Post-Quantum Asynchronous Remote Key Generation for FIDO2 Account Recovery

Jacqueline Brendel, Sebastian Clermont, and Marc Fischlin

Technical University of Darmstadt {first.last}@tu-darmstadt.de

Abstract. The Fast IDentity Online (FIDO) Alliance develops open standards to replace password-based authentication by token-based solutions. The latest protocol suite FIDO2 provides a promising alternative which many key players have already adopted or are planning 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 credential recovery.
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 proposed and analyzed Diffie-Hellman based schemes, showing 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.
In this work here we show a generic construction based on (anonymous) KEMs and signature schemes. In particular, using post-quantum secure instances like Kyber and Dilitihium, one immediately obtains a post-quantum secure solution. In passing, we observe that the security definitions brought forward by Frymann et al., especially the privacy notion, do not appropriately capture the intuitive security goals of the FIDO2 protocol. We thus strengthen the notions and prove our general scheme to satisfy the stronger definitions.

Keywords: FIDO2, post-quantum, account recovery, passwordless

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 credential. 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 for a succesfull authentication is referred to as FIDO2 credential or as a passkey [22].

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 [13, 3]. 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.

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* have been created. These mechanisms rely on a low-entropy recovery PIN created by a user which encrypts a high-entropy key stored on a distributed hardware security modul (HSM). This high-entropy key is then used for the encryption of the user 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.

On one hand, the approach above is unsatisfactory, as the entire security relies on a low-entropy PIN chosen by a users and requires trust in cloud vendors and their HSMs to behave as promised. Furthermore, 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 approaches unusable for recovery.

When considering backup methods for the FIDO2 setting, there are two scenarios for the 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 [22]. 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 policy. 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 the 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; possession of the backup authenticator is required to obtain the secrets.

Yubico proposed a Diffie-Hellman based protocol for instantiating these backup authenticators. The proposal has been formalized and analyzed under the term asynchronous remote key generation (ARKG) by Frymann et al. [8]. Frymann et al. also gave alternative protocols based on pairings [10] and a proposal how one could achieve a post-quantum secure version [7] based on split-KEMs [6]. Unfortunately, the instantiations of such split-KEMs from lattices currently only achieve a weak security notion, called, nn-IND-CCA. This does not allow for ARKG instantiations achieving resilience against adaptive attackers. A different post-quantum secure instantiation using key blinding was proposed by Spencer Wilson in [23].

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. [8] 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 used by honest users to recover their account. 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 public key of its choice (honest vs. malicious). The other property is public-key unlinkability which should prevent relying parties to link users across different services 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 [8, 7, 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 [8], 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 strictly stronger requirement of key recovery is often not considered.

As in the case of key security, the privacy definition of [8] gives the adversary not enough power. The definition only gives the adversary access to the public keys derived by the backup authenticator. However, relying parties also store the recovery information along with the derived public key, which could help in succesfuly linking users. Deanonymization attacks using this strategy are currently not captured in the public-key unlinkability model in [8].[1] Note that all prior solutions [8, 7, 10, 23] target the original security notions, which we argue to be inaccurate at capturing the adversarial capabilities and security goals.

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 recovery key pair is seperate from the regular FIDO2 key pair, which is generated and stored as usual. Figure 1 illustrates how the ARKG messages can be integrated into a FIDO2 registration session.

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

[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.

particular, the MAC enters the security statement neither for key security nor for privacy in Frymann's analysis. Furthermore, it does not seem to provide protection against malicious behavior, since the MAC key is derived from the joint DH key, and can thus be easily derived by an adversary injecting its DH share. The MAC is used in all protocol versions, the initial discrete-log based solution [8], the pairing based solution [10], as well as in the post-quantum based approach [7].

The post-quantum ARKG protocol in [7] is based on the split key encapsulation mechanism (KEM) approach in [6], 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 [6]. Hence, it remains unclear if one can actually derive post-quantum ARKG protocols for the security model brought forward by Frymann et al.

1.4 Our Contributions

We set off by giving stronger security notions for key security and public-key unlinkability. For security we strengthen the notion and only demand that the adversary cannot produce signature forgeries under backup keys. In practice this means an attacker 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 only protect the secret key, but also 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 primary 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. During of recovery, the backup authenticator can retrieve the randomness and re-generate the signing key.

To achieve authentication security, we need IND-CCA security of the KEM and EUF-CMA security of the signature scheme as well as and pseudorandomness of an intermediate key deriviation function. 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, namely Kyber [21] for the KEM, and Dilithium [18], Falcon [20], 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 anonymity, or ANON-CCA of the KEM scheme, meaning that one cannot distinguish ciphertexts created under one public key from ones created under another public key. This notion has first been proposed by Bellare et al. [2] and recently been analyzed in more detail for the case of post-quantum schemes [24, 11]. Fortunately, the aforementioned Kyber KEM is ANON-CCA without any modifications [19].

