Post-Quantum Asynchronous Remote Key Generation for FIDO2

Jacqueline Brendel, Sebastian Clermont, and Marc Fischlin

Technische Universität Darmstadt, Darmstadt, Germany
mail@jbrendel-info.de
sebastian.clermont@tu-darmstadt.de
marc.fischlin@tu-darmstadt.de

Abstract. The Fast IDentity Online (FIDO) Alliance has developed the widely adopted FIDO2 protocol suite that allows for passwordless online authentication. Cryptographic keys stored on a user's device (e.g. their smartphone) are used as credentials to authenticate to services by performing a challenge-response protocol. Yet, this approach leaves users unable to access their accounts in case their authenticator is lost.
The device manufacturer Yubico thus proposed a FIDO2-compliant mechanism that allows to easily create backup authenticators. Frymann et al. (CCS 2020) have first analyzed the cryptographic core of this proposal by introducing the new primitive of *Asynchronous Remote Key Generation* (ARKG) and accompanying security definitions. Later works instantiated ARKG both from classical and post-quantum assumptions (ACNS 2023, EuroS&P 2023).
As we will point out in this paper, the security definitions put forward and used in these papers do not adequately capture the desired security requirements in FIDO2-based authentication and recovery. This issue was also identified in independent and concurrent work by Stebila and Wilson (AsiaCCS 2024), who proposed a new framework for the analysis of account recovery mechanisms, along with a secure post-quantum instantiation from KEMs and key-blinding signature schemes.
In this work, we propose alternative security definitions for the primitive ARKG when used inside an account recovery mechanism in FIDO2. We give a secure instantiation from KEMs and standard signature schemes, which may in particular provide post-quantum security. Our solution strikes a middle ground between the compact, but (for this particular use case) inadequate security notions put forward by Frymann et al., and the secure, but more involved and highly tailored model introduced by Stebila and Wilson.

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

1 Introduction

FIDO2 encompasses a set of pioneering industry standards for passwordless authentication on the web [5, 14] that is driven and supported by a wide consortium

of major industry players and stakeholders. At its core, passwords are replaced by a sophisticated combination of public-key cryptography and hardware devices such as smartphones or dedicated security keys. In a FIDO2 authentication ceremony, users authenticate themselves by proving in a challenge-response fashion that they control the secret key of a previously registered public-key credential.

There are three main actors: the *authenticator*, the *client*, and the *relying party*. The relying party is the service that a user wants to log in to, i.e., the party that will verify the user's identity. The authenticator is the device holding the secret key material used for signing the cryptographic authentication challenge issued by the relying party. The client is the device that the user uses to access the authenticator and to communicate with the relying party.

For example, suppose a user is logging into their Gmail account using their laptop with a Yubico security key. In that case, their Yubikey is the authenticator, the laptop (and the browser on it) is the client, and Google is the relying party. The set of cryptographic information required by a user for a successful authentication is referred to as a *passkey* [21] or simply a (FIDO2) credential.

1.1 The Need for Credential Recovery

When using devices such as a smartphone or a USB security key for authentication, the possibility of transferring or recovering credentials must be considered. Essentially, there are two relevant scenarios:

1. One is the graceful transition from an old authenticator to a new one when the previously registered credentials are still accessible to the user, e.g., when switching smartphones.
2. The other, more challenging, situation occurs when the original credentials are no longer available, e.g., in case the smartphone is lost, stolen, or broken.

While the first scenario can be easily accommodated by logging on with the old device and then adding the new device as an authenticator, the second case is much trickier.

For the secure recovery of *symmetric* secrets, mechanisms have already been created. Signal's *Secure Value Recovery* or Apple's *Secure iCloud Keychain Recovery* serve as two examples. These mechanisms rely on a low-entropy recovery PIN created by the user, which encrypts a high-entropy key stored on a distributed hardware security module (HSM). This high-entropy key is then used to encrypt the user's 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.

For the secure recovery of *asymmetric secrets*, as they are used within FIDO2, the aforementioned approach should not, and often cannot, be applied. For once, the entire security relies on a low-entropy PIN chosen by the user. Furthermore, the approach requires trust in cloud vendors and their HSMs to behave as promised, which is an unsatisfactory assumption at best. Lastly, dedicated hardware authenticators, such as FIDO2 security keys, usually generate their

cryptographic secrets locally and do not allow for their extraction from the device.

