Post-Quantum Account Recovery for Passwordless Authentication

by

Spencer MacLaren Wilson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

WebAuthn is a passwordless authentication protocol which allows users to authenticate to online services using public-key cryptography. Users prove their identity based on possession of a private key, which is stored on a device such as a cell phone or a USB security token. This approach avoids many of the common security problems with password-based authentication. The reliance on possession as opposed to knowledge leads to a usability issue, however: a user who loses access to their authenticator device either loses access to their accounts or is required to fall back on a weaker authentication mechanism for recovery. Yubico has proposed a protocol which allows a user to link two tokens in such a way that one (the primary authenticator) can generate public keys on behalf of the other (the backup authenticator). This allows users to use WebAuthn with a single token, only using their backup token if necessary for account recovery. However, Yubico's protocol relies on the hardness of the discrete log problem for its security and hence is vulnerable to an attacker with a powerful enough quantum computer.

We present a WebAuthn backup protocol which can be instantiated with quantum-safe primitives. We also critique the security model used in previous analysis of Yubico's protocol, proposing a new framework which we use to evaluate the security of both the group-based and the post-quantum protocol. This leads us to uncover a weakness in Yubico's proposal which escaped detection in prior work but was revealed by our model. In our security analysis, we find that a number of novel security properties of cryptographic primitives underlying the protocols are required; we formalize these and prove that well-known algorithms satisfy the properties required for analysis of our post-quantum protocol. For the group-based protocol, we require a novel Diffie–Hellman-like assumption; we leave further evaluation of this property to future work.

Acknowledgements

A great many people in the C&O department supported me throughout the duration of my Master's degree. I would like to single out a few for especial thanks. Carol Seely-Morrison and Melissa Cambridge patiently dealt with many administrative queries and requests. Alfred Menezes and David Jao not only took the time to serve on my thesis committee but also provided me with invaluable academic and career guidance throughout my time at Waterloo. My official and unofficial grad school mentors, Evelyne Smith-Roberge and Valerie Gilchrist, showed me the ropes and introduced me to Camryn Steckel, Jonathan Gold, and Pravek Sharma, who were the best friends and classmates that anyone could hope for. Finally, none of this would have been possible without the insight, patience, and generosity of my supervisor, Douglas Stebila.

I am also indebted, as always, to my parents, Ian Wilson and Mary-Jo Rosenquist; to my brother, Duncan Wilson; and to my girlfriend, Valerie Bustos, for their love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Passwords have provided the dominant method of user authentication on the Internet for decades. Password-based authentication is easy for users to understand, does not require significant infrastructure, and is resilient to device loss. However, the apparent simplicity of password-based authentication masks serious security problems. Some of these arise from implementation pitfalls: properly storing and verifying passwords is a more difficult task than it seems at first glance. Moreover, password-based authentication is, without additional tools such as password managers, inherently non-user-friendly: strong, unique passwords are difficult to remember.

Passwordless authentication attempts to avoid problems with password-based authentication by simply getting rid of passwords. It has gained traction in recent years, driven by the efforts of the FIDO Alliance, a consortium of organizational stakeholders with a shared interest in secure user authentication solutions.

The FIDO Alliance has proposed a number of passwordless authentication solutions. Some of these are intended as a second factor to strengthen password-based authentication; others are intended as standalone authentication solutions. The most notable of the latter class is FIDO2. FIDO2 is the composition of two subprotocols, WebAuthn and CTAP. The former specifies interactions among servers (referred to as "relying parties"), clients, and authenticators (for example, USB security keys). The latter specifies communication between clients and authenticators. Both protocols have been subjected to security analysis and are beginning to see widespread deployment.

WebAuthn introduces new problems of its own, however. One of these is the issue of account recovery. An individual who loses the device they use to authenticate has no built-in method for recovering access to their account. This is a major obstacle preventing users from embracing FIDO2 [18]. The lack of a built-in backup solution can lead to weaker means of authentication—such as security questions or SMS verification—being used to enable account recovery, which undermines the protocol's security. Current advice to users is to purchase two tokens and register them both at each site: if you lose one, then you can still log in with the other [12]. Besides making account registration more complicated, this is lacking as a backup solution: since users must be in physical possession of both tokens whenever they create an account, they are likely to lose both at the same time.

Yubico, a manufacturer of security tokens, has proposed a solution whereby two tokens can be linked in such a way that one can generate recovery credentials for the other. The user can link their two tokens, store the backup in a safe place, and use the primary authenticator for day-to-day authentications. If the primary token is lost, the user retrieves the backup and uses it to recover access to accounts. The proposed protocol has undergone security analysis in [10]. The authors proposed an abstraction for its "cryptographic core", which they called Asynchronous Remote Key Generation, and developed a security model for it, proving security of Yubico's proposal under this model. However, Yubico's protocol relies heavily on elliptic curve cryptography; in particular, its security depends on the hardness of the Diffie–Hellman problem, which makes it vulnerable to an adversary with access to a quantum computer.

1.2 Contributions

In this work, we describe and analyze a quantum-safe recovery protocol for WebAuthn. We also highlight a number of weaknesses in the security analysis of Yubico's protocol. Notably, we describe a simple attack which escapes detection under the model in [10]. To address these weaknesses, we propose a new security model which more accurately captures the required security properties of a WebAuthn recovery solution. We analyze the security of both Yubico's protocol and our post-quantum protocol under this new model. Notably, our model makes no reference to specific details of WebAuthn. This means that our recovery protocol can be piggybacked on top of any authentication protocol with a similar challenge-response structure.

In order to prove security of our protocol, we require the underlying primitives to satisfy a number of non-standard security properties. For example, we require KEM decapsulation to be collision resistant or, in some proofs, pseudorandom. We provide formal definitions

of these security properties and prove that they are satisfied by notable post-quantum cryptographic algorithms. We additionally introduce a new Diffie–Hellman-like assumption required to establish the security of Yubico's protocol under our new model; however, we are unable to reduce this new assumption to any standard ones.

1.3 Structure

In Chapter 2, we describe the cryptographic primitives which are used to instantiate Yubico's and our recovery protocols. Notably, we describe non-standard security properties which we will require in subsequent security analysis. We describe the basics of our construction and show that it is secure under the model from [10] in Chapter 3. Chapter 4 contains the novel security model for account recovery. In Chapter 5, we give the full details of the group-based and post-quantum protocols, whose security we analyze in Chapter 6. In Chapter 7, we provide concrete examples of cryptographic primitives which meet the non-standard security properties required for analysis of the post-quantum protocol. We evaluate space requirements and efficiency in Chapter 8 and then provide closing comments in Chapter 9.

1.4 Related Work

Our contributions are adjacent to a number of recent efforts. Several papers have examined the provable security of WebAuthn, beginning with [3]. It was expanded upon by [14], which was the first to analyze the protocol's privacy properties; [14] also advances a protocol for revocation which is strikingly similar to Yubico's recovery proposal. Of particular relevance to our work is [5], which proposed a provably secure post-quantum version of WebAuthn. Previous work on WebAuthn account recovery includes [1], which proposes a solution based on group signatures, and [10], on which our work builds directly.

Our work makes prominent use of signature schemes with key blinding, which are discussed at length in [9] and [8]. We discuss a novel anonymity property of key encapsulation mechanisms; related analysis is done in [13], [22], and [19].

After our research was completed, we became aware of [7] and [11], two recent efforts to solve the WebAuthn backup problem. Our work was done independently from these and takes a different approach. Notably, neither of the constructions in [7] and [11] is provably secure under the strictest definitions from [10], while ours is.

Chapter 2

Preliminaries

In this chapter, we lay out the cryptographic primitives on which Yubico's recovery protocol and our post-quantum version rely. We recall familiar definitions, reintroduce more specialized concepts on which we build, and describe several novel security properties. We reserve proofs of these novel properties—when applicable—for Chapter 7.

2.1 Pseudorandom Function

A *pseudorandom function (PRF)* takes as input a key and a label and outputs a pseudorandom value. The outputs of a PRF with a fixed, uniformly sampled key should be indistinguishable from those of a random function. In the security analysis of our post-quantum recovery protocol, we will also require a non-standard (but reasonable) security property of a PRF: collision resistance, where the key is included in the input.

To analyze the security of Yubico's protocol under our new model, we introduce a new security property for PRFs, which we refer to as the "shifted group PRF" assumption, or sgPRF. Given a finite group $\mathbb{G}$ of order q and a finite set S, we say that a function $F\colon \mathbb{G} \to \mathbb{Z}_q \times S$ satisfies the sgPRF property if the function $F'\colon \mathbb{Z}_q \times \mathbb{G} \setminus \{1\} \to \mathbb{Z}_q \times S$ defined by $F'(s, E) = (F(E^s)_0 + s, F(E^s)_1)$ is a PRF, where s is regarded as the key and E as the label. Readers will notice similarities between the sgPRF problem and the PRF-ODH family of problems, introduced in [16] and thoroughly summarized in [6]; however, we have been unable to reduce sgPRF to any well-studied security assumption.

2.2 Message Authentication Code

A *message authentication code (MAC)* scheme $\Sigma = (\mathsf{MAC}, \mathsf{Verify})$ consists of a pair of algorithms: MAC, which takes as input a key and a message and (possibly probabilistically) outputs a tag, and Verify, which takes as input a key, a message, and a tag and outputs a single bit. The scheme is correct if Verify outputs 1 for all key-message-tag tuples obtained by calling MAC. A MAC scheme satisfies SUF-CMA security if it is infeasible for an adversary with access to a MAC oracle to compute a tag on a message for a uniformly sampled key without receiving the tag from the oracle.

