcal{B}$ against PKE that makes use of $\mathcal{A}$ and breaks the IND-CPA security of PKE.

Let $\mathcal{B}$ be given a public key pk which is part of a key pair $(\mathsf{pk}, \mathsf{sk})$. Then by definition of IND-CPA, $\mathcal{B}$ picks two messages, receives the encryption of one of the messages, and has to distinguishing which one was encrypted. To achieve this, $\mathcal{B}$ runs $\mathcal{A}$ with its own challenge public key pk.

Probing Phase. To make $\mathcal{A}$ run the attack, $\mathcal{B}$ needs to simulate the probing phase of the active attack with the public key pk, in which $\mathcal{A}$ plays the role of a verifier and can submit challenges to the prover, which is played by $\mathcal{B}$. In the random oracle model, $\mathcal{B}$ can simulate a prover without knowledge of the secret key as follows. In the response computation, $\mathcal{B}$ samples a random value h, and returns h to the verifier, in this case $\mathcal{A}$. After receiving λ, $\mathcal{B}$ checks, whether the challenge of $\mathcal{A}$ was generated honestly. If the check holds, $\mathcal{B}$ is now in possession of the message $m = F(\lambda)$. Then $\mathcal{B}$ picks a random value r and programs the random oracle H to take (r, m) to h. It is impossible for $\mathcal{A}$ to detect this reprogramming, unless it requested the value $H(r, m)$ earlier. However, $\mathcal{B}$ has access to all random oracle calls of $\mathcal{A}$ and can check which values of the form (r, m) have been queried by $\mathcal{A}$. As even for a fixed m, there are exponentially many pairs (r, m), $\mathcal{B}$ can always find a pair which has not been queried before.

Impersonation Phase. The idea for the impersonation phase is that $\mathcal{B}$ will send its own challenge ciphertext (from the IND-CPA game) to the adversary.

During the response computation, $\mathcal{A}$ has to commit to a message $\tilde{m}$ in form of sending $h = H(r, \tilde{m})$. This enables $\mathcal{B}$ to extract the message $\tilde{m}$ from the random oracle queries by $\mathcal{A}$; note that in the random oracle model that any (successful) adversary has to send r that was used to compute h. This enables $\mathcal{B}$ to run the impersonation phase up to the point where $\mathcal{A}$ has sent its commitment h and extracts the message from this.

However, simply injecting the ciphertext from the IND-CPA game does not correspond to challenge ciphertexts in the protocol as they are generated independently of the random oracles $F, \mathcal{G}$. Since the IND-CPA game allows $\mathcal{B}$ to choose arbitrary messages, it can simply compute those as outputs of F. But coins are chosen by the IND-CPA challenger, independently of any random oracle. This means, that $\mathcal{B}$ simulates the impersonation phase (up to the point where $\mathcal{A}$ sends h) for independently chosen coins. Detecting this simulation boils down to querying the random oracle $\mathcal{G}$ on λ, however, $\mathcal{A}$ does not have any information about it; even recovering m from c does not help due to the one-wayness of F. Since $\mathcal{A}$ makes q queries to $\mathcal{G}$, its probability of detecting the simulation is at most $\frac{q}{2^\alpha}$.

Now we can give the reduction $\mathcal{B}$. It picks λ_0, λ_1 uniformly at random from $\{0,1\}^\alpha$ to compute messages $m_b = F(\lambda_b)$, for $b \in \{0,1\}$. The messages m_0, m_1 are sent to the IND-CPA challenger which responds with $c_b = \mathtt{Enc}(\mathsf{pk}, m_b; \mathsf{coins})$ for coins chosen uniformly at random. Then $\mathcal{B}$ sends c_b to $\mathcal{A}$. When $\mathcal{A}$ outputs h, $\mathcal{B}$ will check for a query (r^*, m^*) to H. If $m^* = m_0$, $\mathcal{B}$ outputs 0, if $m^* = m_1$, $\mathcal{B}$ outputs 1. If neither check passes, i.e., $\mathcal{A}$ is not successful, $\mathcal{B}$ outputs a uniformly random bit. $\square$