The FIDO2 Alliance suggests the usage of so-called *multi-device credentials* [21] to ensure reliable access to relying parties. Multi-device credentials are not constrained to the physical devices they were generated on but can be copied across a user's devices to ensure a consistent login experience. While this is an easy-to-use approach, it has numerous security drawbacks.

In principle, it is possible for users to then have multiple, functionally-identical authenticators, by synchronizing the passkeys over a vendor's cloud service. This solves the recovery problem straightforwardly: If one authenticator is lost, a new one can be acquired, and the multi-device credentials can be copied to this new authenticator. However, this also means that:

- The fine-grained revocation of credentials is impossible. All authenticators behave identically and use the same secrets, i.e., a user cannot revoke access for the lost credential, since the credentials on his new authenticator are identical.
- Hardware binding is impossible. A user can never be certain of being the sole owner of their secret keys and thus their account access, as it might have been copied to a different device without their knowledge or consent.

Furthermore, the relying party may insist on single-device passkeys for high-security use cases. This functionality is specified as part of the FIDO2 protocol suite and is thus incompatible with this recovery approach. Lastly, as already mentioned above, dedicated hardware security keys are built to be *tamperproof* which in particular means that they may not support the export of secret key material.

1.2 Introducing Backup Authenticators

In light of this, Yubico, one of the leading manufacturers of hardware security devices, takes the stance that allowing secret key material to leave the authenticators is an inherent weakness in the system and must be avoided [13]. Consequently, Yubico does not support the creation of multi-device FIDO2 passkeys and strictly follows a one-device, one-credential policy.

Yubico has thus suggested an alternative approach, using so-called *backup authenticators* (BA) to facilitate account recovery when a user is faced with lost or broken authenticators [16]. This solution is an easy-to-implement extension to the WebAuthn protocol in FIDO2 and bridges the gap in account recovery mechanisms, as it eliminates the need for trusted third parties, while also maintaining the full autonomy of the authenticators.

A backup authenticator is a hardware device that is initialized *once* by creating a cryptographic key pair. The public key of this key pair is then transferred once to the *primary authenticator* (PA), which is a regular authenticator that is used for day-to-day authentications to relying parties. After this initial pairing between backup and primary authenticator, the backup can be stored offline in a

safe location and will only be required in case a user needs to recover credentials, e.g., when they have lost their primary authenticator.

In particular, when registering a new credential with a relying party, and in any subsequent login, the user only needs their primary authenticator. To enable this some additional protected recovery information is stored at the relying party when the FIDO2 credential is first registered. The backup authenticator uses its secret key to recover the authentication data from this externally stored recovery information.

In more detail, the solution by Yubico is a Diffie–Hellman-based protocol, which roughly lets the backup authenticator and the primary authenticator each generate the public part of a Diffie–Hellman (DH) key share. The primary authenticator stores its public DH share as recovery information at the relying party, along with the public key of the joint DH key derived from the PA's and BA's DH shares. To serve a recovery request, the backup authenticator computes the joint DH secret by combining the externally stored public DH share of the primary authenticator with its own DH secret. The resulting joint DH secret key can then be used to authenticate the recovery request to the relying party.

Note, that the recovery DH keys are separate from the "regular" FIDO2 credentials, which are generated independently and are used for authentications with the primary authenticator during regular protocol execution.

1.3 Security of FIDO2 and Yubico's Proposal

Backup authenticators and how to create them from the newly introduced primitive for asynchronous remote key generation is not part of the FIDO2 standard. Consequently, there exist no established security requirements, let alone any formal definitions. The cryptographic core of the account recovery extension to WebAuthn has first been formalized and analyzed by Frymann et al. [8] through the introduction of a new cryptographic primitive which they termed *Asynchronous Remote Key Generation*, or ARKG, for short. Since then, a subset of the authors of [8] has also introduced alternative instantiations of the ARKG primitive based on pairings [10] and lattices [7]. The latter makes use of the (equally new) primitive of *split-KEMs*, which was first introduced by Brendel et al. [6] in the context of post-quantum asynchronous key exchange.