2.3 Asynchronous Remote Key Generation

Previous security analysis of Yubico's WebAuthn recovery extension in [10] focused on the "cryptographic core" of the proposed protocol: a means by which a primary authenticator can generate public keys for which only the backup authenticator can produce signatures that verify. This mechanism was dubbed *asynchronous remote key generation (ARKG)*. An ARKG scheme $\mathsf{ARKG} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DerivePK}, \mathsf{DeriveSK}, \mathsf{Check})$ consists of five algorithms:

- $\mathsf{Setup}(1^\lambda)$: deterministically outputs the parameters pp for the scheme.
- $\mathsf{KeyGen}(\mathsf{pp})$: outputs a seed key pair $(\mathsf{sk}, \mathsf{pk})$. The public key is shared with the primary authenticator, while the private key is held by the backup authenticator.
- $\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux})$: probabilistically outputs a public key pk' and a corresponding credential cred bound to the input aux.
- $\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$: deterministically recovers the secret key corresponding to the credential cred, returning $\perp$ if the credential is invalid.
- $\mathsf{Check}(\mathsf{pp}, \mathsf{sk}', \mathsf{pk}')$: ouputs a bit indicating whether or not the provided sk' and pk' form a valid derived keypair.

The scheme is correct if for all inputs aux and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\Pr\left[\mathsf{Check}(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}') = 1 \;:\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{cred}) \leftarrow\!\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux}) \\ \mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})\end{array}\right] = 1.$$

In Yubico's protocol, the backup authenticator generates a seed key pair and shares the public key with the primary authenticator. The primary authenticator uploads derived public keys and the corresponding recovery credentials to WebAuthn servers. To recover an account at a server, the backup authenticator receives a recovery credential from the server, derives the associated secret key, and proves its identity by signing a challenge with this derived key.

The security of an ARKG scheme as defined in [10] has two components: public-key unlinkability and private-key security. The former property requires that derived keypairs should not be linkable to a seed public key. For a scheme to satisfy the latter, it should be infeasible to create a valid credential and derived keypair for a given seed keypair without knowledge of the seed private key. These properties are desireable in the context of WebAuthn, where private key proof-of-posession is used for authentication but user credentials should not be correlatable.

2.3.1 Public-Key Unlinkability

An adversary for public-key unlinkability is challenged to distinguish between a fixed distribution $\mathcal{D}$ (in [10], the distribution of seed keypairs) and the distribution of derived keypairs. The adversary is provided with the seed public key and an oracle which outputs either derived keypairs or samples from the distribution $\mathcal{D}$. This security experiment is defined formally in Figure 2.1. The adversary's advantage is defined to be

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

2.3.2 Private-Key Security

Private-key security has four strength levels, categorized as either "honest" or "malicious" and either "strong" or "weak". The strongest of these is malicious strong key security, denoted msKS; it implies the other three security levels. An msKS-adversary is provided with the seed public key, a DerivePK oracle, and a DeriveSK oracle which can only be queried with credentials previously output by the DerivePK oracle. The adversary wins if it can produce a valid public key-private key-credential tuple. This security experiment is defined formally in Figure 2.2. The "weak" security variants remove the DeriveSK oracle, and the "honest" variants require that the adversary output a credential obtained from its

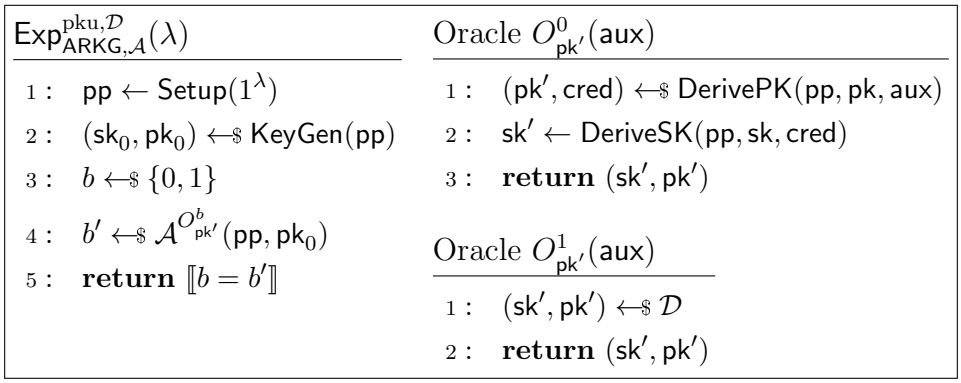

Figure 2.1: The PK-unlinkability experiment for ARKG

DerivePK oracle. The adversary's advantage is defined to be

$$\mathsf{Adv}^{\mathrm{msKS}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) = \Pr\big[\mathsf{Exp}^{\mathrm{msKS}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) = 1\big].$$

2.4 Key Encapsulation Mechanism

A *key encapsulation mechanism (KEM)* $\Pi = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms: KeyGen, which samples a keypair; Encaps, which takes as input a public key and (probabilistically) outputs a ciphertext and a key; and Decaps, which takes as input a private key and a ciphertext and outputs a key. Generally, KEMs fall into two classes, depending on how they handle invalid decapsulation queries. KEMs of the first type perform *explicit rejection*, return a reserved value $\perp$ when given invalid input. KEMs of the second type perform *implicit rejection*, instead returning a pseudorandom value (typically the output of a PRF keyed with a portion of the secret key). For the rest of this document, we will only consider KEMs that perform implicit rejection.

A KEM is correct if

$$\Pr\big[\mathsf{Decaps}(\mathsf{sk}, c) = k \ : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda), (c, k) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk})\big] = 1.$$

A KEM satisfies IND-CCA security if it is infeasible for an adversary given access to a public key, a challenge ciphertext-key secret pair, and a decapsulation oracle to determine whether the key is the decapsulation of the ciphertext or a random value. A related notion is *strong pseudorandomness under chosen ciphertext attack*, or SPR-CCA security, where

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

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{PKList} \leftarrow \emptyset$
$\mathsf{SKList} \leftarrow \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{pp})$
$(\mathsf{sk}^*, \mathsf{pk}^*, \mathsf{cred}^*) \leftarrow\!\$\ \mathcal{A}^{O_{\mathsf{pk}'}, O_{\mathsf{sk}'}}(\mathsf{pp}, \mathsf{pk})$
$\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred}^*)$
return $\mathsf{Check}(\mathsf{sk}^*, \mathsf{pk}^*)$
$\quad \wedge\ \mathsf{Check}(\mathsf{sk}', \mathsf{pk}^*)$
$\quad \wedge\ [\![\mathsf{cred}^* \notin \mathsf{SKList}]\!]$

Oracle $O_{\mathsf{pk}'}(\mathsf{aux})$

$(\mathsf{pk}', \mathsf{cred}) \leftarrow\!\$\ \mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux})$
$\mathsf{PKList} \leftarrow \mathsf{PKList} \cup \{(\mathsf{pk}', \mathsf{cred})\}$
return $(\mathsf{pk}', \mathsf{cred})$

Oracle $O_{\mathsf{sk}'}(\mathsf{cred})$

if $(\cdot, \mathsf{cred}) \notin \mathsf{PKList}$ **then return** $\perp$
$\mathsf{SKList} \leftarrow \mathsf{SKList} \cup \{\mathsf{cred}\}$
return $\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}, \mathsf{cred})$

Figure 2.2: The msKS experiment for ARKG

$\mathsf{Exp}^{\text{CR-CCA}}_{\Pi,\mathcal{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$
$(c_0, c_1) \leftarrow\!\$\ \mathcal{A}^{\mathsf{Decaps}}(\mathsf{pk})$
return $[\![\mathsf{Decaps}(\mathsf{sk}, c_0) = \mathsf{Decaps}(\mathsf{sk}, c_1)]\!]$

Figure 2.3: The CR-CCA experiment for a KEM Π

both ciphertext and key are sampled independently of the public key, as defined in [22]. For analysis of our post-quantum protocol, we require two novel security properties of KEMs. In Chapter 7, we show that these properties are satisfied by CRYSTALS-Kyber.

2.4.1 Collision Resistance

The first property we require is per-key collision resistance: it should be difficult for a CCA adversary to produce two ciphertexts which decapsulate to the same value. The corresponding security experiment is described in Figure 2.3. We define the adversary's advantage to be

$$\mathsf{Adv}^{\text{CR-CCA}}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\text{CR-CCA}}_{\Pi,\mathcal{A}}(\lambda) = 1\right].$$

$\mathsf{Exp}^{\text{KEM-UL}}_{\Pi,\mathcal{A}}(\lambda)$	Oracle $O_1^E()$	Oracle $O_1^D(c)$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$	1 : $c \leftarrow\!\!\$\ \mathcal{C}$	1 : **if** $(c, k') \in \mathcal{L}$ **then return** k'
2 : $\mathcal{L} \leftarrow \emptyset$	2 : $k \leftarrow\!\!\$\ \mathcal{K}$	2 : $k \leftarrow\!\!\$\ \mathcal{K}$
3 : $b \leftarrow\!\!\$\ \{0,1\}$	3 : $\mathcal{L} \leftarrow \mathcal{L} \cup (c, k)$	3 : $\mathcal{L} \leftarrow \mathcal{L} \cup (c, k)$
4 : $b' \leftarrow\!\!\$\ \mathcal{A}^{O_b^E, O_b^D}()$	4 : **return** (c, k)	4 : **return** (c, k)
5 : **return** $[\![b' = b]\!]$		

Figure 2.4: The KEM-UL experiment for a KEM Π. The oracles O_0^E and O_0^D are defined by $\mathsf{Encaps}(\mathsf{pk})$ and $\mathsf{Decaps}(\mathsf{sk}, \cdot)$, respectively.

