Post-Quantum Asynchronous Remote Key Generation for FIDO2 Account Recovery

Jacqueline Brendel
Technical University Darmstadt
jacqueline.brendel@tu-darmstadt.de

Sebastian Clermont
Technical University Darmstadt
sebastian.clermont@tu-darmstadt.de

Marc Fischlin
Technical University Darmstadt
marc.fischlin@tu-darmstadt.de

ABSTRACT

The Fast IDentity Online (FIDO) Alliance develops open standards to replace password-based authentication methods by token-based solutions. The latest protocol suite FIDO2 provides such a promising alternative which many key players have already adopted or are willing to. The central authentication mechanism WebAuthn uses cryptographic keys stored on the device to authenticate clients to a relying party via a challenge-response protocol. Yet, this approach leaves several open issues about post-quantum secure instantiations and methods for recovery of credentials.

Recently Frymann et al. (CCS 2020, ACNS 2023, EuroS&P 2023) made significant progress to advance the security of FIDO2 systems. Following a suggestion by device manufacturer Yubico, they considered a WebAuthn-compliant mechanism to store recovery information at the relying party. If required, the client can recover essential data with the help of a backup authenticator device. They analyzed the Diffie-Hellman based scheme, showing that it provides basic authentication and privacy features. One of their solutions also provides a post-quantum secure variant, but only for a weaker version of authentication security.

Our starting point is to note that the security definitions of Frymann et al., especially the privacy notion, do not seem to capture real threats appropriately. We thus strengthen the notions. Despite this strengthening, we show a generic construction based on (anonymous) KEMs and signature schemes. It follows that, using post-quantum secure instances, like Kyber and Dilitihium, one immediately obtains a post-quantum and strongly secure solution.

KEYWORDS

FIDO2, Webauthn, post-quantum, account recovery, multifactor authentication

1 INTRODUCTION

FIDO2 encompasses a set of pioneering industry standards for passwordless authentication on the web [4, 15]. The approach replaces passwords by a sophisticated combination of public-key cryptography and so-called authenticators (e.g., smart phones or dedicated security keys). The authenticator is the (hardware) device holding the secret key material used for authentication. The client is the device which a user is using to authenticate themselves towards a service, and the relying party is the service which wants to confirm the user's identity. For example, if a user is logging into its Google Account using its computer with a Yubico FIDO2 key, then the Yubico FIDO2 key is the authenticator, its computer is the client, and Google is the relying party.

In a FIDO2 authentication ceremony, users authenticate themselves by proving that they control the key of a previously-registered public key credentials. This is done by having the authenticator produce a valid cryptographic signature on a challenge and associated data, issued by the relying party. The set of cryptographic information required by a user is referred to as FIDO2 credential or as a passkey [23].

Post-Quantum Efforts. Migrating the FIDO2 set of standards to a post-quantum setting is an ongoing effort. The latest protocol suite FIDO2 was recently analyzed cryptographically [1] and with formal methods in [12], with extensions of the results in terms of capturing other protocol versions and modes, as well as advanced security features, in [3, 13]. In particular, [13] discusses privacy features and a protocol modification via blockchains to achieve revocation. The work by Bindel et al. [3] analyzes a full post-quantum instantiation of the latest iteration of the FIDO2 standards CTAP 2.1 and WebAuthn 2. While requiring minor extensions to the protocol, the instantiation provided is provably secure. This shows that the functionality of the FIDO2 protocol family can be achieved securely without any classical hardness assumptions. The post-quantum security of the underlying protocol provides the foundation for our extension to include backup mechanisms withstanding quantum adversaries, too.

Recent work by Frymann et al. [8] provides an alternative generic instantiation of ARKG which also achieves post-quantum security. Their core building block are split-KEMs, first introduced by Brendel et al. in [5]. Unfortunately, the instantiation of such split-KEMs from lattice primitives is not well researched yet. While the original authors, as well as Frymann et al., provide instantiations based on Kyber and Frodo, they only achieve the weakest security notion for split-KEMs, nn-IND-CCA. This does not allow for ARKG instantiations achieving resilience against adaptive attackers. Our solution uses only well-established building blocks while achieving all relevant security goals.

1.1 Authentication Recovery

Involving authentication on a device such as a smart phone imperatively requires to consider the possibility to transfer or recover login credentials.

For the secure recovery of symmetric secrets, mechanisms such as Signal's *Secure Value Recovery* or Apple's *Secure iCloud Keychain Recovery*. These protocols rely on a low-entropy recovery PIN created by a user which encrypts a high-entropy key stored on a distributed HSM. This high-entropy key is used for the encryption of the users secrets. Recovery of the high-entropy key is governed by the HSM, which only allows for a limited amount of retries to prevent brute-force attacks. This approach is unsatisfactory, as the entire security relies on a low-entropy PIN chosen by a users and

Conference'17, July 2017, Washington, DC, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

requires trust in cloud vendors to handle the encryption secrets as promised.

Dedicated hardware authenticators, such as FIDO2 devices, usually generate their cryptographic secrets locally and do not allow for their extraction from the device. This renders the aforementioned approach unusable.

There are two scenarios for thee migration of hardware authenticators. One is the graceful transition from an old device to a new one, when the previous authentication data is still accessible. The other, more challenging situation occurs in case of lost, stolen, or broken authenticators when the original credentials are no longer available.

A hardware-based solution to the loss of devices is to use multi-device credentials, as suggested for instance by the FIDO2 Alliance [23]. Multi-device credentials are not constrained to the physical authenticator they were generated on, but can be copied across a user's devices to ensure a consistent login experience. This straightforwardly solves the recovery problem, as a user can have multiple, functionally identical authenticators. In case one of them is lost, a new authenticator can be acquired and the multi-device credentials can be synchronized to this new authenticator. While this is an easy-to-use approach, it does come with numerous security drawbacks. First of all, revocation of a lost authenticator is not possible, as all authenticators behave identically. Then, a user can never be certain of being sole owner of their private key, as it might have been copied to a different device without his knowledge or consent, since hardware binding is impossible in this setting. Lastly, for some high-security use cases, the relying party may insist on single-device passkeys.

As a manufacturer of hardware security devices, Yubico takes the stance that allowing secret material to leave the security devices is an inherent weakness to the system and should be avoided [14]. As a result, Yubico does not support the creation of multi-device FIDO2 passkeys and strictly follows a one-device one-credential approach. Hence, in order to deal with lost or broken devices, Yubico suggests an alternative approach with so-called backup authenticators. Basically, such a backup authenticator is a hardware device which is initialized once (creating a cryptographic key pair), but then goes offline and is disconnected from any subsequent interactions, similar to a cold wallet. The (single-device) authenticator uses the backup authenticator's public key to store some protected recovery information at the relying party when registering. Only if recovery should take place, the secret key of backup authenticator can be used to extract the authentication data from the externally stored recovery information. Note that, unlike Google's Authenticator, the externally stored recovery information does not grant the relying party access to the secrets; this is only possible when holding the backup authenticator.

Yubico proposed a Diffie-Hellman based protocol for backup authenticators. The proposal has been analyzed by Frymann et al. [6] under the term asynchronous remote key generation (ARKG). Frymann et al. also gave alternative protocols based on pairings [10] and a proposal how one could achieve a post-quantum secure version [9] based on split KEMs [5]. These works are the starting point of our approach.

1.2 Security Notions for Backup Authenticators

The idea of backup authenticators as proposed in combination with the ARKG protocol is not part of the FIDO2 standard at this time and as such has no established security definitions. Frymann et al. [6] define two security notions for ARKG protocols. One is authentication security, called secret-key security or simply key security, which says that the adversary cannot get hold of the secret key of honest users. This notion comes in slightly different flavors, depending on whether the adversary can communicate with the backup authenticator or not (strong vs. weak), and whether the adversary needs to attack a given public key or can attempt to fool the relying party with a chosen public key (honest vs. malicious). The other property is public-key unlinkability which should prevent relying parties to link users via the backup authenticator's public keys. This is captured by an indistinguishability notion where one learns a backup authenticator's long-term public key and either receives derived keys under the long-term public key or independently generated keys. The definitions in [6, 9, 10] all coincide.

Our first observation is that both security definitions are too restrictive from our point of view to adequately capture threats. Key security, as defined in [6], requires the adversary to find the user's secret key for a successful attack. However, authentication in FIDO2 would already be broken if the adversary is able to forge a signature under the user's public key. This is similar to security of signature schemes where the notion of existential unforgeability is paramount, while the stronger requirement of key recovery is often not considered.

As in the case of key security, the privacy definition of [6] gives the adversary not enough power. The definition only gives the adversary access to the backup authenticator's public key in order to distinguish. But relying parties, aiming to link users via the the backup authenticator data, also store the recovery information. Individual relying parties, and even more so colluding relying parties, could potentially use the recovery data to identify users. Such attacks are currently not captured in the public-key unlinkability model in [6].[1]

1.3 Protocols for Backup Authenticators

Yubico's orginal Diffie-Hellman based protocol [17] roughly lets the backup authenticator and the primary authenticator each generate the public part of a DH share. The primary authenticator stores its public DH part as recovery information at the relying party. In case of a recovery request the backup authenticator can compute the joint DH key with the help of its secret DH part and the primary authenticator's externally stored recovery information. Note that the primary authenticator also registers a separate key pair for authentication.

The original Yubico proposal attached a message authentication code (MAC) to the recovery information. This MAC allows to identify the relevant recovery credentials in case of a backup recovery, so rather works as a checksum. In particular, the MAC does not enter the security statements for key security nor for privacy. Nor does it seem to give protection against malicious behavior, since the

[1] For instance, in Google's Authenticator case where the backup authenticator's secret is available to Google, it is very easy to distinguish different public keys, while given only the public key as in the suggested experiment, it remains hard.

MAC key is derived from the joint DH key, and can thus be easily derived by an adversary injecting its DH share. The MAC appears in all protocol versions, the initial discrete-log based solution [6], the pairing based solution [10], as well as in the post-quantum based approach [9].

The post-quantum ARKG protocol in [9] is based on the split key encapsulation mechanism (KEM) approach in [5], and specifically also on the LWE problem. Besides the aforementioned deficiency with respect to the security model, the solution in the malicious key security case is currently not substantiated by concrete schemes. While the honest key security case only requires an IND-CPA secure split KEM —which we know how to build— the malicious case demands an IND-CCA secure split KEM —for which we currently do not have promising candidates, as pointed out in [5]. Hence, it remains unclear if one can actually derive post-quantum ARKG protocols with regards to appropriate security models.