3.3 Extension to the Quantum Random Oracle Model

We briefly argue how the proof can be translated to the quantum random oracle model (QROM) [9]. In the probing phase, the reduction can no longer look up the queries that the adversary has made to the random oracle. This thwarts to simply choose a value r such that the adversary has not queries (r, m) to the random oracle. Instead the reduction will simply pick r at random and reprogram the random oracle on (r, m) to h, where m is the message it obtains after receiving the seed λ and h is the uniformly random value which the reduction send to the adversary after receiving the challenge ciphertext. The O2H lemma [2] allows to upper bound the chance that the adversary can detect this reprogramming where it still holds that the adversary has no knowledge of the value r, which was chosen uniformly at random and independent of everything else. This step, however, induces another term into the bound since the reprogramming cannot be made certain to happen at a point the adversary has not queried.

For the impersonation phase, the reduction extracts the query (r, m) from the hash value h it receives from the adversary; this does not work in the QROM, when the adversary makes its queries in superposition. Luckily, the technique by Targhi and Unruh [48] allows to circumvent the problem. The reduction simulates the random oracle using a $2q$-wise independent function (e.g., a polynomial of degree $2q$) which was shown to be indistinguishable up to q superposition queries by Zhandry [50]. Upon receiving the classical value h, the reduction can extract candidates for (r, m) by computing the roots of the polynomial and, if one of the candidates equals either of the messages, the reduction outputs the corresponding bit. Additionally, the adversary might be able to notice the simulation via the IND-CPA security game, where coins are generated independently of the random oracle $\mathcal{G}$. This step also boils down to applying the O2H lemma [2] and the fact that the adversary has no knowledge about λ. More recent variants of the O2H lemma [2, 8, 36] and other QROM extraction techniques [17] allow to achieve better bounds.

4 An Identification Protocol Based on Kyber

In this section we analyze an instantiation of the identification protocol with $\mathsf{Kyber.CPAPKE}$ from various perspectives. First, we deduce the security of the identification protocol from the general result in Section 3. We then provide design rationales that we used as orientation to create an appropriately protected and lightweight lattice-based identification protocol. Finally, we describe an implementation on a Cortex-M4 32-bit microcontroller, and compare our identification protocol with various other constructions based on lattices, including a discussion on side-channel protection.

4.1 Security and Design Rationales

The security of the protocol is given in the corollary below, which is a direct consequence of Theorem 3.1.

Corollary 4.1. *Let Π be the identification protocol described in Fig. 1 instantiated with* $\mathsf{Kyber.CPAPKE}$*. Then, in the random oracle model, for any adversary $\mathcal{A}$ against Π, making q queries to the random oracles, there exists an adversary $\mathcal{C}$ against* $\mathsf{Kyber.CPAPKE}$ *such that*

$$\mathbf{Adv}_{\Pi}^{\mathsf{AS}}(\mathcal{A}) \leq 2\,\mathbf{Adv}_{\mathsf{Kyber.CPAPKE}}^{\text{IND-CPA}}(\mathcal{C}) \ + \frac{q}{2^{\alpha}}\,,$$

where the hash functions are modeled as random oracles.

Design Rationales. In what follows, we describe our approach with the view on highlighting the main design features. Specifically we compare the given ID protocol to the one constructed from CCA-secured encryption schemes when the CCA security is a result of the Fujisaki-Okamoto transform [23].