2.4.2 KEM Unlinkability

The second property is less straightforward: we require that an adversary without knowledge of the public key should learn absolutely no information about the public key. Concretely, it should be unable to distinguish between encapsulations and decapsulations and random sampling from the ciphertext and key spaces. We refer to this property as KEM unlinkability. The corresponding security experiment is described in Figure 2.4. We define the adversary's advantage to be

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

2.5 Key-Blinding Signature Scheme

As defined in [9], a *key-blinding signature scheme* Δ consists of four algorithms defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: generates an *identity* or *seed* key pair $(\mathsf{pk}, \mathsf{sk})$ from which blinded keys will be derived.
- $\mathsf{BlindPK}(\mathsf{pk}, \tau)$: deterministically computes a blinded public key pk'.
- $\mathsf{Sign}(\mathsf{sk}, \tau, m)$: computes (possibly probabilistically) a signature σ on m for τ.
- $\mathsf{Verify}(\mathsf{pk}', m, \sigma)$: outputs 1 if σ is a valid signature on m for a blinded public key pk'.

A key-blinding signature scheme is correct if signatures under τ always verify under the public key blinded with τ. Two security properties are required. The first, *existential unforgeability under chosen message and epoch attack*, or EUF-CMEA, stipulates that an adversary with a key blinding oracle and a signing oracle should not be able to produce a forgery for any (m, τ) not queried to the signing oracle.[1] "Epoch" refers to the blinding factor τ. The second, *unlinkability under chosen message and epoch attack*, or UL-CMEA, stipulates that an adversary with a key blinding oracle and a signing oracle for a fixed keypair should not be able to distinguish between a blinding with the fixed keypair or a fresh keypair. We refer to [9] for formalizations of these properties.

2.6 Key Blinding Scheme

The security analysis of Yubico's protocol in [10] decouples signing from key blinding. In particular, it allows the adversary to obtain the secret keys corresponding to blinded public keys—for which no interface is provided in [9]—and makes no reference to their involvement in a signature scheme. Desiring a compatible abstraction, we define a *key blinding scheme* $\Delta = (\mathsf{KeyGen}, \mathsf{BlindPK}, \mathsf{BlindSK}, \mathsf{Check})$ to be a collection of four algorithms defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: generates an *identity* or *seed* key pair $(\mathsf{pk}, \mathsf{sk})$ from which blinded keys will be derived.
- $\mathsf{BlindPK}(\mathsf{pk}, \tau)$: deterministically computes a blinded public key pk_τ.
- $\mathsf{BlindSK}(\mathsf{sk}, \tau)$: deterministically computes a blinded secret key sk_τ.
- $\mathsf{Check}(\mathsf{pk}', \mathsf{sk}')$: checks if $(\mathsf{pk}', \mathsf{sk}')$ is a valid blinded keypair.

A key blinding scheme is correct if for all blinding factors τ

$$\Pr\left[\mathsf{Check}(\mathsf{BlindPK}(\mathsf{pk}, \tau), \mathsf{BlindSK}(\mathsf{sk}, \tau)) \ : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)\right] = 1.$$

The group-based key blinding scheme utilized in [10] satisfies a number of additional properties which facilitate the proofs of ARKG security:

[1] The definition in [9] provides the adversary with a single oracle which, given (m, τ), outputs both $\mathsf{BlindPK}(\mathsf{pk}, \tau)$ and $\mathsf{Sign}(\mathsf{sk}, \tau, m)$; however, it is easy to see that there is no difference between this single-oracle formulation and one in which the adversary receives two separate oracles.

1. It is difficult to recover the seed private key given the seed public key. This is the natural minimum security notion for a public-key cryptosystem.
2. Every blinded public key passes Check with exactly one secret key, and vice versa. This allows us to refer to "the" corresponding secret key for a public key.
3. It is easy to recover the seed private key given an arbitrary blinding factor and the corresponding secret key. This typically arises via some sort of homomorphic relationship between public and private keys, which is present in a number of key blinding schemes.
4. The distribution of blinded keypairs for any seed keypair is indistinguishable from the distribution of seed keypairs. This property naturally leads to a strong notion of unlinkability.

All four of the blinded signature schemes in [9] naturally give rise to key blinding schemes, although only one satisfies property 4. Two of them—blLegRoast and blPicnic—include the seed secret key as part of the blinded secret key, and the distribution of blDilithium-QROM blinded secret keys is dependent on the seed secret key. However, blCSI-FiSh does satisfy all of these properties, as we will show in Chapter 7. This makes it a suitable drop-in replacement for the group-based key blinding scheme.

We now give formal definitions of these four properties for subsequent use in proofs.

2.6.1 Key-Recovery Security

We refer to property 1 as *key-recovery security*, which we denote by KR. We define it with a game in which the adversary is challenged to compute the seed private key given only the seed public key, depicted in Figure 2.5. We define the advantage of a KR adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\mathrm{KR}}_{\Delta,\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathrm{KR}}_{\Delta,\mathcal{A}}(\lambda) = 1\right].$$

The KR-security of the key blinding scheme used in Yubico's proposal is equivalent to the discrete log assumption on the underlying group.

2.6.2 Unique Blinding

If a key blinding scheme satisfies property 2, we say that it provides *unique blinding*. Formally, we say that Δ provides unique blinding if for every seed keypair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$$

$\mathsf{Exp}^{\mathrm{KR}}_{\Delta,\mathcal{A}}(\lambda)$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \Delta.\mathsf{KeyGen}(1^\lambda)$
2 : $\mathsf{sk}' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$
3 : **return** $[\![\mathsf{sk} = \mathsf{sk}']\!]$

Figure 2.5: The KR experiment for a key blinding scheme Δ

$\mathsf{KeyGen}(1^\lambda)$ and every blinding factor τ,

$$\mathsf{Check}(\mathsf{BlindPK}(\mathsf{pk}, \tau), \mathsf{sk}') = 1 \iff \mathsf{sk}' = \mathsf{BlindSK}(\mathsf{sk}, \tau)$$

and

$$\mathsf{Check}(\mathsf{pk}', \mathsf{BlindSK}(\mathsf{sk}, \tau)) = 1 \iff \mathsf{pk}' = \mathsf{BlindPK}(\mathsf{pk}, \tau).$$

The key blinding scheme used in [10] provides unique blinding because a derived private key is the discrete log of the associated public key.

2.6.3 Private-Key Unblinding

If a key blinding scheme satifies property 3, we say that it supports *private-key unblinding*. Formally, we require the existence of an efficiently computable function $\mathsf{UnblindSK}$ such that for all blinding factors τ

$$\Pr[\mathsf{UnblindSK}(\tau, \mathsf{sk}_\tau) = \mathsf{sk} \ : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}, \mathsf{sk}_\tau \leftarrow \mathsf{BlindSK}(\mathsf{sk}, \tau)] = 1.$$

At face value, this appears to be a strange (and perhaps even undesirable) security property. Nonetheless, we will see that private-key unblinding in tandem with unique blinding allows a straightforward security reduction for our generic ARKG construction.

2.6.4 Strong Independent Blinding

For property 4, we say that a key blinding scheme provides *strong independent blinding* if the distribution of randomly blinded keypairs for a seed keypair is indistinguishable from the distribution of freshly generated keypairs, even to an adversary with access to the seed

$\mathsf{Exp}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{A}}(\lambda)$	Oracle $O_0()$	Oracle $O_1()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \Delta.\mathsf{KeyGen}(1^\lambda)$	$\tau \leftarrow\!\!\$\ \{0,1\}^\lambda$	**return** $\Delta.\mathsf{KeyGen}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$	$\mathsf{pk}_\tau \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}, \tau)$	
$b' \leftarrow\!\!\$\ \mathcal{A}^{O_b}(\mathsf{pk})$	$\mathsf{sk}_\tau \leftarrow \Delta.\mathsf{BlindSK}(\mathsf{sk}, \tau)$	
return $[\![b = b']\!]$	**return** $(\mathsf{pk}_\tau, \mathsf{sk}_\tau)$	

Figure 2.6: The strong independent blinding experiment for a key blinding scheme Δ

public key. This is formally captured by the security experiment given in Figure 2.6. We define the advantage of a S-Ind-Blind adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{A}}(\lambda) = \left| \Pr\left[\mathsf{Exp}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

Chapter 3

Quantum-Safe ARKG

We now provide a quantum-safe construction of the ARKG primitive defined in [10] and prove its security. Our ARKG scheme follows a similar structure as Yubico's, replacing *ad hoc* group-based mechanisms with generic components which can be instantiated from post-quantum security assumptions.

Group arithmetic is used in Yubico's scheme first to arrive at a shared secret, using Diffie–Hellman key exchange, from which a key-blinding factor can be derived; subsequently, it is used to perform the blinding itself. However, it is not necessary for shared secret computation and key blinding to be coupled together using the same key material. This allows Diffie–Hellman key exchange to be replaced by KEM operations, at the expense of introducing a second, independent keypair for the KEM. This leaves only the key blinding scheme as a group-based component; the final step is to replace it with a quantum-safe key blinding scheme. Yubico's scheme also relies on a handful of other generic primitives, which only require slight modifications for the quantum-safe version.

To prove security under the model defined in [10], we introduce a number of assumptions on the key blinding scheme, as described in Section 2.3.