As we will point out in more detail shortly, we believe that the ARKG security notions proposed by Frymann et al. [8] are not adequate for the proposed use case of FIDO2 account recovery. In independent and concurrent work to ours, Stebila and Wilson [22], have also arrived at this conclusion. While we opt to "fix" the security notions for ARKG, Stebila and Wilson go a step further and propose a new abstraction for account recovery mechanisms in place of ARKG. Their security analysis is closer to a protocol security analysis than the analysis of a stand-alone cryptographic primitive. As such, no straightforward implications between our security notions and theirs can be shown.

The underlying FIDO2 protocol was first analyzed by Barbosa, Boldyreva, Chen, and Warinschi [1] using the provable security paradigm. A formal analysis

of FIDO2 has been conducted by Guan, Li, He, and Zhao [11]. More comprehensive analyses capturing newer protocol versions and modes, as well as advanced security features have been conducted by Hanzlik, Loss, and Wagner [12] as well as Bindel, Cremers and Zhao [3]. In [4], Bindel, Gama, Guasch, and Ronen analyzed the different attestation modes specified for FIDO2. In particular, [3] proposed and analyzed a full post-quantum instantiation of the latest iteration of the FIDO2 standards CTAP 2.1 and WebAuthn 2, requiring only minor extensions to the protocol, thus showing that the functionality of the FIDO2 protocol family can be achieved without any classical hardness assumptions.

1.4 Our Contributions

We have found the security notions for ARKG put forward in [8] and further used in the subsequent works [7, 10] to be a poor fit *when used inside the FIDO2 framework*.[1]

Frymann et al. [8] have defined two security definitions for ARKG. One covers the security of the backup authentication mechanism, called *secret-key security* or *key security*, which intuitively requires that the adversary cannot produce the entire 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 second property they propose is *public-key unlinkability*. It captures the relying parties' inability to track users across different services via the backup authenticator's public keys. This is formalized by an indistinguishability notion where the adversary learns a backup authenticator's long-term public key and either receives keys derived from this long-term public key or independently generated keys.

We set off by giving more befitting security notions for both key security and public-key unlinkability, which may also prove to be advantageous in other use cases of ARKG beyond FIDO2. In the course of these adaptations, we change the names of the security properties to *authentication security* and *unlinkability*, since the first property not only protects the secret key but also prevents forgeries, and the privacy property now also takes other available data beyond the public key into account.

In a bit more detail, for authentication security, we strengthen the adversary in line with the FIDO2 use case by counting the adversary already as successful if it can produce signature forgeries under backup keys. In practice, this means an adversary cannot access the recovery mechanism and register new credentials on behalf of the user, locking them out of their accounts.

For unlinkability, we follow a left-or-right approach where the adversary cannot decide which of two backup keys has been generated given derived public

[1] We stress that we have not considered other contexts in which ARKG may be employed, for which the originally proposed security properties may be appropriate.

keys and recovery information. This extends the previous definition to now also include the recovery data which is accessible to the adversary in the real-world use case. We will give a detailed discussion of the shortcomings in the analysis of [8] and our adjustments in Section 3.

Our second contribution is then the proposal of a FIDO2-compatible ARKG instantiation from standard KEMs and signature schemes in Section 4 to provide post-quantum security. 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 store this ciphertext externally at the relying party. During recovery, the backup authenticator can retrieve the randomness and re-generate the signing key.

In particular, we do not rely on the only recently-introduced notions of split KEMs as necessary in [7] or key-blinding signature schemes as in [22]. We note, that both of these primitives, split KEMs and key-blinding signatures, are not by themselves standardized, which may be a requirement for some future implementations and use cases.

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].

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 $\mathrm{prefix}(x) = y$ 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., $\mathsf{Sign}(\mathsf{sk}, m; r)$ 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.

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

Terminology In FIDO2, services are referred to as *relying parties* (*RP*), to which users can authenticate via so-called *authenticators*. In this work, we view a user with their authenticator(s) as a singular actor and abstract away the client that is located between the authenticator and the relying party. This abstraction removes the analysis of the CTAP protocol but comes without loss of generality

as account recovery with ARKG is a WebAuthn extension, which is exactly the protocol executed between client and 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.

