Technische Universität Darmstadt
Department of Computer Science
Cryptography and Complexity Theory

Post Quantum Asynchronous Remote Key Generation

Master's Thesis by
Sebastian A. Clermont

Examiner: Prof. Dr. phil. nat. Marc Fischlin
Adviser: Dr. rer. nat. Jacqueline Brendel

Date of Submission: July 17, 2022

Thesis Statement pursuant to § 22 paragraph 7 and § 23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Sebastian A. Clermont, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§ 38 paragraph 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content.

Darmstadt, July 17, 2022

Sebastian A. Clermont

Abstract

Relying exclusively on passwords does not provide secure authentication. In the last ten years, efforts to replace passwords with more appropriate mechanisms have been made. These efforts culminated in the creation of the FIDO2 standard, which allows for secure passwordless authentication by using hardware authenticators.

However, there is currently no mechanism in place to recover access to online services after losing an authenticator, unless the user has registered multiple authenticators with each account. In a previous contribution, Frymann et al. tackled this problem by introducing a new cryptographic primitive called Asynchronous Remote Key Generation [FGK+20]. This primitive allows for pairing two authenticators, denoting one as the primary authenticator while the other becomes the backup authenticator. Should the primary authenticator be lost, the backup authenticator can then be used to recover all accounts where the primary authenticator is registered. The recovery can take place offline, meaning the backup authenticator can be stored in a secure location until it is needed. The security requirements public key unlinkability and key security for the scheme are also provided.

The same work also introduces an instantiation realizing the functionality of Asynchronous Remote Key Generation, along with security proofs, demonstrating the security of the proposed algorithms. However, this instantiation relies on elliptic curves, resulting in a scheme that becomes insecure in a post-quantum setting.

This thesis introduces a different instantiation of the same cryptographic scheme, this time providing post-quantum security. To achieve this, we have used Key Encapsulation Mechanisms (KEMs) in combination with digital signatures as our main building blocks. For both, there are post-quantum instantiations available, with an official standardization of these instantiations expected by 2024.

In addition to the new instantiation, we propose modifications to the security games introduced by Frymann et al. The modified security games reflect the real-world threats to which the scheme is exposed more closely. We show that our post-quantum instantiation of the primitive is provably secure these security requirements. Our instantiation is a suitable drop-in replacement, providing secure recovery of accounts even after the advent of quantum computers.

Contents

1 | Introduction

Secure authentication is a fundamental requirement for securely using personalized online services. Nowadays, one can manage all aspects of personal and professional life online. From working over banking up to shopping. For each service, a user account is needed. The account's role is to link the user's real-world identity with his only identity for this service. Forging an authentication allows an attacker to assume a user's identity and to act in its name. When everything can be done online, the potential damage from a compromised digital identity is monumental. Nonetheless, people are not adhering to safe password practices, even if the password is the only form of authentication for their account.

A recent study by IBM revealed that almost half of all global respondents value convenience more than security or privacy [IBM21]. When given the choice, they would rather use an online service than go to a physical location for the same service, even if they had concerns about the security or privacy of the online service.

This desire for convenience also manifests in multiple other online behavior aspects. Over 80% of users habitually reuse passwords for multiple online services. 60% of the respondents will reconsider creating an account with an online service if the registration process takes more than five minutes.

Under these circumstances, it is challenging to promote safe online behavior successfully. To tackle the challenge of online authentication, the Fast Identity Online (FIDO) Alliance has created multiple sets of standards for a more convenient and secure online authentication experience.

FIDO U2F provides an open interface for using a second factor during authentication [SDE+17]. U2F allows for a consistent experience across all services implementing the standard, making users more likely to activate the second factor. The next iteration, FIDO2, improves these ideas and allows for a completely passwordless login experience [CAJ+19; JJM+21]. This is realized by employing an authenticator token. It increases security by relying on randomly generated asymmetric keys stored on the authenticator. Additional measures in the protocol design also provide resistance against phishing and social engineering, which cause yearly damage of over $120 billion per year in the United States alone [SK19]. The authenticator tokens can either be implemented as a dedicated hardware token or in software. Examples of hardware implementations include the various offerings by Yubico[1] and SoloKeys[2]. For software implementations, IDmelon [IDm] is one example that works on both Android and iOS. The FIDO Alliance certifies authenticators that adhere to the standard. A certified FIDO2 authenticator guarantees conformance and interoperability with the FIDO2 standard. When choosing an authenticator, it is highly recommended to choose an authenticator with official certification.

The functionality offered by the FIDO2 specifications is unique in that it improves both security as well as convenience. To leverage this enormous potential, Google, Apple, and Microsoft have recently announced they are expanding support for passwordless sign-in using the FIDO2 standard [Gla22a]. Consequently, we expect to see a steady increase in the usage of FIDO2 compliant authenticators for convenient and secure online authentication. For a smooth mainstream adoption of FIDO2, it is imperative to avoid any security mishaps that cast doubt on the scheme's security. A formal security analysis is required to prove the security of FIDO2 and its two components, CTAP and WebAuthn. Such an analysis was carried out by Barbosa et al. in 2020 [BBC+21] and identified several shortcomings, for which stronger protocols were proposed. There are also other works analyzing the WebAuthn protocol [GH18] and the previous protocol version, FIDO U2F [JK21; PMN+17].

This thesis does not directly cover the security of FIDO2 but a related issue: Account recovery in the case of authenticator loss. With the authenticator facilitating a passwordless login, a user is completely locked out of his account in the event of authenticator loss, unless previous arrangements for recovery have been made. The current recommendation by the FIDO Alliance is to register two independent authenticator devices with each service. If one authenticator is lost, the user can then use the second authenticator to access his account. To provide a more convenient scheme without compromising security, Yubico has come up with an extension for WebAuthn [Lun20]. The scheme allows two FIDO2 compliant authenticators are linked to become primary authenticator and backup authenticator. The primary authenticator is then used normally, while the backup authenticator can be stored in a safe location. In the event of authenticator loss, the backup authenticator can recover access to all accounts a user has set up after the initial pairing interaction.

[1] `yubico.com` [04.07.2022]
[2] `solokeys.com` [04.07.2022]

This scheme was then formalized by Frymann et al. in [FGK+20] and given the name Asynchronous Remote Key Generation (ARKG). The formalization is split into two parts. First, they create a new cryptographic primitive for capturing the functionality provided by Asynchronous Remote Key Generation and define corresponding security properties. Then, they interpret the protocol proposed by Yubico as one instantiation of this primitive, which is then used in the security proofs.

In this thesis, we will first look at other related schemes to increase the security of online authentication, such as the time-based one-time-pad, before introducing FIDO2 in Section 2. We will then look at the primitive for ARKG and the security requirements proposed by Frymann et al. in Section 5. Next, the original instantiation as proposed by Yubico will be introduced. With all prerequisites in place, we will cover the instantiation proposed by Yubico and Frymann et al.

We will then move on to our proposed post-quantum instantiation, which is presented in Section 6. This Section also includes new security games, which model the real world threats more closely. This thesis will conclude with the security proofs of our instantiation for the updated security notions.

2 | Related Work

This section will provide an overview of previous approaches for realizing multifactor authentication as well as the standards U2F and FIDO2 published by the FIDO Alliance. It will also examine the impact of quantum computers on today's cryptography and give insights into the currently ongoing standardization process for post-quantum cryptographic schemes.

2.1 Multifactor Authentication

There is broad consensus on the necessity of multifactor authentication for proper authentication, but no single approach has yet been singled out as the go-to solution. Available technologies range from a simple 8-digit number, communicated over an insecure channel, all the way to truly passwordless communication with hardware tokens using public key cryptography to authenticate users.

One of the most recent drivers for the adoption of multifactor authentication (MFA) is the Payment Service Directive 2 (PSD2)[Deu19], mandated by the European Union (EU). It mandates strong authentication of customers before they are allowed to authorize any online transaction. Strong authentication is defined as follows:

Definition 2.1 (Strong Authentication in PSD2)**.** *For a user to achieve strong authentication, it has to present security features from at least two of the three domains*

- *Knowledge, something the user knows, such as a password*
- *Possession, something the user owns, such as a hardware token*
- *Inherence, something the user is, for having a certain fingerprint*

For example, scanning a fingerprint on a device the user has linked to his account would prove possession and inherence and therefore constitute a strong authentication.

Before we dive into modern authentication, we will have a look at the history of securing online authentication using additional factors.

2.1.1 Traditional Approaches

The first implementations of MFA have relied on some insecure channel to transmit an independent second factor. Popular choices for a secondary channel were a pager, highlighted in the original patent by AT&T [BGM+96], as well as text messages and emails. While most people no longer own a pager, sending a verification code via text message or email is still commonplace. For some services, this is done to cater to customers that do not own any specialized hardware or are unwilling to set up the corresponding application on their phone, while others are simply behind in modernizing their authentication process.

While being more secure than not using a second factor at all, attacks such as sim swapping [Yub22] can be used to compromise the second channel, thus completely circumventing the security measure.

2.1.2 HOTP and TOTP

HMAC-based one-time password (HOTP) and time-based one-time password (TOTP) are two more advanced mechanisms to strengthen authentication and represent the next evolutionary step in the history of MFA solutions. HOTP and TOTP rely on six to eight-digit numbers that a user has to obtain from an authenticator and manually type in after providing his password. These numbers are generated by hardware tokens or special authenticator apps, making the attacks previously discussed inapplicable. HOTP is the older of the two algorithms and has been standardized in RFC 4226 [VMH+05]. The most important building block is the HMAC function, which is a MAC function based on hash functions [KBC97]. HOTP relies on static symmetric keys established between the two parties during the pairing phase as well as a strictly increasing counter. Let k be some key and c some counter value, then HOTP can be defined as

$$\text{HOTP}(k, c) = \text{truncate}(\text{HMAC}(k, c))$$

The dynamic truncate operation reduces the output of the HMAC to six to eight digits, depending on the configuration, which the user can supply to the application.

The counter value is used to avoid collisions in the hash function's input. On the server, the counter is incremented with each successful authentication, while the client increases its counter each time a new value is generated. This leads to a divergence between the two counters. To counteract this divergence, a lookahead window is implemented on the server. Depending on a parameter set by the administrator, the server precomputes the next few HOTP values and compares them to the user input. If there is a match on any of the values, the server can validate the authentication and resynchronize its counter.

Time-based one-time passwords extend the idea of HOTP, solving the issue of resynchronizing the counter. Instead of relying on a counter, it uses the current time as the second input. TOTP is standardized by RFC 6238 [VRP+11]. With HOTP as defined above, TOTP can be defined as

$$\mathrm{TOTP}(k) = \mathrm{HOTP}(k, T)$$

where T is an integer and represents the number of time steps between the current time and some starting time T0, usually the Unix epoch. A time step is a time interval of a fixed length, 30 seconds by default, after which T is incremented. Client and server can now compute T independently and non-interactively. When using TOTP, a server no longer needs to employ a lookahead window to align counters with the client.

While TOTP is the more modern approach, it cannot be deployed in scenarios where the client has no access to a clock. This might be the case for embedded devices with hardware limitations. Both HOTP and TOTP can be implemented in mobile apps, such as Google Authenticator [Goo22b], or hardware authenticators, such as Yubikey [Yubb]. Hardware authenticators can store symmetric secrets in their secure element, offering additional security. While they do not have a built-in clock, they obtain the current time from a phone or computer before outputting the one-time password.

2.1.3 Modern Authentication with FIDO

The age of modern authentication on the web began in 2013 with the founding of the Fast Identity Online Alliance, or FIDO Alliance for short. Its mission is the development of open-source industry standards for secure and easy-to-use online authentication. Since its inception, the FIDO Alliance has standardized several authentication schemes that enjoy broad adaption in modern IT products.

First, we introduce some definitions from the FIDO Alliance's glossary [LDB15].

- **Relying party (RP)** A website or other entity that uses a FIDO protocol to directly authenticate users
- **(FIDO) Authenticator** An authentication entity that meets the FIDO Alliance's requirements and which has related metadata. A FIDO Authenticator is responsible for user verification, and maintaining the cryptographic material required for authenticating with the relying party (RP).

FIDO Universal 2nd Factor (U2F) The FIDO Alliance started by addressing two factor authentication (2FA) by creating a universal standard called FIDO Universal 2nd Factor (U2F) [SDE+17]. In a 2FA setting, a user is asked to provide a second after successfully using his regular password. FIDO U2F defines an open standard that can be implemented on security tokens to act as a second factor.

Unlike HOTP and TOTP, the process is mostly automated and requires little manual input from the user. The U2F token and the browser communicate directly via the Client to Authenticator Protocol (CTAP) protocol. The user is only required to demonstrate presence, for example by touching the authenticator to prevent accidental use.

FIDO2 FIDO U2F has been expanded and refined to become a new standard, FIDO2 [CAJ+19; JJM+21]. Unlike the previous iteration, which could only serve as a second factor, a FIDO2 compliant authenticator allows for a truly passwordless login. This means that FIDO2 compliant devices enable a user to securely authenticate himself without using any password at all. The entire authentication process is built around the authenticator, which performs all the heavy lifting. A RP can choose how strongly it wants its users to be authenticated. This setting is called user verification and determines if possession of an authenticator is sufficient or if the authenticator has to be unlocked by entering a PIN or providing some biometric feature. Only when user verification is required, a user is forced to provide some second factor. Otherwise, only the possession of the authenticator is sufficient for a successful login. To achieve strong authentication, as defined by the PSD2, the RP must require user verification.

The FIDO2 standard itself is comprised of a set of specifications for secure authentication in various web-based contexts. The two core specifications are the *CTAP* and *WebAuthn* protocols. Both are characterized by being platform agnostic, allowing developers to implement them on a wide range of platforms. An authenticator can be implemented as a hardware token, which is connected via USB, but also as a mobile phone with a secure element, which is connected via Bluetooth. Figure 1 illustrates the relationship between CTAP and WebAuthn, as well as their relation to FIDO2.

Client To Authenticator Protocol (CTAP) CTAP defines the communication of an external authenticator and some platform that is communicating with the RP [CAJ+19]. In a setting where a laptop communicates with some website, the website is the RP and interacts with the platform, in this case the browser, via WebAuthn, while the laptop communicates with the external authenticator using CTAP. Currently, FIDO2 supports the authenticator to communicate with its platform via USB, Near Field Communication (NFC), and Bluetooth Low Energy (BLE). The CTAP standard is published and maintained by the FIDO alliance. There exist two versions of the CTAP standard, CTAP1 for communicating with U2F devices and CTAP2 for FIDO2 compliant authenticators. The full specification can be found on the website of the FIDO Alliance [CAJ+19].

CTAP itself is split into three parts: An authenticator API, message encoding and transport-specific binding.