Indeed, given an encryption scheme, there is a simple construction of an identification protocol. In such a protocol, the verifier encrypts a random message, sends the ciphertext to the prover, the prover decrypts the ciphertext with the secret key and provides the message to the verifier. However, due to ciphertext malleability [21] an attacker could break such a scheme when it is based on common lattice-based CPA-secured KEMs and PKEs. During the probing phase, the honest prover acts as a decryption oracle for the adversary, as it decrypts any ciphertext it receives as a challenge. This entails that the used encryption scheme has to achieve CCA security for the protocol to be sufficiently secure; any scheme achieving only CPA security can be broken by performing a CCA attack against the underlying encryption during the probing phase of the protocol. The typical way of achieving CCA security is to design a CPA-secured encryption scheme and applying the FO transform to it. However, the FO transform adds overhead and—more importantly—is very hard to secure against side-channel attacks [4, 7, 43].

The fundamental idea of the FO transform is to avoid maliciously generated ciphertexts by re-encrypting the decrypted message and comparing it with the received ciphertext. The decrypted message is only outputted if the re-encryption results in the given ciphertext, otherwise, the ciphertext is rejected as an invalid one. The re-encryption procedure comes with a huge overload when used with lattice constructions. For example, it requires the sampling from a noise distribution, which is notoriously hard to secure against side-channels [13, 35, 40, 43, 45, 51].

Our approach mimics the idea to check that the challenge ciphertext is generated honestly. However, instead of using the decrypted message, we achieve this check independently of the secret key. In fact, an honest challenge in the present identification protocol is generated by means of a seed. This seed is provided to the prover only after the prover commits to its response by sending the hash value of its response. Then the seed can be used to check whether the ciphertext received after the first communication is indeed generated with the presented seed.

Note that the commitment to the response does not reveal any information about the secret unless either the hash function is broken, or the response computation leaked information. Thus, the CPA decryption still needs to be secured against side-channel attacks.

The benefit comes into play in the challenge verification step. As the computation uses the seed only and is independent of the secret key or any result of the response computation, the verification does not need to be secured against side channels. As will be discussed below (see Table 2), the challenge verification takes the greater computational costs of the prover, but in contrast to the CCA version, does not need to be side-channel secured. Also note that after the commitment in terms of the hash of the message with a random value, there is no need for the verifier to keep the message secret. Thus, the verifier can send the seed to the prover, who can check whether the challenge was generated honestly. This seed allows the prover to verify the challenge, without using the secret key or any values derived from the secret key.

The described benefits are achieved by adding a marginally larger communication cost given in terms of an additional hash value and the seed being transmitted in the intermediate steps.

Note that we are only interested in side-channel leakage on the prover side, which possesses a long-term secret. One could, of course, consider side-channel leakage on the verifier side, but the relevance is questionable. Assume that an

Table 1. Parameter sets and NIST security level for Kyber and the implementation of the identification protocol with Kyber.

Reference Kyber	Security	
	Bit	Level
Kyber512 $(k = 2, \eta_1 = 3, \eta_2 = 2)$	118	I
Kyber768 $(k = 3, \eta_1 = 2, \eta_2 = 2)$	183	III
Kyber1024 $(k = 4, \eta_1 = 2, \eta_2 = 2)$	256	V

adversary can obtain the challenge message m via some side-channel from the challenge computation. This would immediately allow to identify. However, it would only enable a single identification and be useless afterwards. It would also require to obtain this side-channel information from a single trace.

4.2 Parameter Sets

The instantiation of our identification protocol with Kyber.CPAPKE comes with the NIST security levels I, III, and V corresponding to the Kyber parameter sets Kyber512, Kyber768, and Kyber1024, see Table 1. All parameter sets share the common MLWE structure instantiated with $n = 256$ and $q = 3329$. For our security analysis we rely on the core-SVP classical hardness that is also used by Kyber [46] version 3.02.

4.3 Implementation

In general, performance measurements for common PQC schemes can be performed with a portable and easier to maintain implementation (e.g., pq-clean [32]) or an implementation that is optimized for the target platform (e.g., pqm4 [31]) and that uses assembly instructions or CPU-specific operations. We evaluate our implementation using both approaches on an ARM Cortex-M4 32-bit microcontroller and use ARM GCC version 6.3.1. Our target device is an STM32F407 that is mounted on the popular STM32F4-DISCOVERY board[3]. For the evaluation, we set the clock frequency to 24 MHz and do not use the maximum frequency of 168 MHz to reduce the impact of caches or delays caused by wait states stemming from the particular non-volatile memory (NVM) technology.