2 Preliminaries of Asynchronous Remote Key Generation

We start by introducing the notation, terminology and main definitions used in this paper. In particular, we revisit the definition of Asynchronous Remote Key Generation (ARKG) which was first proposed by Frymann et al. in [8]

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 $\text{prefix}(x) = y$ to indicate that y is a prefix of x. We assume classical algorithms for implementing schemes, with the common notion of *efficiency* if the algorithms run in probabilistic or quantum polynomial time in the length of the security parameter λ, denoted by PPT and QPT, respectively. Explicit randomness is indicated in an algorithm's input using a semicolon, e.g., $\text{Sign}(\mathsf{sk}, m; r)$ denotes the execution of the signing algorithm with randomness r.

We assume QPT adversaries $\mathcal{A}$. Since the honest parties use classical algorithms, $\mathcal{A}$ may only interact classically with honest parties. We write $\mathcal{A}^{\mathcal{O}}$ to denote that $\mathcal{A}$ has access to the oracle $\mathcal{O}$. We use $\cdot$, to represent required input to an algorithm, i.e., $\mathsf{O}(\cdot, \cdot)$ denotes that the algorithm O takes two inputs.

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

2.3 ARKG Syntax

We now recall the notion of asynchronous remote key generation schemes as introduced by Frymann et al. [8] but slightly change notation to make it more aligned with the intended purpose. We assume that the *BA* generates a long-term key pair $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$ via the algorithm KGen. Key pairs on the PA are denoted as $(\mathsf{pk}, \mathsf{sk})$ and are generated together with recovery information rec via the algorithm DerivePK in such a way that allows the backup authenticator *BA* to recover the secret key with the help of $\mathsf{sk}_{\mathsf{BA}}$ via the algorithm DeriveSK. Both algorithms are linked through an algorithm Check to identify matching public and secret keys. Instead of calling the recovery information a *credential* denoted by cred as in [8] we call it recovery information, or rec, for short, resembling the externally stored session resumption data in TLS.

Definition 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.,* $\mathsf{pp} \leftarrow \mathrm{Setup}(1^{\lambda})$*.*
- KGen *takes as input the public parameters* pp *and output a public/secret key pair* $(\mathsf{pk}_{BA}, \mathsf{sk}_{BA}) \leftarrow\!\!\$\ \mathrm{KGen}(\mathsf{pp})$ *for the backup authenticator.*
- DerivePK *takes as input the public parameters* pp*, a public key* pk_{BA} *and auxiliary information aux*[2] *and outputs a derived public key* pk' *and associated credential information* rec*, i.e.,* $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathrm{DerivePK}(\mathsf{pp}, \mathsf{pk}_{BA}, aux)$.
- DeriveSK *takes as input the public parameters* pp*, a secret key* sk_{BA} *and recovery information* rec*. It outputs either a secret key* sk'*, i.e.,* $\mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{BA}, \mathsf{rec})$*, 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* $(\mathsf{pk}, \mathsf{sk})$ *and returns 1 if* $(\mathsf{pk}, \mathsf{sk})$ *forms a valid public/secret key pair, and 0 otherwise.*

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

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

Remark 1. Frymann et al. include the algorithm $\mathrm{Check}(\mathsf{pk}, \mathsf{sk})$ 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

[2] We assume that in the context of FIDO2 account recovery as treated in this paper, *aux* is a unique identifier *rpid* of the relying party for which the public key and credential are derived.

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.

2.4 ARKG in the Context of FIDO2

As also mentioned in [8], the ARKG primitive 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 [9]. Here we focus on the original purpose for FIDO2 account recovery.

In Section 3 we will present our modified security notions for ARKG. To motivate the changes to the security definitions given in [8], we recap the basics of passwordless authentication via WebAuthn in FIDO2 [15] and how ARKG fits into this flow. Roughly speaking, the role of the ARKG primitive within the context of FIDO2 account recovery is two-fold:

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.

On the *BA*: To 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.

In more detail, ARKG has three phases: pairing, registration, and recovery. These phases are illustrated in Figure 1 and we briefly describe them next.

Pairing At the beginning, ARKG 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 *BA* is transferred to the *PA* which stores it.

Registration At some point, the primary authenticator *PA* then begins to register credentials with relying parties. This registration happens over 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, which 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 which contains (among other information) the challenge ch and the new public

[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*.