The *Authenticator API* provides a certain degree of abstraction, defining interfaces that need to be implemented by each platform. The implementation depends on the platform and has to be realized by the developer of an authenticator.

Message encoding defines the format of messages exchanged between platform and authenticator via the aforementioned API. It mandates the encoding to be Concise Binary Object Representation (CBOR) and defines command codes, status codes, and a few utility functions.

The *Transport-specific binding* section defines the low-level communication for all possible communication interfaces. As stated previously, this currently supports USB, NFC, and BLE.

WebAuthn WebAuthn defines standardized interfaces for the communication between some client and a RP with the goal of authenticating a user with the RP [JJM+21]. It defines an API for "the creation and use of strong, attested, scoped, public key-based credentials by web applications, for the purpose of strongly authenticating users". The WebAuthn specification [JJM+21] is, unlike the CTAP specification, published by the W3C. The purpose of WebAuthn is to provide a stable interface for the communication between a RP and some client, abstracting away from the underlying platform of the client. Currently, all major browsers, as well as Android and Windows 10 implement the WebAuthn standard [Shi].

To fulfill its purpose, the WebAuthn specification defines API interfaces to handle all aspects of a credential's lifecycle, from registration over authentication up to decommissioning at the end of an authenticators lifecycle.

There are also some security requirements imposed by the WebAuthn specification. Most notably, the API may only be used in "secure contexts". Secure contexts have their own definition [Wes21], but for the scope of this thesis it is sufficient for them to be considered equivalent to TLS.

Consequently, the security properties of TLS [Res18] have been met before any communication between an authenticator device and a relying party takes place. This includes peer authentication, confidentiality, and integrity of the exchange.

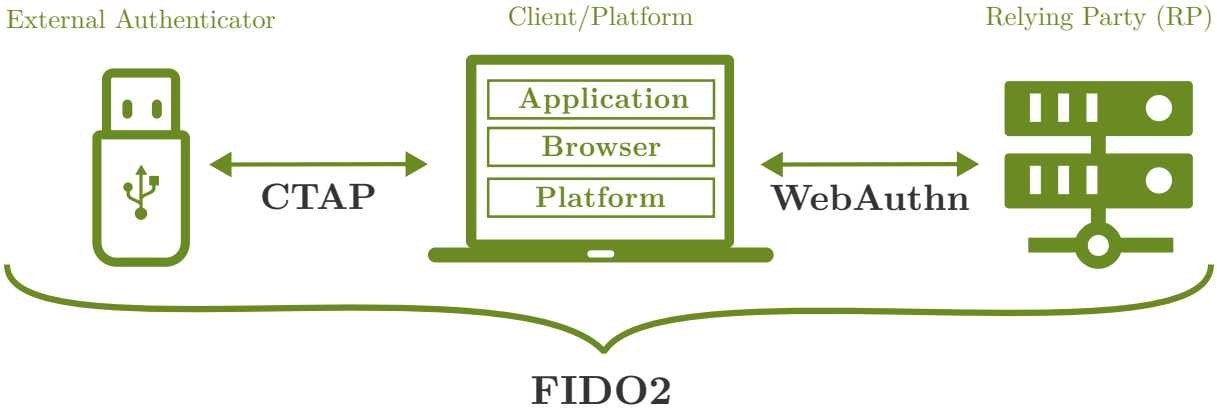


Figure 1: Relation between the FIDO2 specification and its two components, WebAuthn and CTAP [Sin19]

Authenticating with FIDO2 We will now explain the steps a single FIDO2 authentication process is composed of. The paragraph's numbering corresponds to the visualization steps in Figure 2.

1. After receiving a login request, the RP sends a challenge back to the client. The challenge consists of at least 16 bytes of randomness, which were generated by the server.
2. Next, the platform contacts its authenticator using the CTAP protocol to obtain a signature on the challenge. The authenticator is sent the relying party id (rpId) and a hash of the data to be signed.
3. Should user verification be required, the authenticator prompts the user to be unlocked. A user can unlock its authenticator using biometrics or a pin. Should user verification not be requested, a touch is required to demonstrate presence and prevent accidental use. Once unlocked, the authenticator retrieves the key pair for the provided rpId and creates the requested signature.
4. The data generated by the authenticator is sent back to the platform.
5. The browser processes and enriches the response from the authenticator and sends the result back to the RP.
6. Lastly, the server validates the response. This includes verifying the correct challenge was signed by the authenticator, the validity of the signature, and checking that the RP is the intended recipient of the response.

If all of the above steps have been successfully completed, a passwordless authentication with FIDO2 has been accomplished.

2.2 Hardware Authenticators

Using hardware devices for sensitive operations is common practice to achieve an enhanced level of security. This includes but is not limited to the domain of authentication. For example, Hardware Security Modules (HSMs) are used to securely manage cryptographic keys for larger organizations [BB19] and in the domain of cryptocurrency hardware wallets, such as Trezor [Tre22], are recommended for the secure long-term storage of cryptographic assets.

The primary advantage of cryptographic devices is their enhanced security. They are purpose-built to exactly fulfill their intended purpose, without any additional bells and whistles. Integration is made easy, as the devices perfectly adhere to the relevant standard, while the limited functionality reduces complexity and subsequently drastically reduces the attack surface. The standard interface provided by these devices also often protects users from themselves. Secret keys are usually securely generated on the device and remain there for their entire lifecycle. There are interfaces available for all operations one might want to perform with the secret key, but there is no interface for extracting the key.

Security of the internal secrets is not only protected by not providing any interface for extracting them, but also by additional measures in the hardware design. For normal storage devices, it is possible to desolder individual chips and extract data using special hardware. Cryptographic hardware has additional

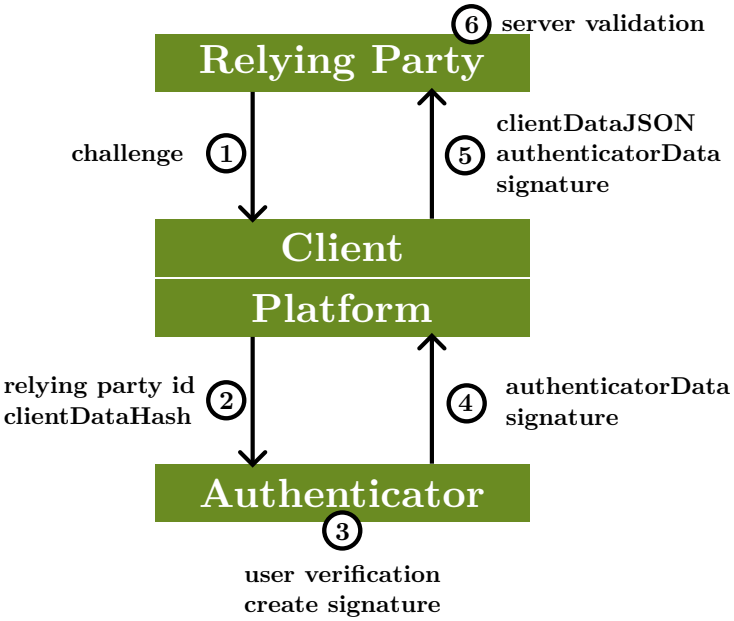


Figure 2: A diagram of the interactions taking place when authenticating with FIDO2

safeguards in place against such attacks. This usually involves, but is not limited to, making it impossible to disassemble the device without destroying the data storage. We call this property tamper-resistance.

It is reasonable to assume that secrets stored on a cryptographic device cannot be extracted by an adversary. In the context of FIDO2, this assumption has previously been made by [BBC+21], this thesis will adopt the same assumption and consider FIDO2 hardware authenticators to be tamper-proof.

2.3 Impact of Quantum Computers on Cryptography

The discussion on addressing adversaries with access to local quantum computing power was initiated in the late 1990s by the publication of two milestone papers.

2.3.1 Grover's Algorithm

In 1996, Grover published a paper describing a quantum search algorithm, which can locate a single item in an unstructured database with N entries using only $\mathcal{O}\left(\sqrt{N}\right)$ lookup operations [Gro96]. This is remarkable, as on a classical computer the most efficient approach would be randomly probing values. On average, such an algorithm finds a specific value after $N/2$ lookup operations, resulting in a much higher complexity of $\mathcal{O}(N)$ lookup operations.

For symmetric cryptography, this unstructured search can be interpreted as a brute force attack on some symmetric security scheme. In a setting using 128 bit symmetric keys, the size of the key space to be searched is 2^{128}. Randomly testing keys on a classical computer would take $\frac{2^{128}}{2} = 2^{127}$ attempts on average. Grover's algorithm can complete the same search in just $\sqrt{2^{128}} = 2^{64}$ operations. To achieve the original security level against an adversary with local quantum computing power, key lengths of symmetric schemes have to be doubled. In the example, a 256 bit key yields the original 128 bits of security against a quantum adversary: $\sqrt{2^{256}} = 2^{128}$.

2.3.2 Shor's Algorithm

Even more devastating for today's most prevalent cryptographic algorithms is the quantum algorithm described by Shor, published one year later in 1997 [Sho97]. In this paper, Shor describes a quantum algorithm for efficiently solving the period finding problem.

Unfortunately, both the discrete logarithm problem as well as efficient factoring of large integers can be formulated as a period finding problem. As a result, both hardness assumptions, on which all classical instantiations rely, are no longer hard in the presence of a quantum adversary.

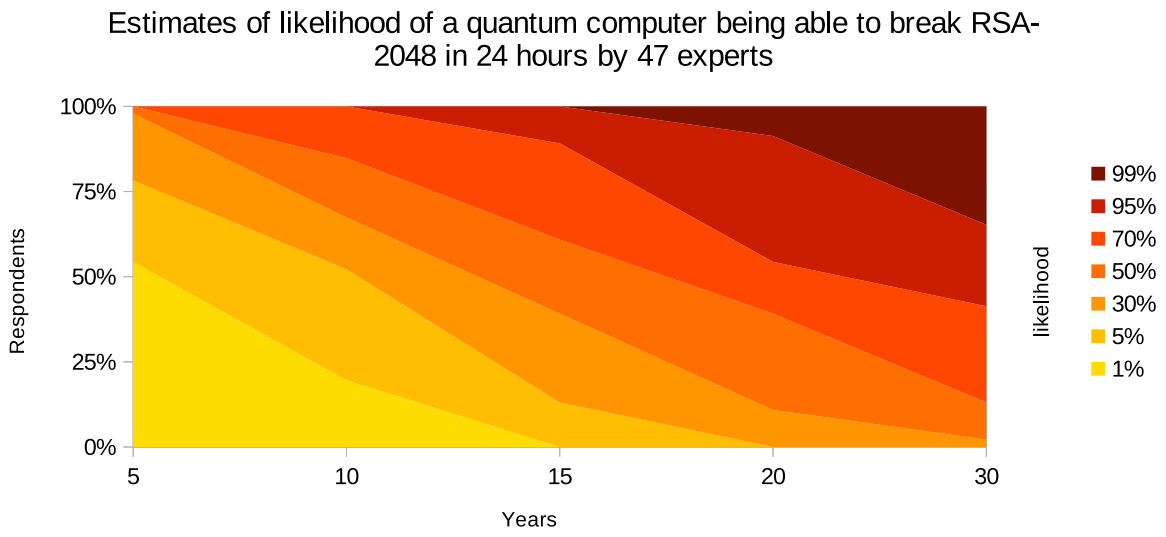


Figure 3: Results of expert survey on the likelihood of quantum computers breaking RSA-2048 in 24 hours within the next 30 years [Pia21]

Both, key agreement using Diffie-Hellman-style protocols, as well as all RSA-based signature schemes can be broken efficiently by an adversary with sufficient quantum computing power.

2.3.3 Expected Timeline for the Development of Quantum Computers

While at the time of their publication in 1996 and 1997 quantum computers were a rather theoretical construct, nowadays, real-world quantum computers have been built by a few of the world's leading technology companies, such as IBM [IBM15] and Google [Goo22c]. In 2019, Google demonstrated its quantum computer to be many orders of magnitude faster than a classical computer in sampling the output of a pseudo-random quantum circuit [AAB+19]. While this is certainly a great achievement, it should be pointed out that the problem was chosen specifically to cater to the strengths of a quantum computer.

To break RSA-2048, a recent estimate states a requirement of 20 million noisy qubits and eight hours of computation [GE21]. At the moment, a quantum computer with that many qubits seems like something from a distant future. Currently, IBM's largest quantum computer has just 127 qubits [IBM22]. By the end of 2023, IBM expects this number to increase an entire order of magnitude to 1121 qubits [IBM15].

When to expect a quantum computer with sufficient capabilities to break a widespread key length has been the subject of many discussions ever since the publications by Grover and Shor. As there is no way to precisely predict when this will happen, the best available guess is a survey of some of the world's leading cryptographers [Pia21]. The result of this survey can be seen in Figure 3. More than half of the surveyed experts estimate the likelihood of a quantum computer being able to break RSA-2048 within 24 hours to exist within the next 15 years at 50% or higher. When looking at a 30-year timeframe, the majority of respondents estimated the likelihood of such a quantum computer to exist at 95% or higher.

2.4 Post-Quantum Cryptography

As discussed in the previous sections, quantum computers with enough capacity to break classical hardness assumptions will likely be available within the next few decades. As the impact would be devastating on the current security infrastructure, it is imperative to explore measures to facilitate secure online communication in a world where our classical hardness assumptions no longer hold. This research is currently taking place and is labeled Post-Quantum Cryptography (PQC).

The term PQC refers to cryptographic algorithms that can be implemented on classical computers and are resistant to attackers with quantum computing power. They are also referred to as "quantum-safe" or "quantum-resistant".

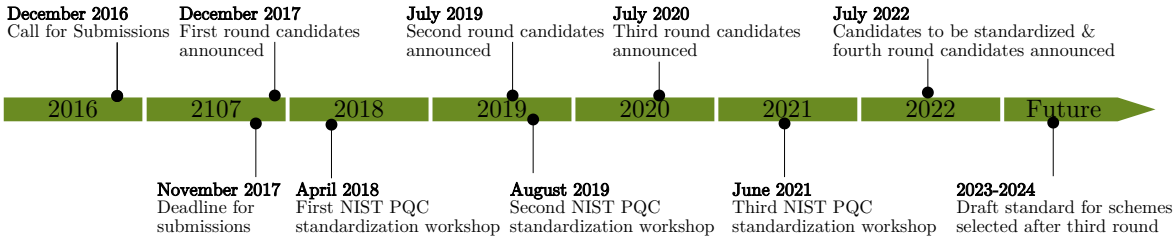


Figure 4: Timeline of the NIST PQC standardization [JMM+22]

A few new potentially hard problems have been identified over the years that could replace classical hardness assumptions. These new problems are the foundation for the construction of PQC schemes, which rely on one or more of these new hardness assumptions. They can be classified into five distinct categories [CJL+16]: Lattice-based, Code-based, Multivariate, Isogeny-based and Hash-based.