1.4 Our Contributions

We set off by giving stronger security notions for key security and public-key unlinkability. For security we now only demand that the adversary cannot produce signature forgeries under backup keys and thus cannot register new keys on behalf of the user. For public-key unlinkability we follow a left-or-right approach where the adversary cannot decide to which of two backup keys generated public keys and recovery information belong to. This extends the previous definition to now also include the recovery data. In the course of this we slightly adapt the names of the security properties to authentication security and unlinkability, since the first property does not protect the secret key anymore but prevents forgeries, and the privacy property now also takes other available data beyond the public key into account.

We then propose a protocol based on (ordinary) KEMs and signature schemes. The idea is to let the authenticator generate the key pair of a signature scheme, encapsulate the key-generating randomness under the backup authenticator's public key, and to store this ciphertext externally at the relying party. In case of recovery, the backup authenticator can retrieve the randomness and re-generate the signing key.

As for our strongest notion of authentication security we need IND-CCA security of the KEM and EUF-CMA of the signature scheme (and pseudorandomness of an intermediate key deriviation function for generating the key generation randomness from the encapsulated KEM key). Note that these are standard properties of schemes. In particular, we do not rely on split KEMs. Due to the recent PQC standardization effort of NIST, appropriate candidates like Kyber [22] for the KEM and Dilithium [18], Falcon [21], or SPHINCS+ [16] for the signature scheme exist. They are all proven to satisfy the required security properties under reasonable assumptions.

For our stronger notion of unlinkability we also draw on a property called IND-CCA anonymity of the KEM scheme, meaning that one cannot distinguish ciphertexts created under one public key from ones created under another public key. This notions has recently been considered in more detail for the case of post-quantum schemes [11, 24]. Fortunately, the aforementioned Kyber KEM also supports this property [20].

2 ASYNCHRONOUS REMOTE KEY GENERATION

The primitive of Asynchronous Remote Key Generation (ARKG) was first proposed by Frymann et al. in [6] in the context of Yubico's proposal to enable recovery of accounts protected by FIDO2 authenticators in case of authenticator loss. Their initial instantiation in [6] was based on elliptic curves, but since then a construction based on pairings [9] and a post-quantum version based on lattices [10] have been introduced.

2.1 Notation

We write $y \leftarrow \mathsf{Alg}(x)$ and $y \leftarrow\!\!\$\ \mathsf{Alg}(x)$ for the deterministic, resp. probabilistic execution of an algorithm Alg on input x with output y. We write $\mathsf{prefix}(x) = y$ to indicate that y is a prefix of x. We usually assume classical algorithms for implementing schemes, with the usual notion of efficiency if the algorithms run in polynomial-time in the length of the security parameter λ. Explicit randomness is indicated in an algorithm's input using a semicolon. For instance, $\mathsf{Sign}(\mathsf{sk}, m; r)$ denotes the execution and outcome of the signing algorithm with secret key sk on m when run with randomness r.

We assume quantum polynomial-time (QPT) adversaries $\mathcal{A}$ and say that the adversary is efficient if it runs in QPT in the security parameter λ. Since the honest parties use classical algorithms the adversary can only interact classically with honest parties (Q1 setting). We write $\mathcal{A}^{O}$ to denote that the adversary $\mathcal{A}$ has access to the oracle O. We use the dot notation, denoted by $\cdot$, to represent required input to an algorithm. For example, $\mathsf{O}(\cdot,\cdot)$ denotes that the algorithm O takes two input parameters.

We further use $y \leftarrow x$ to denote the assignment of a value x to a variable y. In security games we use $[\![\text{expression}]\!]$ to denote boolean evaluation of an expression *expression*. The special symbol $\perp$ shall denote rejection or an error, usually output by an algorithm; in particular $\perp \notin \{0,1\}^*$.

2.2 Terminology

In the FIDO2 context, services are referred to as *relying parties* (*RP*) to which users can authenticate via so-called *authenticators*. For our analysis here, we identify a user with its authenticator(s) and abstract away the client that sits between the authenticator and the relying party. When ARKG is employed, authenticators are split into two different classes: *backup* authenticators (*BA*), which hold the long-term secrets $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$ and are used for account recovery, and *primary* authenticators (*PA*) which derive (public) keys pk' and recovery information rec from the long-term public key and are used to authenticate the user to relying parties.

2.3 Syntax

We now recall the notion of asynchronous remote key generation schemes as introduced by Frymann et al. [6] but slightly change notation of the involved algorithms to make it more aligned with the intended purpose. We also consider ARKG schemes in the setting of common parameters pp generated by the Setup algorithm. We assume that the backup authenticator generates a key pair $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$ via algorithm KGen. Key pairs for the authenticator are denoted as $(\mathsf{pk}, \mathsf{sk})$ and are generated together with recovery information rec via algorithm DerivePK in such a way that allows the backup authenticator to recover the secret key with the help of $\mathsf{sk}_{\mathsf{BA}}$ via algorithm DeriveSK. The former algorithm has been denoted as DerivePK in [6] and the latter DeriveSK. Both algorithms were linked through an algorithm Check to identify matching public and secret keys. We omit the latter here and assume that DeriveSK re-establishes the correct secret key. Instead of calling the recovery information a *credential* denoted by cred as in [6] we call it recovery information, or rec, for short, resembling the externally stored session resumption data in TLS.

*Definition 2.1 (*ARKG*).* A scheme for asynchronous remote key generation, or ARKG for short, consists of four algorithms (Setup, KGen, DerivePK, DeriveSK, Check) such that

Setup takes as input the security parameter λ in unary and outputs the public parameters, i.e., $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$.

KGen takes as input the public parameters pp and output a public/secret key pair $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ for the backup authenticator.

DerivePK takes as input the public parameters pp, a public key $\mathsf{pk}_{\mathsf{BA}}$ and auxiliary information *aux*[2] and outputs a derived public key pk' and associated credential information rec, i.e., $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$.

DeriveSK takes as input the public parameters pp, a secret key $\mathsf{sk}_{\mathsf{BA}}$ and recovery information rec. It outputs either a secret key sk', i.e., $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$, or the dedicated symbol $\perp$, in case no valid sk' can be computed for pk' associated with rec.

Check takes as input a public-secret key pair (pk, sk) and returns 1 if (pk, sk) forms a valid public/secret key pair, and 0 otherwise.

We say that an asynchronous remote key generation scheme ARKG = (Setup, KGen, DerivePK, DeriveSK, Check) is ϵ*-correct*, if for all λ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$ and $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, and auxiliary information *aux* we have $\Pr[\mathsf{Check}(\mathsf{pk}', \mathsf{sk}') = 0] \leq \epsilon$ where we have $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$ as well as $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$.

If the scheme is ϵ-correct for $\epsilon = 0$ then we say that the scheme is (perfectly) correct.

Frymann et al. include the algorithm Check(pk, sk) as part of their ARKG syntax, which is necessary to define correctness in the ARKG setting. In public-key cryptography, you can always leverage the randomness that went into key generation to implement such a check. That is, we define the secret key to be the randomness during key generation and, if required, reconstruct the actual secret key by re-running key generation. Then one can easily check that the public key matches given the randomness as secret key. Indeed, our generic construction follows this approach, such that we do not give a concrete instantiation of Check.

We defer the discussion of security properties of ARKG schemes to Section 3.

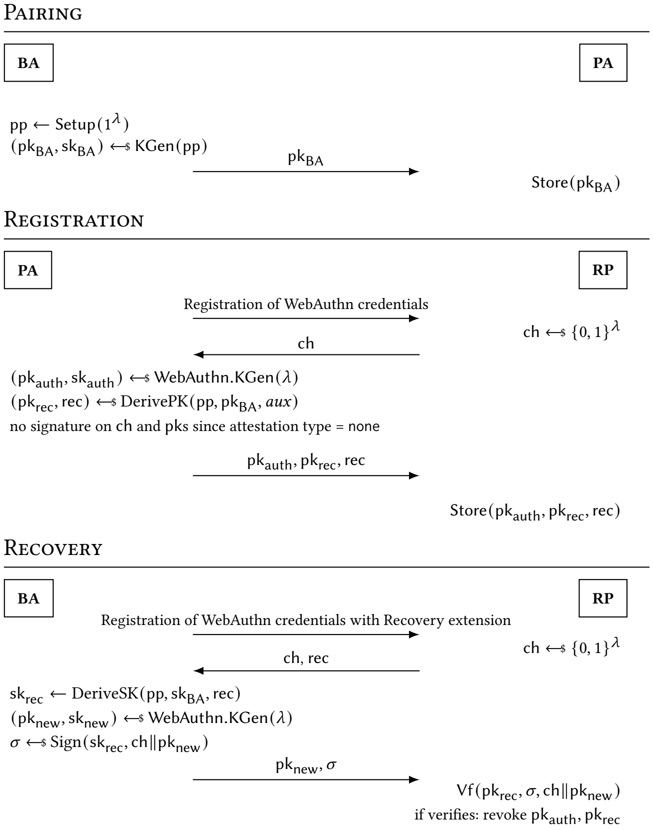

Pairing
BA
PA
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$
$(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
$\mathsf{pk}_{\mathsf{BA}}$
$\mathsf{Store}(\mathsf{pk}_{\mathsf{BA}})$
Registration
PA
RP
Registration of WebAuthn credentials
$\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
ch
$(\mathsf{pk}_{\mathsf{auth}}, \mathsf{sk}_{\mathsf{auth}}) \leftarrow\!\!\$\ \mathsf{WebAuthn.KGen}(\lambda)$
$(\mathsf{pk}_{\mathsf{rec}}, \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$
no signature on ch and pks since attestation type = none
$\mathsf{pk}_{\mathsf{auth}}, \mathsf{pk}_{\mathsf{rec}}, \mathsf{rec}$
$\mathsf{Store}(\mathsf{pk}_{\mathsf{auth}}, \mathsf{pk}_{\mathsf{rec}}, \mathsf{rec})$
Recovery
BA
RP
Registration of WebAuthn credentials with Recovery extension
$\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
ch, rec
$\mathsf{sk}_{\mathsf{rec}} \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
$(\mathsf{pk}_{\mathsf{new}}, \mathsf{sk}_{\mathsf{new}}) \leftarrow\!\!\$\ \mathsf{WebAuthn.KGen}(\lambda)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}_{\mathsf{rec}}, \mathsf{ch} \| \mathsf{pk}_{\mathsf{new}})$
$\mathsf{pk}_{\mathsf{new}}, \sigma$
$\mathsf{Vf}(\mathsf{pk}_{\mathsf{rec}}, \sigma, \mathsf{ch} \| \mathsf{pk}_{\mathsf{new}})$
if verifies: revoke $\mathsf{pk}_{\mathsf{auth}}, \mathsf{pk}_{\mathsf{rec}}$

Figure 1: Simplified illustration how ARKG integrates into WebAuthn registration flows

2.4 ARKG in the Context of FIDO2

As also mentioned by Frymann et al., the primitive of asynchronous remote key generation may be applicable to use cases outside of account recovery for FIDO2. Most notably, privacy-preserving proxy signatures with unlinkable warrants can be generically constructed from ARKG [7]. For this paper, we focus on account recovery via ARKG in the context of FIDO2 to allow for a meaningful analysis.