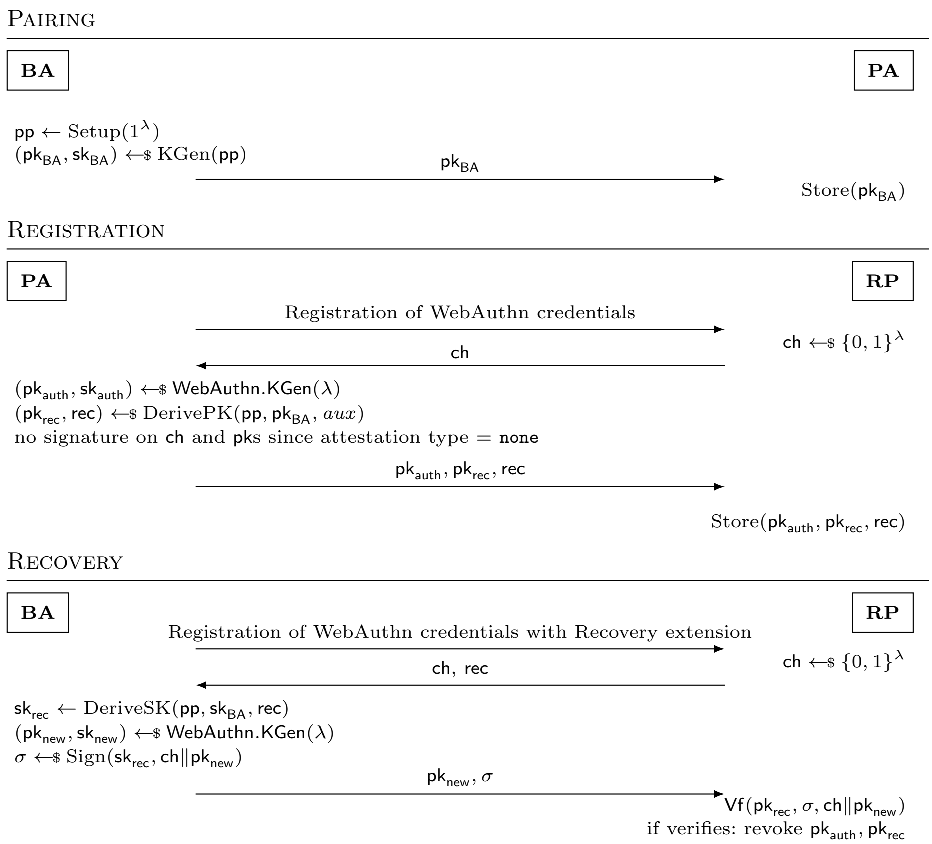

Fig. 1: Simplified illustration how ARKG integrates into WebAuthn registration flows

key $\mathsf{pk}_{\mathsf{auth}}$). The signature is created using the long-term secret key that is embedded in the authenticator at production time. Since no attestation is the proposed default, we chose to omit this signature from Figure 1.

When the recovery extension is present, the *PA* 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 *RP*.

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. 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 which then checks the signature wrt. its stored information. If the signature verifies, RP stores $\mathsf{pk}_{\mathsf{new}}$ and should revoke the old stored credentials.

Remark 2. After registration, 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.

3 Security of ARKG Schemes

In this section we discuss the security properties of ARKG schemes. When first introducing ARKG and in later works, Frymann et al. [8, 7, 10] described security 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 mentioned before, we chose 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. [8] 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. Their notion demands that in order to break the scheme, an adversary must be able to recover and *entire* secret key. However, in the context of FIDO2, recovery is broken if an adversary can successfully convince a relying party that it is authorized to register new credentials following a recovery. For this, the adversary does not necessarily need knowledge of the full secret key. Thus, we switch to a notion based on the adversary's (in)ability to successfully *authenticate.*

With regards to public-key unlinkability, we note that the definition in [8], 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: 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. 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, i.e., it may not query any oracles in superposition.

We note that it is generally assumed 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 also 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. [8, 7, 10] implicitly make this assumption in the weak form of their key-security property of ARKG, where they do not provide the adversary with an oracle that outputs derived secret keys for previously generated derived public keys. Nevertheless, we provide the adversary with oracles that leak the derived secret keys to achieve stronger notions of security by default, analogous to the strong version in Frymann et al.'s works [8, 7, 10].

We assume that the initial pairing between the BA and the PA(s) is in a trusted setting 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 (cf. [3]). In the context of ARKG this is especially important during registration, where the derived public keys and recovery information 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 takes place 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. [8] 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. Since our main contribution is a generic post-quantum secure ARKG construction for account recovery in FIDO2 authentications, we opt to define security in the latter sense as follows.

We discuss the differences between this notion, and the one given by Frymann et al. in more detail in Appendix A.

$\underline{\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})\text{:}}$

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

$\underline{\textsc{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, \cdot) \text{ on input } aux\text{:}}$

$(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathrm{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$
$\mathcal{L}_{\mathrm{keys}} \leftarrow \mathcal{L}_{\mathrm{keys}} \cup \{(\mathsf{pk}', \mathsf{rec}, aux)\}$
$\mathbf{return}\ (\mathsf{pk}', \mathsf{rec})$

$\underline{\textsc{Chall-auth}(\cdot) \text{ on input } aux\text{:}}$

$\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$\mathcal{L}_{\mathsf{ch}} \leftarrow \mathcal{L}_{\mathsf{ch}} \cup \{(\mathsf{ch}, aux)\}$
$\mathbf{return}\ \mathsf{ch}$

$\underline{\textsc{Sign}(\cdot,\cdot) \text{ on input } (\mathsf{rec}, m)\text{:}}$

$\mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
$\mathbf{if}\ \mathsf{sk}' = \perp\text{: abort}$
$\sigma \leftarrow\!\!\$\ \mathrm{Sign}(\mathsf{sk}', m)$
$\mathcal{L}_\sigma \leftarrow \mathcal{L}_\sigma \cup \{(\mathsf{rec}, m)\}$
$\mathbf{return}\ \sigma$

$\underline{\textsc{LeakSK}(\cdot) \text{ on input } \mathsf{rec}\text{:}}$

$\mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$
$\mathcal{L}_{\mathsf{sk}'} \leftarrow \mathcal{L}_{\mathsf{sk}'} \cup \{\mathsf{rec}\}$
$\mathbf{return}\ \mathsf{sk}'$

Fig. 2: Our security definition for authentication security of ARKG schemes.