2.5 Standardizing Post Quantum Cryptographic Algorithms

The leading body for the standardization of cryptographic schemes is the National Institute of Standards and Technology (NIST), which is an agency of the United States of America. For the past 50 years, it has published standards for secure storage and transmission of data, from the Data Encryption Standard (DES) in 1977 [Nat99] over AES is 2001 [DBN+01] up to elliptic curve cryptography in 2018 [BCR+18].

With the advent of quantum computing, NIST has started another standardization process to identify cryptographic schemes for a future where neither calculating discrete logarithm nor the factorization of large numbers is hard [CML17]. The objective of this process is the standardization of new cryptographic algorithms that remain secure against adversaries with quantum computing power for the following two functionalities:

- Public-key encryption and key establishment algorithms
- Digital signature algorithms

Note that public-key encryption schemes and key-establishment algorithms are closely related. The Fujisaki-Okamoto transformation (FO transformation) [FO99] is a generic transformation that can be used to transform a public-key encryption scheme and a symmetric encryption scheme into an IND-CCA secure hybrid encryption scheme. This means that any post-quantum public-key encryption scheme can be combined with some symmetric scheme, which are not as heavily affected by quantum computers, to obtain a post-quantum key establishment algorithm. In fact, the only Key Encapsulation Mechanism (KEM) scheme selected to be standardized after the third round, CRYSTALS-Kyber [BDK+18], is constructed using this very idea.

Unlike previous standardization efforts by NIST, such as SHA-3 [Tec15b] and AES [DBN+01], where only one scheme was selected and standardized, the goal for the PQC schemes is to provide multiple secure standards for each functionality. The standardization process is designed to be as inclusive as possible, giving everyone the chance to submit proposals or comments about suggested candidates. Ultimately, the goal is to ensure a thorough analysis of each candidate and to build trust with the community on the quality of the schemes that will ultimately be selected.

There are multiple rounds in the standardization process. The objective is to give the scientific community the chance to thoroughly analyze suggested schemes in multiple iterations. At the end of each round, the maintainers of the scheme are allowed to introduce tweaks to their submissions, such as modifying parameter sets. A complete timeline of the process can be seen in Figure 4.

As with classical schemes, the confidence of the community in the security of the proposed schemes plays an important role. For the discrete logarithm and factorization, one can have very high confidence that these are actually hard problems. The new PQC algorithms have yet to stand the test of time. As is to be expected with new schemes that have not yet been standardized, some surprising weaknesses will be found. For example, Beullens published an attack breaking the Rainbow signature scheme [DS05] within a few days with just a laptop [Beu22].

With an increasing focus of the scientific community on Post-Quantum Cryptography, these proposed schemes can hopefully demonstrate their resilience to researchers' attacks and thus increase the overall confidence level. As of July 2022, the third round has just concluded with the announcement of schemes to be standardized, as well as candidates for a fourth round [Moo22]. For key encapsulation, there is one scheme that will be standardized with four more that will be further refined with the goal of later standardization. For digital signatures, three schemes have been selected for standardization. The current timeline expects the publication of the completed standards by 2024.

3 | Preliminaries

This chapter will introduce the cryptographic building blocks, hardness assumptions, and notation used throughout this thesis.

3.1 Definitions and Notation

First, we fix some general notation.

Let λ denote the security parameter for all security notions. To indicate equality, we use the symbol $=$. For definition, the symbol $:=$ is used. Concatenation of two elements x, y is written as $x \| y$. Failure events are represented using the symbol $\perp$, for example, to indicate invalid decryption of some ciphertext.

For assignment, we use the symbol $\leftarrow$. $x \leftarrow y$ assigns the value of variable y to variable x. The output of algorithms, such as adversaries, is denoted in the same way. If the algorithm $\mathcal{A}$ is deterministic, we denote the output using $\leftarrow$. $b \leftarrow \mathcal{A}$ denotes an algorithm $\mathcal{A}$ deterministically outputting a value b.

For a probabilistic adversary, we use $\leftarrow\!\!\$$ and write $b \leftarrow\!\!\$ \mathcal{A}$. To denote an algorithm $\mathcal{A}$ operating on some input x we write $\mathcal{A}(x)$. We also use $\leftarrow\!\!\$$ to denote random sampling from some set or distribution. The term $b \leftarrow\!\!\$ \{0,1\}$ describes the variable b being randomly assigned a value of either 0 or 1.

We use a semicolon to make randomness explicit in a probabilistic algorithm. If any parameters are listed after the semicolon, such as $\mathcal{A}(x; y)$, we can interpret y as the randomness used by $\mathcal{A}$. The output becomes deterministic, and multiple evaluations subsequently yield the same output.

Oracles O can have some secret parameters, which are fixed throughout the game, and some mutable parameters, which can be chosen by the algorithm for each execution. Immutable parameters are listed as parameters of the oracle, while mutable parameters are indicated by "$\cdot$", followed by a description of the expected input parameter. An oracle O with the immutable input k and mutable input x is subsequently written as "$\mathsf{O}(k, \cdot)$ on input x". Subsequent usage of this oracle can be abbreviated to $\mathsf{O}(k, \cdot)$ or even just O for brevity.

Should an algorithm have access to one or more oracles O, we list the oracles in superscript as $\mathcal{A}^{\mathsf{O}}$. Adversaries can have an arbitrary number of oracles and inputs, which will be denoted in superscript and parentheses, respectively. $\mathcal{A}^{\mathsf{O}_1, \mathsf{O}_2}(x, y)$ denotes an algorithm with access to oracle O_1 and O_2 and two input parameters x and y.

Unless explicitly stated otherwise, all algorithms in this thesis run in quantum polynomial time and have access to some randomness, allowing them to behave probabilistically. To highlight this fact, we sometimes write "QPT algorithm $\mathcal{A}$". However, just writing $\mathcal{A}$ implies the QPT property.

For message space, key space, ciphertext space, and key distribution we use the symbols $\mathbb{M}, \mathbb{K}$, $\mathbb{C}$, and $\mathbb{D}$, respectively.

Boolean evaluations are denoted using $[\![$ and $]\!]$. As an example, to obtain a boolean value for the equality of b and b', we write $[\![b = b']\!]$.

For security games we write $\mathsf{Exp}^{\mathsf{prop}}_{\sqcap, \mathcal{A}}(\lambda)$ to describe the security property prop of some cryptographic scheme $\sqcap$ against an adversary $\mathcal{A}$ under the security parameter λ. The event where the adversary wins the security game is denoted as $\mathsf{Exp}^{\mathsf{prop}}_{\sqcap, \mathcal{A}}(\lambda) = 1$. The probability for some event to occur is expressed as $\Pr\left[\mathsf{Exp}^{\mathsf{prop}}_{\sqcap, \mathcal{A}}(\lambda) = 1\right]$, in this case, to denote the probability of the adversary winning the security game. To obtain a more compact notation, we can directly state the adversaries advantage as $\mathsf{Adv}^{\mathrm{prop}}_{\sqcap, \mathcal{A}}(\lambda)$.

Some security games require a challenger to distinguish between interacting with a random instantiation or the scheme's real instantiation. Similarly, for some security properties, such as PRF security, the challenger has to distinguish between a truly random function and an actual PRF. Which instantiation the challenger is interacting with is denoted by some bit b. For consistency, we use $b = 0$ when arguing about the ideal or random instantiation, and $b = 1$ to denote interaction with the real instantiation.

Should multiple schemes provide the same function, such as KEM and Sig both having a KGen algorithm, we write KEM.KGen or Sig.KGen to clarify which scheme we are referring to.

All algorithms are always implicitly provided the security parameter λ as an input. We omit explicitly giving the security parameter for every execution of an algorithm.

3.2 Negligible Functions

A negligible function is a function that, from a specific point N onwards, is always smaller than the inverse of any polynomial. In cryptography, negligible functions serve as an upper bound for any advantage an unsuccessful adversary is allowed to have while still being considered unsuccessful. For an adversary to be

$\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$

1 : $MacList \leftarrow \emptyset$
2 : $k \leftarrow\!\!\$\ KGen(1^\lambda)$
3 : $(m^*, \tau^*) \leftarrow \mathcal{A}^{\mathsf{O}_{tag}(k,\cdot)}$
4 : **return** $[\![\text{Vrfy}(k, m^*, \tau^*) \wedge m^* \notin MacList]\!]$

$\mathsf{O}_{tag}(k, \cdot)$ on input m

1 : $\tau \leftarrow\!\!\$\ \text{Mac}(k, m)$
2 : $MacList \leftarrow MacList \cup m$
3 : **return** τ

Figure 5: Game definition for existential unforgeability under chosen message attacks on MAC schemes

successful, it is required for it to achieve a non-negligible advantage over randomly guessing. Formally, a negligible function is defined as

Definition 3.1 (Negligible Functions)**.** *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible, if for every polynomial $p : \mathbb{N} \to \mathbb{R}^+$ there exists an $N \in \mathbb{N}$, such that for all $n \geq N$ we have $\epsilon(n) \leq \frac{1}{p(n)}$.*

Note that the sum of two negligible functions is still negligible and that subtracting a negligible function from a non-negligible function still yields a non-negligible function. To indicate that something is negligible in the security parameter λ we write $\mathsf{negl}(\lambda)$.

3.3 Building Blocks

Definitions of the primitives are heavily influenced by [KL14].

3.3.1 Message Authentication Codes

A Message Authentication Code (MAC) is a symmetrical scheme to ensure the integrity of messages transmitted over an unsecure channel. A Message Authentication Code (MAC) scheme consists of three algorithms $\mathsf{MAC} = (\text{KGen}, \text{Mac}, \text{Vrfy})$.

- **KGen** KGen is used to generate keys for the MAC scheme. It takes the security parameter λ as input and outputs a key k, which is sampled from some key space $\mathbb{K}$.

$$k \leftarrow\!\!\$\ \text{KGen}(1^\lambda)$$

- **Mac** Mac takes as input a message m from some message space $\mathbb{M}$ and a key k output by KGen and returns a tag τ. The Mac algorithm may be randomized.

$$\tau \leftarrow\!\!\$\ \text{Mac}(k, m)$$

- **Vrfy** Vrfy takes as input a message m, a key k, and a tag τ. It outputs a bit b, indicating whether the tag is a valid or not. An output of 1 indicates a valid tag, and an output of 0 an invalid tag.

$$b \leftarrow \text{Vrfy}(k, m, \tau)$$

For correctness, it is required that the following holds:

$$\forall k \in \mathbb{K}, \forall m \in \mathbb{M} : \text{Vrfy}(k, m, \text{MAC}(k, m)) = 1$$

Security is defined via unforgeability. An adversary with access to an oracle providing valid tags for any valid input message must not be able to produce a valid tag for any new message of his choice.

Definition 3.2 (EUF-CMA security of MAC schemes)**.** *Given the security game in Figure 5, a Message Authentication scheme* $\mathsf{MAC} = (\text{KGen}, \text{Mac}, \text{Vrfy})$ *is* EUF-CMA *secure, if the advantage* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = 1\right]$ *is negligible in λ for any* QPT *adversary $\mathcal{A}$.*

$\underline{\mathsf{Exp}^{\mathrm{PRF}}_{F,\mathcal{A}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 1$
  $k \leftarrow\!\!\$\ \mathbb{K}, f \leftarrow F(k,\cdot)$
else
  $f \leftarrow\!\!\$\ \{\text{functions } f : \{0,1\}^{\iota(\lambda)} \to \{0,1\}^{\omega(\lambda)}\}$
endif
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}(f,\cdot)}$
return $[\![b' = b]\!]$

$\underline{\mathsf{O}(f,\cdot) \text{ on input x}}$

return $f(x)$

Figure 6: Game definition for PRF security

3.3.2 Pseudorandom Functions

A Pseudorandom Function (PRF) is used to derive random-looking output from some input. A PRF takes two parameters, a key, and some input string, and returns an output of fixed length. Input and output lengths are not necessarily related. A PRF can be compressing, length-preserving, or extending. A PRF can be formalized as $\mathrm{F} : \{0,1\}^{\kappa(\lambda)} \times \{0,1\}^{\iota(\lambda)} \to \{0,1\}^{\omega(\lambda)}$. F is an efficient keyed function with key length $\kappa(\lambda)$, input length $\iota(\lambda)$, and output length $\omega(\lambda)$.

The key is kept secret and defines the behavior of the PRF for the given input. Security is based on an adversariy's inability to distinguish if it is interacting with a pseudorandom function or a truly random function. This is formalized by Definition 3.3.

Depending on some randomly sampled bit b, the adversary is either given access to a PRF with a random key or a truly random function sampled from the set of all random functions. The expected output is a guess whether the adversary is interacting with a truly random function, corresponding to $b = 0$ or a PRF, in which case $b = 1$ would be output.

Definition 3.3 (PRF Security)**.** *Let $F : \{0,1\}^{\kappa(\lambda)} \times \{0,1\}^{\iota(\lambda)} \to \{0,1\}^{\omega(\lambda)}$ be an efficient keyed function with key length $\kappa(\lambda)$, input length $\iota(\lambda)$ and output length $\omega(\lambda)$. Given the security experiment in Figure 6, a PRF is secure, if for all QPT adversaries $\mathcal{A}$ the following holds*

$$\begin{aligned}
&\mathsf{Adv}^{\mathrm{prf}}_{F,\mathcal{A}}(\lambda) \\
&\qquad := \left|\Pr\left[\mathsf{Exp}^{\mathrm{PRF}}_{F,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \\
&\qquad = \left|\Pr\left[1 \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}(f,\cdot)}(\lambda) \;\middle|\; b = 1\right] - \Pr\left[1 \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}(f,\cdot)}(\lambda) \;\middle|\; b = 0\right]\right| \\
&\qquad \leq \mathsf{negl}(\lambda)
\end{aligned}$$

3.3.3 Key Derivation Functions

A Key Derivation Function (KDF) is used to transform some fixed length, non-uniformly distributed input, such as the result of a DH key exchange, and deterministically extracts symmetric key material y. It takes as input a key k and a label x. We write $\mathrm{KDF}(k,x)$ to denote the execution of a KDF on the input key k and the label x. The syntax of a KDF is defined as

$$y \leftarrow \mathrm{KDF}(k,x)$$

While k must be kept secret to achieve the desired security goals, the label can be considered public information. We treat a KDF as a kind of PRF, with key length $\psi(\lambda)$, label length $\theta(\lambda)$ and output length $\phi(\lambda)$. The security is given by PRF security, given in Definition 3.3. An adversary must not be able to distinguish between a key output by a KDF and a randomly sampled key of the same length.