2.1 ARKG Syntax

We now recall the notion of asynchronous remote key generation schemes introduced in [8] but slightly change notation to align it more with the intended purpose of account recovery in FIDO2. 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 $(\mathsf{pk}, \mathsf{sk})$. They are generated together with recovery information rec via the algorithm $\mathsf{DerivePK}$ 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 $\mathsf{DeriveSK}$. 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* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{DerivePK}, \mathsf{DeriveSK}, \mathsf{Check})$ *such that*

- Setup *takes as input the security parameter* λ *in unary and outputs the public parameters, i.e.,* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
- KGen *takes as input the public parameters* pp *and output a public/secret key pair* $(\mathsf{pk}_{BA}, \mathsf{sk}_{BA}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ *for the backup authenticator.*
- $\mathsf{DerivePK}$ *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.,* $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{BA}, aux)$.
- $\mathsf{DeriveSK}$ *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.,* $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{BA}, \mathsf{rec})$*, 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* $(\mathsf{pk}, \mathsf{sk})$ *form a valid public/secret key pair, and 0 otherwise.*

We say that an asynchronous remote key generation scheme $\mathsf{ARKG} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{DerivePK}, \mathsf{DeriveSK}, \mathsf{Check})$ *is* ϵ-correct*, if for all* λ *and* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pk}_{BA}, \mathsf{sk}_{BA}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$*, and auxiliary information* aux *we have that the*

[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.

probability of Check *outputting 0 is bounded by* ϵ*, i.e.,* $\Pr\left[\mathsf{Check}(\mathsf{pk}', \mathsf{sk}') = 0\right] \leq \epsilon$*, for* $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{BA}, aux)$ *and* $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{BA}, \mathsf{rec})$.

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 $\mathsf{Check}(\mathsf{pk}, \mathsf{sk})$ as part of their ARKG syntax, which is necessary to define correctness in the ARKG setting. In public-key cryptography, one can always leverage the randomness that went into key generation to implement such a check. That is, we define the secret key as the randomness used 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 the secret key. Indeed, our generic construction follows this approach, such that we do not give a concrete instantiation of Check.

2.2 ARKG in the Context of FIDO2

As also mentioned in [8], the ARKG primitive itself 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]. This work focuses on the original motivation for the introduction of ARKG: 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 [14] 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(s) be paired with a primary authenticator. We note that it is possible to pair any primary authenticator with multiple backup authenticators, and vice versa. Nonetheless, for ease of presentation, we focus on the case where a single *PA* is paired with a single *BA*. 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* begins to register credentials with relying parties. This registration happens over a secure channel since the user has logged into the relying party via another authentication