For key generation we use the Kyber.CPAPKE key generation as is. For challenge computation, response computation, challenge verification, and response verification we call the Kyber.CPAPKE routines from either pq-clean or pqm4 and also use the hashing routines provided by these libraries.

[3] The source code of our implementations is available at `https://github.com/tpoeppelmann/id_protocol`.

The security analysis of the protocol makes use of different, independent random oracles. For the implementation, we make use of SHAKE and instantiate the different random oracles via domain separation, using different prefixes. This was shown in [5] to provide a sound method of instantiating multiple random oracles from a single one.

To measure the cycle counts we rely on the system timer (SysTick) and confirmed that we obtain the same cycle counts for Kyber768.CPAPKE with our compiler and setup as given in [30] for Kyber768.CPAPKE. In Table 2, we provide measured cycle counts of our implementation for the cryptographic processing (cf. Section 3). Cycles for communication and protocol state handling are excluded as they are highly application specific and depend on the used interface (e.g., contactless, IC2, SPI, CAN).

Table 2. Cycle counts of our implementation on an ARM Cortex-M4 using either pq-clean [32] or pqm4 [31] using the m4fspeed implementation.

Function	Cycles (pqclean)	Cycles (pqm4; m4fspeed)
Key generation	927 412	607 652
Challenge Computation (verifier)	1 097 362	637 251
Response Computation (prover)	244 264	62 497
Challenge Verification (prover)	1 099 267	644 945
Response Verification (verifier)	42 089	38 569

In Table 3 we compare our implementation with standard constructions for the realization of identification protocols based on Kyber.CCA and Dilithium when using different implementations. For Kyber.CCA we assume that the verifier runs encapsulation while the prover runs decapsulation and then provides the encapsulated secret back to the verifier. For the Dilithium instance we assume that the verifier sends a random number (not accounted in cycle counts) and that the prover executes a signing operation and the verifier a signature verification. The average cycle counts for Kyber and Dilithium are obtained from [30] in October 2022. We also provide cycle counts for an insecure instantiation of CPA-secured Kyber768.CPAPKE as ID scheme. The large difference in cycles to the CCA-secured version Kyber.CCA shows the overhead attributed to the FO transform. Another important metric for an ID scheme is the amount of data that has to be transferred. For our approach it is required to transmit 1088 bytes for c, 32 bytes for r, 32 bytes for h and 32 bytes for r, which results in 1184 bytes. When Kyber.CCA is used as ID scheme, it requires $1088 + 32 = 1120$ bytes and Dilithium3 needs $32 + 3293 = 3325$ bytes.

Table 3. Comparison of cycle counts for cryptographic operations when excluding communication.

Library	Function	Cycles verifier	Cycles prover
pq-clean	Our protocol ARM Cortex-M4	1 139 451	1 343 531
	Kyber768.CCA as ID scheme	1 352 393	1 470 514
	Dilithium3 as ID scheme	3 499 388	11 722 059
	Kyber768.CPAKEM as ID scheme (*insecure*)	1 068 876	229 451
pqm4; m4fspeed	Our protocol ARM Cortex-M4	675 820	707 442
	Kyber768.CCA as ID scheme	869 974	795 161
	Dilithium3 as ID scheme	2 691 469	6 610 160
	Kyber768.CPAKEM as ID scheme (*insecure*)	611 076	49 021

4.4 Side-Channel Protection

Some implementations of identification schemes on embedded devices may require protection against physical attacks. For our protocol, we see the benefit that only the Response Computation by the prover is sensitive to side-channel attacks. This is a big advantage compared to KEMs that are using the FO transform where the decapsulation procedure is sensitive [42] and requires costly masking or other countermeasures [4,7,43]. The challenge verification routine is not sensitive as all inputs and the resulting ciphertexts c and c' are known by the prover and verifier. The only added operation on top of a masked Kyber.CPAPKE decryption is the masked computation of $h \leftarrow H(r, \tilde{m})$. This operation needs to be masked as well to prevent leakage of information on the decrypted message $\tilde{m}$. The value h itself is not critical anymore as it is randomized via r. Note that to obtain the cycle counts for the full computation of the prover, one has to add also the non-sensitive cycles for challenge verification.