3.1 Description

We present our post-quantum construction (Figure 3.2) alongside the group-based construction (Figure 3.1) from [10], to highlight both common structure and points of difference. Yubico's construction is instantiated with

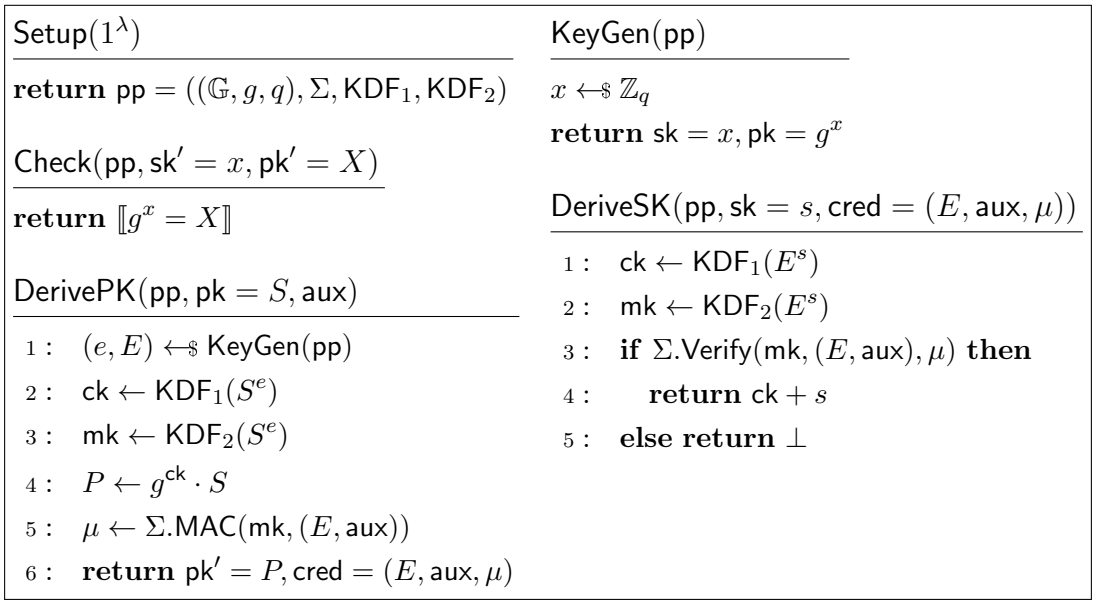

Figure 3.1: Yubico's group-based ARKG construction from [10]

- a cyclic group $\mathbb{G}$ of order q with generator g,
- a MAC scheme Σ, and
- two key derivation functions, $\mathsf{KDF}_1 \colon \mathbb{G} \to \mathbb{Z}_q$ and $\mathsf{KDF}_2 \colon \mathbb{G} \to \{0,1\}^*$.

Our construction replaces the group with a key blinding scheme Δ and a KEM Π and the key derivation functions with a single PRF. Additionally, we do away with the MAC scheme, instead opting to incorporate the information which would be tagged into PRF labels; because of this, we can replace the two KDFs with a single PRF. We will assume that the PRF output binary strings of length at least λ, but we leave this implicit in the protocol description.

The quantum-safe version follows a pattern similar to the group-based construction. Public keys and credentials are generated by computing a pseudorandom shared secret (either the Diffie–Hellman value or the encapsulated key) using the seed public key (either S or $(\mathsf{pk}_\Delta, \mathsf{pk}_\Pi)$) and deriving a blinding factor (either ck or τ). Information necessary to recover the shared secret (the ephemeral Diffie–Hellman public key E or the KEM ciphertext c) is included in the associated credential. Secret keys are derived from credentials by using this information, along with the seed secret key (s or $(\mathsf{sk}_\Delta, \mathsf{sk}_\Pi)$), to recover the shared secret, from which the blinding factor can be recovered.

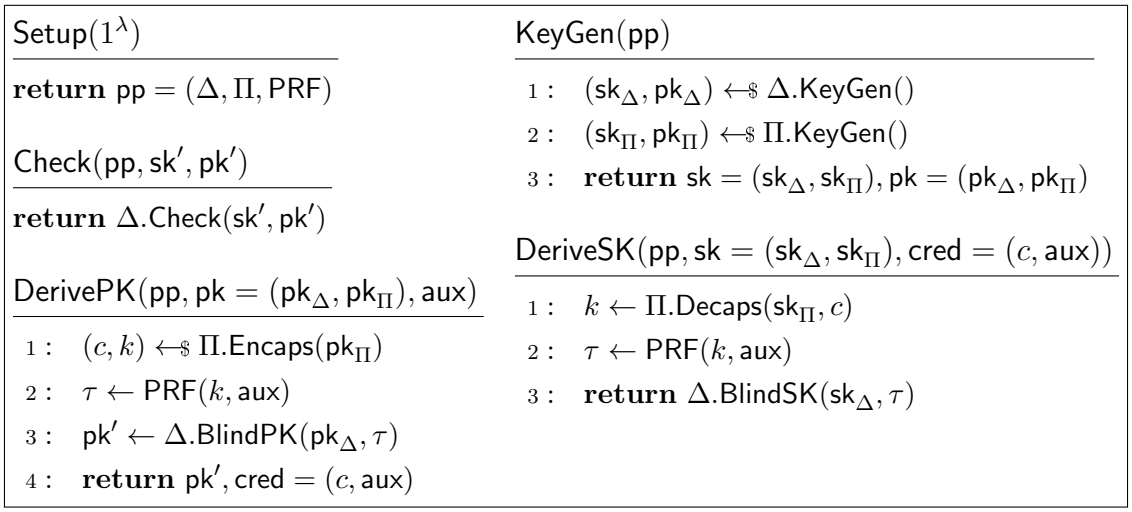

Figure 3.2: Our post-quantum ARKG construction

The correctness of our construction is immediate from the correctness of its components.

3.1.1 Comparison to Prior Work

Our construction is very similar to the one proposed in [7]. In that construction, fresh keypairs are generated using the pseudorandom value τ (see lines 2 and 2) in place of true randomness. This approach has the advantage of not requiring a key-blinding signature scheme, which allows more mainstream algorithms to be used. However, our approach ensures that derived private keys are never computed (and indeed are infeasible to compute) on the primary token, one of the stated goals of Yubico's proposal [15]. Furthermore, the construction in [7] is not proven to be secure under the original definitions from [10], instead working with a modified security model.

The approach taken by [11] differs from ours more significantly, as it is primarily based on a different primitive: a split KEM. As with [7], it is not proven to be securely instantiable under the strongest definitions from [10]. Hence, our work is the first (and to date, only) construction to have a provably secure instantiation using the strongest model from [10]. However, we will argue in Chapter 4 that this model does not accurately capture the security requirements; hence, this is not necessarily a strong argument for preferring our approach.

$\mathsf{Exp}^{\mathrm{pku}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda)$	Oracle $O^0_{\mathsf{pk}'}(\mathsf{aux})$	Oracle $O^1_{\mathsf{pk}'}(\mathsf{aux})$
1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	1 : $(c, k) \leftarrow\!\!\$\ \Pi.\mathsf{Encaps}(\mathsf{pk}_\Pi)$	**return** $\Delta.\mathsf{KeyGen}()$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{pp})$	2 : $\tau \leftarrow\!\!\$\ \mathsf{PRF}(k, \mathsf{aux})$	
3 : $b \leftarrow\!\!\$\ \{0,1\}$	3 : $\mathsf{pk}' \leftarrow \Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \mathsf{ck})$	
4 : $b' \leftarrow\!\!\$\ \mathcal{A}^{O^b_{\mathsf{pk}'}}(\mathsf{pp}, \mathsf{pk})$	4 : $\mathsf{sk}' \leftarrow \Delta.\mathsf{BlindSK}(\mathsf{sk}_\Delta, \mathsf{ck})$	
5 : **return** $[\![b = b']\!]$	5 : **return** $(\mathsf{sk}', \mathsf{pk}')$	

Figure 3.3: The PK-unlinkability experiment for pqARKG

3.2 Security Analysis

3.2.1 Public-Key Unlinkability

We now prove that the post-quantum ARKG construction described in Figure 3.2 satifies PK-unlinkability with the base distribution $\mathcal{D}$ equal to the distribution of identity keypairs in the blinded signature scheme used, that is, the distribution of outputs of $\Delta.\mathsf{KeyGen}$.

As a reference, in Figure 3.3 we repeat the PK-unlinkability security experiment described in Figure 2.1 with our scheme, denoted by pqARKG, and the distribution $\mathcal{D}$ inlined. Since DeriveSK is only called immediately after DerivePK and in the same oracle, we omit the line which performs decapsulation of the just-encapsulated value.

Theorem 3.1. *Let* pqARKG *denote the ARKG construction in Figure 3.2, and let* $\mathcal{D}$ *be the distribution of outputs of* $\Delta.\mathsf{KeyGen}$*. For any efficient adversary* $\mathcal{A}$ *making at most* n *queries to oracle* $O^b_{\mathsf{pk}'}$*, there exist efficient algorithms* $\mathcal{B}_0$*,* $\mathcal{B}_1$*, and* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{pku},\mathcal{D}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_0}(\lambda) + n \cdot \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{B}_2}(\lambda).$$

Proof. We proceed by a sequence of games, beginning with the original security experiment as Game 0. We define the adversary's advantage in Game 0 in the same fashion as its advantage in the original experiment; its advantages in subsequent games are similar. We will follow this same method in subsequent proofs.

Game 0. This game is precisely the security experiment $\mathsf{Exp}^{\mathrm{pku}}_{\mathsf{pqARKG},\mathcal{A}}$, as described in Figure 3.3. Hence

$$\mathsf{Adv}^{\mathrm{pku},\mathcal{D}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

Hybrid games. In order to reduce to the strong independent blinding property of Δ, we must replace the derived τ-values with truly random values. We first replace the encapsulated keys k with truly random values (relying on the IND-CCA security of Π), then the derived keys τ (relying on the pseudorandomness of PRF). We accomplish this via a sequence of hybrid games $\mathcal{H}_i$, beginning with $\mathcal{H}_0 = \mathcal{G}_0$. For $i \geq 1$, we define

- $\mathcal{H}_{2i-1}$ to be identical to $\mathcal{H}_{2i-2}$ except that in the ith iteration of $O^0_{\mathsf{pk}'}$ the call to $\Pi.\mathsf{Encaps}$ on line 1 is replaced by taking a random sample from the key space of Π.
- $\mathcal{H}_{2i}$ to be identical to $\mathcal{H}_{2i-1}$ except that in the ith iteration of $O^0_{\mathsf{pk}'}$ the output τ of PRF on line 2 is replaced by a truly random value.