Game description The formal description of the authentication game $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ can be found in 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, Sign, and 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 the relying parties that are 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 recovery information rec, the backup authenticator BA has no means to distinguish between credential information that had been honestly generated by a primary authenticator and recovery 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 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 $(\mathsf{pk}^\star, \mathsf{rec}^\star, aux^\star, m^\star, \sigma^\star)$ and wins the game $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$, if it is able to produce a valid signature on a message containing the challenge posed by a relying party. The valid signature must fulfill the following requirements:

- $(\mathsf{pk}^\star, \mathsf{rec}^\star)$ 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
- The secret key associated with $\mathsf{rec}^\star$ has not been given to the adversary.

Definition 2. *Let* $\mathsf{ARKG} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{DerivePK}, \mathrm{DeriveSK})$ *be an async. remote key generation scheme. We say that* ARKG *is* AUTH-secure*, if for every QPT adversary* $\mathcal{A}$ *the advantage in winning the game* $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ *described in Figure 2, defined as* $\mathsf{Adv}^{\mathrm{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A}) = 1\right] \right|$*, is negligible in the security parameter* λ.

Remark 3. Note that a weaker notion of authentication security, where $\mathcal{A}$ does not have access to the LeakSK oracle to learn other derived keys, could be defined. However, at least in our instantiation from KEMs we gain nothing from this modification as both notions require the same assumptions on the primitives.

3.3 Unlinkability

Unlinkability aims to fulfill a requirement in the WebAuthn standard [15] which recommends authenticators to 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, we 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. In essence, their definition allows to prove ARKG schemes as 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 a 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 ciphertext to the oracle LeakSK-u, even if the auxiliary information in the recovery credential has been modified. This does not impose any undue limitation, because during an honest execution the auxiliary information is the unique identifier of the relying party and thus remains unchanged. In the end, $\mathcal{A}$ will output a bit b', guessing whether the challenge was derived from $\mathsf{pk}^0_{\mathsf{BA}}$ or $\mathsf{pk}^1_{\mathsf{BA}}$ and wins if correct. More formally,

Definition 3. *As before, let* $\mathsf{ARKG} = (\mathrm{Setup}, \mathrm{KGen}, \mathrm{DerivePK}, \mathrm{DeriveSK})$ *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* $\mathsf{Exp}^{\mathrm{unl}}_{\mathsf{ARKG}}(\mathcal{A})$ *described in Figure 3, defined as* $\mathsf{Adv}^{\mathrm{unl}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathrm{unl}}_{\mathsf{ARKG}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right|$*, is negligible in the security parameter* λ*.*

Recall that [8] in their (public-key) unlinkability game ask to distinguish 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. Looking ahead to a post-quantum instantiation, the post-quantum KEM Kyber can also achieve this notion. The reason is that Kyber provides *strong pseudorandomness* under CCA [24], as shown in [19].

$\underline{\mathsf{Exp}^{\mathrm{unl}}_{\mathsf{ARKG}}(\mathcal{A})}$:

$\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda)$
$b \leftarrow\$ \{0,1\}$
$(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{sk}^0_{\mathsf{BA}}) \leftarrow\$ \mathrm{KGen}(\mathsf{pp})$
$(\mathsf{pk}^1_{\mathsf{BA}}, \mathsf{sk}^1_{\mathsf{BA}}) \leftarrow\$ \mathrm{KGen}(\mathsf{pp})$
$b' \leftarrow\$ \mathcal{A}^{\text{1-CHALL-U},\text{LEAKSK-U}}(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{pk}^1_{\mathsf{BA}})$
return $[\![b = b']\!]$

$\underline{\text{1-CHALL-U}(\cdot) \text{ on input } aux}$:

$(\mathsf{pk}', \mathsf{rec}) \leftarrow \mathrm{DerivePK}(\mathsf{pp}, \mathsf{pk}^b_{\mathsf{BA}}, aux)$
  with $\mathsf{rec} = (c^*, aux)$
$\mathsf{sk}' \leftarrow\$ \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}^b_{\mathsf{BA}}, \mathsf{rec})$
return $(\mathsf{pk}', \mathsf{sk}', \mathsf{rec})$

$\underline{\text{LEAKSK-U}(\cdot) \text{ on input } (\beta, \mathsf{rec})}$:

$(\mathrm{c}, aux) \leftarrow \mathsf{rec}$
if $\mathsf{rec} = (c^*, \cdot)$: abort
$\mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{pp}, \mathsf{sk}^\beta_{\mathsf{BA}}, \mathsf{rec})$
return sk'

Fig. 3: Our security definition for unlinkability of ARKG schemes.