3.3.4 Key Encapsulation Mechanisms

A KEM scheme is a public key based scheme to generate and communicate a shared secret over an unsecure channel. The primary use case for KEMs is key establishment. KEMs are non-interactive, meaning only

$\underline{\mathsf{Exp}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathrm{KGen}(1^\lambda)$
2 : $k_0 \leftarrow\!\!\$\ \mathbb{K}$
3 : $(c, k_1) \leftarrow\!\!\$\ \mathrm{Encaps}(\mathsf{pk})$
4 : $b \leftarrow\!\!\$\ \{0,1\}$
5 : $b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}, c, k_b)$
6 : **return** $[\![b' = b]\!]$

Figure 7: Game definition for IND-CPA security of KEMs

one party can contribute randomness. The length of the key as well as the ciphertext are dependent on the security parameter and can be expressed as $\Gamma(\lambda)$ for the length of the key and $\Theta(\lambda)$ for the length of the ciphertext. The receiving party cannot influence on the key generation process and has to trust the generating party to use adequate randomness. A key encapsulation scheme KEM consists of three algorithms $\mathsf{KEM} = (\mathrm{KGen}, \mathrm{Encaps}, \mathrm{Decaps})$.

- **KGen** KGen is a probabilistic algorithm that takes the security parameter λ as input and probabilistically outputs a key pair (pk, sk).

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathrm{KGen}(1^\lambda)$$

- **Encaps** Encaps is a probabilistic algorithm and takes as input a public key pk, where $\mathsf{pk} \leftarrow \mathrm{KGen}(1^\lambda)$, and outputs a key k as well as a ciphertext c. The ciphertext c encapsulates the key k.

$$(k, c) \leftarrow\!\!\$\ \mathrm{Encaps}(\mathsf{pk})$$

- **Decaps** Decaps is a deterministic algorithm and takes a secret key sk and a ciphertext c as input, outputting either a key k or $\perp$ to indicate failure.

$$k \leftarrow \mathrm{Decaps}(\mathsf{sk}, c)$$

Definition 3.4 (Correctness of KEM schemes)**.** *A key encapsulation scheme* $\mathsf{KEM} = (\mathrm{KGen}, \mathrm{Encaps}, \mathrm{Decaps})$ *is* δ*-correct, if for all* $(\mathsf{sk}, \mathsf{pk})$ *key pairs output by the* KGen *algorithm*

$$\Pr[\mathrm{Decaps}(\mathsf{sk}, c) = k : (c, k) \leftarrow\!\!\$\ \mathrm{Encaps}(\mathsf{pk})] \geq 1 - \delta$$

If $\delta = 0$*,* KEM *is called perfectly correct.*

Note that no KEM scheme currently under consideration by NIST for standardization is perfectly correct.

Security of a KEM scheme is defined over CPA indistinguishability of derived keys and random keys. A challenger is provided a triple $(pk, c, \hat{k})$, where c is output by $(c, k) \leftarrow \mathrm{Encaps}(\mathsf{pk})$ and $\hat{k}$ is either sampled uniformly as $\{0,1\}^\Gamma(\lambda)$ or the actual key, which was output by the encapsulation algorithm. A challenger is successful if it can decide whether the given $\hat{k}$ is randomly sampled or generated by the encapsulation algorithm with non-negligible probability.

Definition 3.5 (IND-CPA security of KEM schemes)**.** *A key encapsulation scheme* KEM *is* IND-CPA *secure, if for all* QPT *adversaries* $\mathcal{A}$

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

For this thesis it is important to note, that KEM instantiations, which are resistant to quantum adversaries, exist.

$\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$

1 : $messages \leftarrow \emptyset$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \text{KGen}(1^\lambda)$
3 : $(\sigma', m') \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}_{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
4 : **return** $[\![\text{Vrfy}(\mathsf{pk}, \sigma', m') \wedge m' \notin messages]\!]$

$\mathsf{O}_{Sign}(\mathsf{sk}, \cdot)$ on input m

1 : $\sigma \leftarrow\!\!\$\ \text{Sign}(\mathsf{sk}, m)$
2 : $messages \leftarrow messages \cup m$
3 : **return** σ

Figure 8: Game definition for EUF-CMA security of signature schemes

3.3.5 Signature Schemes

A signature scheme is a public key based scheme to generate unforgeable signatures over arbitrary messages. Valid signatures can be generated only by a party with access to the secret key, while any party with access to the public key can validate signatures. A signature scheme Sig consists of three algorithms $\mathsf{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$.

- **KGen** KGen is a probabilistic algorithm used to generate a key pair for the signature scheme. It takes the security parameter λ as input and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \text{KGen}(1^\lambda)$$

- **Sign** Depending on its implementation, Sign is either a probabilistic or deterministic algorithm used to create signatures over some message. It takes as input a secret key sk and some message m, and outputs a signature σ.

$$\sigma \leftarrow\!\!\$\ \text{Sign}(\mathsf{sk}, m)$$

- **Vrfy** Vrfy is a deterministic algorithm that is used to verify signatures generated by Sign. It takes as input a public key pk, a signature σ and a message m, and outputs a boolean value of either 1 or 0, denoting a valid or invalid signature under the provided public key.

$$b \leftarrow \text{Vrfy}(\mathsf{pk}, \sigma, m)$$

For correctness, we require that

$$\Pr\big[0 \leftarrow \text{Vrfy}(\mathsf{pk}, \sigma, m) : (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \text{KGen}(1^\lambda), \sigma \leftarrow\!\!\$\ \text{Sign}(\mathsf{sk}, m)\big] \leq \mathsf{negl}(\lambda)$$

Intuitively, correctness requires that the verification of a valid signature fails only with negligible probability.

Security of signature schemes is defined over the notion of unforgeability. For the basic notion of existential unforgeability under chosen message attack (EUF-CMA) we require an adversary with access to a signing oracle to be unable to forge a signature for a message not previously queried to the oracle. This notion is formalized in the following definition

Definition 3.6 (EUF-CMA security of signature schemes)**.** *Given the security definitions from Figure 8, a public key signature scheme* Sig *is* EUF-CMA *secure, if*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\Big[\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = 1\Big] \leq \mathsf{negl}(\lambda)$$

4 | Asynchronous Remote Key Generation

This section will present the ARKG primitive, $\mathcal{ARKG}$, as introduced by Frymann et al. [FGK+20]. It will introduce syntax and semantics of its algorithms as well as a description of the interactions that can be realized using this primitive. Two different instantiations, as well as two different sets of security properties for this primitive, will be introduced in Sections 5 and 6.

4.1 Motivation

Multi-Factor authentication is the main tool for providing a more robust authentication for any digital system. The most secure option for a second factor are hardware authenticators, such as FIDO2 compliant devices. Unlike traditional passwords, they cannot be forgotten or leaked and are even resistant to phishing attacks.

However, having a token always come with the possibility of losing said token. For this reason, some recovery options must be put in place. The trivial solution is copying secrets between authenticators, creating an identical backup. While the FIDO2 standard itself does not prohibit copying secret key material between authenticators, this functionality is currently hardly being implemented [Gla22b]. Additionally, allowing the extraction of secrets from an authenticator comes with a high risk of accidentally leaking the secret.

Yubico, one of the largest suppliers of FIDO compliant hardware tokens, suggests having two hardware tokens and manually registering both with each new service [Yuba] as the go-to backup solution. While this works, it is inconvenient as it requires a user to undergo the same registration procedure twice in a row. Furthermore, if you always have both authenticators with you at the same place, you are likely to lose both of them at once, still leaving you stranded.

Backup codes are an alternative solution for account recovery. When setting up MFA, one-time passwords are generated by the service and output to the user. They can be used to recover access in case the second factor is lost. However, backup codes suffer from similar shortcomings as traditional passwords, as they have to be stored somewhere.

In order to provide a secure way of recovering an account in the event of authenticator loss, Yubico has proposed a set of algorithms they call Asynchronous Remote Key Generation (ARKG) [Lun20]. ARKG is meant to address those weaknesses and provide a secure way to recover access after the loss of an authenticator. The proposal describes how a primary authenticator device can register additional credentials on behalf of a backup authenticator (BA). Should the primary authenticator (PA) become unavailable, the BA can asynchronously recover the secrets for the credentials registered by the PA. This allows the BA to successfully complete the authentication process, thus recovering access to the account.

4.2 Notation

Before diving into details, we first fix some general definitions and notation that will be used henceforth. The definitions are kept as close as possible to the current WebAuthn standard published by the W3C [JJM+21].

Parties All of the interactions within an $\mathcal{ARKG}$ scheme happen between three distinct parties.

- **Primary Authenticator** (PA) The primary authenticator is the primary security token a user possesses. Under normal circumstances, it is used by the user to authenticate himself to his services.
- **Backup Authenticator** (BA) The backup authenticator is an additional security token a user might have in case his PA is lost. It is not used under normal circumstances and only becomes relevant when the PA is no longer accessible for any reason.
- **Relying Party** (RP) The RP is the entity whose web application utilizes the Web Authentication API to register and authenticate users. Simply put, it is the service where a user has registered his authenticator for passwordless authentication.

Interactions There are three types of interactions that can happen between the aforementioned three parties. Each interaction only involves two parties. A visualization of the interactions is provided in Figure 9.

- **Pair** The process of pairing links a PA with a BA and allows the PA to register credentials on behalf of the BA. The interacting parties are two authenticators. This has to happen only once for each pair of authenticators.
- **Register** The process of registering is characterized by a PA registering credentials on behalf of a BA at a RP. This step happens between a PA and a RP.
- **Recover** Recovery happens in case the PA has become unavailable. It is performed by the BA using the credentials which were registered by the PA on its behalf. This interaction takes place between a BA and a RP.

Common Variables There are a few variable names commonly used throughout this thesis, which will now be defined.

- $(\mathsf{pk}_0, \mathsf{sk}_0)$ denote the long-term secrets generated by PA and BA during the pairing phase. The key pair is generated by the BA and the public key is shared with the corresponding PA allowing it to register credentials on BA's behalf.
- $(\mathsf{pk}', \mathsf{sk}')$ denote a key pair that has been derived from some long-term public key, such as pk_0.
- cred denotes some cryptographic material generated by the PA during the registration phase that is stored by the RP. It is required by the BA during the recovery phase to recover the secret corresponding to the public key stored with the RP on its behalf.
- aux denotes some auxiliary data that contains miscellaneous additional information. Typically the remote party identifier rpId is contained in aux, but depending on the definition of the instantiation, it might contain more information.
- rpId is short for remote party identifier and is a parameter defined by the WebAuthn specification. It denotes some information to uniquely identify the RP with which an authenticator is communicating. By default, this value is the effective domain of the rpId.

4.3 Syntax

This subsection will introduce syntax and correctness requirement of Asynchronous Remote Key Generation. These requirements are shared among both primitives introduced in the following two sections.

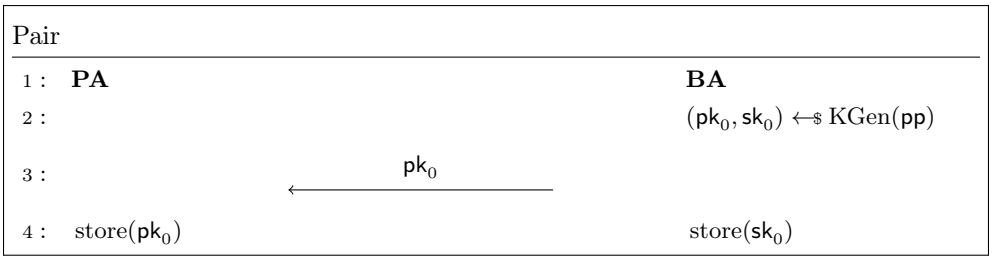
Definition 4.1 ($\mathcal{ARKG}$ Syntax)**.** *A scheme for asynchronous remote key generation and recovery consists of the following algorithms:*

- $\text{Setup}(1^\lambda)$ *generates and outputs public parameters* pp *for the security parameter* λ*.*
- $\text{KGen}(\mathsf{pp})$*, on input* pp*, computes and returns a key pair* $(\mathsf{sk}, \mathsf{pk})$*.*
- $\text{Check}(\mathsf{sk}, \mathsf{pk})$*, on input* $(\mathsf{sk}, \mathsf{pk})$*, returns 1 if the tuple* $(\mathsf{pk}, \mathsf{sk})$ *forms a valid private-public key pair, otherwise it returns 0.*
- $\text{DerivePk}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux})$ *probabilistically returns a new public key* pk' *together with a key handle* cred*. The input* aux *is always required but may be empty.*
- $\text{DeriveSk}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$ *computes and outputs either the new secret key* sk'*, corresponding to the public key* pk' *using* cred*, or* $\perp$ *on error.*

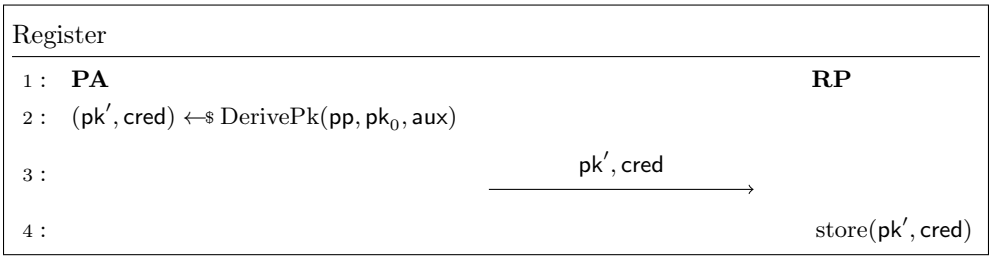
In Figure 9, the additional function "store" is used to indicate an authenticator storing values for later use.