As shown in Table 4, the overhead of a 1^{st}-order masked Kyber decryption (including masked FO transform) is already roughly a factor of 3 ($\approx 2\,200\,000$ cycles) but increases massively for second or higher orders protection. And it is important to note that a first order masked scheme is not sufficient in practice, as practical attacks have already been shown that exploit in particular properties of the FO transform. Such a scheme would at least need to be combined with hiding measures to counter known attacks.

In Table 4, we also provide measurements for an implementation of the Response Computation using the open-source first-order masked implementation of Kyber presented in [27]. In addition, we do performance estimations of our scheme based on results reported in [12]. Such an approach using an estimation is necessary as the source code of [12] is not available but sufficient to reach a general impression about the benefits of our proposal as we mainly call Kyber as a subroutine.

The Response Computation of the prover is a masked Kyber.CPAPKE decapsulation (indcpa_dec in [12]) and a masked hashing operation (e.g., hashg in [12]). Therefore, the 1st-order Response Computation of the prover is estimated to be roughly 174 000 + 118 000 + 62 497 = 354 497 cycles, which fits to the results obtained via [27]. A 2nd-order protection implementation can be estimated with 2 916 000 + 1 543 000 + 62 497 = 4 521 497 cycles. This is roughly 8.5 times better than the approach of using Kyber768.CCA as ID scheme with 2nd-order protection (when also accounting for the non-sensitive 644 945 cycles for challenge verification the prover has to perform in our approach as well).

For a fair comparison, Table 4 also provides the full cycle count of the Prover. For the identification protocol based on Kyber768.CCA, there is no difference to the cycle count of the response computation. This is because the validity is already been checked by the Kyber768.CCA decryption algorithm. For our protocol, the full cycle count of the prover consists of the cycle count for the (masked) response computation plus the fixed cycle count of 644 945 (cf. Table 1) for the challenge verification, which does not need to be protected.

Table 4. Comparison of cycle counts.

Masking	Scheme	Cycles (Prover)		Speedup
		Resp. Comp.	Full Comp.	
none	Our protocol	62 497	707 442	$\approx$ 1.12
	Kyber768.CCA as ID scheme	795 168	795 168	
1st-order [27]	Our protocol	241 887	886 832	$\approx$ 3.35
	Kyber768.CCA as ID scheme	2 978 441	2 978 441	
1st-order [12]	Our protocol	$\approx$ 354 497	$\approx$ 999 442	$\approx$ 3.12
	Kyber768.CCA as ID scheme	3 116 000	3 116 000	
2nd-order [12]	Our protocol	$\approx$ 4 521 497	$\approx$ 5 166 442	$\approx$ 8.58
	Kyber768.CCA as ID scheme	44 347 000	44 347 000	
3rd-order [12]	Our protocol	$\approx$ 12 009 497	$\approx$ 12 654 442	$\approx$ 9.12
	Kyber768.CCA as ID scheme	115 481 000	115 481 000	

Comparison of cycle counts for response computation and the full computation performed by the prover between our protocol and the identification protocol based on Kyber768.CCA. For our protocol, only the response computation, as shown in the Resp. Comp. column, is required to be secured against side channels. The full computation cycles result from the cycles for the response computation and the (non-sensitive) 644 945 cycles for the challenge verification. For Kyber768.CCA as ID scheme, the response calculation is equivalent to decryption; the full computation is the same as there is no separate challenge verification. The listed speedup is based on the cycle count for the full computation. The comparison is based on the implementation using pqm4; m4fspeed.