We now bound the increase in advantage between games $\mathcal{H}_{2i-2}$ and $\mathcal{H}_{2i-1}$ and the increase in advantage between games $\mathcal{H}_{2i-1}$ and $\mathcal{H}_{2i}$. For the former, consider an IND-CCA adversary $\mathcal{B}^i$ for Π which challenges $\mathcal{A}$ in $\mathcal{H}_{2i-2}$, replacing the generated Π public key with the one received from its IND-CCA challenger and inserting its challenge ciphertext-key pair at line 1 in the ith iteration of $O^0_{\mathsf{pk}'}$, returning the value returned by $\mathcal{A}$. The adversary $\mathcal{B}^i$ faithfully simulates game $\mathcal{H}_{2i-2}$ if its challenge key is a valid decapsulation and faithfully simulates game $\mathcal{H}_{2i-1}$ if its challenge key is a random sample. Then

$$\mathsf{Adv}^{\mathcal{H}_{2i-2}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}^i}(\lambda) + \mathsf{Adv}^{\mathcal{H}_{2i-1}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

For the latter, an adversary $\mathcal{C}^i$ for PRF can similarly challenge $\mathcal{A}$ in $\mathcal{H}_{2i-1}$, using its oracle at line 2 in the ith iteration of $O^0_{\mathsf{pk}'}$ and returning 1 if and only if $\mathcal{A}$ wins the game. This gives the bound

$$\mathsf{Adv}^{\mathcal{H}_{2i-1}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{C}^i}(\lambda) + \mathsf{Adv}^{\mathcal{H}_{2i}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

Combining the previous two inequalities, we obtain

$$\mathsf{Adv}^{\mathcal{H}_0}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_0}(\lambda) + n \cdot \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{H}_{2n}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda),$$

where $\mathcal{B}_0$ is the most successful of the adversaries $\mathcal{B}^i$ and $\mathcal{B}_1$ is the most successful of the adversaries $\mathcal{C}^i$ for $1 \leq i \leq n$.

Game 1. This game is identical to Game 0 except that all PRF outputs τ on line 2 of the $O^0_{\mathsf{pk}'}$ oracle are replaced by truly random values. Since $\mathcal{A}$ makes at most n queries to $O^b_{\mathsf{pk}'}$, Game 1 is identical to game $\mathcal{H}_{2n}$, so

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{H}_{2n}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

We now reduce Game 1 to the S-Ind-Blind experiment for Δ, described in Figure 2.6. Let $\mathcal{B}_2$ be a S-Ind-Blind adversary for Δ, challenged to guess a bit b. From its challenger, it receives a Δ-public key and oracle which either (when $b = 0$) samples a random blinding factor τ and outputs the associated blinded keypair or (when $b = 1$) outputs a fresh Δ keypair. The S-Ind-Blind adversary $\mathcal{B}_2$ acts as the challenger for Game 1 with $\mathcal{A}$, using its oracle to answer $\mathcal{A}$'s queries to $O^b_{\mathsf{pk}'}$. Since the blinding factor τ in $O^0_{\mathsf{pk}'}$ is now sampled randomly, $\mathcal{B}_2$ faithfully simulates Game 1 for $\mathcal{A}$, providing the oracle $O^0_{\mathsf{pk}'}$ when $b = 0$ and $O^1_{\mathsf{pk}'}$ when $b = 1$. It follows that $\mathcal{B}_2$ wins precisely when $\mathcal{A}$ does, implying that

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{B}_2}(\lambda).$$

The desired security statement is given by combining these bounds. □

3.2.2 Private-Key Security

We will show that the post-quantum ARKG construction satisfies msKS-security, from which it follows that it also satisfies the other three security levels.

As before, in Figure 3.4 we repeat the msKS security experiment from Figure 2.2 with pqARKG inlined.

Theorem 3.2. *Let* pqARKG *denote the ARKG construction in Figure 3.2, and suppose that the key blinding scheme Δ provides unique blinding and supports private-key unblinding. For any efficient adversary $\mathcal{A}$ making at most n queries to oracle $O_{\mathsf{pk}'}$, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ such that*

$$\mathsf{Adv}^{\text{msKS}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_0}(\lambda) + n \cdot \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{KR}}_{\Delta,\mathcal{B}_3}(\lambda).$$

Proof. We proceed by a sequence of games.

Game 0. This game is precisely the security experiment $\mathsf{Exp}^{\text{msKS}}_{\mathsf{pqARKG},\mathcal{A}}$ as described in Figure 3.4. Hence

$$\mathsf{Adv}^{\text{msKS}}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0 except for the following changes:

- The encapsulated key k on line 1 of $O_{\mathsf{pk}'}$ is replaced by a truly random value.
- The output τ of PRF on line 2 of $O_{\mathsf{pk}'}$ is replaced by a truly random value. The challenger stores this value alongside (c, aux).

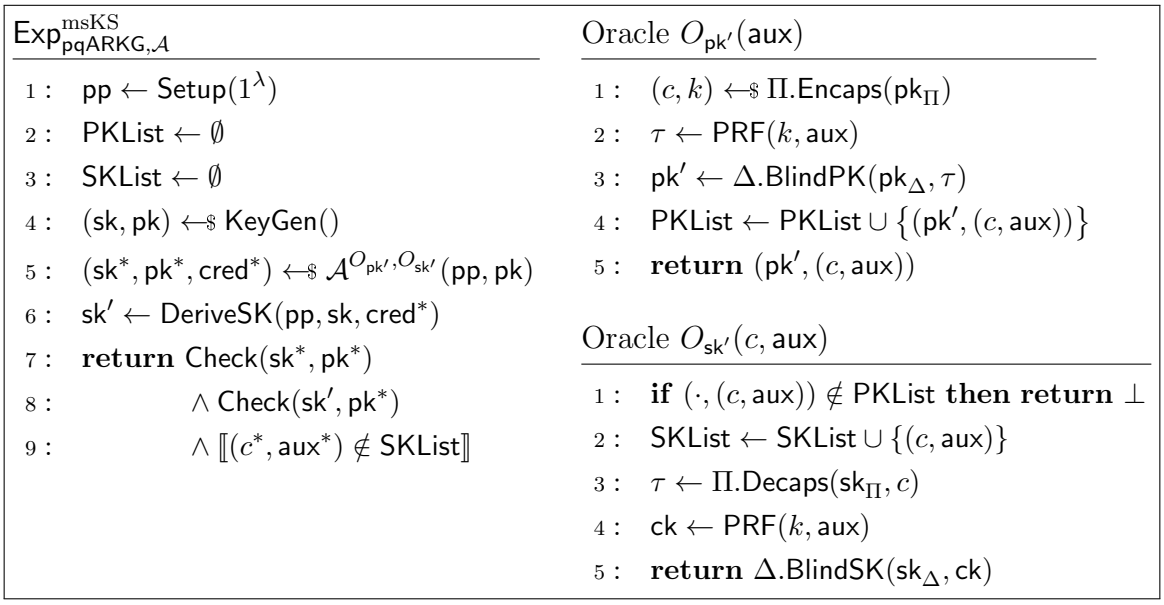

Figure 3.4: The msKS experiment for pqARKG

- The call to PRF on line 4 of $O_{\mathsf{sk}'}$ is replaced by looking up the value of τ that was sampled and stored alongside the given credential (c, aux).

A similar hybrid argument to the one given in the proof of Theorem 3.1 shows that there exist efficient algorithms $\mathcal{B}_0$ and $\mathcal{B}_1$ such that

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{B}_0}(\lambda) + n \cdot \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

Game 2. This game is identical to Game 1 except for the following changes:

- The call to $\Delta.\mathsf{BlindPK}$ on line 3 of $O_{\mathsf{pk}'}$ is replaced by a call to $\Delta.\mathsf{KeyGen}$. The challenger stores the generated secret key as sk' alongside (c, aux) and τ.
- The call to $\Delta.\mathsf{BlindSK}$ on line 5 of $O_{\mathsf{sk}'}$ is replaced by looking up the value of sk' that was generated and stored alongside (c, aux) and τ.

A S-Ind-Blind adversary for Δ which uses its key generation oracle in place of $\Delta.\mathsf{BlindPK}$ and $\Delta.\mathsf{BlindSK}$ faithfully simulates Game 1 when its challenge bit is 0 and Game 2 when

its challenge bit is 1. Hence, the increase in advantage introduced by these changes is bounded by the S-Ind-Blind security of Δ:

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{S-Ind-Blind}}_{\Delta,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqARKG},\mathcal{A}}(\lambda).$$

We now define a KR adversary $\mathcal{B}_3$ for Δ which calls $\mathcal{A}$ as an oracle, reducing Game 2 to the KR experiment for Δ. From the KR challenger, $\mathcal{B}_3$ receives a freshly generated Δ-public key. The KR-adversary $\mathcal{B}_3$ acts as the challenger for Game 2, substituting this public key for the Δ-public key provided to $\mathcal{A}$. Both $O_{\mathsf{pk}'}$ and $O_{\mathsf{sk}'}$ are independent of the Δ-secret key, so $\mathcal{B}_3$ simulates Game 2 faithfully. When $\mathcal{A}$ returns $(\mathsf{sk}^*, \mathsf{pk}^*, \mathsf{cred}^*)$, the algorithm $\mathcal{B}_3$ computes the blinding factor τ^* corresponding to cred^*. Since Δ supports private-key unblinding, $\mathcal{B}_3$ then computes $\mathsf{sk} \leftarrow \mathsf{UnblindSK}(\tau^*, \mathsf{sk}^*)$ and returns sk to the KR challenger.