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 4 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 [21, 18, 20, 16].

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 $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}})$, and transfers $\mathsf{pk}_{\mathsf{BA}}$ to the *PA*.

During registration, which is exclusively done by the *PA*, an encapsulation operation is performed under $\mathsf{pk}_{\mathsf{BA}}$ to obtain a random key, which is then input

Pairing

BA | **PA**

Setup (1^λ)
return $\mathsf{pp} = (\mathsf{KEM}, \mathsf{KDF}, \mathsf{Sig})$

KGen (pp)
$(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\$ \mathsf{KEM}.\mathrm{KGen}(\mathsf{pp})$

$\xrightarrow{\mathsf{pk}_{\mathsf{BA}}}$

Registration

PA | **RP**

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

$\xrightarrow{\mathsf{pk}', \mathsf{rec}}$

Recovery

BA | **RP**

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

$\xleftarrow{\mathsf{rec}}$

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

Setup (1^λ):

```
return pp = (KEM, KDF, Sig)
```

KGen(pp):

```
(pk_BA, sk_BA) ←$ KEM.KGen(pp)
```

DerivePK(pp, $\mathsf{pk}_{\mathsf{BA}}$, aux):

```
(c, K) ←$ KEM.Encaps(pk_BA)
r ← KDF(K, aux)
(pk', sk') ← Sig.KGen(pp; r)
rec ← (c, aux)
```

DeriveSK(pp, $\mathsf{sk}_{\mathsf{BA}}$, rec):

```
(c, aux) ← rec
K ← KEM.Decaps(sk_BA, c)
r ← KDF(K, aux)
(pk', sk') ← Sig.KGen(pp; r)
```

Fig. 4: Our PQ-ARKG instantiation from KEMs, Signatures and KDFs

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 pk'.

During recovery, the BA retrieves the ciphertext from the 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 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 [8] 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 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.

Authentication Security We first show authentication security.

Theorem 1 *Let* ARKG *be the generic instantiation of* ARKG *as given in Figure 4,* KEM *be an* IND-CCA *secure and* ϵ*-correct KEM scheme,* Sig *be an* EUF-CMA *secure signature scheme and* KDF *a secure key derivation function modeled as a* PRF*. Then* ARKG *provides* ϵ*-correctness and authentication security as defined in Definition 2. 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*

$$\mathsf{Adv}^{\text{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \quad \leq \quad q \ \cdot \ \left(\epsilon + \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\right)$$

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

Proof. Correctness with parameter ϵ for ARKG directly follows from ϵ-correctness of KEM: If one is able to decapsulate the right key, then one can also derive the same key pair. We will prove the authentication property of 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 $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$.

$\mathsf{Game}_2(\lambda)$**:** In this game we assume that KEM decapsulation for honestly generated public keys and ciphertexts never fails. This is always the case, except for a negligible failure probability ϵ for each of the at most q generated keys, given by Definition 1. Thus, we get the bound

$$\mathsf{Adv}^{\mathsf{game_1}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \epsilon + \mathsf{Adv}^{\mathsf{game_2}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

$\mathsf{Game}_3(\lambda)$**:** 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 $\mathcal{L}_{\mathrm{keys}}$. 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

$$\mathsf{Adv}^{\mathsf{game_2}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq q \cdot \mathsf{Adv}^{\mathsf{game_3}}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

$\mathsf{Game}_4(\lambda)$**:** 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 4).

We show that any efficient adversary $\mathcal{A}$, which can distinguish between game_3 and game_4 implies the existence of an efficient adversary $\mathcal{B}_1$ against the IND-CCA security of KEM. $\mathcal{B}_1$ receives the KEM challenge $(\mathsf{pk}^*, k^*, c^*)$ and initializes $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ with pk^* as $\mathsf{pk}_{\mathsf{BA}}$.

During the execution of DerivePK that has been guessed in game_3, algorithm $\mathcal{B}_1$ modifies the behavior of the algorithm by plugging in its own challenge: In line 1 of the DerivePK algorithm, pk^* is used for encapsulation; in line 2 k^* is used as input to the KDF . The ciphertext, which is output as a component of rec, is replaced with the ciphertext c^*.

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. As we are only retrieving stored keys, we implicitly eliminate all decryption failures, which is already captured by the transition to game_2. Queries to the Sign oracle for inputs that have not been generated by the DerivePK algorithm can be answered using the Decaps oracle provided by the IND-CCA challenger. Due to the guess in game_3, the challenge key and ciphertext is always embedded in an output of the DerivePK oracle, and therefore such a query to the Decaps oracle does not coincide with the challenge ciphertext, which subsequently means that the game's own Decaps oracle always answers.