Definition 4.2 ($\mathcal{ARKG}$ Correctness)**.** *An asynchronous remote key generation scheme is delta-correct, if for all security parameters* λ *and* $\mathsf{pp} \leftarrow \text{Setup}(1^\lambda)$*, the probability* $\Pr\big[\text{Check}(\mathsf{pk}', \mathsf{sk}') = 1\big] = 1 - \delta$*, if*

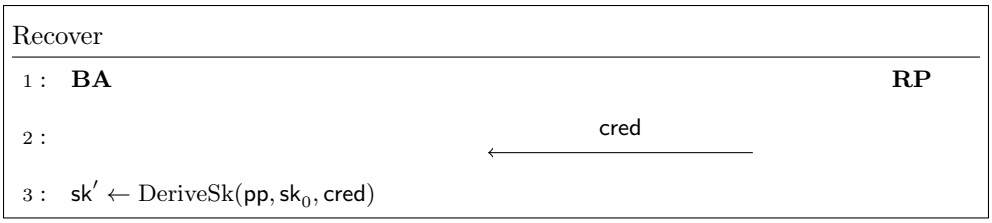
$$
\begin{aligned}
(\mathsf{pk}, \mathsf{sk}) &\leftarrow\!\!\$\ \text{KGen}(\mathsf{pp}) \\
(\mathsf{pk}', \mathsf{cred}) &\leftarrow\!\!\$\ \text{DerivePK}(\mathsf{pp}, \mathsf{pk}, \cdot) \\
\mathsf{sk}' &\leftarrow \text{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})
\end{aligned}
$$



(a) Initial pairing of PA with its BA



(b) PA registering credentials on behalf of the BA



(c) BA recovering credentials registered on its behalf from a RP

Figure 9: The three interactions that can take place as part of $\mathcal{ARKG}$

4.4 A Primitive for ARKG

Definition 4.3 ($\mathcal{ARKG}$ primitive)**.** *The primitive $\mathcal{ARKG}$ consists of the set of algorithms $\mathcal{ARKG}$ =* (Setup, KGen, Check, DerivePk, DeriveSk)*. In combination with the syntax defined by Definition 4.1 and the correctness requirement from Definition 4.2, this defines the new primitive $\mathcal{ARKG}$.*

4.5 Intuition for Security Requirements

To ensure the security of their proposed interactions, Yubico has partnered with Frymann et al. to create a formal security analysis of their proposal. First, Frymann et al. have modeled Asynchronous Remote Key Generation as a new cryptographic primitive, which we have introduced in Definition 4.3. Then, they define corresponding security properties, which we will now introduce.

The new primitive for Asynchronous Remote Key Generation, which we denote as $\mathcal{ARKG}$, has two security properties.

- Security of the derived key pairs
- Unlinkability of all derived public keys

We will start by providing intuition for these security properties and subsequently give a formal definition. Security of the derived secret key pairs requires a derived key pair to be just as secure as a freshly generated key pair. What exactly secure means is challenging to define. The differentiating properties of a derived key pair compared to a regular key pair are the key handle cred and the initial public key pk_0, from which the pair was derived. Thus, the intuitive requirement is that the additional information contained in cred and pk_0 does not leak any information about the secret key. In the following two sections, we will see two different formalizations of this requirement.

Unlinkability of all derived public keys is a more straightforward requirement. Unlinkability prevents platforms from learning which authenticator has generated which public keys. This could be used to track a user across various servers or detect the creation of multiple accounts for the same person. The intuition for the formal security requirements is as follows: Given some initial public key and some key pair, an adversary should be unable to decide whether the given key pair was derived from the provided initial public key or generated independently.

5 | ARKG by Frymann et al.

This section will expand on the primitive introduced at the end of the previous section, introducing the formal security requirements proposed by Frymann et al. as well as their elliptic-curve-based instantiation.

5.1 Definitions and Security Properties

We will now formalize the intuitions for the security requirements we have introduced in Section 4.5.

Definition 5.1 (Key Security of $\mathcal{ARKG}$)**.** *We say an $\mathcal{ARKG}$ scheme provides secret key security with $\overline{\mathsf{ks}}$ $\in \{mwKS, hwKS, msKS, hsKS\}$, if the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\overline{\mathsf{ks}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\overline{\mathsf{ks}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) = 1\right]$$

Intuitively, this definition captures the probability, that an adversary is able to derive any valid key pair $(\mathsf{pk}', \mathsf{sk}')$ and corresponding key handle cred from the given pk, without knowing the corresponding sk.

Frymann et al. consider four different types of adversaries: Honest and malicious adversaries, as well as weak and strong adversaries. Combining these attributes results in the four types honest weak (hw), honest strong (hs), malicious weak (mw) and malicious strong (ms).

The honest flavors of SK-security force the adversary to provide a secret key corresponding to a $(\mathsf{pk}^*, \mathsf{cred}^*)$ tuple obtained from the derived public key oracle. Effectively, this requires an adversary to extract the secret key sk^* from the tuple $(\mathsf{pk}^*, \mathsf{cred}^*)$.

Strong adversaries have access to a secret key oracle, $\overline{\mathsf{O}_{\mathsf{sk}'}}$, which provides secret keys corresponding to the public keys obtained from the derived public key oracle. For a strong adversary, an additional condition is required, which prevents the adversary from using the secret key oracle to trivially win the game. This restriction is introduced in line 8 of the $\overline{\mathsf{ks}}$ experiment.

Remark 5.2. *It can be shown via reduction that the relation between the different key security flavors is*

$$msKS \Rightarrow mwKS \Rightarrow hwKS$$

and

$$msKS \Rightarrow hsKS \Rightarrow hwKS$$

Definition 5.3 (Public Key Unlinkability of $\mathcal{ARKG}$)**.** *We say an $\mathcal{ARKG}$ scheme provides public key unlinkability if the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\overline{\mathsf{pku}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\overline{\mathsf{pku}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$$

The unlinkability property captures the inability of an adversary to decide whether key pairs are derived from some given pk_0 or sampled randomly. To formalize this idea, a challenge oracle $\overline{\mathsf{O}^b_{\mathsf{pk}'}}$ is introduced. Depending on the bit b of the game $\overline{\mathsf{pku}}$, it either returns derived key pairs or randomly sampled key pairs. The QPT adversary can run a polynomial number of queries, giving him a polynomial number of key pairs before it has to make a decision. The security property $\overline{\mathsf{pku}}$ does not differentiate between different classes of adversaries.

5.2 Instantiating ARKG with Elliptic Curves

The instantiation of $\mathcal{ARKG}$ by Frymann et al., $\overline{\mathsf{ARKG}}$, is given by Definition 5.4.

Definition 5.4 ($\overline{\mathsf{ARKG}}$)**.** $\overline{\mathsf{ARKG}} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{DerivePk}, \mathrm{DeriveSk})$ *is an instantiation of $\mathcal{ARKG}$. The instantiation for the algorithms of $\mathcal{ARKG}$ is given in Figure 11.*

$\overline{\mathsf{ARKG}}$ is based on Elliptic Curve Cryptography (ECC) and relies on the hardness of the discrete logarithm problem. Group operations in $\mathbb{G}$ are realized on the P-256 curve. For signatures, the ECDSA standard is used [Inf13]. HKDF [KE10], in combination with SHA-256 [Tec15a], is used as the KDF.

As none of the aforementioned schemes are quantum-resistant, $\overline{\mathsf{ARKG}}$ offers no security in a post-quantum setting. Without the presence of a quantum adversary, the instantiation ARKG is msKS-secure and satisfies pku.

$\mathsf{Exp}^{\overline{\mathsf{ks}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
2 : $SKList \leftarrow \emptyset, PKList \leftarrow \emptyset$
3 : $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\$ \mathrm{KGen}(\mathsf{pp})$
4 : $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{cred}^*) \leftarrow\$ \mathcal{A}^{\overline{\mathsf{O}_{\mathsf{pk}'}}, \boxed{\overline{\mathsf{O}_{\mathsf{sk}'}}}}(\mathsf{pp}, \mathsf{pk}_0)$
5 : $\mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}*)$
6 : **return** $[\![\mathrm{Check}(\mathsf{sk}^*, \mathsf{pk}^*) = 1$
7 : $\wedge \mathrm{Check}(\mathsf{sk}', \mathsf{pk}^*)$
8 : $\boxed{\wedge \mathsf{cred}^* \notin SKList}$
9 : $\boxed{\wedge (\mathsf{pk}^*, \mathsf{cred}^*) \in PKList}\,]\!]$

(a) Security experiment defining the key security property of ARKG. The four flavors of key security {mwKS, hwKS, msKS,hsKS} can be achieved by omitting or including the dashed and dotted boxes. Presence of dashed boxes give strong variants, presence of the dotted box the honest variants

$\mathsf{Exp}^{\overline{\mathsf{pku}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
2 : $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\$ \mathrm{KGen}(\mathsf{pp})$
3 : $b \leftarrow\$ \{0, 1\}$
4 : $b' \leftarrow\$ \mathcal{A}^{\overline{\mathsf{O}^b_{\mathsf{pk}'}}}(\mathsf{pp}, \mathsf{pk}_0)$
5 : **return** $[\![b = b']\!]$

(b) Security experiment defining the public key unlinkability property of ARKG

$\overline{\mathsf{O}_{\mathsf{pk}'}}(\mathsf{pk}_0, \cdot)$ on input aux

1 : $\mathsf{pk}', \mathsf{cred} \leftarrow\$ \mathrm{DerivePk}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{aux})$
2 : $PKList \leftarrow PKList \cup (\mathsf{pk}', \mathsf{cred})$
3 : **return** $(\mathsf{pk}', \mathsf{cred})$

$\overline{\mathsf{O}_{\mathsf{sk}'}}(\mathsf{sk}_0, \cdot)$ on input cred

1 : **if** $(\cdot, \mathsf{cred}) \in PKList$
2 : $\quad \mathsf{sk}' \leftarrow \mathrm{DeriveSk}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{cred})$
3 : $\quad SKList \leftarrow SKList \cup \mathsf{cred}$
4 : $\quad$ **return** sk'
5 : **else return** $\perp$

$\overline{\mathsf{O}^b_{\mathsf{pk}'}}(\mathsf{pk}_0, \mathsf{sk})$

1 : $\quad \mathsf{pk}'_0, \mathsf{cred} \leftarrow\$ \mathrm{DerivePk}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{aux})$
2 : $\quad \mathsf{sk}'_0 \leftarrow \mathrm{DeriveSk}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$
3 : $\quad \mathsf{pk}'_1, \mathsf{sk}'_1 \leftarrow\$ \mathbb{D}$
4 : **return** $(\mathsf{pk}'_b, \mathsf{sk}'_b, \mathsf{cred})$

Figure 10: Security Experiments for $\mathcal{ARKG}$

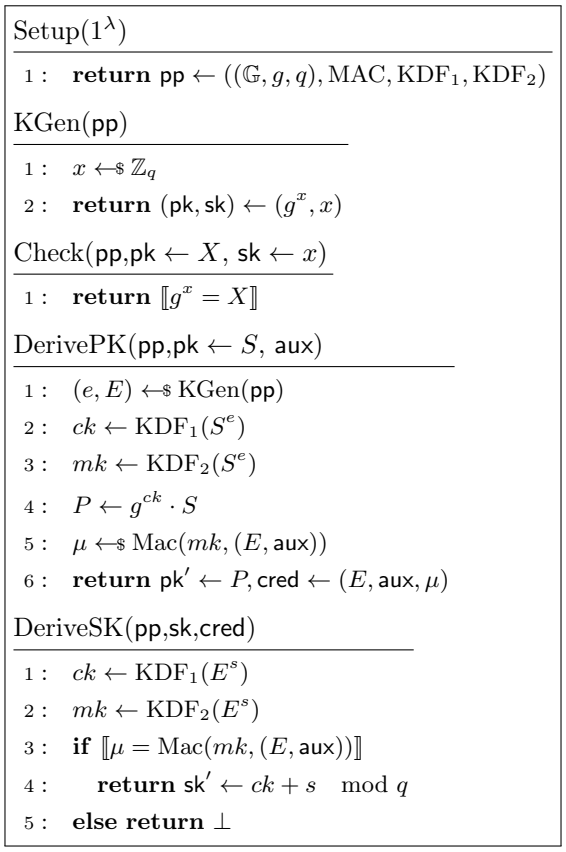


Figure 11: The original instantiation proposed in [FGK+20]. It is based on the hardness of the discrete logarithm problem and, as such, not secure in a post-quantum setting

6 | Post-Quantum ARKG

This section will present the core contribution of this thesis: ARKG, a post-quantum instantiation of the primitive $\mathcal{ARKG}$ discussed in the previous sections, redefined security games, as well as complete security proofs. While the intuitive security requirements remain the same, the formalization has slightly changed to better match the real-world requirements and to work with our instantiation. We argue that the real-life security of Asynchronous Remote Key Generation (ARKG) remains unchanged by our modified formalization. We will first introduce our modified security games $\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ and provide reasoning for the modifications. Afterward, our post-quantum instantiation will be defined in Definition 6.3. To conclude the chapter, we will provide security proofs for our instantiation of the modified primitive.

6.1 Approach

In the original instantiation $\overline{\mathsf{ARKG}}$, proposed in [FGK+20] and defined in Definition 5.4, Frymann et al. relied on a Diffie-Hellman style key exchange implemented on an elliptic curve to establish a shared secret between PA and BA. This shared secret is the key to maintaining the security of the derived keys.

As discussed in Section 2.4, this style of key agreement is not resistant to attackers with local quantum computing power. To achieve our goal of a post-quantum secure scheme, we are restricting ourselves to using algorithms that are currently being standardized by NIST's PQC standardization initiative [Moo22]. For key agreement, this leaves us with two options, either using a public key encryption scheme or Key Encapsulation Mechanisms. The decision for KEMs is motivated by our use case. We intend to establish a shared secret between PA and BA, which can be used for key generation. Using a public key encryption scheme would make the protocol more complicated, as key generation and communication are two separate concerns. KEMs provide a solution for both problems in a single algorithm. This makes it easier to use and yields simpler, more compact security proofs.

The ongoing standardization of PQC schemes by NIST, discussed in Section 2.5, has identified several instantiations for Key Encapsulation Mechanisms (KEMs), which remain secure against an adversary with quantum computing power. The first algorithms are expected to be fully standardized by 2024 [Moo22]. With this motivation, we are introducing a modified instantiation of the primitive $\mathcal{ARKG}$, which changes the main building block from Diffie-Hellman (DH) key exchanges to KEMs.

This modification will also affect the security games presented in the previous section.

6.2 Redefining Security Games

This subsection will introduce two new security games for the security requirements public key unlinkability and key security discussed in Section 4.5. We call them ks and pku. They are designed to replace the formalization by Frymann et al. introduced in Definition 5.1 and Definition 5.3, closer modelling the capabilities of real-world adversaries.

Afterward we will explain the thought process behind the updated security games and highlight important differences and similarities.