5 Conclusion

This article presents a novel lattice-based identification protocol using an interactive challenge-response protocol. It is lightweight, efficient, and simple to implement, making it well-suited for use in IoT devices, microcontrollers, and constrained devices. The protocol is designed in a way that supposedly allows easier protection against side-channel attacks than generic constructions using KEMs as it avoids rejection sampling and the FO transform.

It might be of interest to investigate possible variations of the proposed protocol that may be able to realize identity-based identification [20]. As lattice-based constructions allow identity-based encryption schemes as shown in [25, 50], a natural question is whether it is possible to extend the present scheme to develop an identity-based identification scheme.

Acknowledgements

We thank the anonymous reviewers for valuable comments. This work has been supported by the the German Ministry of Education and Research in the context of the project Aquorypt (grant number 16KIS1022 and 16KIS1020), the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297, and the Bavarian State Ministry of Science and the Arts in the framework of the bidt Graduate Center for Postdocs.

References

1. M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly-secure signatures from lossy identification schemes. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Heidelberg, Apr. 2012. 3
2. A. Ambainis, M. Hamburg, and D. Unruh. Quantum security proofs using semi-classical oracles. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, Aug. 2019. 9, 10
3. H. Anada and S. Arita. Identification schemes from key encapsulation mechanisms. In A. Nitaj and D. Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 59–76. Springer, Heidelberg, July 2011. 2, 3
4. M. Azouaoui, J. W. Bos, B. Fay, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal. Surviving the FO-calypse: Securing PQC implementations in practice, 2022. `https://iacr.org/submit/files/slides/2022/rwc/rwc2022/48/slides.pdf`. 2, 3, 11, 14
5. M. Bellare, H. Davis, and F. Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 3–32. Springer, Heidelberg, May 2020. 4, 13
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 4

7. S. Bhasin, J.-P. D'Anvers, D. Heinz, T. Pöppelmann, and M. Van Beirendonck. Attacking and defending masked polynomial comparison. *IACR TCHES*, 2021(3):334–359, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8977`. 2, 3, 11, 14
8. N. Bindel, M. Hamburg, K. Hövelmanns, A. Hülsing, and E. Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Heidelberg, Dec. 2019. 10
9. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, Dec. 2011. 9
10. A. Boorghany and R. Jalili. Implementation and comparison of lattice-based identification protocols on smart cards and microcontrollers. Cryptology ePrint Archive, Report 2014/078, 2014. `https://eprint.iacr.org/2014/078`. 3
11. A. Boorghany, S. B. Sarmadi, and R. Jalili. On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards. Cryptology ePrint Archive, Report 2014/514, 2014. `https://eprint.iacr.org/2014/514`. 3
12. J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal. Masking kyber: First- and higher-order implementations. *IACR TCHES*, 2021(4):173–214, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/9064`. 14, 15
13. L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In B. Gierlichs and A. Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 323–345. Springer, Heidelberg, Aug. 2016. 11
14. W. Castryck and T. Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975, 2022. `https://eprint.iacr.org/2022/975`. 3
15. W. P. Consortium. The QI authentication system, 2021. `https://www.wirelesspowerconsortium.com/qi-authentication/`. 2
16. L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Heidelberg, Dec. 2020. 3
17. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Online-extractability in the quantum random-oracle model. In *EUROCRYPT 2022, Part III*, LNCS, pages 677–706. Springer, Heidelberg, June 2022. 10
18. M. S. Dousti and R. Jalili. Efficient statistical zero-knowledge authentication protocols for smart cards secure against active & concurrent attacks. Cryptology ePrint Archive, Report 2013/709, 2013. `https://eprint.iacr.org/2013/709`. 3
19. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, Aug. 2013. 3
20. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987. 16
21. S. Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016. `https://eprint.iacr.org/2016/085`. 10