To simulate 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 its own decryption oracle of the IND-CCA KEM to answer these different requests.
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_3 when the KEM challenge is real and game_4 when the KEM challenge is random. Consequently, we obtain the following bound:

$$\mathsf{Adv}^{\mathsf{game}_3}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{game}_4}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

$\mathsf{Game}_5(\lambda)$**:** 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 $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ for $\mathcal{A}$ as specified, but modifies the behavior of the KDF used as part of the DerivePK algorithm in line 2 (cf. Figure 4). 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:

$$\mathsf{Adv}^{\mathsf{game}_4}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{game}_5}_{\mathsf{ARKG},\mathcal{A}}(\lambda)\,.$$

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

$$\mathsf{Adv}^{\mathsf{game}_5}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) \,.$$

To conclude the proof, we sum up the advantages:

$$\mathsf{Adv}^{\text{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \epsilon + q \cdot \Big(\mathsf{Adv}^{\text{ind-cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\Big) \,.$$

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.

Unlinkability We next discuss unlinkability of our scheme. This follows from the fact that the underlying KEM scheme is anonymous [2].

Theorem 2 *Let* ARKG *be the instantiation of* ARKG *as shown in Figure 4 and* KEM *be an* ANON-CCA *secure* KEM *scheme. Then* ARKG *provides unlinkability security as described in Definition 3. 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* $\mathsf{Adv}^{\text{unl}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{anon-cca}}_{\mathsf{KEM},\mathcal{B}}(\lambda)$.

Proof. We prove Theorem 2 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 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 8). 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 4), the challenge key k^* is used as input to the KDF in combination with aux provided by the inner adversary $\mathcal{A}$. The KDF output is then used as input to the key generation algorithm 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 including about the challenge value c^* 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 is efficient and 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:

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

4.3 Overhead and Instantiation

Implementing ARKG requires relatively low additional computations and storage at both the relying party and the authenticator itself, with the overhead dependent on the instantiation of the underlying primitives. Crucially, during the most frequent operation, namely authentication, no additional computations are necessary. Per registration of an authenticator at a relying party only a single additional KEM key generation and encapsulation are performed. Similarly, the initial pairing (only done once) and account recovery require a KEM key generation and one single other operation (KEM decapsulation and signing, respectively). In terms of storage, the authenticator needs to store the backup authenticator's public key and the relying party needs to store the public key of the recovery credential and the recovery information rec.

Our solution can be instantiated with any suitable primitive that satisfies the security requirements stated in the theorems. 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. [8] 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 [7, 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.

Acknowledgements

We thank Varun Maram for pointing out flaws in the security proofs of a previous version of this work as well as the anonymous reviewers for their valuable comments. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1119-236615297 and the German Federal Ministry of Education and Research (BMBF) under reference 16KISQ074.

References

1. Barbosa, M., Boldyreva, A., Chen, S., Warinschi, B.: Provable security analysis of FIDO2. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 125–156. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84252-9˙5
2. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1˙33
3. Bindel, N., Cremers, C., Zhao, M.: FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation. In: IEEE Symposium on Security and Privacy (SP). pp. 674–693 (2023)
4. Bradley, J., Hodges, J., Jones, M.B., Kumar, A., Lindemann, R., Verrept, J., Antoine, M., Bharadwaj, V., Birgisson, A., Brand, C., Czeskis, A., Duboucher, T., Ehrensvärd, J., Ploch, M.J., Powers, A., Armstrong, C., Georgantas, K., Kaczmarczyck, F., Satragno, N., Sung, N.: Client to Authenticator Protocol (CTAP) (Jun 2022), https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html
5. Brendel, J., Clermont, S., Fischlin, M.: Post-quantum asynchronous remote key generation for fido2 account recovery. Cryptology ePrint Archive, Paper 2023/1275 (2023), https://eprint.iacr.org/2023/1275, https://eprint.iacr.org/2023/1275
6. Brendel, J., Fischlin, M., Günther, F., Janson, C., Stebila, D.: Towards post-quantum security for Signal's X3DH handshake. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 404–430. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0˙16
7. Frymann, N., Gardham, D., Manulis, M.: Asynchronous remote key generation for post-quantum cryptosystems from lattices. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroSP). pp. 928–941. IEEE Computer Society, Los Alamitos, CA, USA (jul 2023). https://doi.org/10.1109/EuroSP57164.2023.00059, https://doi.ieeecomputersociety.org/10.1109/EuroSP57164.2023.00059
8. Frymann, N., Gardham, D., Kiefer, F., Lundberg, E., Manulis, M., Nilsson, D.: Asynchronous remote key generation: An analysis of yubico's proposal for W3C WebAuthn. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 939–954. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417292
9. Frymann, N., Gardham, D., Manulis, M.: Unlinkable delegation of WebAuthn credentials. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part III. LNCS, vol. 13556, pp. 125–144. Springer, Heidelberg (Sep 2022). https://doi.org/10.1007/978-3-031-17143-7˙7
10. Frymann, N., Gardham, D., Manulis, M., Nartz, H.: Generalised asynchronous remote key generation for pairing-based cryptosystems. In: Applied Cryptography and Network Security: 21st International Conference, ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part I. p. 394–421. Springer-Verlag, Berlin, Heidelberg (2023), https://doi.org/10.1007/978-3-031-33488-7_15
11. Grubbs, P., Maram, V., Paterson, K.G.: Anonymous, robust post-quantum public key encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 402–432. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2˙15
12. Guan, J., Li, H., Ye, H., Zhao, Z.: A formal analysis of the FIDO2 protocols. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ES-