Definition 6.1 (Modified Key Security of $\mathcal{ARKG}$)**.** *Let the security experiment* $\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ *be defined by Figure 12. We say an* $\mathcal{ARKG}$ *scheme provides key security, if*

$$\mathsf{Adv}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

Definition 6.2 (Modified Public Key Unlinkability of $\mathcal{ARKG}$)**.** *Let the security experiment* $\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ *be defined by Figure 12. We say an* $\mathcal{ARKG}$ *scheme provides public key unlinkability, if*

$$\mathsf{Adv}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

6.2.1 Key Security

The most striking difference between the two security properties capturing the notion of key security, ks and $\overline{\mathsf{ks}}$, is the omission of the different classes of adversaries. The adversary plying $\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ does

$\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
2 : $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow\!\$\ \mathrm{KGen}(\mathsf{pp})$
3 : $(\mathsf{pk}'_0, \mathsf{cred}) \leftarrow\!\$\ \mathrm{DerivePk}(\mathsf{pp}, \mathsf{pk}_0)$
4 : $\mathsf{sk}'_0 \leftarrow \mathrm{DeriveSk}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{cred})$
5 : $(\mathsf{pk}'_1, \mathsf{sk}'_1) \leftarrow\!\$\ \mathbb{D}$
6 : $b \leftarrow\!\$\ \{0,1\}$
7 : $b' \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}'_b, \mathsf{sk}'_b, \mathsf{cred})$
8 : **return** $[\![b = b']\!]$

(a) Security Experiment for the public key unlinkabiliy property

$\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
2 : $PKList \leftarrow \emptyset$
3 : $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\$\ \mathrm{KGen}(\mathsf{pp})$
4 : $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow\!\$\ \mathcal{A}^{\mathsf{O}_{\mathsf{pk}'}(\mathsf{pk}_0,\cdot)}(\mathsf{pp}, \mathsf{pk}_0)$
5 : **return** $[\![\mathrm{Check}(\mathsf{sk}^*, \mathsf{pk}^*) = 1$
6 : $\quad \wedge\, \mathsf{pk}^* \in PKList]\!]$

(b) Security experiment for the key security property

$\mathsf{O}_{\mathsf{pk}'}(\mathsf{pk}_0, \cdot)$ on input aux

1 : $(\mathsf{pk}', \mathsf{cred}) \leftarrow\!\$\ \mathrm{DerivePk}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{aux})$
2 : $PKList \leftarrow PKList \cup \mathsf{pk}'$
3 : **return** $(\mathsf{pk}', \mathsf{cred})$

(c) Oracle for the security definitions of $\mathcal{ARKG}$

Figure 12: Modified security experiments for the security of $\mathcal{ARKG}$. The thought process behind these changes is further elaborated in Section 6.2

not have access to a decryption oracle and is limited to outputting a secret key corresponding to a public key previously output by the public key oracle. Additionally, outputting the key handle cred^* is no longer required.

It would seem like this security definition corresponds to the weakest class of adversaries in $\overline{\mathsf{ks}}$, the honest weak adversary. However, we argue that an adversary $\mathcal{A}$ playing the new security game $\mathsf{Exp}^{\mathsf{ks}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ is in no way restricted beyond the real-world setting this game is mean to represent.

Strong Adversaries First, we have a look at the implications of the existence of the secret key oracle $\overline{\mathsf{O}_{\mathsf{sk}'}}$. In [FGK+20], adversaries with this capability are referred to as strong adversaries. An interaction with the secret key oracle simulates the real-world setting where an adversary has access to the internal secrets of the BA. This is because the derived secret key sk' is only output by the algorithm DeriveSk, which requires the initial secret key sk_0 as input. Exactly this key is stored on the BA only. Furthermore, the derivation of the keys also takes place on the authenticator and no algorithm allows for the keys' extraction. As we discussed in Section 2.2, we consider FIDO2 authenticators to be tamper-proof, making exactly this extraction of internal secrets impossible. Consequently, giving an adversary access to a secret key oracle does not simulate a real-world scenario, and the notion of the strong adversary can be omitted.

Note that for any other initial key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$ an adversary can derive as many key pairs as it likes. This simulates the real-world case, where it has an authenticator with a different public key than the one it is trying to attack.

Honest and Malicious Adversaries Next, we look at the implications of a malicious adversary playing against $\mathsf{Exp}^{\overline{\mathsf{ks}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$. We have previously established that a malicious adversary is successful if it can output any derived key pair along with a corresponding cred. This corresponds to an adversary successfully deriving any key pair.

When considering the real-world implication of this attack, it does not become evident how an attacker can leverage deriving some arbitrary key pair into impersonating a user in a FIDO2 session. To obtain access to an account, it must recover the secret key corresponding to the public key generated by the PA during the pairing phase.

Circling back to the intuition we provided for the security property key security, we stated that a derived key pair must be just as "secure" as an independently generated key pair. However, we did not

define what exactly we meant by "secure". When looking at a recovery interaction of $\mathcal{ARKG}$, as shown in Figure 9, we see that only cred is output by the RP. In combination with the challenge public key pk', which can be assumed to be publicly available, an adversary now has the triple $(\mathsf{pk}_0, \mathsf{pk}', \mathsf{cred})$ at his disposal.

We now use this insight to refine "secure": The additional information contained in cred, must not increase the likelihood of an adversary extracting the secret key sk' from the public key pk' by any non-negligible amount. For this reason, the adversary is only required to output a secret key sk^* and the public key pk^* from which it extracted the secret key.

Why the Derived Public Key Oracle? We have already established in the previous paragraphs that anyone with access to some initial public key pk_0 can derive as many keys as it desires, by running $(\mathsf{pk}', \mathsf{cred}) \leftarrow\!\$\ \mathrm{DerivePk}(\mathsf{pp}, \mathsf{pk}_0)$. Why do we need this oracle if any adversary can simulate it? The sole purpose of the oracle $\mathsf{O}_{\mathsf{pk}'}$ is to track which derived public keys have been output to the adversary. They are tracked in the set $PKList$. Public key unlinkability prevents us from verifying if the key pair $(\mathsf{pk}^*, \mathsf{sk}^*)$ is a derived key pair or a freshly sampled key pair. If we were not to keep track of public keys output to the adversary, it could generate a fresh key pair $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow\!\$\ \mathrm{KGen}(\mathsf{pp})$ and return this key pair to win the game trivially.

6.2.2 Public Key Unlinkability

Recall that intuitively, an adversary has to decide whether a given key pair is derived from some given initial public key pk_0, or sampled randomly from the distribution of all key pairs.

The original game $\mathsf{Exp}^{\overline{\mathsf{pku}}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ provides a challenge oracle to the adversary that behaves one way or the other, depending on some internal bit b. Any QPT adversary is allowed to obtain a polynomial amount of key pairs with corresponding key handle cred by querying this oracle. In contrast, an adversary playing $\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$ is only given one key pair and has to decide based on this information. Having access to a more significant number of key pairs would allow an adversary to draw some conclusions on the distribution from which the key pairs were sampled. Nonetheless, this does not affect the success probability of an adversary $\mathcal{A}$ playing $\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{ARKG},\mathcal{A}}(\lambda)$.

We will now argue why this is the case. In some experiments, it is sufficient to determine from which distribution values are sampled for an adversary to win the game. One example of such an experiment is the PRF security definition, introduced in Section 3.

However, this is not relevant for the pku game. As the random key is sampled from the same distribution as the derived keys, an adversary cannot use this feature to distinguish which game he is playing. As a result, identifying the distribution from which key pairs are sampled does not aid an adversary in winning the ks security game.

Furthermore, the real-world capabilities of an attacker also support the omission of the public key oracle. In practice, an adversary would be able to obtain derived public keys by tricking a PA to register with him. He would then receive cred and some pk' as part of the registration interaction. At no point will the adversary have access to the secret key, as it is generated on the BA only and cannot be extracted from the authenticator. Any adversary can trivially simulate the event of a PA registering with him by running the algorithm $\mathrm{DerivePk}(\mathsf{pk}_0)$, which also outputs the tuple $(\mathsf{cred}, \mathsf{pk})$.

This means providing the adversary with the derived secret key does not reflect any real-world attack, while the other information returned by the challenge oracle can be calculated directly by any adversary with knowledge of the initial public key pk_0.

6.3 A Post-Quantum Instantiation for $\mathcal{ARKG}$

We now introduce the formal definition of the post-quantum instantiation ARKG of the primitive $\mathcal{ARKG}$. To achieve post-quantum security, we use KEMs and digital signatures as the main building blocks for our instantiation. Both, KEMs and digital signatures, can be instantiated in a way that they are resistant to quantum adversaries, as discussed in Section 2.5. This section will introduce the new interactions and associated security games, give reasoning for the design choices, and provide proof of the security of our designs.

Definition 6.3 (ARKG)**.** *We say* $\mathsf{ARKG} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{Check}, \mathrm{DerivePk}, \mathrm{DeriveSk})$ *is an instantiation of the primitive* $\mathcal{ARKG}$ *when the algorithms are instantiated, as described in Figure 13.*

An intuition for the functionality of the algorithms is provided in Section 6.4.

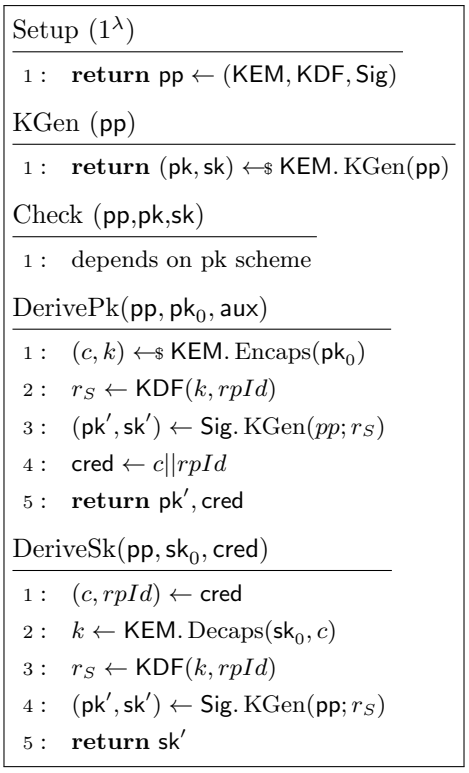


Figure 13: Algorithms for the post-quantum instantiation ARKG

6.4 Intuition

We start by giving an intuition for the behavior of our instantiation ARKG. Like in the original instantiation, $\overline{\mathsf{ARKG}}$, PA is provided a public key for a KEM scheme by the BA during the pairing phase. The PA can use this public key pk as input to a KEM scheme to generate tuples of (c, k). The ciphertext c is then sent to the RP as part of cred. During recovery, the key handle cred, containing c, is relayed to the BA for decapsulation. As PA and BA cannot directly communicate after the initial pairing, relaying the key becomes the responsibility of the RP. When account recovery is initiated, the RP provides the information required for a successful recovery to the BA. This allows the BA to retroactively establish a shared symmetric secret with the PA, which is no longer available at this time. The symmetric secret is then used as input for a KDF to obtain a random key of the appropriate length. Subsequently, this (psuedo-) randomness, which all parties with knowledge of the shared symmetric secret can obtain, is used to derive a (pk, sk) key pair, by using it as randomness in a call to Sig. KGen. Due to BA and PA both having the same input for the KDF, they both obtain the same key pair. This concludes the objective of the Asynchronous Remote Key Generation, as both authenticators have asynchronously generated the same key pair. These interactions are visualized in Figure 9.

6.5 ARKG Security Proofs

This section will analyze the security of our proposed KEM-based interaction regarding the updated security properties ks and pku.

6.5.1 Key Security

Theorem 6.4. *Let* ARKG *be the instantiation of $\mathcal{ARKG}$ defined in Definition 6.3,* $\mathsf{KEM} = (\mathrm{KGen}, \mathrm{Encaps}, \mathrm{Decaps})$ *an* IND-CPA *secure KEM scheme,* $\mathsf{Sig} = (\mathrm{KGen}, \mathrm{Sign}, \mathrm{Vrfy})$ *an* EUF-CMA *secure signature scheme and* KDF *a secure key derivation function modeled as a PRF.*

Then ARKG *fulfills the security requirements imposed by the key security game introduced in Definition 6.1. For any efficient* QPT *adversary $\mathcal{A}$, we derive efficient adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, such that*

$$\mathsf{Adv}^{\mathrm{ks}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot (\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_2}(\lambda)) \leq \mathsf{negl}(\lambda)$$

Proof. We will prove Theorem 6.4 using game hopping.

$\mathsf{Game}_1(\lambda)$**:** The original key security game $\mathsf{Exp}^{\mathsf{ks}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$.

$\mathsf{Game}_2(\lambda)$**:** We guess for which call to $\mathsf{O}_{\mathsf{pk}'}$ the adversary will compute the secret key sk^*. We define the number of total queries to the oracle as q. Therefore we will guess the correct oracle call with a chance of $\frac{1}{q}$. Thus we obtain a bound of

$$\mathsf{Adv}^{\mathrm{g}_1}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \mathsf{Adv}^{\mathrm{g}_2}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$$

$\mathsf{Game}_3(\lambda)$**:** We now modify the behavior of the oracle $\mathsf{O}_{\mathsf{pk}'}$. During the oracle call guessed in the previous game we replace the key, which is input to the KDF, with a random key of the same length as the key previously generated by running Encaps(pk_0).

We claim that it is impossible to distinguish between Game_2 and Game_3 with a non-negligible advantage, as such a distinguisher $\mathcal{A}$ could be used as a subroutine by an adversary $\mathcal{B}_1$ to efficiently break the IND-CPA security of the underlying KEM scheme KEM.

$\mathcal{B}_1$ receives as input the IND-CPA challenge $(\mathsf{pk}, c, \hat{k})$. In order to decide if the bit b of the IND-CPA challenge is 0, i.e. $\hat{k} \leftarrow\!\!\$\ \mathbb{K}$ or 1, which would mean $\hat{k} = \mathrm{Decaps}(\mathsf{sk}, c)$, the adversary $\mathcal{B}_1$ uses $\mathcal{A}$ as a subroutine.

First, it initializes $\mathcal{A}$ with the public key pk from its own challenge. As a response to one of the oracle calls to $\mathsf{O}_{\mathsf{pk}'}$, $\mathcal{B}_1$ does not replace the key of the KDF with a random value but with $\hat{k}$ from its own IND-CPA challenge.

Note that if the bit b of the IND-CPA game is $b = 1$, we have $\hat{k} = \mathrm{Decaps}(\mathsf{sk}, c)$, which corresponds precisely to the behavior of Game_2. Otherwise, $\hat{k}$ is sampled randomly from the distribution, which is exactly the behavior of Game_3.

Therefore, $\mathcal{A}$ being able to distinguish between the two games efficiently would imply $\mathcal{B}_1$ being able to break IND-CPA security efficiently. The advantage of $\mathcal{A}$ can be bound by

$$\mathsf{Adv}^{\mathrm{g}_2}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{g}_3}_{\mathsf{ARKG},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda)$$

We now have created a situation, where the challenge key pk' given to the adversary is completely independent of the KEM ciphertext c contained in cred. This means the only way for an adversary to beat the challenge is to directly derive the corresponding secret key from the given public key.