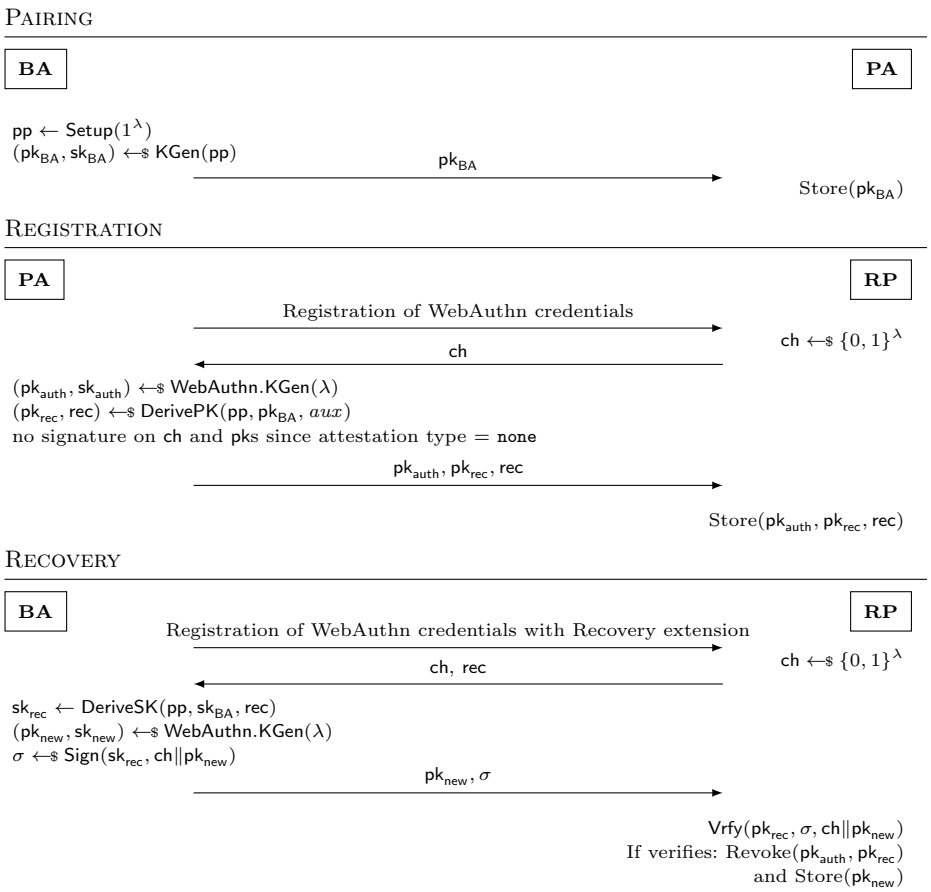

Fig. 1: Simplified illustration of how ARKG integrates into WebAuthn registration flows with default attestation type `none`.

method, typically with a username and password, and has established a TLS connection to the server of the *RP*.

In a regular WebAuthn registration procedure, 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 that contains (among other information) the challenge ch and the new public key $\mathsf{pk}_{\mathsf{auth}}$. This signature is created using the (highly non-unique) 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 $\mathsf{DerivePK}$; $(\mathsf{pk}_{\mathsf{rec}}, \mathsf{rec})$ 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 the relying party. Finally, 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 the initial registration, the user can use their 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 is valid, the user is authenticated and logged onto the service.

3 Security 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 should guarantee that an adversary is not able to successfully complete the account recovery process without access to the secret key stored on the backup authenticator, whereas public-key unlinkability shall ensure that users cannot be tracked across services via their registered public-key credentials. Unfortunately, neither of the previously proposed notions adequately provides these guarantees in the FIDO2 setting.

The key security notion in [8] demands that in order to break the scheme, an adversary must be able to recover the *entire* secret key. However, in the context of FIDO2, recovery is broken if an adversary can successfully convince a relying party that they are authorized to register new credentials following a recovery. For this, the adversary does not 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.

In the following, we provide the security notions of *authentication security* and *unlinkablity* that fix the just mentioned shortcomings. Before we formally define these security properties for ARKG schemes, we first state our basic assumptions on the adversary's power and capabilities.

The formal definitions from [8] can be found in Appendix A.

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 tamperproof, i.e., they do not leak information on the secret keys stored on them, even if they are in the adversary's possession. 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., might 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. 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 [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 into 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. This assumption is in line with the security model for FIDO2.

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.

We remind the reader of the possibility of pairing any primary authenticator with multiple backup authenticators, and vice versa. For the former case, all interactions are now carried out for each of the registered backup authenticators in parallel. This introduces a minor side channel by revealing the number of registered backup authenticators to relying parties. For the latter case, no change in the protocol is required and the security remains unaffected.

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 2.2, 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.

Game description The formal description of the authentication game $\mathsf{Exp}^{\mathrm{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ is given 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*.

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

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

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

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

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

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

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

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

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

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

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

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 some 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 signs the provided message m, returning the signature σ. If the secret key derivation fails, the oracle simply aborts.