To substantiate our choice of security notions for ARKG, which we will present in detail in the following Section 3, we briefly describe how web authentication via WebAuthn in FIDO2 works [15] and how ARKG fits into this flow. On a high level, the role of the ARKG primitive within the context of FIDO2 account recovery is two-fold:

1. On the *PA*: To create a signature key pair and recovery information from the *BA*'s long-term public key to register with relying parties such that no interaction with the *BA* is necessary to do so.
2. On the *BA*: Use the long-term secret and the recovery information from relying parties to derive a signing key to authenticate to the respective relying party and recover account access.

[2] We assume that in the context of FIDO2 account recovery as treated in this paper, *aux* contains at least a unique identifier *rpID* that uniquely identifies for which relying party the public key and credential are derived

A simplified illustration of this is given in Figure 1, which we will elaborate on briefly:

2.4.1 Pairing. At the beginning, asynchronous remote key generation requires that the backup authenticator is paired with a primary authenticator.[3] During the pairing process, the long-term public key $\mathsf{pk}_{\mathsf{BA}}$ of the backup authenticator is transferred to the primary authenticator which stores it to derive keys from.

2.4.2 Registration. At some point, the primary authenticator then begins to register credentials with relying parties. This registration happens inside a secure channels since the user has logged into the relying party via another authentication method, typically with user name and password and has established a TLS connection to the server of the RP.

In a "normal" WebAuthn registration, the relying party sends a challenge value to the authenticator and the authenticator then derives a key pair $(\mathsf{pk}_{\mathsf{auth}}, \mathsf{sk}_{\mathsf{auth}})$ and sends $\mathsf{pk}_{\mathsf{auth}}$ to the relying party. Depending on the chosen attestation type, the authenticator's response may also include a signature on a message with contains (among other information) the challenge ch and the new public key $\mathsf{pk}_{\mathsf{auth}}$). The signature is established with the long-term secret key that is embedded in the authenticator at production time. Since no attestation is the proposed default, we choose to omit this signature from our simplified illustration.

When the recovery extension is present, the primary authenticator will also derive a recovery public key $\mathsf{pk}_{\mathsf{rec}}$ and recovery information rec from $\mathsf{pk}_{\mathsf{BA}}$ via the ARKG algorithm DerivePK. $(\mathsf{pk}_{\mathsf{rec}}, \mathsf{rec})$ are then also transmitted to the relying party.

WebAuthn authentication. From then on, the user can use its primary authenticator with the secret $\mathsf{sk}_{\mathsf{auth}}$ to sign WebAuthn authentication challenges in a passwordless manner. ARKG is not involved in this phase, thus we did not include it in the figure. As usual, this happens via a challenge-response protocol in which the relying party sends a challenge value to the user, and the user then signs the challenge with the secret key stored on the authenticator. The RP then verifies the signature with respect to the public key it had received during registration. If the signature verifies, the user is authenticated and is logged onto the service.

2.4.3 Recovery. While the primary authenticator acts as the "standard" authenticator of the user when signing in to services, the backup authenticator comes into play should the user lose access to its *PA*. Until that point the *BA* can be stored offline.

Note that the recovery process is a regular WebAuthn registration ceremony with the recovery extension. Thus, the flow is again as in 2.4.2. When the recovery is triggered by the *BA*, the relying party sends out a challenge ch to the authenticator along with recovery information rec for the user in question. The *BA* then uses its long-term secret $\mathsf{sk}_{\mathsf{BA}}$ to recover the derived secret key $\mathsf{sk}_{\mathsf{rec}}$ associated with rec. It then generates a new key pair $(\mathsf{pk}_{\mathsf{new}}, \mathsf{sk}_{\mathsf{new}})$ to replace the lost $(\mathsf{pk}_{\mathsf{auth}}, \mathsf{sk}_{\mathsf{auth}})$ and signs (among other information) the new public key $\mathsf{pk}_{\mathsf{new}}$ and the challenge provided by rec. It sends the new public key and the signature to the relying party. The *RP* then verifies the signature with respect to its stored information and if it verifies store the new public key and should revoke the old credentials for authentication and recovery.

3 SECURITY OF ARKG SCHEMES

In this section we discuss the security properties of asynchronous remote key generation schemes. When introducing the ARKG primitive and in later works, Frymann et al. [6, 9, 10] described security of ARKG schemes in terms of an adversary's inability to recover a derived secret key in various adversarial settings (honest/malicious, weak/strong) and the unlinkability of derived public keys. The former aims to guarantee that an adversary is not able to successfully complete the account recovery process without access to the secret keys stored on the backup authenticator, whereas public-key unlinkability shall ensure that users cannot be tracked across services via their registered public-key credentials.

As explained in the introduction we choose to assume different security properties which, we believe, capture the real-world setting for ARKG usage in FIDO2 account recovery more adequately than the ones in the original work. The formal definitions by Frymann et al. [6] and a more in-depth comparison with our security notions can be found in Appendix A.

In particular, we deem their key security notions to be too restrictive and switch to a notion based on the adversary's (in)ability to successfully *authenticate* to relying parties during the account recovery process instead of the adversary's (in)ability to recover an entire secret key.

With regards to public-key unlinkability, we note that the definition by Frymann et al., which states that derived keys are indistinguishable from randomly sampled keys, does not take the adversary's actual view during the execution of the protocol into account. This omission gives a false sense of security: As we explain in Appendix A, one can have ARKG schemes that provide public-key unlinkability wrt. Frymann et al.'s definition that trivially link derived public keys when employed in the envisioned setting.

But before we formally define our security properties for ARKG schemes, we first state our basic assumptions on the adversary's power and capabilities.

3.1 Adversarial model

Recall that we assume a quantum polynomial-time (QPT) adversary since our ARKG construction aims to provide post-quantum security, interacting classically with the honest parties. Thus, we cannot rely on classical hardness assumptions, however due to the nature of the presented protocol, the QPT adversary does not get to query any of the oracles in superposition.

We generally assume that authenticators are tamper-proof, i.e., they do not leak information on the secret keys stored on them, even if they are in possession of the adversary. This assumption was also made for the FIDO2 analysis by Barbosa et al. [1] and is intuitively reasonable in our setting where we assume the primary authenticator has been lost, i.e., may be in the hands of the adversary. If (primary) authenticators leaked secret keys, the adversary could immediately log into services and reset credentials such that account recovery would not be possible anymore.

Frymann et al. [6, 9, 10] implicitly make this assumption in the weak form of their key-security property of ARKG, where they do

[3] We note that it is possible to pair any primary authenticator with multiple backup authenticators, and vice versa. However, for ease of presentation we focus on the case where a single *PA* is paired with a single *BA*.

not provide the adversary with an oracle that outputs derived secret keys for previously generated derived public keys. Nevertheless, we do (optionally) provide the adversary with oracles that leak derived secret keys to achieve stronger notions of security, analogous to the strong version of Frymann et al.

We assume that the initial pairing between the backup authenticator and the primary authenticator(s) is in a trusted stage such that an adversary is not able to inject its own long-term public key to the user's primary authenticator. This is a reasonable assumption since this pairing only happens once, is of short duration, and is executed locally at the user with no information going over public network channels.

Backup authenticators are typically offline and should only come online during account recovery. Since we cannot rule out that an adversary intercepts the user's account recovery attempts, we do nevertheless grant the adversary access to a signing oracle where the *BA*'s long-term secrets are employed.

We assume that WebAuthn registrations (with extensions) are secure against active adversaries [2]. In the context of ARKG this is especially important for the registration procedure, where the derived public keys and recovery information for account recovery are transmitted from the *PA* to the relying party. If the adversary were able to inject its own account recovery credentials here, all is lost. It is reasonable to assume this interaction to happen over a secure channel. Typically, upon registration of FIDO2 credentials, a user has previously logged in to the service using other means of authentication, e.g., with username and password and has established an authenticated connection [1]. Thus, we assume that the adversary remains passive during the registration of credentials with a relying party. During the account recovery process, however, the user is not authenticated to the relying party and no secure channel exists. Thus, we allow the adversary to actively interfere, i.e., it may drop, modify, or inject messages.

As usual for reliable authentication, we assume that public keys are globally unique. This can be accomplished by including the relying party's identity *rpid* and a unique user identifier (or pseudonym) *uid* in the public key.

3.2 Authentication Security

Viewed merely from the cryptographic primitive level, the main functionality of ARKG schemes is to derive public-secret key pairs $(\mathsf{pk}', \mathsf{sk}')$ along with additional recovery information rec from a long-term public key $\mathsf{pk}_{\mathsf{BA}}$ such that sk′ can only be recovered with knowledge of the long-term secret $\mathsf{sk}_{\mathsf{BA}}$. Frymann et al. [6] thus describe the main security property of ARKG schemes as one where an adversary may not be able to derive valid public-secret key pairs (and recovery information) without knowledge of the long-term secret key.

As elaborated in Section 2.4, ARKG schemes were originally introduced to support account recovery in case of primary authenticator loss in FIDO2 authentication procedures, i.e., in a challenge-response-based protocol using digital signatures. Viewed in this context, the main security goal of ARKG schemes should be the adversary's inability to create a valid response, i.e., a valid signature on a given challenge value (and new public key credential) during an account recovery procedure.