We use this idea to conclude the proof by reducing the advantage of an adversary against Game_3 to the EUF-CMA advantage of the underlying signature scheme Sig. We do this by constructing an adversary $\mathcal{B}_2$ as a direct reduction. This reduction is depicted in Figure 14. An adversary $\mathcal{A}$ that is able to defeat Game_3 outputs a secret key for some given public key. The idea of the reduction is to use this adversary as a subroutine to obtain the secret key corresponding to the public key provided as part of the EUF-CMA challenge. It can then generate an arbitrary message and create a valid signature using the secret key obtained from $\mathcal{A}$, completing the reduction.

Evidently, the advantage of the reduction is bound by EUF-CMA security of the signature scheme.

$$\mathsf{Adv}^{\mathrm{g}_3}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_2}(\lambda)$$

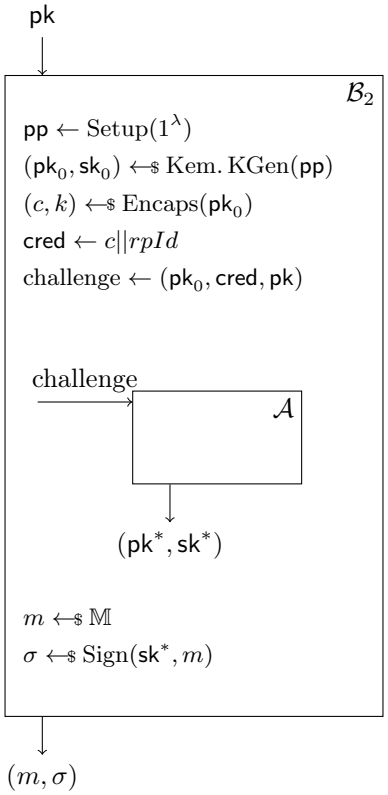


Figure 14: Reduction for the last step of the proof for Theorem 6.4

To conclude the proof, we summarize the advantages:

$$\mathsf{Adv}^{\text{ks}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot (\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_2}(\lambda))$$

With our initial assumption that KEM is an IND-CPA secure KEM scheme, and Sig is an EUF-CMA secure signature scheme we find that

$$\mathsf{Adv}^{\text{ks}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot (\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_2}(\lambda)) \leq \mathsf{negl}(\lambda)$$

which concludes the proof. □

Remark 6.5. *The reduction in Figure 14 does not make use of the Sign oracle, which is provided as part of the* EUF-CMA *security game. Therefore, the weaker security notion* EUF-KO*, which does not provide any oracles to the challenger, would be sufficient for a secure instantiation of* ARKG*. However,* EUF-CMA *is generally considered the minimum security requirement for a signature scheme to be used in a real-world scenario. Consequently, NIST is only considering candidates for the PQC standardization that provide at least* EUF-CMA *security.*

6.5.2 Public Key Unlinkability

Theorem 6.6. *Let* ARKG *be the instantiation of $\mathcal{ARKG}$ defined in Definition 6.3,* KEM = (KGen, Encaps, Decaps) *an* IND-CPA *secure KEM scheme,* Sig = (KGen, Sign, Vrfy) *an* EUF-CMA *secure signature scheme and* KDF *a secure key derivation function modeled as a PRF.*

Then ARKG *fulfills the security requirements imposed by the public key unlinkabiliy game introduced in Definition 6.2. For any efficient* QPT *adversary $\mathcal{A}$, we derive an efficient adversary $\mathcal{B}$, such that*

$$\mathsf{Adv}^{\text{pku}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},B}(\lambda) \leq \mathsf{negl}(\lambda)$$

Proof. We prove Theorem 6.6 using a direct reduction to the IND-CPA security of the underlying KEM. Assuming the existence of an adversary $\mathcal{A}$ that can defeat the pku security game in polynomial time with

$\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
2 : $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow\!\!\$\ \mathrm{KGen}(\mathsf{pp})$
3 : $(c, k) \leftarrow\!\!\$\ \mathsf{KEM}.\mathrm{Encaps}(\mathsf{pk}_0)$
4 : $r_{S_0} \leftarrow \mathrm{KDF}(k; rpId)$
5 : $\tilde{k} \leftarrow\!\!\$\ \mathbb{K}$
6 : $r_{S_1} \leftarrow\!\!\$\ \mathrm{KDF}(\tilde{k}; rpId)$
7 : $b \leftarrow\!\!\$\ \{0,1\}$
8 : $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig}.\mathrm{KGen}(\mathsf{pp}; r_{S_b})$
9 : $\mathsf{cred} \leftarrow c||rpId$
10 : $b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{pk}', \mathsf{sk}', \mathsf{cred})$
11 : **return** $[\![b = b']\!]$

Figure 15: Security Experiment for the public key unlinkability property pku expanded for the instantiation ARKG

non-negligible probability, we construct a reduction B that can break IND-CPA security using $\mathcal{A}$ as a subroutine.

To increase readability, we have implemented the pku game with the algorithms from the instantiation ARKG in Figure 15. It is equivalent to the experiment defining pku security, but the algorithms have been replaced with their instantiation in ARKG.

The reduction B, given by Figure 16, works by perfectly simulating the pku security game for $\mathcal{A}$. Behavior will depend solely on the bit b of the IND-CPA game B itself is playing. Should B play in a setting where $b = 0$, it will perfectly simulate a setting for $\mathcal{A}$ where $b = 0$ and likewise for $b = 1$. Its own guess will be identical to the guess of $\mathcal{A}$, which has a non-negligible success probability. To determine the advantage of B, we separately inspect the game's behavior for the cases b=0 and b=1.

Case b=0 If the bit b in the IND-CPA game is 0, then the provided key $\hat{k}$ is the result of decapsulating the provided ciphertext c. B will generate r_S like in line 6 of the security experiment defined in Figure 12, thus providing a perfect simulation of the pku game when $b = 0$. With B simply forwarding the output of $\mathcal{A}$, the advantage of B against IND-CPA is greater or equal to the advantage of $\mathcal{A}$.

Case b=1 If the bit b of the IND-CPA game is 1, the input value $\hat{k}$ is randomly sampled from the distribution. The random key $\hat{k}$ is used as input to the KDF, resulting in a value for $r_S \leftarrow\!\!\$\ \mathrm{KDF}(\hat{k}, rpId)$ that is independent of c. This behavior corresponds exactly to lines five and six of $\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$, defined in Figure 12, thus perfectly simulating an execution of $\mathsf{Exp}^{\mathsf{pku}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$ for b=1. Therefore, the advantage of B against IND-CPA is greater or equal to the advantage of $\mathcal{A}$.

Summary Regardless of the actual value of b, the reduction B can perfectly simulate the behavior of the pku game. In both cases, the output of $\mathcal{A}$ is forwarded by B as the output of its own security game. Due to the perfect simulation, the success probability of B is at least equal to that of its subroutine $\mathcal{A}$. With KEM being IND-CPA secure, we achieve the following security bound:

$$\mathsf{Adv}^{\mathrm{pku}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM},B}(\lambda) \leq \mathsf{negl}(\lambda)$$

□

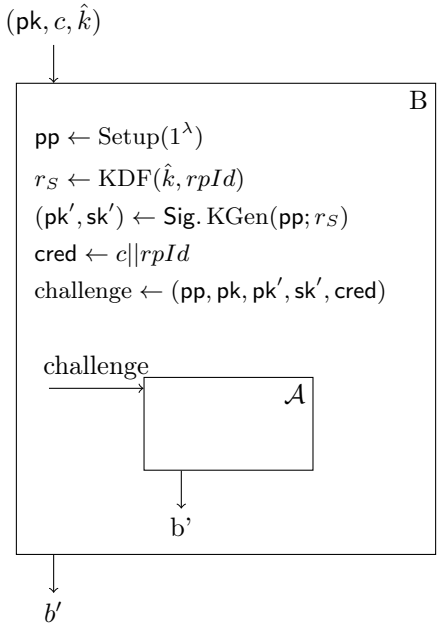


Figure 16: Reducing pku security of ARKG to IND-CPA security of KEM

7 | Discussion

This Section will address a few aspects that have been intensively discussed during the creation of our post-quantum instantiation ARKG. It should clarify some question regarding design decisions we made along the way.

7.1 Omitting the MAC and Reflection Attacks

In the original instantiation $\overline{\mathsf{ARKG}}$, cred contained a MAC in addition to the ciphertext. For our post-quantum instantiation, we have decided to omit the MAC. One of the motivations for using a MAC is to prevent so-called reflection attacks. In a reflection attack, an adversary impersonates some authentication server and tricks a user into signing a challenge. It then uses this signature to impersonate this user. This concept is described more thoroughly in [Mit89]. Note that using a MAC does not automatically prevent a reflection attack, but authenticating queries can play an important role in their prevention.

Two aspects of our scheme prevent an attacker from carrying out a reflection attack:

- The remote party id $rpId$ is part of the input to the KDF deriving randomness for the secret key. During recovery, the BA has to actively provide the identifier of the RP for which it wants to perform the recovery. In the WebAuthn specification, the $rpId$ is required to reflect the effective domain of the RP [JJM+21]. This allows for a check of the provided $rpId$ and the actual domain of the RP, which could be implemented in a future version of the WebAuthn specification. If the domain of the RP and the rpId do not match, a reflection attack is likely taking place, and the interaction should be aborted.

- The WebAuthn specification [JJM+21] requires the API for interacting with the authenticators to be used exclusively in secure contexts. This requires establishing secure channel between RP and the platform, before any interactions from $\mathcal{ARKG}$ can take place. Section 2.1.3 provides further details on this.

For the reasons outlined above, we argue that authenticity is ensured at another point in the interaction outside the scope of $\mathcal{ARKG}$. Therefore the MAC can be safely omitted.

7.2 Alternative Definition for Public Key Unlinkability

When looking at the intuition for the public key unlinkability property, formalized by the experiments pku and $\overline{\mathsf{pku}}$, one might get the feeling they do not adequately capture the unlinkability property.

While crafting the security requirements, we have explored the option of providing an adversary two initial public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ as well as a derived key pair $(\mathsf{pk}', \mathsf{sk}')$ and key handle cred, without disclosing which of the two initial public keys the game used to derive the key pair. The adversary would then have to output a guess b, indicating whether pk_0 or pk_1 was used as input to the algorithm DerivePk. Such a game would ensure that a derived key and the key handle cred hold no information on the initial public key it was derived from.

We have realized that distinguishing a derived key pair from some other derived key pair is less universal than distinguishing it from any other randomly generated key pair. For this reason, we have decided against the variation described above. Like the adversary playing against $\overline{\mathsf{pku}}$, our adversary playing against pku has to distinguish the provided key from a random key.

A nice side effect of this decision is that it simplifies the security proof. While it has not played a role in our design decision-making, the reduction becomes a lot cleaner when distinguishing between a real and a random key pair.

7.3 Secret Key Recovery

In our security definition for the derived keys, we require an attacker to recover the secret key to consider him successful. Regarding to the real world setting $\mathcal{ARKG}$ will be deployed in, this requirement might be considered too strong. An adversary can illegitimately authenticate itself without access to the secret key.

Authentication via WebAuthn is a challenge-response protocol, as covered in Section 2. The RP generates some cryptographic challenge to be signed by the authenticator and sends it to the platform. The user must then use its authenticator with the corresponding secret key to sign the challenge. To

successfully complete the challenge, a valid forgery of the signature is sufficient. Consequently, reducing the security of derived key pairs to the EUF-CMA property of some underlying signature scheme might be more precise in capturing the real world security requirements of $\mathcal{ARKG}$. However, following the security definitions introduced in [FGK+20] we have decided to stick with secret key recovery for the scope of this thesis.

Reducing the security of the derived public keys to EUF-CMA will be explored as a follow-up project to this thesis.

7.4 Multi-Device FIDO Credentials

To make using FIDO2 authenticators even more convenient, the FIDO Alliance has recently begun promoting the idea of multi-device credentials [Gla22b]. A multi-device FIDO credential, also referred to as a passkey, provides users with the option to synchronize their credentials among multiple authenticator devices. This synchronization is realized by copying the secrets across all relevant devices. For example, the platform authenticator of a user's Android device might be synchronized with the Windows 10 platform of the user's computer to allow for a seamless authentication with all services on both devices.

To facilitate the synchronization of these multi-device credentials, the FIDO Alliance has enlisted the support of the tech giants Apple, Google, and Microsoft [Gla22a]. The goal is to tie the FIDO credential to the user's Apple/Google/Microsoft account, making it available on every device where it is logged in with such an account. Using multi-device FIDO credentials trivially solves the issue of account recovery discussed throughout this thesis but comes at a cost.

Such an approach breaks with the core principle "one device, one key", to which all dedicated hardware authenticator tokens adhere. Nonetheless, no part of the FIDO2 specification prohibits sharing secret keys between multiple authenticators. Outsourcing storage of the secret key requires trust in the platform to act in the users' best interests.

For many users having ultimate control over their secret keys and not having to trust some platform is the main selling point for hardware authenticators. They are not left out in the rain, as authenticators are free to continue implementing the "one device, one key" principle. In fact, current hardware authenticators do not come with the possibility of extracting the secret key and are unaffected by the entire discussion.

To promote trust in their platforms, Apple and Google have announced employing end-to-end encryption for the storing multi-device FIDO credentials [Goo22a; App22], allowing users to retain full control over their key material. Microsoft is yet to announce its stance on this issue.

Using Asynchronous Remote Key Generation instead of multi-device FIDO credentials has additional benefits. The explicit recovery operation forces users to disclose the loss of their authenticator to the relying parties, allowing them to revoke access for the old authenticator. This forces users to actually do something about their compromised credentials. On the other hand, when losing one of the devices where the FIDO credentials are synced to, authentication could commence as usual with one of the copies. Not only does the RP never learn the user's identity could be compromised, but changing the registered keys would be very inconvenient, as all other devices would suddenly lose access until the keys are synchronized again.

After all, multi-device credentials are a trade-off between convenience and security. While some users will enjoy the simplicity of FIDO2 credentials being seamlessly integrated into their platform, others will likely choose to stick with their "one device, one key" authenticators.

8 | Conclusion

Authentication on the internet remains a challenging topic. While protocols like FIDO2 exist to supersede the traditional password, they come with their own challenges. One of these challenges is recovering access to an account in the event of authenticator loss.

This thesis has extended the work by Frymann et al. [FGK+20] and introduces a post-quantum instantiation of the primitive Asynchronous Remote Key Generation, $\mathcal{ARKG}$.

This primitive allows a user to pair two authenticators, designating one as a primary authenticator (PA) and the other as a backup authenticator (BA). In the event of authenticator loss, the BA can be used to recover access to all services where PA was registered as an authenticator after the pairing step.