We claim that $\mathcal{B}_3$ wins the KR experiment whenever $\mathcal{A}$ wins Game 2. If $\mathcal{A}$ submits a winning tuple, then the value sk' that would be computed on line 6 must be the blinding of the unknown seed private key with τ^*. By unique blinding, $\mathsf{sk}^* = \mathsf{sk}'$, so $\mathsf{UnblindSK}$ does indeed recover the unknown seed private key. It follows that

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{KR}}_{\Delta,\mathcal{B}_3}(\lambda).$$

The desired security statement is given by combining these bounds. □

Chapter 4

Credential-Based Recovery

While the ARKG abstraction models the so-called "cryptographic core" of Yubico's proposed standard, we argue that it does not capture the practical security properties required of the protocol or a post-quantum replacement. Some of this is due to focusing on the core and ignoring real-world details like server and token policy—for instance, the unlinkability definition considers only the adversary's ability to distinguish two distributions of keypairs, ignoring non-mathematical sources of information. At times, however, even the cryptographic details seem out of step with reality. In particular, the ARKG adversary is given at once too much power and not enough.

Consider the msKS security model, presented in Figure 2.2. In this experiment, the adversary can obtain derived secret keys corresponding to public keys that have already been generated. In reality, these keys never leave the backup token; an adversary who can retrieve them has powers that would render WebAuthn insecure. Of course, granting the adversary more power than is realistic doesn't inherently constitute a weakness in analysis. However, this is the only way in which it can interact with the derived secret keys. Notably, the adversary cannot obtain signatures made with the derived keys—which is a more natural interface available in the real world.

The lack of a signature oracle could be explained by the composability result obtained in [10]: an ARKG scheme which satisfies PK-unlinkability with some distribution (see Figure 2.1) can be securely composed with protocols using keypairs distributed according to this distribution. According to this result, using derived keypairs to produce signatures in WebAuthn should not weaken the protocol's security. However, the process by which keypairs are derived reveals more information to a potential attacker than is given to the PK-unlinkability adversary. The adversary's powers are limited to observing freshly

derived keypairs. In particular, it is not given the ability to view the recovery credentials which enable the backup token to derive these secret keys, nor may it interact either the primary token or the backup token in any other way.[1] These values are computed and sent to servers alongside derived public keys before the derived secret keys are used to provide signatures for WebAuthn.

These points are all theoretical, but they have a practical impact. The key-revealing power given to the adversary in PK-unlinkability and msKS security precludes the use of cryptographic primitives which do not guarantee security under such compromises, such as all but one of the blinded signatures from [9], even though such primitives may be secure for practical use in the recovery extension. At the same time, the PK-unlinkability adversary is too underpowered to model a real-world attacker. This can be illustrated concretely: in Chapter 5 we highlight an attack on the unlinkability of recovery credentials which the ARKG security model fails to capture.

For these reasons, we argue that security analysis of the WebAuthn recovery extension under a new model more aligned with real-world use is required before the protocol's security can be assured. We attempt to provide such a security model here.

4.1 Protocol Model

We propose an abstraction which captures both the credential generation scheme and the associated protocol, which we call *credential-based recovery*. To simplify security analysis we make the simplifying assumption that each user has exactly one primary token and exactly one backup token, and that these tokens are not shared among users. In contrast, Yubico's protocol allows users not only to have multiple backup tokens for a single primary token but also to link a single backup token with multiple primary tokens. Our model does not capture the full functionality of Yubico's protocol; however, it does provide guarantees for users who use only a single primary token and a single backup token.

Our model considers client, human user, and token to be a single entity, which we simply call a user. This amalgamation assumes that the client and the token have a secure connection. In the FIDO2 passwordless authentication protocol, of which WebAuthn is one

[1] There is some room for doubt about the PK-unlinkability security definition in [10]. The definition of the $O^b_{\mathsf{pk}'}$ oracle indicates that it only returns a keypair and not a credential; however, the proof of PK-unlinkability seems to include some steps which attempt to prove the unlinkability of credentials. The oracle's pseudocode is never provided in full. We have chosen to take the interpretation that only a keypair is returned, as this is compatible with the way the PK-unlinkability security definition is used later, and to do otherwise would be to speculate about undefined security properties.

component, this connection is provided by the Client To Authenticator Protocol, or CTAP. Yubico's proposed recovery standard additionally specifies extensions to CTAP, which we do not consider as part of this model. We additionally assume that registration of recovery credentials occurs over an authenticated channel. This is a reasonable assumption: the recovery protocol is intended to back up a secure authentication protocol, which should allow for such a transaction to take place. We make no such assumptions during the recovery phase.

A *credential-based recovery* protocol CBR defines an interaction between a user U and a server S. The protocol has two components: a interface of stateless algorithms for performing operations on credentials and keys, and a set of protocol actions which consume and manipulate state and which make calls to the interface. We refer to an execution of the protocol at a party (either a user or a server) as a session. Each party maintains some long-term state as well as short-term state associated with individual sessions.

The stateless interface consists of the following algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: outputs a seed keypair $(\mathsf{pk}, \mathsf{bk})$. The primary key pk will be used by the primary authenticator to generate recovery credentials for the backup authenticator, which retains the backup key bk.
- $\mathsf{CredGen}(\mathsf{pk}, \mathsf{aux})$: inputs a primary key pk and some auxiliary information aux, outputs a recovery credential rc bound to aux and its identifier rcid.
- $\mathsf{Response}(\mathsf{bk}, \mathsf{rcid}, \mathsf{aux}, \mathsf{ch}, \mathsf{nc})$: inputs a secret key bk, a recovery credential identifier rcid, auxiliary information aux, a new credential nc, and a challenge ch; outputs a response rsp.
- $\mathsf{Verify}(\mathsf{rc}, \mathsf{aux}, \mathsf{nc}, \mathsf{ch}, \mathsf{rsp})$: inputs a recovery credential rc, auxiliary information aux, a new credential nc, a challenge ch, and a response rsp; outputs a decision bit b.

We refer to keys as "backup" and "primary" instead of "private" and "public" because Yubico's specification indicates that the primary ("public") key should not be exposed to the server. Indeed, if this key is made public the unlinkability of the scheme is severely weakened.

The protocol actions are the following:

- $\mathsf{Register}$: The user generates a recovery credential which the server stores.
- $\mathsf{UserBegin}$: The user requests to initiate the recovery process by providing their username to the server.

- ServerBegin: The server initiates the recovery process by returning recovery information to the user for identification purposes. The server may also provide data to be used to establish a new permanent credential.
- UserComplete: The user proves their identity using the provided recovery information. The user may also provide data to be used to establish a new permanent credential.
- ServerComplete: The server verifies the user's response. If successful, the parties have established a new permanent credential, restoring the user's access to their account on the server.

Each non-Register action consumes the party's long-term state st and the session state π, both of which it may manipulate, and some input data. The format of data is protocol-specific. The Register action occurs over an authenticated channel; hence, we model it as a joint action which consumes some input data and the long-term state of each party.

Session state π consists of the following variables, which are common to all protocol actions:

- selfid: the identifier used by the session owner,
- peerid: the identifier used by the session peer,
- role: the role played by the session owner (one of user or server),
- status: the session status (one of recover, accept, or reject),
- sid: the session identifier, and
- st: additional state for the session, to be used as defined by specific protocols.

We make no assumptions about variables defined in long-term state, but we do assume that all state is initialized to $\perp$ or $\emptyset$ as appropriate.

We next formalize two security goals for credential-based recovery. The first, which we call *recovery authentication*, stipulates that a server only completes the recovery process for a registered user if the user also completes the recovery process, and the two parties agree on each other's identities, which credential was used for recovery, and information to be used for future authentication. Additionally, this user should be the only user which satisfies these requirements: that is, two different users should not both be authenticated to the same server with the same session transcript. Similarly, the same user should

not repeat the same transcript at the same server in two different sessions. The second security goal, which we call *unlinkability*, stipulates broadly that an adversary should be unable to differentiate users based on information obtained via observing (and interfering with) protocol execution.

4.1.1 Recovery Authentication

We model the recovery authentication security of a credential-based recovery protocol CBR with the experiment $\mathsf{Exp}^{\mathrm{rec}}_{\mathsf{CBR}}$, formally defined in Figure 4.1. The adversary can make calls to any of the following oracles, which we denote collectively by $\mathcal{O}$:

- NewUser: inputs a party U. Initializes U as a user.
- NewServer: inputs a party S and a string serverID. Initializes S as a server with the given ID, if no other server has the same ID.
- ORegister: inputs a user U, a server S, and a username uid. Attempts to register U at S with the given username, returning the output of Register. If the registration is successful, (U, S, uid) is added to a list of registered accounts.
- Action: inputs a party P, an index i, and data data. Proceeds with the next action of the recovery process for session π^i_P with data as input, returning the output of whichever action is called.

Similarly to security analyses of FIDO2 and WebAuthn in [3], [14], and [5], we rely on the notion of matching sessions to define security. Intuitively, two sessions match if they represent two different sides of the same protocol interaction. Formally, we say that sessions π_1 and π_2 match if

- one of $\pi_1.\mathsf{role}$ and $\pi_2.\mathsf{role}$ is user and the other is server,
- $\pi_1.\mathsf{status} = \mathsf{accept} = \pi_2.\mathsf{status}$,
- $\pi_1.\mathsf{selfid} = \pi_2.\mathsf{peerid}$ and $\pi_2.\mathsf{selfid} = \pi_2.\mathsf{peerid}$, and
- $\pi_1.\mathsf{sid} = \pi_2.\mathsf{sid}$.