This paper defines a post-quantum instantiation of ARKG for the explicit purpose of account recovery in FIDO2. We thus decide to forgo Frymann et al.'s more general definition of security with respect to key recovery and introduce a more tailored notion of authentication security, or auth security, for short. We discuss the differences between this notion, and the one given by Frymann et al. in more detail in Appendix A.

```
$\mathsf{Exp}^{\boxed{\mathsf{s}}\mathsf{-auth}}_{\mathsf{ARKG}}(\mathcal{A})$:
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
    $\mathcal{L}_{\mathsf{keys}}, \mathcal{L}_{\mathsf{ch}}, \mathcal{L}_{\mathsf{sk}'}, \mathcal{L}_{\sigma} \leftarrow \emptyset$;
    $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
    $(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star, m^\star, \sigma^\star) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{DerivePK},\textsc{Chall-auth},\textsc{Sign},\boxed{\textsc{LeakSK}}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}})$
    return $[\![(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star) \in \mathcal{L}_{\mathsf{keys}} \wedge \exists(\mathsf{ch}, aux^\star) \in \mathcal{L}_{\mathsf{ch}} : \mathsf{prefix}(m^\star) = \mathsf{ch} \wedge$
        $\mathsf{Vrfy}(\mathsf{pk}^\star, \sigma^\star, m^\star) \wedge (\mathsf{rec}^\star, m^\star) \notin \mathcal{L}_{\sigma} \boxed{\wedge \mathsf{rec}^\star \notin \mathcal{L}_{\mathsf{sk}'}}\,]\!]$
```

```
$\textsc{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, \cdot)$ on input $aux$:
    $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$
    $\mathcal{L}_{\mathsf{keys}} \leftarrow \mathcal{L}_{\mathsf{keys}} \cup \{(\mathsf{pk}', \mathsf{rec}, aux)\}$
    return $(\mathsf{pk}', \mathsf{rec})$
```

```
$\textsc{Chall-auth}(\cdot)$ on input $aux$:
    $\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}^\lambda$
    $\mathcal{L}_{\mathsf{ch}} \leftarrow \mathcal{L}_{\mathsf{ch}} \cup \{(\mathsf{ch}, aux)\}$
    return ch
```

```
$\textsc{Sign}(\cdot,\cdot)$ on input $(\mathsf{rec}, m)$:
    $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
    if $\mathsf{sk}' = \bot$: abort
    $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}', m)$
    $\mathcal{L}_{\sigma} \leftarrow \mathcal{L}_{\sigma} \cup \{(\mathsf{rec}, m)\}$
    return $\sigma$
```

```
$\textsc{LeakSK}(\cdot)$ on input rec:
    $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
    $\mathcal{L}_{\mathsf{sk}'} \leftarrow \mathcal{L}_{\mathsf{sk}'} \cup \{\mathsf{rec}\}$
    return sk′
```

```
$\mathsf{Exp}^{\mathsf{unl}}_{\mathsf{ARKG}}(\mathcal{A})$:
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
    $b \leftarrow\!\!\$\ \{0,1\}$
    $(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{sk}^0_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
    $(\mathsf{pk}^1_{\mathsf{BA}}, \mathsf{sk}^1_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
    $b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{1-Chall-u},\textsc{LeakSK-u}}(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{pk}^1_{\mathsf{BA}})$
    return $[\![b = b']\!]$
```

```
$\textsc{1-Chall-u}(\cdot)$ on input $aux$:
    $(\mathsf{pk}', \mathsf{rec}) \leftarrow \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}^b_{\mathsf{BA}}, aux)$
    $\mathsf{sk}' \leftarrow\!\!\$\ \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}^b_{\mathsf{BA}}, \mathsf{rec})$
    return $(\mathsf{pk}', \mathsf{sk}', \mathsf{rec})$
```

```
$\textsc{LeakSK-u}(\cdot)$ on input $(\beta, \mathsf{rec})$:
    if $\mathsf{rec}' = \mathsf{rec}$: abort
    $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}^\beta_{\mathsf{BA}}, \mathsf{rec}')$
    return sk′
```

Figure 2: Our security definitions for authentication (top) and unlinkability (bottom) of ARKG schemes. The oracle LeakSK in the experiment $\mathsf{Exp}^{\mathsf{s-auth}}_{\mathsf{ARKG}}(\mathcal{A})$ (highlighted in gray) gives a stronger form of authentication security in the presence of secret key leakage.

Game description. The formal description of the authentication game $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ can be found in the upper part of Figure 2. The adversary $\mathcal{A}$ gets as input the public parameters pp and the long-term public key $\mathsf{pk}_{\mathsf{BA}}$ from the backup authenticator *BA*. $\mathcal{A}$ then has access to the oracles DerivePK, Chall-auth, and Sign, and optionally, in the strong variant LeakSK.

The oracle DerivePK takes as input auxiliary data *aux* and derives a public key pk′ and recovery information rec for the relying party specified in *aux* from the long-term public key $\mathsf{pk}_{\mathsf{BA}}$. This simulates the honest generation of derived public keys and recovery information on the primary authenticator *PA* when registering with relying parties specified in the auxiliary data *aux*.

As they are by default not authenticated, account recovery processes may be triggered by the adversary. Thus, $\mathcal{A}$ gets access to the challenge oracle CHALL-AUTH, which takes as input auxiliary data *aux* and outputs a uniformly random challenge value ch. This challenge value corresponds to the challenges sent out by relying parties specified via *aux* in the account recovery process. The adversary eventually has to create a valid signature on a message containing one of these challenges, more specifically on a message m that starts with a challenge value and has not been queried to SIGN with respect to the secret key.

When it receives credential information rec, the backup authenticator *BA* has no means to distinguish between credential information that had been honestly generated by a primary authenticator and credential information that the adversary sends to it. The *BA* will simply use its long-term secret $\mathsf{sk}_{\mathsf{BA}}$ to derive the secret key sk' and sign the response with it. Thus, we grant $\mathcal{A}$ access to an oracle SIGN which takes as input recovery information rec and a message m. The oracle then tries to derive a secret key sk', and if it fails will abort. If an sk' was derived, it will then use it to generate a signature on the provided message m and return the signature σ.

The optional LEAKSK oracle models the leakage of derived secret keys. The adversary may provide recovery information rec and the oracle will return the output of DeriveSK, which is either $\perp$ if the derivation failed or the derived secret key sk'.

The adversary outputs $(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star, m^\star, \sigma^\star)$ wins the game $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$, or $\mathsf{Exp}^{\mathsf{s-auth}}_{\mathsf{ARKG}}(\mathcal{A})$, if LEAKSK is present, if it is able to produce a valid signature on a message containing the challenge posed by a relying party. Valid here means:

- $(\mathsf{pk}^\star, \mathsf{rec}^\star)$ was honestly generated for the relying party specified in $aux^\star$,
- there exists an honestly generated challenge ch for $aux^\star$ such that ch is a prefix of $m^\star$,
- $\sigma^\star$ is a valid signature on $m^\star$ with respect to $\mathsf{pk}^\star$,
- the adversary has not received a signature on $m^\star$ with respect to the secret key associated with $\mathsf{rec}^\star$,
- and, if LEAKSK is present, the secret key associated with $\mathsf{rec}^\star$ has not been given to the adversary.

More formally,

Definition 3.1. Let ARKG = (Setup, KGen, DerivePK, DeriveSK) be an asynchronous remote key generation scheme. We say that ARKG is auth-*secure*, if for every QPT adversary $\mathcal{A}$ the advantage in winning the game $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ described in the top of Figure 2, defined as

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

is negligible in the security parameter λ. When replacing $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ with $\mathsf{Exp}^{\mathsf{s-auth}}_{\mathsf{ARKG}}(\mathcal{A})$, we say that ARKG = (Setup, KGen, DerivePK, DeriveSK) is strongly auth-secure.

3.3 Unlinkability

Unlinkability aims to provide the requirement in the WebAuthn standard [15] which recommends authenticators ensure that the credential IDs and credential public keys of different public-key credentials cannot be correlated as belonging to the same user. We note that this is a non-normative requirement, i.e., WebAuthn implementations that do not provide this unlinkability are still considered as conforming to the standard. As mentioned in the beginning, we also deviate from Frymann et al.'s definition for public-key unlinkability, which was based on the adversary's inability to distinguish derived from randomly sampled key pairs. In Appendix A.2, we elaborate on the issue with their definition, as it allows to prove ARKG schemes public-key unlinkable although they trivially link public-key credentials when employed in account recovery.

In our definition, two long-term key pairs $(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{sk}^0_{\mathsf{BA}})$ and $(\mathsf{pk}^1_{\mathsf{BA}}, \mathsf{sk}^1_{\mathsf{BA}})$ are generated and the public keys are given to the adversary. A bit $b \leftarrow\$ \{0, 1\}$ is sampled uniformly at random. The adversary may once query auxiliary information of its choice to the oracle 1-CHALL-U. The oracle then derives public key pk' and credential information rec either from $\mathsf{pk}^0_{\mathsf{BA}}$ (if $b = 0$), or $\mathsf{pk}^1_{\mathsf{BA}}$ (if $b = 1$). It then derives the corresponding secret key sk' and outputs $(\mathsf{pk}', \mathsf{sk}', \mathsf{rec})$ as challenge to the adversary. Note that with a standard hybrid argument one may lift this definition to a setting with multiple challenges. Additionally, the adversary may learn derived secret keys sk' for credential information of its choice, where it can also specify via a bit β which of the long-term secrets $\mathsf{sk}^\beta_{\mathsf{BA}}$ shall be used in the oracle's internal DeriveSK call. Note that if secret key derivation fails, DeriveSK outputs $\perp$ and this is then returned to the adversary as sk'. Of course, the adversary may not query its challenge recovery information to the oracle LEAKSK-U.

In the end, the adversary will output a bit b', guessing whether the challenge was derived from $\mathsf{pk}^0_{\mathsf{BA}}$ or $\mathsf{pk}^1_{\mathsf{BA}}$. The adversary wins if this guess is correct. More formally,

Definition 3.2. Let ARKG = (Setup, KGen, DerivePK, DeriveSK) be an asynchronous remote key generation scheme. We say that ARKG provides *unlinkability*, or is UNL-*secure*, for short, if for every QPT adversary $\mathcal{A}$, the advantage in winning the game $\mathsf{Exp}^{\mathsf{unl}}_{\mathsf{ARKG}}(\mathcal{A})$ described in the bottom of Figure 2, defined as

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

is negligible in the security parameter λ.