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* $\mathsf{ARKG} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{DerivePK}, \mathsf{DeriveSK})$ *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* $\mathsf{Exp}^{\text{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ *described in Figure 2, defined as*

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

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 [14] 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 of public-key unlinkability, which was based on the adversary's inability to distinguish derived from randomly sampled key pairs. 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 the 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

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

1 $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 $b \leftarrow\!\!\$\ \{0,1\}$
3 $(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{sk}^0_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
4 $(\mathsf{pk}^1_{\mathsf{BA}}, \mathsf{sk}^1_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
5 $b' \leftarrow\!\!\$\ \mathcal{A}^{\text{1-CHALL-U},\text{LEAKSK-U}}(\mathsf{pk}^0_{\mathsf{BA}}, \mathsf{pk}^1_{\mathsf{BA}})$
6 **return** $[\![b = b']\!]$

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

7 $(\mathsf{pk}', \mathsf{rec}) \leftarrow \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}^b_{\mathsf{BA}}, aux)$
8 **with** $\mathsf{rec} = (c^*, aux)$
9 $\mathsf{sk}' \leftarrow\!\!\$\ \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}^b_{\mathsf{BA}}, \mathsf{rec})$
10 **return** $(\mathsf{pk}', \mathsf{sk}', \mathsf{rec})$

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

11 $(\mathrm{c},\ aux) \leftarrow \mathsf{rec}$
12 **if** $\mathsf{rec} = (c^*, \cdot)$: abort
13 $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}^\beta_{\mathsf{BA}}, \mathsf{rec})$
14 **return** sk'

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

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* $\mathsf{ARKG} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{DerivePK}, \mathsf{DeriveSK})$ *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 asks to distinguish genuinely generated public keys against independently sampled ones. This requires defining a distribution of 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 ML-KEM Kyber, currently being standardized as FIPS 203, achieves this notion. The reason is that Kyber provides *strong pseudorandomness* under CCA [23], as shown in [18].

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 [20, 17, 19, 15]. Formal definitions of the building blocks used in this chapter can be found in Appendix B. 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*.

$\underline{\mathsf{Setup}\ (1^\lambda)\text{:}}$
1 **return** $\mathsf{pp} = (\mathsf{KEM}, \mathsf{KDF}, \mathsf{Sig})$

$\underline{\mathsf{KGen}(\mathsf{pp})\text{:}}$
1 $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KEM.KGen}(\mathsf{pp})$

$\underline{\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)\text{:}}$
1 $(c, K) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk}_{\mathsf{BA}})$
2 $r \leftarrow \mathsf{KDF}(K, aux)$
3 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$
4 $\mathsf{rec} \leftarrow (c, aux)$

$\underline{\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})\text{:}}$
1 $(c, aux) \leftarrow \mathsf{rec}$
2 $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{sk}_{\mathsf{BA}}, c)$
3 $r \leftarrow \mathsf{KDF}(K, aux)$
4 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$

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

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

During recovery, the *BA* retrieves the ciphertext from the *RP* 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

Our instantiation of ARKG achieves both authentication security, as well as unlinkability, as reflected in the following theorems. The standard definitions of security for the employed cryptographic primitives can be found in Appendix B.

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*

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

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 $\mathsf{Exp}^{\text{auth}}_{\mathsf{ARKG}}(\mathcal{A})$.

$\mathsf{Game}_2(\lambda)$**:** 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 $\mathcal{L}_{\text{keys}}$. 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 $1/q$ 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 $\mathsf{Exp}^{\text{auth}}_{\mathsf{ARKG}}(\mathcal{A})$ 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:

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

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 the 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:

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

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. For each 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.

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, Cham, 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, Berlin, 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. Bindel, N., Gama, N., Guasch, S., Ronen, E.: To attest or not to attest, this is the question - provable attestation in FIDO2. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VI. LNCS, vol. 14443, pp. 297–328. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8736-8˙10
5. 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
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, Cham (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)
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, Cham (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)
11. 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.) ESORICS 2022, Part III. LNCS, vol. 13556, pp. 3–21. Springer, Cham (Sep 2022). https://doi.org/10.1007/978-3-031-17143-7˙1
12. Hanzlik, L., Loss, J., Wagner, B.: Token meets wallet: Formalizing privacy and revocation for FIDO2. In: 2023 IEEE Symposium on Security and Privacy. pp. 1491–1508. IEEE Computer Society Press (May 2023). https://doi.org/10.1109/SP46215.2023.10179373
13. Harell, C.: Yubikeys, passkeys and the future of modern authentication (03 2022), https://www.yubico.com/blog/passkeys-and-the-future-of-modern-authentication/
14. 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
15. 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 Tech-