ORICS 2022, Part III. LNCS, vol. 13556, pp. 3–21. Springer, Heidelberg (Sep 2022). https://doi.org/10.1007/978-3-031-17143-7˙1
13. Hanzlik, L., Loss, J., Wagner, B.: Token meets wallet: Formalizing privacy and revocation for FIDO2. Cryptology ePrint Archive, Report 2022/084 (2022), https://eprint.iacr.org/2022/084
14. Harell, C.: Yubikeys, passkeys and the future of modern authentication (03 2022), https://www.yubico.com/blog/passkeys-and-the-future-of-modern-authentication/
15. Hodges, J., Jones, J., Jones, M.B., Kumar, A., Lundberg, E., Bradley, J., Brand, C., Langley, A., Mandyam, G., Satragno, N., Steele, N., Tan, J., Weeden, S., West, M., Yasskin, J.: Web Authentication: An API for accessing Public Key Credentials - Level 3 (Apr 2021), https://www.w3.org/TR/webauthn-3
16. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS$^+$. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
17. Lundberg, E., Nielsson, D.: WebAuthn Recovery Extension (2019), https://github.com/Yubico/webauthn-recovery-extension
18. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
19. Maram, V., Xagawa, K.: Post-quantum anonymity of Kyber. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 3–35. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4˙1
20. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
21. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D., Ding, J.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
22. Shikiar, A.: Charting an Accelerated Path Forward for Passwordless Authentication Adoption (03 2022), https://fidoalliance.org/charting-an-accelerated-path-forward-for-passwordless-authentication-adoption/
23. Wilson, Spencer MacLaren: Post-Quantum Account Recovery for Passwordless Authentication. Master's thesis (2023), http://hdl.handle.net/10012/19316
24. Xagawa, K.: Anonymity of NIST PQC round 3 KEMs. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 551–581. Springer, Heidelberg (May / Jun 2022). 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. [8, 7, 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 [8] in the *honest* setting in Figure 5.

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

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

$\underline{\mathcal{O}_{\mathsf{pk}'}(\mathsf{pp}, \mathsf{pk}_0, \cdot) \text{ on input } aux\text{:}}$

7 $(\mathsf{pk}', \mathsf{rec}) \leftarrow\$ \mathrm{DerivePK}(\mathsf{pp}, \mathsf{pk}_0, aux)$
8 $\mathcal{L}_{\mathrm{keys}} \leftarrow \mathcal{L}_{\mathrm{keys}} \cup (\mathsf{pk}', \mathsf{rec})$
9 **return** $(\mathsf{pk}', \mathsf{rec})$

$\underline{\mathcal{O}_{\mathsf{sk}'}(\cdot) \text{ on input } \mathsf{rec}\text{:}}$

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

Fig. 5: SK-Security as defined in [8] (honest variants).[4]

Honest vs. malicious security For this security definition, the term *honest* refers to the first requirement in Line 6 in Figure 5, which enforces that $(\mathsf{pk}^\star, \mathsf{rec}^\star)$ must be in $\mathcal{L}_{\mathrm{keys}}$, i.e., that the tuple was honestly generated via DerivePK. This same requirement is mirrored in our check that $(\mathsf{pk}', aux) \in \mathcal{L}_{\mathrm{keys}}$ 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 note in [7], 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 5. If it is present

[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. [8].

(the strong setting), the adversary is required to output a valid secret key $\mathsf{sk}^\star$ for a pair $(\mathsf{pk}', \mathsf{rec}^\star)$ 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 AUTH-security notion, which takes the place of key security, 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 their *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-security definition provides capabilities to an adversary roughly comparable to the strong setting. The provided oracles correlate to the case, where there is virtually unlimited access to the backup authenticator with all its capabilities, except of course for trivial attacks. Such an attacker is very powerful and a weaker notion could be defined, but as observed in Remark 3 relaxing the notion would not ease any of the requirements for the underlying primitives.

A.2 Public-Key Unlinkability

Figure 6 gives a complete pseudocode description of the so-called public-key unlinkability as proposed by Frymann et al. [8]. 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.

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

$$
\begin{array}{l}
\mathsf{pp} \leftarrow \mathrm{Setup}(1^\lambda) \\
(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\!\$\ \mathrm{KGen}(\mathsf{pp}) \\
b \leftarrow\!\!\$\ \{0,1\} \\
b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}^b_{\mathsf{pk}'}}(\mathsf{pp}, \mathsf{pk}_0) \\
\textbf{return } [\![b = b']\!]
\end{array}
$$

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

$$
\begin{array}{l}
\textbf{if } b = 0 \\
\quad (\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathrm{DerivePK}(\mathsf{pk}_{\mathsf{BA}}, aux) \\
\quad \mathsf{sk}' \leftarrow \mathrm{DeriveSK}(\mathsf{sk}_{\mathsf{BA}}, \mathsf{rec}) \\
\textbf{else} \\
\quad (\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{D} \\
\textbf{return } (\mathsf{pk}', \mathsf{sk}')
\end{array}
$$

Fig. 6: Public-key unlinkability as defined in [8].[5]

[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. [8]. In particular,

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.