Note that Frymann et al. [6] in their (public-key) unlinkability game compare genuinely generated public keys against independently sampled ones. This requires to define a distribution on public keys. We have opted here for the common left-or-right notion. In principle we could also cover such real-or-random scenarios and for the post-quantum KEM Kyber one could also achieve this notion. The reason is that Kyber provides *strong pseudorandomness* under CCA [24], as shown in [19]. A more detailled comparison between the two different security models is provided in A.

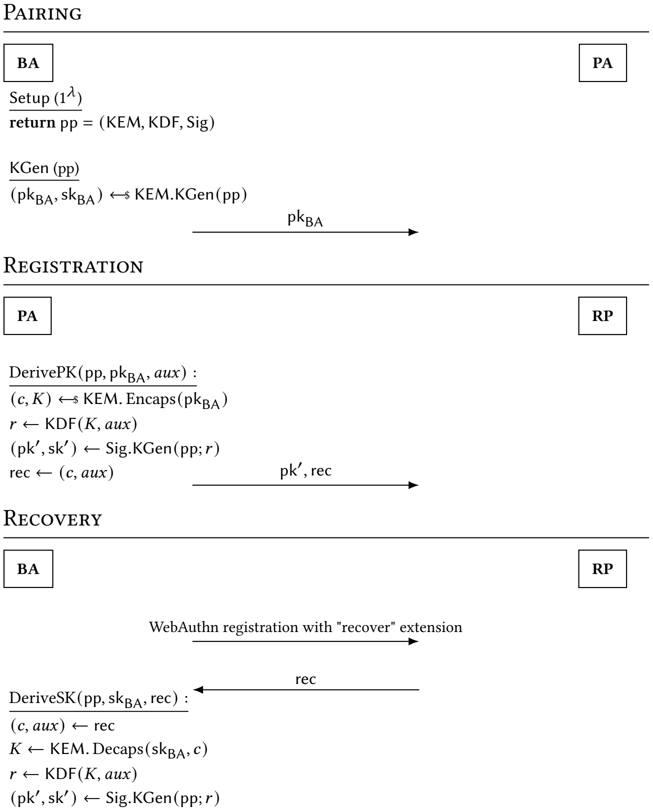
4 POST-QUANTUM ASYNCHRONOUS REMOTE KEY GENERATION

This section will introduce our instantiation for PQ-ARKG, built from generic primitives and provide security proofs in the setting discussed in Section 3. Choosing generic primitives for the instantiation allows us to provide a general security proof independent of the actual instantiation of the primitives. The result ensures ARKG is secure within the specified scenario, as long as the underlying primitives achieve the respective security properties.

4.1 The PQ-ARKG scheme

In Figure 3 we provide the generic instantiation for all algorithms required for an ARKG scheme. The key building blocks of the proposed ARKG instantiation are key encapsulation mechanisms, digital signatures and key derivation functions, all of which allow for multiple concrete instantiations believed to be resistant to a QPT attacker with high confidence [16, 18, 21, 22].

Conceptually, the interactions that make up a full ARKG protocol execution work as follows: During pairing, the *BA* generates a KEM key pair, denoted as $(\mathsf{pk_{BA}}, \mathsf{sk_{BA}})$, and transfers $\mathsf{pk_{BA}}$ to the *PA*.



Pairing

BA PA

Setup (1^λ)
return pp = (KEM, KDF, Sig)

KGen (pp)
$(\mathsf{pk_{BA}}, \mathsf{sk_{BA}}) \leftarrow\$ \mathsf{KEM.KGen(pp)}$
$\xrightarrow{\mathsf{pk_{BA}}}$

Registration

PA RP

$\mathsf{DerivePK(pp, pk_{BA}}, aux)$:
$(c, K) \leftarrow\$ \mathsf{KEM.Encaps(pk_{BA})}$
$r \leftarrow \mathsf{KDF}(K, aux)$
$(\mathsf{pk'}, \mathsf{sk'}) \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$
$\mathsf{rec} \leftarrow (c, aux)$ $\xrightarrow{\mathsf{pk'}, \mathsf{rec}}$

Recovery

BA RP

$\xrightarrow{\text{WebAuthn registration with "recover" extension}}$

$\xleftarrow{\mathsf{rec}}$
$\mathsf{DeriveSK(pp, sk_{BA}, rec)}$:
$(c, aux) \leftarrow \mathsf{rec}$
$K \leftarrow \mathsf{KEM.Decaps(sk_{BA}}, c)$
$r \leftarrow \mathsf{KDF}(K, aux)$
$(\mathsf{pk'}, \mathsf{sk'}) \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$

Figure 3: Our PQ-ARKG instantiation from KEMs, Signatures and KDFs

During registration, which is exclusively done by the *PA*, an encapsulation operation is performed under $\mathsf{pk_{BA}}$ to obtain a random key, which is then input to a KDF which outputs a random seed in the desired format. This seed is then used to deterministically generate a new signature key pair $(\mathsf{pk'}, \mathsf{sk'})$. The ciphertext resulting from the encapsulation operation is sent to the relying party for safekeeping along with the newly derived public key $\mathsf{pk'}$.

During recovery, the *BA* retrieves the ciphertext from the relying party and performs a decapsulation operation to obtain the key used as input to the KDF. By executing the PRF it obtains the seed used for the key generation. This allows *BA* to regenerate the original signature key pair, which critically includes the secret key $\mathsf{sk'}$. As a result, *BA* now has access to the same signing key pair as *PA* had during the registration, without any direct communication from *PA* to *BA*.

In short, we use KEM ciphertexts stored at the relying parties to securely relay seeds for the creation of recovery credentials between *PA* and *BA*.

A minor difference between the instantiation proposed in [6] and in this work is the fact that the primary authenticator temporarily has access to the full recovery key pair $(\mathsf{pk'}, \mathsf{sk'})$. Nonetheless, the secret key material is immediately discarded by the primary authenticator after the generation of $\mathsf{pk'}$. This does not pose a security risk, as the primary authenticator is also in possession of the primary credentials used during regular FIDO2 sessions. Consequently, an attacker with access to the primary authenticator's internal secrets could authenticate himself using a regular FIDO2 interaction while completely disregarding the recovery extension.

The fact that recovery credentials are generated by the primary authenticator but only ever used by the backup authenticator therefore also holds for our instantiation.

4.2 Security Analysis

We will now show that our instantiation achieves the two security properties authentication security and unlinkability, which are required from an ARKG instantiation.

4.2.1 Authentication Security. We first show security in the authentication case where the adversary does not have access to the LeakSK oracle to learn other derived keys. In this case IND-CPA security of the KEM scheme suffices:

Theorem 1. Let ARKG be the generic instantiation of ARKG as given in Figure 3, KEM be an IND-CPA secure KEM scheme, Sig be an EUF-CMA secure signature scheme and KDF a secure key derivation function modeled as a PRF. Then ARKG provides authentication security as defined in Definition 3.1. More precisely, for any QPT adversary $\mathcal{A}$ against auth, there exist QPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ with approximately the same running time as $\mathcal{A}$ such that

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \Big(\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\Big)$$

where q is the maximum number of calls to the DerivePK oracle.

Proof. We will prove Theorem 1 using game hopping. We denote by $\mathsf{Adv}^{\mathsf{Game_i}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$ the advantage of the adversary in the corresponding game.

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

$\mathsf{Game}_2(\lambda)$**:** In this game we guess for which call of DerivePK the adversary will output the forgery $(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star, m^\star, \sigma^\star)$ for the key $\mathsf{pk}^\star$ output by DerivePK. Note that, by definition, the adversary must succeed for one of the keys in $\mathcal{L}_{\mathsf{keys}}$. We denote the number of oracle calls of $\mathcal{A}$ to DerivePK with q. Consequently, the correct oracle call is guessed with a probability of $\frac{1}{q}$ and hence it follows that

$$\mathsf{Adv}^{\mathsf{Game_1}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \mathsf{Adv}^{\mathsf{Game_2}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) .$$

$\mathsf{Game}_3(\lambda)$**:** In this game we modify the behavior of the DerivePK algorithm for the execution guessed during the previous game: The input to the KDF, which previously was a KEM ciphertext, is replaced with a random value. This substitution takes place in line 2 of the DerivePK algorithm (cf. Figure 3).

We show that any efficient adversary $\mathcal{A}$, which can distinguish between Game_2 and Game_3 implies the existence of an efficient adversary against the IND-CPA security of KEM. The reduction $\mathcal{B}_1$ receives the KEM challenge $(\mathsf{pk}^*, k^*, c^*)$ and initializes $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ with pk^* as $\mathsf{pk}_{\mathsf{BA}}$.

During the execution of DerivePK that has been guessed in Game_2, algorithm $\mathcal{B}_1$ modifies the behavior of the algorithm by plugging in its own challenge: In line 1 the challenge pk^* is used for encapsulation and the challenge key k^* is used as input to the KDF in line 3. The ciphertext output as a component of rec is replaced with the ciphertext c^* from the KEM challenge. To simulate the Sign Oracle, the reduction keeps a list of derived secret keys, which are also generated as part of the DerivePK algorithm, but discarded during normal operation. Chall-auth can be trivially simulated, as it has no secret inputs. Finally, $\mathcal{A}$ terminates and outputs a guess b, which the reduction $\mathcal{B}_1$ outputs as its own answer to the KEM challenger.

Clearly, $\mathcal{B}_1$ perfectly simulates Game_2 when the KEM challenge is real and Game_3 when the KEM challenge is random. Consequently, we obtain the following bound:

$$\mathsf{Adv}^{\mathsf{Game}_2}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{Game}_3}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

$\mathsf{Game}_4(\lambda)$**:** In this game the execution of the DerivePK algorithm is further modified. The variable r, which was previously assigned the output of a KDF, is now sampled uniformly at random.