nology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
16. Lundberg, E., Nielsson, D.: WebAuthn Recovery Extension (2019), https://github.com/Yubico/webauthn-recovery-extension
17. 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
18. 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, Cham (May 2023). https://doi.org/10.1007/978-3-031-31368-4˙1
19. 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
20. 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
21. 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/
22. Stebila, D., Wilson, S.: Quantum-safe account recovery for webauthn. Cryptology ePrint Archive, Paper 2024/678 (2024). https://doi.org/10.1145/3634737.3661138, https://eprint.iacr.org/2024/678, to appear at AsiaCCS '24
23. 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, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2˙20

A Comparison to the Different Security Definitions

In the following, we review the security definitions of ARKG schemes as proposed by Frymann et al. [8, 7, 10] and compare them to our proposed notions.

A.1 Key Security

Intuitively, key security formalizes the confidentiality of the derived secret keys and assures that no party can learn the key without knowledge of the BAs secret key. The notions by Frymann et al. [8] 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 $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 $\mathcal{L}_{\mathrm{keys}}, \mathcal{L}_{\mathsf{sk}'} \leftarrow \emptyset$
3 $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
4 $(\mathsf{pk}^\star, \mathsf{sk}^\star, \mathsf{rec}^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{pk}'}, \mathcal{O}_{\mathsf{sk}'}}(\mathsf{pp}, \mathsf{pk}_0)$
5 $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{rec}^\star)$
6 **return** $[\![(\mathsf{pk}^\star, \mathsf{rec}^\star) \in \mathcal{L}_{\mathrm{keys}} \wedge \mathsf{Check}(\mathsf{pk}^\star, \mathsf{sk}^\star) = 1 \wedge \mathsf{Check}(\mathsf{pk}^\star, \mathsf{sk}') = 1 \wedge \mathsf{rec}^\star \notin \mathcal{L}_{\mathsf{sk}'}]\!]$

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

7 $(\mathsf{pk}', \mathsf{rec}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_0, aux)$
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 $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{rec})$
11 $\mathcal{L}_{\mathsf{sk}'} \leftarrow \mathcal{L}_{\mathsf{sk}'} \cup \mathsf{rec}$
12 **if** $(\cdot, \mathsf{rec}) \notin \mathcal{L}_{\mathrm{keys}}$: abort
13 **return** sk'

Fig. 5: SK-Security as defined in [8] (honest variants).[3]

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 $\mathsf{DerivePK}$. 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".

[3] We note that the pseudocode descriptions of $\mathcal{O}_{\mathsf{pk}'}$ and $\mathcal{O}_{\mathsf{sk}'}$ have not been given before and thus correspond merely to our interpretation of the prose description. [8].

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 (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 argue that the formalization by Frymann et al. is inadequate in the context of ARKG within FIDO2. An adversary that can output the triple of values specified by the security game cannot complete a malicious account recovery. In other words, a successful attacker against the security notions is unable to mount an actual attack in the real world. This problem persists independent of the weak/strong and honest/malicious definitions.

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 under the honestly generated public key it has stored as the 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

The security notion of unlinkability has the goal of capturing an adversary's inability to identify multiple derived public keys belonging to the same backup authenticator.

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.

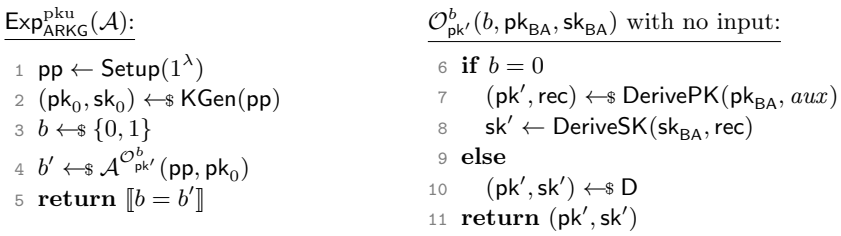

Fig. 6: Public-key unlinkability as defined in [8].[4]

Delineation On its own, the definition by Frymann et al. 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 under Frymann et al.'s definition can output trivially linkable keys, which is also observed in [22] This is because the definition above does not take into account the actual information an adversary has at 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.