22. E. Fujisaki. New constructions of efficient simulation-sound commitments using encryption and their applications. In O. Dunkelman, editor, *CT-RSA 2012*, volume 7178 of *LNCS*, pages 136–155. Springer, Heidelberg, Feb. / Mar. 2012. 3
23. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, Aug. 1999. 10
24. S. D. Galbraith, C. Petit, and J. Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2017. 3
25. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. 16
26. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, Sept. 2012. 3
27. D. Heinz, M. J. Kannwischer, G. Land, T. Pöppelmann, P. Schwabe, and D. Sprenkels. First-order masked kyber on ARM cortex-M4. Cryptology ePrint Archive, Report 2022/058, 2022. `https://eprint.iacr.org/2022/058`. 14, 15
28. S. Heyse, E. Kiltz, V. Lyubashevsky, C. Paar, and K. Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In A. Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 346–365. Springer, Heidelberg, Mar. 2012. 4
29. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In B.-Y. Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 19–34. Springer, Heidelberg, Nov. / Dec. 2011. 3
30. M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. `https://github.com/mupq/pqm4`. 13
31. M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM cortex-M4. Cryptology ePrint Archive, Report 2019/844, 2019. `https://eprint.iacr.org/2019/844`. 12, 13
32. M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers. Improving software quality in cryptography standardization projects. In *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops*, pages 19–30. IEEE Computer Society, 2022. Cited for PQClean: Clean, portable, tested implementations of post-quantum cryptography, see `https://github.com/PQClean/PQClean`. 12, 13
33. J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. 4
34. E. Kiltz, K. Pietrzak, D. Cash, A. Jain, and D. Venturi. Efficient authentication from hard learning problems. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, Heidelberg, May 2011. 4
35. S. Kim and S. Hong. Single trace analysis on constant time CDT sampler and its countermeasure. *Applied Sciences*, 8(10):1809, 2018. 11
36. V. Kuchta, A. Sakzad, D. Stehlé, R. Steinfeld, and S. Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 703–728. Springer, Heidelberg, May 2020. 10

37. V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In R. Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 162–179. Springer, Heidelberg, Mar. 2008. 2, 3
38. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, Dec. 2009. 3
39. V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`. 3
40. S. Marzougui, N. Wisiol, P. Gersch, J. Krämer, and J.-P. Seifert. Machine-learning side-channel attacks on the GALACTICS constant-time implementation of BLISS. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–11, 2022. 11
41. V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque. Masking Dilithium - efficient implementation and side-channel evaluation. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 344–362. Springer, Heidelberg, June 2019. 3
42. T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu. Practical CCA2-secure masked Ring-LWE implementations. *IACR TCHES*, 2018(1):142–174, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/836`. 14
43. P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR TCHES*, 2020(3):307–335, 2020. `https://tches.iacr.org/index.php/TCHES/article/view/8592`. 2, 3, 11, 14
44. K. Sakumoto, T. Shirai, and H. Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 706–723. Springer, Heidelberg, Aug. 2011. 3
45. T. Schneider, C. Paglialonga, T. Oder, and T. Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. In D. Lin and K. Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 534–564. Springer, Heidelberg, Apr. 2019. 11
46. P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`. 12
47. J. Stern. A new identification scheme based on syndrome decoding. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, Aug. 1994. 3
48. E. E. Targhi and D. Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 192–216. Springer, Heidelberg, Oct. / Nov. 2016. 9
49. USB 3.0 Promoter Group. Universal serial bus security foundation specification, 2019. `https://www.usb.org/document-library/usb-authentication-specification-rev-10-ecn-and-errata-through-january-7-2019`. 2
50. M. Zhandry. Secure identity-based encryption in the quantum random oracle model. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, Heidelberg, Aug. 2012. 10, 16
51. R. K. Zhao, R. Steinfeld, and A. Sakzad. FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers. *IEEE Transactions on Computers*, 69(1):126–137, 2019. 11