Our post-quantum instantiation retains the syntax of the original instantiation, allowing for its use as a drop-in replacement. With the long lifetime of authenticator tokens and consumers' unwillingness to update a running system, the advent of quantum computers threatens the long-term security of Asynchronous Remote Key Generation (ARKG) is implemented using elliptic curves. Using a post-quantum implementation from the get-go ensures the longevity and security of the backup keys.

Additionally, we have updated the security definitions to relate to the real-world threat landscape more closely. We have provided reasoning for our modifications and related them to the definitions created by Frymann et al. We also show that our instantiation is provably secure under the new security definitions.

Our contribution enables the creation of future-proof, non-interactive backup keys that can be stored offline in a secure location until needed.

References

[AAB+19] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (Oct. 2019). Number: 7779 Publisher: Nature Publishing Group, pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: https://www.nature.com/articles/s41586-019-1666-5 (visited on 05/29/2022).

[App22] Apple. *About the security of passkeys*. Apple Support. July 6, 2022. URL: https://support.apple.com/en-us/HT213305 (visited on 07/06/2022).

[BB19] Elaine Barker et al. *Recommendation for key management:: part 2 – best practices for key management organizations*. NIST SP 800-57pt2r1. Gaithersburg, MD: National Institute of Standards and Technology, May 2019, NIST SP 800–57pt2r1. DOI: 10.6028/NIST.SP.800-57pt2r1. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf (visited on 07/09/2022).

[BBC+21] Manuel Barbosa et al. "Provable Security Analysis of FIDO2". In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 125–156. ISBN: 978-3-030-84252-9. DOI: 10.1007/978-3-030-84252-9_5.

[BCR+18] Elaine Barker et al. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. NIST Special Publication (SP) 800-56A Rev. 3. National Institute of Standards and Technology, Apr. 16, 2018. DOI: 10.6028/NIST.SP.800-56Ar3. URL: https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final (visited on 06/30/2022).

[BDK+18] Joppe Bos et al. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018 IEEE European Symposium on Security and Privacy (EuroS&P). London: IEEE, Apr. 2018, pp. 353–367. ISBN: 978-1-5386-4228-3. DOI: 10.1109/EuroSP.2018.00032. URL: https://ieeexplore.ieee.org/document/8406610/ (visited on 06/28/2022).

[Beu22] Ward Beullens. "Breaking Rainbow Takes a Weekend on a Laptop". In: *Cryptology ePrint Archive* (2022). URL: https://eprint.iacr.org/2022/214 (visited on 06/27/2022).

[BGM+96] Greg E. Blonder et al. "Transaction authorization and alert system". European pat. 0745961 (A2). At & T Corp. Dec. 4, 1996. URL: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=19961204&DB=worldwide.espacenet.com&locale=en_EP&CC=EP&NR=0745961A2&KC=A2&ND=4 (visited on 07/05/2022).

[CAJ+19] Christiaan Brand et al. *Client to Authenticator Protocol (CTAP)*. Jan. 30, 2019. URL: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html (visited on 06/30/2022).

[CJL+16] Lily Chen et al. *Report on Post-Quantum Cryptography*. NIST IR 8105. National Institute of Standards and Technology, Apr. 2016, NIST IR 8105. DOI: 10.6028/NIST.IR.8105. URL: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf (visited on 06/20/2022).

[CML17] Lily Chen et al. *Call for Proposals - Post-Quantum Cryptography | CSRC | CSRC*. CSRC | NIST. Jan. 3, 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/call-for-proposals (visited on 06/30/2022).

[DBN+01] Morris J. Dworkin et al. "Advanced Encryption Standard (AES)". In: *NIST* (Nov. 26, 2001). Last Modified: 2021-03-01T01:03-05:00 Publisher: Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, James F. Dray Jr. URL: https://www.nist.gov/publications/advanced-encryption-standard-aes (visited on 07/10/2022).

[Deu19] Deutsche Bundesbank. *PSD2*. 2019. URL: https://www.bundesbank.de/de/aufgaben/unbarer-zahlungsverkehr/psd2/psd2-775434 (visited on 06/30/2022).

[DS05] Jintai Ding et al. "Rainbow, a New Multivariable Polynomial Signature Scheme". In: *Applied Cryptography and Network Security*. Ed. by John Ioannidis et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 164–175. ISBN: 978-3-540-31542-1. DOI: 10.1007/11496137_12.

[FGK+20] Nick Frymann et al. "Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. New York, NY, USA: Association for Computing Machinery, Oct. 30, 2020, pp. 939–954. ISBN: 978-1-4503-7089-9. DOI: 10.1145/3372297.3417292. URL: https://doi.org/10.1145/3372297.3417292 (visited on 12/01/2021).

[FO99] Eiichiro Fujisaki et al. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 537–554. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_34.

[GE21] Craig Gidney et al. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (Apr. 15, 2021), p. 433. ISSN: 2521-327X. DOI: 10.22331/q-2021-04-15-433. arXiv: 1905.09749[quant-ph]. URL: http://arxiv.org/abs/1905.09749 (visited on 05/29/2022).

[GH18] Iness Guirat et al. "Formal verification of the W3C web authentication protocol". In: HoTSoS '18: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security. Apr. 10, 2018, pp. 1–10. ISBN: 978-1-4503-6455-3. DOI: 10.1145/3190619.3190640.

[Gla22a] Lori Glavin. *Apple, Google and Microsoft Commit to Expanded Support for FIDO Standard to Accelerate Availability of Passwordless Sign-Ins*. FIDO Alliance. May 5, 2022. URL: https://fidoalliance.org/apple-google-and-microsoft-commit-to-expanded-support-for-fido-standard-to-accelerate-availability-of-passwordless-sign-ins/ (visited on 07/04/2022).

[Gla22b] Lori Glavin. *White Paper: Multi-Device FIDO Credentials*. FIDO Alliance. Mar. 17, 2022. URL: https://fidoalliance.org/white-paper-multi-device-fido-credentials/ (visited on 07/04/2022).

[Goo22a] Google. *FIDO authentication with passkeys | Google Identity | Google Developers*. June 21, 2022. URL: https://developers.google.com/identity/fido (visited on 07/06/2022).

[Goo22b] Google. *Google Authenticator - Apps on Google Play*. 2022. URL: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=en&gl=US (visited on 07/13/2022).

[Goo22c] Google. *Google Quantum AI Hardware*. Google Quantum AI. 2022. URL: https://quantumai.google/hardware (visited on 06/30/2022).

[Gro96] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *arXiv:quant-ph/9605043* (Nov. 19, 1996). arXiv: quant-ph/9605043. URL: http://arxiv.org/abs/quant-ph/9605043 (visited on 12/01/2021).

[IBM15] IBM. *IBM Quantum Computing*. Oct. 1, 2015. URL: https://www.ibm.com/quantum (visited on 06/30/2022).

[IBM21] IBM. *IBM Survey: Pandemic-Induced Digital Reliance Creates Lingering Security Side Effects*. IBM Newsroom. June 15, 2021. URL: https://newsroom.ibm.com/2021-06-15-IBM-Survey-Pandemic-Induced-Digital-Reliance-Creates-Lingering-Security-Side-Effects (visited on 07/04/2022).

[IBM22] IBM Research. *IBM Quantum roadmap to build quantum-centric supercomputers | IBM Research Blog*. 2022. URL: https://research.ibm.com/blog/ibm-quantum-roadmap-2025 (visited on 06/30/2022).

[IDm] IDmelon. *IDmelon Authenticator*. IDmelon Technologies Inc. URL: https://www.idmelon.com/idmelon-authenticator-details/ (visited on 07/13/2022).

[Inf13] Information Technology Laboratory. *Digital Signature Standard (DSS)*. NIST FIPS 186-4. National Institute of Standards and Technology, July 2013, NIST FIPS 186–4. DOI: 10.6028/NIST.FIPS.186-4. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (visited on 07/06/2022).

[JJM+21] Jeff Hodges et al. *Web Authentication: An API for accessing Public Key Credentials - Level 2*. Apr. 8, 2021. URL: https://www.w3.org/TR/webauthn-2/#relying-party-identifier (visited on 06/29/2022).

[JK21] Charlie Jacomme et al. "An Extensive Formal Analysis of Multi-factor Authentication Protocols". In: *ACM Transactions on Privacy and Security* 24.2 (Feb. 2021), pp. 1–34. ISSN: 2471-2566, 2471-2574. DOI: 10.1145/3440712. URL: https://dl.acm.org/doi/10.1145/3440712 (visited on 07/04/2022).

[JMM+22] David Joseph et al. "Transitioning organizations to post-quantum cryptography". In: *Nature* 605.7909 (May 2022). Number: 7909 Publisher: Nature Publishing Group, pp. 237–243. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04623-2. URL: https://www.nature.com/articles/s41586-022-04623-2 (visited on 06/05/2022).

[KBC97] Hugo Krawczyk et al. *HMAC: Keyed-Hashing for Message Authentication*. Request for Comments RFC 2104. Num Pages: 11. Internet Engineering Task Force, Feb. 1997. DOI: 10.17487/RFC2104. URL: https://datatracker.ietf.org/doc/rfc2104 (visited on 07/07/2022).

[KE10] Hugo Krawczyk et al. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. Request for Comments RFC 5869. Num Pages: 14. Internet Engineering Task Force, May 2010. DOI: 10.17487/RFC5869. URL: https://datatracker.ietf.org/doc/rfc5869 (visited on 07/06/2022).

[KL14] Jonathan Katz et al. *Introduction to Modern Cryptography, Second Edition*. 2nd. Chapman & Hall/CRC, 2014. 603 pp. ISBN: 978-1-4665-7026-9.

[LDB15] Rolf Lindemann et al. *FIDO Technical Glossary*. May 14, 2015. URL: https://fidoalliance.org/specs/u2f-specs-1.0-bt-nfc-id-amendment/fido-glossary.html (visited on 07/05/2022).

[Lun20] Emil Lundberg. *Yubico proposes WebAuthn protocol extension to simplify backup security keys*. Yubico. Nov. 16, 2020. URL: https://www.yubico.com/blog/yubico-proposes-webauthn-protocol-extension-to-simplify-backup-security-keys/ (visited on 06/27/2022).

[Mit89] C. Mitchell. "Limitations of challenge-response entity authentication". In: *Electronics Letters* 25.17 (1989), p. 1195. ISSN: 00135194. DOI: 10.1049/el:19890801. URL: https://digital-library.theiet.org/content/journals/10.1049/el_19890801 (visited on 06/29/2022).

[Moo22] Dustin Moody. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST IR 8413. Gaithersburg, MD: National Institute of Standards and Technology, 2022, NIST IR 8413. DOI: 10.6028/NIST.IR.8413. URL: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf (visited on 07/10/2022).

[Nat99] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. Federal Information Processing Standard (FIPS) 46-3 (Withdrawn). U.S. Department of Commerce, Oct. 25, 1999. URL: https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25 (visited on 06/30/2022).

[Pia21] Marco Piani. "QUANTUM THREAT TIMELINE REPORT 2021". In: (2021), p. 87.

[PMN+17] Christoforos Panos et al. "A Security Evaluation of FIDO's UAF Protocol in Mobile and Embedded Devices". In: *Digital Communication. Towards a Smart and Secure Future Internet*. Ed. by Alessandro Piva et al. Communications in Computer and Information Science. Cham: Springer International Publishing, 2017, pp. 127–142. ISBN: 978-3-319-67639-5. DOI: 10.1007/978-3-319-67639-5_11.

[Res18] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Request for Comments RFC 8446. Num Pages: 160. Internet Engineering Task Force, Aug. 2018. DOI: 10.17487/RFC8446. URL: https://datatracker.ietf.org/doc/rfc8446 (visited on 06/30/2022).

[SDE+17] Sampath Srinivas et al. *Universal 2nd Factor (U2F) Overview*. Apr. 11, 2017. URL: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html (visited on 07/17/2022).

[Shi] Andrew Shikiar. *FIDO2: Web Authentication (WebAuthn)*. FIDO Alliance. URL: https://fidoalliance.org/fido2-2/fido2-web-authentication-webauthn/ (visited on 07/05/2022).

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. arXiv: quant-ph/9508027. URL: http://arxiv.org/abs/quant-ph/9508027 (visited on 12/01/2021).

[Sin19] Anna Sinitsyna. *Beyond Passwords: FIDO2 and WebAuthn in Practice*. inovex GmbH. Dec. 16, 2019. URL: https://www.inovex.de/de/blog/fido2-webauthn-in-practice/ (visited on 06/04/2022).

[SK19] Fatima Salahdine et al. "Social Engineering Attacks: A Survey". In: *Future Internet* 11 (Apr. 2, 2019). DOI: 10.3390/fi11040089.

[Tec15a] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. Federal Information Processing Standard (FIPS) 180-4. U.S. Department of Commerce, Aug. 4, 2015. DOI: 10.6028/NIST.FIPS.180-4. URL: https://csrc.nist.gov/publications/detail/fips/180/4/final (visited on 07/06/2022).

[Tec15b] National Institute of Standards and Technology. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standard (FIPS) 202. U.S. Department of Commerce, Aug. 4, 2015. DOI: 10.6028/NIST.FIPS.202. URL: https://csrc.nist.gov/publications/detail/fips/202/final (visited on 07/10/2022).

[Tre22] Trezor. *Trezor Hardware Wallet (Official) | The original and most secure hardware wallet.* 2022. URL: https://trezor.io/ (visited on 07/11/2022).

[VMH+05] Mountain View et al. *HOTP: An HMAC-Based One-Time Password Algorithm*. Request for Comments RFC 4226. Num Pages: 37. Internet Engineering Task Force, Dec. 2005. DOI: 10.17487/RFC4226. URL: https://datatracker.ietf.org/doc/rfc4226 (visited on 06/30/2022).

[VRP+11] Mountain View et al. *TOTP: Time-Based One-Time Password Algorithm*. Request for Comments RFC 6238. Num Pages: 16. Internet Engineering Task Force, May 2011. DOI: 10.17487/RFC6238. URL: https://datatracker.ietf.org/doc/rfc6238 (visited on 06/30/2022).

[Wes21] Mike West. *Secure Contexts*. Aug. 18, 2021. URL: https://w3c.github.io/webappsec-secure-contexts/#secure-contexts (visited on 07/05/2022).

[Yuba] Yubico. *Spare YubiKeys*. Yubico. URL: https://www.yubico.com/spare/ (visited on 07/06/2022).

[Yubb] Yubico. *USB-A YubiKey 5 NFC Two Factor Security Key*. Yubico. URL: https://www.yubico.com/product/yubikey-5-nfc (visited on 07/17/2022).

[Yub22] Yubico. *What is a Sim Swap?* Yubico. 2022. URL: https://www.yubico.com/resources/glossary/sim-swap/ (visited on 06/30/2022).