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.

[4] 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, it is underspecified how *aux* in Line 7 is chosen.

B.1 Key Encapsulation Mechanisms

A KEM scheme is a public key based scheme to generate and communicate a shared secret over an unsecure channel. The primary use case for KEMs is key establishment. KEMs are non-interactive, meaning only one party can contribute randomness. The length of the key as well as the ciphertext are dependent on the security parameter and can be expressed as $\Gamma(\lambda)$ for the length of the key and $\Theta(\lambda)$ for the length of the ciphertext. The receiving party cannot influence on the key generation process and has to trust the generating party to use adequate randomness. A key encapsulation scheme KEM consists of three algorithms $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$.

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 $\mathsf{pk} \leftarrow \mathsf{KGen}(1^\lambda)$, and outputs a key k as well as a ciphertext c. The ciphertext c encapsulates the key k. Decaps is a deterministic algorithm and takes a secret key sk and a ciphertext c as input, outputting either a key k or $\perp$ to indicate failure.

Definition 1 (Correctness of KEM schemes) *A key encapsulation scheme* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* δ-correct*, if for all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ *we have* $\Pr[\mathsf{Decaps}(\mathsf{sk}, c) = k : (c, k) \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})] \geq 1 - \delta$ *If* $\delta = 0$ *holds, the scheme is called* perfectly correct*.*

The security of a KEM scheme is defined over the 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* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* IND-ATK *secure for* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$*, 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* $\mathcal{A}$*.*

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* $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$*, 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* $\mathcal{A}$*.*

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

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

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

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

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

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

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

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

1 $b \leftarrow\!\$\ \{0,1\}$
2 $(\mathsf{pk}_0,\mathsf{sk}_0) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$
3 $(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$
4 $(c^*, k^*) \leftarrow\!\$\ \mathsf{Encaps}(\mathsf{pk}_b)$
5 $b' \leftarrow\!\$\ \mathcal{A}(\mathsf{pk}_0, \mathsf{pk}_1, c^*, k^*)$
6 **return** $[\![b = b']\!]$

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

1 $b \leftarrow\!\$\ \{0,1\}$
2 $(\mathsf{pk}_0,\mathsf{sk}_0) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$
3 $(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda)$
4 $(c^*, k^*) \leftarrow\!\$\ \mathsf{Encaps}(\mathsf{pk}_b)$
5 $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{O}_{Dec}}(\mathsf{pk}_0, \mathsf{pk}_1, c^*, k^*)$
6 **return** $[\![b = b']\!]$

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

7 **if** $c = c^*$
8 **return** $\perp$
9 $k = \mathsf{Decaps}(\mathsf{sk}_{id}, c)$
10 **return** k

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

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

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 as 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* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is correct, if* $\Pr\big[0 \leftarrow \mathsf{Vrfy}(\mathsf{pk}, \sigma, m) : (\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(1^\lambda), \sigma \leftarrow\!\$\ \mathsf{Sign}(\mathsf{sk}, m)\big]$ *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* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is*

EUF-CMA *secure, if* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{A}}(\lambda) = 1\right]$ *is negligible in the security parameter* λ *for any QPT adversaries* $\mathcal{A}$*.*

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

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

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

5 $\sigma \leftarrow\!\!\$\ \mathrm{Sign}(\mathsf{sk}, m)$
6 $\mathcal{L}_m \leftarrow \mathcal{L}_m \cup \{m\}$
7 **return** σ

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

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

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

1 $b \leftarrow\!\!\$\ \{0,1\}$
2 **if** $b = 1$
3 $\quad k \leftarrow\!\!\$\ \mathbb{K}, f \leftarrow F(k, \cdot)$
4 **else**
5 $\quad f \leftarrow\!\!\$\ \{\ f : \{0,1\}^{\iota(\lambda)} \to \{0,1\}^{\omega(\lambda)}\}$
6 **endif**
7 $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O}(f,\cdot)}$
8 **return** $[\![b' = b]\!]$

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

9 **return** $f(x)$

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