The adversary wins the game if either of the following conditions hold:

$\mathsf{Exp}^{\mathrm{rec}}_{\mathsf{CBR},\mathcal{A}}(\lambda)$

$\mathcal{L}_{\mathsf{user}}, \mathcal{L}_{\mathsf{server}}, \mathcal{L}_{\mathsf{Register}} \leftarrow \emptyset$
$\mathcal{A}^{\mathcal{O}}(1^\lambda)$
if $\exists (P_1, i_1) \neq (P_2, i_2)$:
  $\pi^{i_1}_{P_1}.\mathsf{sid} = \pi^{i_2}_{P_2}.\mathsf{sid} \neq \bot$
  $\wedge\, \mathsf{Match}(\pi^{i_1}_{P_1}, \pi^{i_2}_{P_2}) \neq 1$
 then return 1
if $\exists (S, i) : \pi^i_S.\mathsf{role} = \mathsf{server}$
  $\wedge\, \pi^i_S.\mathsf{status} = \mathsf{accept}$
  $\wedge\, \nexists (U, j) : \mathsf{Match}(\pi^i_S, \pi^j_U) = 1$
    $\wedge\, (U, S, \pi^i_S.\mathsf{peerid}) \in \mathcal{L}_{\mathsf{Register}}$
 then return 1
return 0

$\mathsf{Match}(\pi_1, \pi_2)$

return $[\![\pi_1.\mathsf{role} \neq \bot]\!]$
  $\wedge\, [\![\pi_2.\mathsf{role} \neq \bot]\!]$
  $\wedge\, [\![\pi_1.\mathsf{role} \neq \pi_2.\mathsf{role}]\!]$
  $\wedge\, [\![\pi_1.\mathsf{status} = \mathsf{accept}]\!]$
  $\wedge\, [\![\pi_2.\mathsf{status} = \mathsf{accept}]\!]$
  $\wedge\, [\![\pi_1.\mathsf{selfid} = \pi_2.\mathsf{peerid}]\!]$
  $\wedge\, [\![\pi_1.\mathsf{peerid} = \pi_2.\mathsf{selfid}]\!]$
  $\wedge\, [\![\pi_1.\mathsf{sid} = \pi_2.\mathsf{sid}]\!]$

$\mathsf{NewUser}(U)$

if $U \in \mathcal{L}_{\mathsf{user}} \cup \mathcal{L}_{\mathsf{server}}$ **then return**
$\mathcal{L}_{\mathsf{user}} \leftarrow \mathcal{L}_{\mathsf{user}} \cup \{U\}$
return

$\mathsf{NewServer}(S, \mathsf{serverID})$

if $S \in \mathcal{L}_{\mathsf{user}} \cup \mathcal{L}_{\mathsf{server}}$ **then return**
if $\exists S' \in \mathcal{L}_{\mathsf{server}} : \mathsf{st}_{S'}.\mathsf{id} = \mathsf{serverID}$ **then**
  return
$\mathsf{st}_S.\mathsf{id} \leftarrow \mathsf{serverID}$
return

$\mathsf{ORegister}(U, S, \mathsf{uid})$

if $U \notin \mathcal{L}_{\mathsf{user}}$ **then return** $\bot$
if $S \notin \mathcal{L}_{\mathsf{server}}$ **then return** $\bot$
$\mathsf{ret} \leftarrow\!\!\$\ \mathsf{Register}(\mathsf{uid}, \mathsf{st}_U, \mathsf{st}_S, \lambda)$
if $\mathsf{ret} \neq \bot$ **then**
  $\mathcal{L}_{\mathsf{Register}} \leftarrow \mathcal{L}_{\mathsf{Register}} \cup \{(U, S, \mathsf{uid})\}$
return ret

$\mathsf{Action}(P, i, \mathsf{data})$

$\mathsf{ret} \leftarrow \bot$
if $\pi^i_P = \bot$ **then**
  if $P \in \mathcal{L}_{\mathsf{user}}$ **then**
    $\mathsf{ret} \leftarrow \mathsf{UserBegin}(\pi^i_P, \mathsf{data}, \mathsf{st}_P)$
  elseif $P \in \mathcal{L}_{\mathsf{server}}$ **then**
    $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{ServerBegin}(\pi^i_P, \mathsf{data}, \mathsf{st}_P)$
elseif $\pi^i_P.\mathsf{status} = \mathsf{recover}$ **then**
  if $\pi^i_P.\mathsf{role} = \mathsf{user}$ **then**
    $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{UserComplete}(\pi^i_P, \mathsf{data}, \mathsf{st}_P)$
  elseif $\pi^i_P.\mathsf{role} = \mathsf{server}$ **then**
    $\mathsf{ret} \leftarrow \mathsf{ServerComplete}(\pi^i_P, \mathsf{data}, \mathsf{st}_P)$
return ret

Figure 4.1: The rec experiment for CBR

- Two distinct non-matching sessions have the same session identifier. In principle, this means that protocol admits some sort of replay attack.
- A server session π accepts without a matching user session. Moreover, the user session must belong to a user registered at S under the username $\pi.\mathsf{peerid}$. In principle, this means that it is possible for someone (registered or otherwise) to authenticate to a server by some means other than following the protocol.

Note that two distinct sessions with the same role and the same session identifier will not match; therefore, the adversary wins if two distinct user sessions (or two distinct server sessions) have the same session identifier. We define the advantage of an adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\mathrm{rec}}_{\mathsf{CBR},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathrm{rec}}_{\mathsf{CBR},\mathcal{A}} = 1\right].$$

Unfortunately, Yubico's proposed standard does not satisfy this security notion, as we will show in Chapter 5. It does, however, satisfy a slightly modified version, in which a user may only have a single account at each server. We refer to this notion as *single-account recovery authentication*, or 1rec. The security experiment is identical except that the adversary fails if any user is registered twice at any server.

4.1.2 Unlinkability

In the unlinkability security experiment $\mathsf{Exp}^{\mathrm{UL}}_{\mathsf{CBR}}$, depicted in Figure 4.2, the adversary is challenged to distinguish between two users of their choice—that is, to determine whether the two users have been switched or not. The adversary is initially provided with the same set $\mathcal{O}$ of oracles as for recovery authentication. Eventually, the adversary selects target users U_0 and U_1. The challenger samples a random bit b and chooses new identifiers U_0^* and U_1^*. The game continues with the adversary receiving the set of oracles $\mathcal{O}_b$, consisting of $\mathsf{NewUser}$, $\mathsf{NewServer}$, $\mathsf{ORegister}_b$, and Action_b, with the latter two defined as follows:

- The oracle $\mathsf{ORegister}_b$ is identical to $\mathsf{ORegister}$, except that on a query with $U = U_0^*$, it will use st_{U_b} when calling $\mathsf{Register}$; on a query with $U = U_{1^*}$, it will use $\mathsf{st}_{U_{1-b}}$.
- The oracle Action_b is identical to Action, except that on a query with $P = U_0^*$, it will use st_{U_b} instead of st_{U_1} when calling one of the protocol actions; on a query with $P = U_1^*$ it will use $\mathsf{st}_{U_{1-b}}$.

The adversary is challenged to guess the value of b.

Of course, the adversary could trivially win the game by observing the behaviour of U_0^* given a recovery credential generated by U_0. Since the adversary has all the information available to a server, it must be able to determine whether or not U_0^* and U_0 are the same user based on the response to this query; if not, the recovery protocol would be useless. We prevent this trivial winning strategy by setting a bit fail if the adversary makes such a query and returning a random bit in the event that fail is set. Since the experiment is designed to be opaque with regards to input data for protocol actions, we detect this condition via session variables, setting fail if at any point one of U_0^* and U_1^* has a session corresponding to a registration for U_0 or U_1, and vice versa. Our experiment creates new identifiers for the challenge users U_0 and U_1 for a similar reason: it prevents the adversary from winning by attempting to begin session i with U_0^* for a value i which corresponds to a session for U_0 but not for U_1.

We define the advantage of an adversary $\mathcal{A}$ to be

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

Yubico's proposed standard fails to meet this notion of unlinkability due to the same weakness that prevents it from satisfying our notion of recovery authentication. It does, however, provide unlinkability under the assumption that a user may have only a single account at each server. We refer to this weaker notion as *inter-domain unlinkability*. We model inter-domain unlinkability with an experiment which is identical to that for unlinkability except for two changes:

- The fail flag is set in $\mathsf{ORegister}$ and $\mathsf{ORegister}_b$ if the given user has already registered at the given server.
- The fail flag is set in $\mathsf{ORegister}$ if the adversary attempts to register U_0^* or U_1^* (respectively U_0 or U_1) at a server where U_0 or U_1 (respectively U_0^* or U_1^*) already has an account.

This experiment is depicted in Figure 4.3. We define the advantage of an adversary $\mathcal{A}$ to be

$$\mathsf{Adv}^{\text{I-UL}}_{\mathsf{CBR},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\text{I-UL}}_{\mathsf{CBR},\mathcal{A}} = 1\right] - \frac{1}{2}\right|.$$

Note that the I-UL security experiment is identical to the UL security experiment except for imposing additional failure conditions on the adversary. Thus, inter-domain unlinkability is implied by unlinkability.