Any efficient adversary $\mathcal{A}$, able to distinguish Game_3 and Game_4 can be used to construct an efficient adversary $\mathcal{B}_2$ against the security of the underlying KDF, whose security we model as a PRF. The construction works similarly as in the previous game hop. $\mathcal{B}_2$ initializes $\mathsf{Exp}^{\mathsf{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ for $\mathcal{A}$ as specified, but modifies the behavior of the KDF used as part of the DerivePK algorithm in line 2 (cf. Figure 3). Instead of directly invoking the key derivation function, $\mathcal{B}_2$ forwards the input to the PRF oracle provided by the PRF challenger.

The simulation of the other oracles works identically as in the previous hop. Finally, $\mathcal{A}$ terminates and outputs a bit b, to indicate whether it is playing against Game_3 or Game_4. $\mathcal{B}_2$ forwards this as its own output to the PRF challenger.

Clearly, $\mathcal{B}_2$ perfectly simulates Game_3 if the oracle is an actual KDF, and Game_4 if the oracle is a random function. Thus, we get the following advantage:

$$\mathsf{Adv}^{\mathsf{Game}_3}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{Game}_4}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

Now we bound the last term on the right hand side. For this we can construct a reduction $\mathcal{B}_3$, which uses an efficient adversary $\mathcal{A}$ against Game_4 as a subroutine and can efficiently win against any EUF-CMA challenger with non-negligible probability. This allows us to bound the advantage of any QPT adversary against Game_4 by the EUF-CMA security of the underlying signature scheme.

The reduction $\mathcal{B}_3$ receives a challenge public key pk^* and a signing oracle Sign from the EUF-CMA challenger. It then initializes the game Game_4 as specified, in particular it holds the backup authenticator's key pair $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$. During the query guessed in the first game hop, it replaces the public key output by DerivePK with the challenge public key $\mathsf{pk}^\star = \mathsf{pk}^*$. Note that this also means that this choice also determines the recovery information $\mathsf{rec}^\star$. Replacing the public key by pk^* is possible, as in Game_4 the output of DerivePK is completely independent of both the key derivation function and the initial public key $\mathsf{pk}_{\mathsf{BA}}$.

Queries by $\mathcal{A}$ to the Sign Oracle of the auth game for the value $\mathsf{rec}^\star$ can be forwarded to the outer Sign Oracle of the EUF-CMA game by the reduction $\mathcal{B}_3$. Since DeriveSK is deterministic, the signature oracle in the attack would recover exactly *the* secret key to pk^*, such that using the external signing oracle is valid. Note that signature queries for any other rec value can be answered with the help of $\mathsf{sk}_{\mathsf{BA}}$, first recovering the derived key and then signing the input message m.

Ultimately, the inner adversary $\mathcal{A}$ terminates and outputs values $(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star, m^\star, \sigma^\star)$, where $\sigma^\star$ is a valid signature under the challenge public key pk^* and an arbitrary message $m^\star$. The reduction can then output the message-signature pair $(m^\star, \sigma^\star)$ as its forgery. Per construction, this constitutes a valid forgery: The only queries forwarded to $\mathcal{B}_3$'s external signing oracle are the ones for $\mathsf{rec}^\star$. Since the adversary $\mathcal{A}$ can only win if $(\mathsf{rec}^\star, m^\star)$ is not in the list of signed pairs $\mathcal{L}_\sigma$, it follows that $m^\star$ must not have been signed before in $\mathcal{B}_3$'s attack.

Consequently, the success probabilities of $\mathcal{A}$ and $\mathcal{B}_3$ are equal. Thus we can conclude that

$$\mathsf{Adv}^{\mathsf{Game}_4}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)\,.$$

To conclude the proof, we sum up the advantages:

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \Big(\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\Big)\,.$$

□

Note that in the proof we have not used the requirement that the forgery needs to be for a random challenge and for the right format. The reason is that we presume existential unforgeability of the signature scheme, such that even forgeries for arbitrary messages should be infeasible. For practical purposes we would only require the relaxed unforgeability notion but do not explore this here further.

We next extend the proof to strong authentication security, where the availability of the LeakSK oracle requires IND-CCA security of the KEM:

Theorem 2. Let ARKG be the generic instantiation of ARKG as given in Figure 3, KEM be an IND-CCA secure KEM scheme, Sig be an EUF-CMA secure signature scheme and KDF a secure key derivation function modeled as a PRF. Then ARKG provides strong authentication security as defined in Definition 3.1. More precisely, for any QPT adversary $\mathcal{A}$ against auth, there exist QPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ with approximately the same running time as $\mathcal{A}$

such that

$$\mathsf{Adv}^{\text{s-auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \Big(\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\Big)$$

where q is the maximum number of calls to the DerivePK oracle.

Proof. The proof is almost identical to the one for the IND-CPA case. We only need to adapt the game hop to Game$_3$, based on the IND-CPA security of the KEM. Since adversary $\mathcal{A}$ now has access to oracle LeakSK our reduction $\mathcal{B}_1$ needs to answer queries rec to LeakSK without knowing the decryption key $\mathsf{sk}_{\mathsf{BA}}$ of the KEM. But since the adversary can only win if the forgery attempt $\mathsf{rec}^\star$ does not lie in $\mathcal{L}_{\mathsf{sk}'}$, reduction $\mathcal{B}_1$ can use the decryption oracle of the IND-CCA security game of the KEM to answer these different requests. The rest of the proofs remains unchanged. □

4.2.2 Unlinkability. We next discuss unlinkability of our scheme. This follows from the fact that the KEM scheme is anonymous.

Theorem 3. Let ARKG be the instantiation of ARKG as given in Figure 3 and KEM be an ANON-CCA secure KEM scheme. Then ARKG provides unlinkability security as described in Definition 3.2. More precisely, for any QPT adversary $\mathcal{A}$ against unl there exists a QPT algorithm $\mathcal{B}$ with approximately the same running time as $\mathcal{A}$, such that

$$\mathsf{Adv}^{\mathsf{unl}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{ANON-CCA}}_{\mathsf{KEM},\mathcal{B}}(\lambda)\,.$$

Proof. We prove Theorem 3 using a direct reduction to the ANON-CCA security of the underlying KEM. Let $\mathcal{A}$ be a QPT adversary against unlinkability. We use $\mathcal{A}$ to construct an efficient reduction, $\mathcal{B}$, that uses $\mathcal{A}$ as a subroutine to win with against ANON-CCA with non-negligible probability.

First, $\mathcal{B}$ receives the challenge set $(\mathsf{pk}_0, \mathsf{pk}_1, c^*, k^*)$ as per the ANON-CCA security definition (cf. Figure 7). Then, $\mathcal{B}$ forwards $(\mathsf{pk}_0, \mathsf{pk}_1)$ to the inner adversary $\mathcal{A}$ as $(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{pk}^1_{\mathsf{BA}})$. Next, $\mathcal{A}$ outputs *aux* to query the 1-Chall-u oracle. The reduction simulates the behavior of 1-Chall-u as follows: During the execution of the algorithm DerivePK (cf. Figure 3), the challenge key k^* is used as input to the KDF in combination with *aux* provided by the inner adversary $\mathcal{A}$. During the subsequent execution of Sig.KGen, the reduction obtains the tuple $(\mathsf{pk}', \mathsf{sk}')$. Lastly, it creates rec as $\mathsf{rec} \leftarrow (c^*, aux)$. Then it returns $(\mathsf{pk}', \mathsf{sk}', \mathsf{rec})$ to the inner adversary. Queries to the LeakSK-u oracle can be answered by $\mathcal{B}$ with the help of its own decapsulation oracle provided by the ANON-CCA challenger; any query about the challenge value is immediately rejected.

Finally, $\mathcal{A}$ outputs a bit b, which the reduction forwards as its guess to the ANON-CCA challenger. Depending on the bit b of the ANON-CCA game, this perfectly simulates either the case where the challenge bit of unl is sampled as 0 or 1.

We have constructed $\mathcal{B}$ in such a way, that it perfectly simulates the unl game for $\mathcal{A}$ and its view depends only on the random bit b chosen by the challenger. Consequently, the success probability of the reduction is equal to that of the inner adversary, which yields the following result:

$$\mathsf{Adv}^{\mathsf{unl}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{ANON-CCA}}_{\mathsf{KEM},\mathcal{B}}(\lambda)\,.$$

□

4.3 Performance

Implementing ARKG in practice requires additional computational capabilites and storage at both the relying party and the authenticator itself. The overhead is completely dependent on the choices of the underlying primitives. Figure 1 illustrates those requirements: Additional storage is required at the authenticator to store the backup authenticator's public key and at the relying party which has to store the public key of the recovery credential as well as recovery information rec. Each regular registration of a FIDO2 authenticator also requires an additional computational step, where one KEM encapsulation and one KEM key generation is performed. Pairing and recovery, which do not happen during regular operation, require a single KEM key generation and one decapsulation, one KEM key generation and one sign operation, respectively. The regular authentication routine is completely unaffected by the modification and should see no changes in performance.

With some viable options available for both KEMs and digital signatures, several choices could be viable. For example, SPHINCS+ could be a viable signature choice due to its small key sizes, which would be beneficial for the storage overhead at the relying parties, however the recovery would take longer than with other options. We leave the ideal tradeoff between storage and computational costs as an open question for future work.

5 CONCLUSION

As elaborated before, asynchronous remote key generation is the preferable approach for account recovery in comparison to multi-device passkeys, especially in security-sensitive settings.

Using hardware authenticators comes with many security upsides, which are partially invalidated by opting for the simpler, but less secure multi-device passkeys. In particular, an ARKG-based solution is compatible with hardware-binding of secret key material, such that full control over the authentication information is retained at all time.

The primitive of asynchronous remote key generation as introduced by Frymann et al. [6] proves useful to provide a mechanism within FIDO2 to support account recovery in case of authenticator less. While their original construction was based on the discrete-logarithm assumption, further works [9, 10] have introduced instantiations from lattices and pairings, respectively. In this work, we have introduced a generic instantiation using key encapsulation mechanisms and digital signatures, which is especially relevant for the post-quantum setting and have proven it secure.

We have refined the security properties required of ARKG schemes when employed in FIDO2 flows to capture real-world adversarial capabilities. In particular, we fixed a shortcoming in Frymann et al.'s definition of public-key unlinkability that falsely categorizes schemes to be public-key unlinkable although they trivially link keys. Furthermore, we no longer use a key security notion which requires the adversary to output a full secret key to a notion of existential unforgeability of signatures. It is quite uncommon to require the adversary to be able to recover entire secret keys in order to win the games and, indeed, the challenge-response based FIDO2-setting would already fail if an adversary were able to existentially forge a signature.