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

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 $(\mathsf{pk}, \mathsf{sk})$. Encaps is a probabilistic algorithm and takes as input a public key pk, where $\mathsf{pk} \leftarrow \mathrm{KGen}(1^\lambda)$, and outputs a key k as well as a ciphertext c. The ciphertext c encapsulates the key k. Decapsis 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* $\mathsf{KEM} = (\mathrm{KGen}, \mathrm{Encaps}, \mathrm{Decaps})$ *is* δ-correct*, if for all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathrm{KGen}$ *we have* $\Pr[\mathrm{Decaps}(\mathsf{sk}, c) = k : (c, k) \leftarrow\$ \mathrm{Encaps}(\mathsf{pk})] \geq 1 - \delta$ *If* $\delta = 0$ *holds, the scheme is called* perfectly correct*.*

Security of a KEM scheme is defined over indistinguishability of derived keys and random keys. A challenger is provided a triple (pk, c, k_b), where c is output by $(c, k) \leftarrow \mathrm{Encaps}(\mathsf{pk})$ 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.

Definition 2 (IND-ATK security of KEM schemes) *Given the security game in Figure 7, a key encapsulation scheme* $\mathsf{KEM} = (\mathrm{KGen}, \mathrm{Encaps}, \mathrm{Decaps})$ *is* IND-ATK *secure for* ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$*, 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 in the security parameter* λ *for any QPT adversary* $\mathcal{A}$*.*

$\mathsf{Exp}^{\text{ind-cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:

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

$\mathsf{Exp}^{\text{ind-cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathrm{KGen}(1^\lambda)$
$k_0 \leftarrow\$ \mathcal{K}$
$(c^*, k_1) \leftarrow\$ \mathrm{Encaps}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{O}_{Dec}(\cdot)}(\mathsf{pk}, c^*, k_b)$
return $[\![b' = b]\!]$

$\mathsf{O}_{Dec}(\cdot)$ on input c:

if $c = c^*$
  return $\perp$
return $\mathrm{Decaps}(\mathsf{sk}, c)$

Fig. 7: Game definition for IND-ATK security of key encapsulation mechanisms with ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$

An additional property some 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 8, a key encapsulation scheme* KEM*, is* ANON-ATK *secure with* ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$*, if the advantage* $\mathsf{Adv}^{\text{ANON-ATK}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\text{ANON-ATK}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$ *is negligible in the security parameter* λ *for any QPT adversary* $\mathcal{A}$*.*

$\underline{\mathsf{Exp}^{\text{anon-cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:

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

$\underline{\mathsf{Exp}^{\text{anon-cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:

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

$\underline{\mathsf{O}_{Dec}(\cdot,\cdot) \text{ on input } id, c}$:

if $c = c^*$
    return $\perp$
$k = \text{Decaps}(\mathsf{sk}_{id}, c)$
return k

Fig. 8: Game definition for ANON-ATK anonymity of key encapsulation mechanisms with ATK $\in$ {CPA, CCA}

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 $\mathsf{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$.

KGen takes as input the security parameter λ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$. 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* $\mathsf{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$ *is correct, if* $\Pr\big[0 \leftarrow \text{Vrfy}(\mathsf{pk}, \sigma, m) : (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$ \text{KGen}(1^\lambda), \sigma \leftarrow\!\!\$ \text{Sign}(\mathsf{sk}, m)\big]$ *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 9, a digital signature scheme* $\mathsf{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$ *is* EUF-CMA *secure, if* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\Big[\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = 1\Big]$ *is negligible in the security parameter* λ *for any QPT adversaries* $\mathcal{A}$.

B.3 PRF Security

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 10, a* PRF *is secure, if for all QPT adversaries* $\mathcal{A}$ *the following holds* $\mathsf{Adv}^{\text{prf}}_{F,\mathcal{A}}(\lambda) := \Big|\Pr\Big[\mathsf{Exp}^{\text{prf}}_{F,\mathcal{A}}(\lambda) = 1\Big] - \frac{1}{2}\Big|$

$\underline{\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda)\text{:}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathrm{KGen}(1^\lambda)$
$\mathcal{L}_m \leftarrow \emptyset$
$(m', \sigma') \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}_{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
return $[\![\mathrm{Vrfy}(\mathsf{pk}, \sigma', m') \land m' \notin \mathcal{L}_m]\!]$

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

$\sigma \leftarrow\!\!\$\ \mathrm{Sign}(\mathsf{sk}, m)$
$\mathcal{L}_m \leftarrow \mathcal{L}_m \cup \{m\}$
return σ

Fig. 9: Game definition for EUF-CMA security of signature schemes

$\underline{\mathsf{Exp}^{\text{prf}}_{F,\mathcal{A}}(\lambda)\text{:}}$

$b \leftarrow\!\!\$\ \{0, 1\}$
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
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}(f,\cdot)}$
return $[\![b' = b]\!]$

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

return $f(x)$

Fig. 10: Game definition for PRF security of a function F