REFERENCES

[1] Manuel Barbosa, Alexandra Boldyreva, Shan Chen, and Bogdan Warinschi. 2021. Provable Security Analysis of FIDO2. In *CRYPTO 2021, Part III (LNCS)*, Tal Malkin and Chris Peikert (Eds.), Vol. 12827. Springer, Heidelberg, Virtual Event, 125–156. https://doi.org/10.1007/978-3-030-84252-9_5

[2] Nina Bindel, Cas Cremers, and Mang Zhao. 2022. FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation. (2022). Report Number: 1029, https://eprint.iacr.org/2022/1029.pdf.

[3] Nina Bindel, Cas Cremers, and Mang Zhao. 2023. FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation. In *IEEE Symposium on Security and Privacy (SP)*. 674–693.

[4] John Bradley, Jeff Hodges, Michael B. Jones, Akshay Kumar, Rolf Lindemann, Johan Verrept, Matthieu Antoine, Vijay Bharadwaj, Arnar Birgisson, Christiaan Brand, Alexei Czeskis, Thomas Duboucher, Jakob Ehrensvärd, Mirko J. Ploch, Adam Powers, Chad Armstrong, Konstantinos Georgantas, Fabian Kaczmarczyck, Nina Satragno, and Nuno Sung. 2022. Client to Authenticator Protocol (CTAP). (June 2022). https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html.

[5] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. 2020. Towards Post-Quantum Security for Signal's X3DH Handshake. In *SAC 2020 (LNCS)*, Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn (Eds.), Vol. 12804. Springer, Heidelberg, 404–430. https://doi.org/10.1007/978-3-030-81652-0_16

[6] Nick Frymann, Daniel Gardham, Franziskus Kiefer, Emil Lundberg, Mark Manulis, and Dain Nilsson. 2020. Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn. In *ACM CCS 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 939–954. https://doi.org/10.1145/3372297.3417292

[7] Nick Frymann, Daniel Gardham, and Mark Manulis. 2022. Unlinkable Delegation of WebAuthn Credentials. In *ESORICS 2022, Part III (LNCS)*, Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng (Eds.), Vol. 13556. Springer, Heidelberg, 125–144. https://doi.org/10.1007/978-3-031-17143-7_7

[8] Nick Frymann, Daniel Gardham, and Mark Manulis. 2023. Asynchronous Remote Key Generation for Post-Quantum Cryptosystems from Lattices. Cryptology ePrint Archive, Paper 2023/419. (2023). https://eprint.iacr.org/2023/419 https://eprint.iacr.org/2023/419.

[9] Nick Frymann, Daniel Gardham, and Mark Manulis. 2023. Asynchronous Remote Key Generation for Post-Quantum Cryptosystems from Lattices. Cryptology ePrint Archive, Paper 2023/419. (2023). https://eprint.iacr.org/2023/419, to appear at EuroS&P 2023.

[10] Nick Frymann, Daniel Gardham, Mark Manulis, and Hugo Nartz. 2023. Generalised Asynchronous Remote Key Generation for Pairing-based Cryptosystems. Cryptology ePrint Archive, Paper 2023/456. (2023). https://eprint.iacr.org/2023/456, to appear at ACNS 2023.

[11] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. 2022. Anonymous, Robust Post-quantum Public Key Encryption. In *EUROCRYPT 2022, Part III (LNCS)*, Orr Dunkelman and Stefan Dziembowski (Eds.), Vol. 13277. Springer, Heidelberg, 402–432. https://doi.org/10.1007/978-3-031-07082-2_15

[12] Jingjing Guan, Hui Li, Haisong Ye, and Ziming Zhao. 2022. A Formal Analysis of the FIDO2 Protocols. In *ESORICS 2022, Part III (LNCS)*, Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng (Eds.), Vol. 13556. Springer, Heidelberg, 3–21. https://doi.org/10.1007/978-3-031-17143-7_1

[13] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. 2022. Token meets Wallet: Formalizing Privacy and Revocation for FIDO2. Cryptology ePrint Archive, Report 2022/084. (2022). https://eprint.iacr.org/2022/084.

[14] Christopher Harell. 2022. YubiKeys, passkeys and the future of modern authentication. (03 2022). https://www.yubico.com/blog/passkeys-and-the-future-of-modern-authentication/.

[15] Jeff Hodges, J.C. Jones, Michael B. Jones, Akshay Kumar, Emil Lundberg, John Bradley, Christiaan Brand, Adam Langley, Giridhar Mandyam, Nina Satragno, Nick Steele, Jiewen Tan, Shane Weeden, Mike West, and Jeffrey Yasskin. 2021. Web Authentication: An API for accessing Public Key Credentials - Level 3. (April 2021). https://www.w3.org/TR/webauthn-3.

[16] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. 2022. *SPHINCS$^+$*. Technical Report. National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[17] Emil Lundberg and Dain Nielsson. 2019. WebAuthn Recovery Extension. (2019). https://github.com/Yubico/webauthn-recovery-extension.

[18] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. 2022. *CRYSTALS-DILITHIUM*. Technical Report. National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[19] Varun Maram and Keita Xagawa. 2022. Post-Quantum Anonymity of Kyber. Cryptology ePrint Archive, Report 2022/1696. (2022). https://eprint.iacr.org/2022/1696.

[20] Varun Maram and Keita Xagawa. 2023. Post-quantum Anonymity of Kyber. In *PKC 2023, Part I (LNCS)*, Alexandra Boldyreva and Vladimir Kolesnikov (Eds.), Vol. 13940. Springer, Heidelberg, 3–35. https://doi.org/10.1007/978-3-031-31368-4_1

[21] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. 2022. *FALCON*. Technical Report. National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. 2022. *CRYSTALS-KYBER*. Technical Report. National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[23] Andrew Shikiar. 2022. Charting an Accelerated Path Forward for Passwordless Authentication Adoption. (03 2022). https://fidoalliance.org/charting-an-accelerated-path-forward-for-passwordless-authentication-adoption/.

[24] Keita Xagawa. 2022. Anonymity of NIST PQC Round 3 KEMs. In *EUROCRYPT 2022, Part III (LNCS)*, Orr Dunkelman and Stefan Dziembowski (Eds.), Vol. 13277. Springer, Heidelberg, 551–581. https://doi.org/10.1007/978-3-031-07082-2_20

A COMPARISON TO THE ORIGINAL SECURITY DEFINITIONS

In the following, we review the security definitions of ARKG schemes as proposed by Frymann et al. [6, 9, 10].

A.1 Key Security

Their notions for *key security* are based on the adversary's inability to output an entire valid secret key $\mathsf{sk}^\star$ needed for account recovery. We have depicted the strong and weak game description for SK-security of [6] in the *honest* setting in Figure 4.

```
$\mathsf{Exp}^{ks}_{\mathsf{ARKG},\mathcal{A}}(\lambda)$:
1  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2  $\mathcal{L}_{\mathsf{keys}}, \mathcal{L}_{\mathsf{sk}'} \leftarrow \emptyset$
3  $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
4  $(\mathsf{pk}^\star, \mathsf{sk}^\star, \mathsf{rec}^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{pk}'}, \boxed{\mathcal{O}_{\mathsf{sk}'}}}(\mathsf{pp}, \mathsf{pk}_0)$
5  $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{rec}^\star)$
6  return $[\![(\mathsf{pk}^\star, \mathsf{rec}^\star) \in \mathcal{L}_{\mathsf{keys}} \wedge \mathsf{Check}(\mathsf{pk}^\star, \mathsf{sk}^\star) = 1 \wedge \mathsf{Check}(\mathsf{pk}^\star, \mathsf{sk}') =$
   $1 \wedge \boxed{\mathsf{rec}^\star \notin \mathcal{L}_{\mathsf{sk}'}}\ ]\!]$

$\mathcal{O}_{\mathsf{pk}'}(\mathsf{pp}, \mathsf{pk}_0, \cdot)$ on input $aux$:
7  $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_0, aux)$
8  $\mathcal{L}_{\mathsf{keys}} \leftarrow \mathcal{L}_{\mathsf{keys}} \cup (\mathsf{pk}', \mathsf{rec})$
9  return $(\mathsf{pk}', \mathsf{rec})$

$\mathcal{O}_{\mathsf{sk}'}(\cdot)$ on input rec:
10 $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{rec})$
11 $\mathcal{L}_{\mathsf{sk}'} \leftarrow \mathcal{L}_{\mathsf{sk}'} \cup \mathsf{rec}$
12 if $(\cdot, \mathsf{rec}) \notin \mathcal{L}_{\mathsf{keys}}$ : abort
13 return $\mathsf{sk}'$
```

Figure 4: SK-Security as defined in [6] (honest variants).[4]

Honest vs. malicious security. For this security definition, the term *honest* refers to the first requirement in Line 6 in Figure 4, which enforces that $(\mathsf{pk}^\star, \mathsf{rec}^\star)$ must be in $\mathcal{L}_{\mathsf{keys}}$, i.e., that the tuple was honestly generated via DerivePK. This same requirement is mirrored in our check that $(\mathsf{pk}', aux) \in \mathcal{L}_{\mathsf{keys}}$ in Line 5 in Figure 2.

In the malicious setting, this requirement is dropped, thus giving the adversary more leeway. However, Frymann et al. themselves

[4] We note that the pseudocode descriptions of $\mathcal{O}_{\mathsf{pk}'}$ and $\mathcal{O}_{\mathsf{sk}'}$ have not been given before and thus corresponds merely to our interpretation of the prose description. [6].

note in [9], that this may be "too strong for many applications". In particular, it is inadequate in the context of ARKG within FIDO2. An adversary that can output a derived secret key to a public key and credential that are not actually registered with any relying party cannot successfully complete account recovery.

Weak vs. strong security. Frymann et al. have another dimension of security in their definition, which they term *weak* and *strong* security, respectively. The distinction is made along the presence of the highlighted oracle $\mathcal{O}_{\mathsf{sk}'}$ in Line 4 and the highlighted condition $\mathsf{rec}^\star \notin \mathcal{L}_{\mathsf{sk}'}$ in Line 6 in Figure 4. If it is present (the strong setting), the adversary is required to output a valid secret key $\mathsf{sk}^\star$ for a $\mathsf{pk}', \mathsf{rec}^\star$ for which it has not already learned a secret key via a query to $\mathcal{O}_{\mathsf{sk}'}$. In the weak setting, the adversary may not learn derived secret keys.

Delineation. We find that this notion of security does not capture the real-world setting, where an adversary is already successful, if it can forge a signature during the recovery process and thus gain access to the user's account. Our (strong) auth-security notion does not require the adversary to output a valid secret key to win, it only requires that the adversary can sign the challenge provided by the relying party during the recovery mechanism. Analogously to the relevant security results by Frymann et al., our definition aligns with the *honest* setting, since an adversary can only be considered successful in account recovery if it can convince a relying party to successfully verify the signature with respect to the honestly generated public key it has stored as recovery credential for the user. With regards to the strong vs. weak setting of Frymann et al. our auth definition operates in the middle grounds. In this version, we do not provide an oracle that can leak derived secret keys to the adversary. This is both due to the fact that FIDO2 authenticators are considered to be tamper-proof,and that the backup authenticator that is able to derive the secret keys never actually comes online unless an account recovery is triggered. However, we do provide the adversary with an interface to the secret-key operations on the backup authenticator, i.e., a signing oracle. In the strong version of our auth security, we provide the adversary with an LeakSK oracle that leaks derived secret keys. But of course the adversary must then forge with respect to a recovery for which the secret key was not leaked.

A.2 Public-Key Unlinkability

Figure 5 gives a complete pseudocode description of the so-called public-key unlinkability as proposed by Frymann et al. [6]. Essentially, the adversary is given the long-term public key $\mathsf{pk}_{\mathsf{BA}}$ of a backup authenticator and may then receive key pairs, which, depending on a hidden bit b, are either derived from this long-term public key or sampled independently from the key-pair distribution D.

Issue with this definition. On its own, the above definition makes sense to formalize that derived public keys do not leak from which long-term public key they were derived. However, one can show that an ARKG scheme that satisfies public-key unlinkability in accordance with Frymann et al.'s definition, can output trivially linkable keys. This is due to the fact that the definition above does not take into account the actual information an adversary has as its disposal.

```
$\mathsf{Exp}^{pku}_{\mathsf{ARKG}}(\mathcal{A})$:

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

$\mathcal{O}^b_{\mathsf{pk}'}(b, \mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$ with no input:

6  **if** $b = 0$
7      $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pk}_{\mathsf{BA}}, aux)$
8      $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
9  **else**
10     $(\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{D}$
11 **return** $(\mathsf{pk}', \mathsf{sk}')$
```

Figure 5: Public-key unlinkability as defined in [6].[5]

During registration of derived public keys pk', not only is pk' sent over the wire, but also the credential information rec, which the relying party also sends back over an insecure channel when account recovery is triggered. This rec may contain $\mathsf{pk}_{\mathsf{BA}}$: there is nothing in the construction per se that forbids this. But then public keys derived from this $\mathsf{pk}_{\mathsf{BA}}$ are all trivially linkable by the adversary.

We want to stress that the linkability is not always as easy to spot (or prevent) as in this example. Especially in the (post-quantum) KEM setting it is not always guaranteed that schemes that provide standard indistinguishably of ciphertexts do not leak information on the public key for which the encapsulation took place. As we show in our results, only KEMs that satisfy ANON-CCA security do provide this guarantee and thus any KEM-based ARKG schemes must ensure this property to provide unlinkability of derived public keys in the presence of recovery information, which we term simply unlinkability.

Delineation. We thus opted to define unlinkability as a game where the adversary gets to see two long-term public keys $\mathsf{pk}^0_{\mathsf{BA}}$ and $\mathsf{pk}^1_{\mathsf{BA}}$ and can as a challenge derive a public key and recovery information with auxiliary information of its choice. Multiple queries would also be easily supported due to a hybrid argument. Furthermore, the adversary may query recovery information of its choice (not the challenge) and let the oracle derive the secret key either from $\mathsf{sk}^0_{\mathsf{BA}}$ or $\mathsf{sk}^1_{\mathsf{BA}}$.

B DEFINITIONS

This appendix will introduce definitions for common building blocks used throughout this work.

B.1 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 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 $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$.

[5] We note that the pseudocode description of $\mathcal{O}^b_{\mathsf{pk}'}$ has not been given before and thus corresponds merely to our interpretation of the prose description. [6]. In particular, it is underspecified how *aux* in Line 7 is chosen. We would allow the adversary to give *aux* as input to the oracle, but refrain from specifying this here.

KGen is a probabilistic algorithm that takes the security parameter λ as input and probabilistically outputs a key pair (pk, sk). Encaps is a probabilistic algorithm and takes as input a public key pk, where pk $\leftarrow$ KGen(1^λ), and outputs a key k as well as a ciphertext c. The ciphertext c encapsulates the key k. 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.

Definition 1 (Correctness of KEM schemes)**.** A key encapsulation scheme KEM = (KGen, Encaps, Decaps) is δ-correct, if for all (sk, pk) key pairs output by the KGen algorithm the following equation holds

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

If $\delta = 0$ holds, the scheme is called 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, k_b), where c is output by $(c, k) \leftarrow \mathsf{Encaps}(\mathsf{pk})$ and k_b 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 k_b is randomly sampled or generated by the encapsulation algorithm with non-negligible probability.

```
$\mathsf{Exp}^{\text{IND-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$k_0 \leftarrow\$ \mathcal{K}$
$(c, k_1) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}(\mathsf{pk}, c, k_b)$
return $[\![b' = b]\!]$

$\mathsf{Exp}^{\text{IND-CCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$k_0 \leftarrow\$ \mathcal{K}$
$(c, k_1) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{O}_{Dec}(\cdot)}(\mathsf{pk}, c, k_b)$
return $[\![b' = b]\!]$

$\mathsf{O}_{Dec}(\cdot)$ on input $c$:
return $\mathsf{Decaps}(\mathsf{sk}, c)$
```

Figure 6: Game definition for IND-ATK **security of Key Encapsulation Mechanisms**

Definition 2 (IND-ATK security of KEM schemes)**.** Given the security game in Figure 6, a key encapsulation scheme KEM = (KGen, Encaps, Decaps) is IND-ATK secure for ATK $\in$ {CPA, CCA}, if the advantage

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

is negligible for any QPT adversary $\mathcal{A}$

An additional property some Key Encapsulation Mechanism (KEM) schemes achieve is anonymity. Intuitively, anonymity requires that the ciphertext obtained during encapsulation does not leak any information on the public key used during the encapsulation operation.

Definition 3 (Anonymity of KEM Schemes)**.** Given the security game in Figure 7, a key encapsulation scheme KEM, is ANON-ATK secure with ATK $\in$ {CPA, CCA}, if the advantage

$$\mathsf{Adv}^{\mathsf{kem},\mathcal{A}}_{\text{ANON-ATK}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\text{ANON-ATK}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$$

is negligible for any QPT adversary $\mathcal{A}$.

```
$\mathsf{Exp}^{\text{ANON-CCA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:
$b \leftarrow\$ \{0,1\}$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$(c^*, k^*) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}_b)$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{O}_{Dec}}(\mathsf{pk}_0, \mathsf{pk}_1, c^*, k^*)$
return $[\![b = b']\!]$

$\mathsf{O}_{Dec}(\cdot,\cdot)$ on input $id, c$:
if $[\![c = c^*]\!]$
    return $\perp$
$k = \mathsf{Decaps}(\mathsf{sk}_{id}, c)$
return $k$

$\mathsf{Exp}^{\text{ANON-CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:
$b \leftarrow\$ \{0,1\}$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$(c^*, k^*) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}_b)$
$b' \leftarrow\$ \mathcal{A}(\mathsf{pk}_0, \mathsf{pk}_1, c^*, k^*)$
return $[\![b = b']\!]$
```

Figure 7: Game definition for anonymity of key encapsulation mechanisms

B.2 Digital Signatures

A digital signature scheme is a public-key scheme that can be used to generate publicly verifiable signatures. It is defined as a triple of PPT algorithms Sig = (KGen, Sign, Vrfy).

KGen takes as input the security parameter λ and outputs a key pair (pk, sk). Sign takes as input a secret key sk and a message m, and computes a signature σ on the message m. Vrfy is used to verify signatures. It takes a input a public key pk, a signature σ and a message m. The output is 1, if σ is a valid signature for the message m under the public key pk, otherwise it returns 0.

Definition 4. A digital signature scheme Sig = (KGen, Sign, Vrfy) is correct, if

$$\Pr\left[0 \leftarrow \mathsf{Vrfy}(\mathsf{pk}, \sigma, m) : (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda), \sigma \leftarrow\$ \mathsf{Sign}(\mathsf{sk}, m)\right]$$

is negligible in the security parameter λ.

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 5 (EUF-CMA security of digital signature schemes)**.** Given the security game in Figure 8, a digital signature scheme Sig = (KGen, Sign, Vrfy) is EUF-CMA secure, if

$$\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = 1\right]$$

is negligible for any QPT adversaries $\mathcal{A}$.

```
$\mathsf{Exp}^{\text{EUF-CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(1^\lambda)$
$\mathcal{L}_m \leftarrow \emptyset$
$(m', \sigma') \leftarrow\$ \mathcal{A}^{\mathsf{O}_{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
return $[\![\mathsf{Vrfy}(\mathsf{pk}, \sigma', m') \wedge m' \notin \mathcal{L}_m]\!]$

$\mathsf{O}_{Sign}(\mathsf{sk},\cdot)$ on input $m$:
$\sigma \leftarrow\$ \mathsf{Sign}(\mathsf{sk}, m)$
$\mathcal{L}_m \leftarrow \mathcal{L}_m \cup m$
return $\sigma$
```

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

B.3 Key Derivation Function

A KDF is used to transform some non-uniformly distributed input of arbitrary length into a fixed length key that can safely be used in cryptographic schemes. We model its security as a PRF using the following definition

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

$\mathrm{O}(f, \cdot)$ on input x:
return $f(x)$
```

Figure 9: Game definition for PRF **security of** KDF**s**

Definition 6 (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 9, a PRF is secure, if for all QPT adversaries $\mathcal{A}$ the following holds

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