$\mathsf{Exp}^{\mathrm{UL}}_{\mathsf{CBR},\mathcal{A}}(\lambda)$

$\mathcal{L}_{\mathsf{user}}, \mathcal{L}_{\mathsf{server}}, \mathcal{L}_{\mathsf{Register}} \leftarrow \emptyset$
$\mathsf{fail} \leftarrow 0$
$(U_0, U_1) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}(1^\lambda)$
$U_0^* \leftarrow U : U \notin \mathcal{L}_{\mathsf{server}} \cup \mathcal{L}_{\mathsf{user}}$
$\mathcal{L}_{\mathsf{user}} \leftarrow \mathcal{L}_{\mathsf{user}} \cup \{U_0^*\}$
$U_1^* \leftarrow U : U \notin \mathcal{L}_{\mathsf{server}} \cup \mathcal{L}_{\mathsf{user}}$
$\mathcal{L}_{\mathsf{user}} \leftarrow \mathcal{L}_{\mathsf{user}} \cup \{U_1^*\}$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_b}(U_0^*, U_1^*)$
if fail **then** $b' \leftarrow\!\!\$\ \{0,1\}$
return $[\![b = b']\!]$

$\mathsf{CheckFail}(U, \pi)$

if $U \in \{U_0, U_1\}$ **then**
  if $\exists (d, S) : \mathsf{st}_S.\mathsf{id} = \pi.\mathsf{peerid}$
    $\wedge\ (U, S, \pi.\mathsf{selfid}) \notin \mathcal{L}_{\mathsf{Register}}$
    $\wedge\ (U_d^*, S, \pi.\mathsf{selfid}) \in \mathcal{L}_{\mathsf{Register}}$
   then return 1
  else return 0
elseif $U \in \{U_0^*, U_1^*\}$ **then**
  if $\exists (d, S) : \mathsf{st}_S.\mathsf{id} = \pi.\mathsf{peerid}$
    $\wedge\ (U, S, \pi.\mathsf{selfid}) \notin \mathcal{L}_{\mathsf{Register}}$
    $\wedge\ (U_d, S, \pi.\mathsf{selfid}) \in \mathcal{L}_{\mathsf{Register}}$
   then return 1
  else return 0
else return 0

$\mathsf{ORegister}_b(U, S, \mathsf{uid})$

if $U \notin \mathcal{L}_{\mathsf{user}}$ **then return** $\perp$
if $S \notin \mathcal{L}_{\mathsf{server}}$ **then return** $\perp$
if $U = U_0^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_b}$
elseif $U = U_1^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_{1-b}}$
else $\mathsf{st} \leftarrow \mathsf{st}_U$
$\mathsf{ret} \leftarrow\!\!\$\ \mathsf{Register}(\mathsf{uid}, \mathsf{st}, \mathsf{st}_S, \lambda)$
if $\mathsf{ret} \neq \perp$ **then**
  $\mathcal{L}_{\mathsf{Register}} \leftarrow \mathcal{L}_{\mathsf{Register}} \cup \{(U, S, \mathsf{uid})\}$
return ret

$\mathsf{Action}_b(P, i, \mathsf{data})$

$\mathsf{ret} \leftarrow \perp$
if $P = U_0^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_b}$
elseif $P = U_1^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_{1-b}}$
else $\mathsf{st} \leftarrow \mathsf{st}_P$
if $\pi_P^i = \perp$ **then**
  if $P \in \mathcal{L}_{\mathsf{user}}$ **then**
    $\mathsf{ret} \leftarrow \mathsf{UserBegin}(\pi_P^i, \mathsf{data}, \mathsf{st})$
    $\mathsf{fail} \leftarrow \mathsf{fail} \vee \mathsf{CheckFail}(P, \pi_P^i)$
  elseif $P \in \mathcal{L}_{\mathsf{server}}$ **then**
    $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{ServerBegin}(\pi_P^i, \mathsf{data}, \mathsf{st})$
elseif $\pi_P^i.\mathsf{status} = \mathsf{recover}$ **then**
  if $\pi_P^i.\mathsf{role} = \mathsf{user}$ **then**
    $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{UserComplete}(\pi_P^i, \mathsf{data}, \mathsf{st})$
  elseif $\pi_P^i.\mathsf{role} = \mathsf{server}$ **then**
    $\mathsf{ret} \leftarrow \mathsf{ServerComplete}(\pi_P^i, \mathsf{data}, \mathsf{st})$
return ret

Figure 4.2: The UL experiment for CBR. For a description of the oracles NewUser, NewServer, ORegister, and Action, see Figure 4.1.

$\mathsf{Exp}^{\text{I-UL}}_{\mathsf{CBR},\mathcal{A}}(\lambda)$

1 : $\mathcal{L}_{\mathsf{user}}, \mathcal{L}_{\mathsf{server}}, \mathcal{L}_{\mathsf{Register}} \leftarrow \emptyset$
2 : $\mathsf{fail} \leftarrow 0$
3 : $(U_0, U_1) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}(1^\lambda)$
4 : $U_0^* \leftarrow U : U \notin \mathcal{L}_{\mathsf{server}} \cup \mathcal{L}_{\mathsf{user}}$
5 : $\mathcal{L}_{\mathsf{user}} \leftarrow \mathcal{L}_{\mathsf{user}} \cup \{U_0^*\}$
6 : $U_1^* \leftarrow U : U \notin \mathcal{L}_{\mathsf{server}} \cup \mathcal{L}_{\mathsf{user}}$
7 : $\mathcal{L}_{\mathsf{user}} \leftarrow \mathcal{L}_{\mathsf{user}} \cup \{U_1^*\}$
8 : $b \leftarrow\!\!\$\ \{0,1\}$
9 : $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_b}(U_0^*, U_1^*)$
10 : **if** fail **then** $b' \leftarrow\!\!\$\ \{0,1\}$
11 : **return** $[\![b = b']\!]$

$\mathsf{ORegister}_b(U, S, \mathsf{uid})$

1 : **if** $U \notin \mathcal{L}_{\mathsf{user}}$ **then return** $\perp$
2 : **if** $S \notin \mathcal{L}_{\mathsf{server}}$ **then return** $\perp$
3 : **if** $(U, S, \cdot) \in \mathcal{L}_{\mathsf{Register}}$ **then** $\mathsf{fail} \leftarrow 1$
4 : **if** $U = U_0^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_b}$
5 : **elseif** $U = U_1^*$ **then** $\mathsf{st} \leftarrow \mathsf{st}_{U_{1-b}}$
6 : **else** $\mathsf{st} \leftarrow \mathsf{st}_U$
7 : $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{Register}(\mathsf{uid}, \mathsf{st}, \mathsf{st}_S, \lambda)$
8 : **if** $\mathsf{ret} \neq \perp$ **then**
9 : $\quad \mathcal{L}_{\mathsf{Register}} \leftarrow \mathcal{L}_{\mathsf{Register}} \cup \{(U, S, \mathsf{uid})\}$
10 : **if** $\exists b_1, b_2 : (U_{b_0}, S, \cdot), (U_{b_1}^*, S, \cdot) \in \mathcal{L}_{\mathsf{Register}}$
11 : **then** $\mathsf{fail} \leftarrow 1$
12 : **return** ret

$\mathsf{ORegister}(U, S, \mathsf{uid})$

1 : **if** $U \notin \mathcal{L}_{\mathsf{user}}$ **then return** $\perp$
2 : **if** $S \notin \mathcal{L}_{\mathsf{server}}$ **then return** $\perp$
3 : **if** $(U, S, \cdot) \in \mathcal{L}_{\mathsf{Register}}$ **then** $\mathsf{fail} \leftarrow 1$
4 : $\mathsf{ret} \leftarrow\!\!\$\ \mathsf{Register}(\mathsf{uid}, \mathsf{st}_U, \mathsf{st}_S, \lambda)$
5 : **if** $\mathsf{ret} \neq \perp$ **then**
6 : $\quad \mathcal{L}_{\mathsf{Register}} \leftarrow \mathcal{L}_{\mathsf{Register}} \cup \{(U, S, \mathsf{uid})\}$
7 : **return** ret

Figure 4.3: The I-UL experiment for CBR. For a description of the oracles NewUser, NewServer, Action, and Action_b, see Figures 4.1 and 4.2.

Chapter 5

Group-Based and Post-Quantum CBR Protocols

Now that we have laid out the desired security properties for a credential-based recovery protocol, we turn our hand to modelling Yubico's proposal in this framework, describing our novel quantum-safe credential-based recovery protocol, and analyzing the security of both. We will denote the group-based CBR protocol by gCBR and the post-quantum protocol by pqCBR.

5.1 Protocol Descriptions

In Figure 5.1, we describe the credential-based recovery scheme defined by Yubico's standard. Readers will note the similarities with the ARKG scheme described in Figure 3.1. Notably, this formulation includes a hash function H, a digital signature scheme Λ, and some bookkeeping around the identity point of $\mathbb{G}$, which are absent in the ARKG description; these elements are present in Yubico's proposal but omitted from the analysis in [10]. Our post-quantum credential-based recovery scheme is described in Figure 5.2. The only notable difference from the post-quantum ARKG formulation is that the key blinding scheme Δ is treated as a key-blinding signature scheme.

Both the group-based and the post-quantum protocols follow a similar structure with regards to using the interface provided by the credential-based recovery scheme. We describe a generic construction in Figure 5.3, with only one point of difference between the two: the post-quantum version uses both server identifier and username in the derivation of

recovery credentials, whereas the group-based version only uses the server identifier. This lack of binding recovery credentials to usernames leads to a minor weakness in Yubico's protocol.

The recovery protocol is intended to be piggy-backed on top of another authentication protocol: its purpose is to allow the user and server to establish a new credential in the event of device loss. Hence, we have attempted to make as few assumptions about the underlying protocol as possible, in order for our results to be more widely applicable. The following assumptions about the underlying protocol are required:

- Users must retain knowledge of their usernames in the event of device loss.
- A server must not repeat a challenge for the same user.
- A user must not generate the same new credential twice for the same account.

Up to the latter two restrictions, we allow challenges and new credentials to be provided by the adversary. This captures protocols which might generate random challenges (which repeat with low probability) as well as those which use a counter. Although we describe the string ch as a challenge, it could also include any data that the server wishes to provide to be agreed upon with the user, for instance, to be used in credential generation. Similarly, although we describe nc as a credential, it could also include any data that the user wishes to agree upon with the server. Concretely, WebAuthn realizes these assumptions by having human-memorable usernames, randomly generated challenges of sufficient length such that a collision is highly unlikely, and a high-entropy public key included in user-generated credentials.

5.2 Weaknesses in the Group-Based Protocol

As previously mentioned, Yubico's protocol does not satisfy our security requirements when users are allowed to have multiple accounts at the same server. This is because recovery credentials are generated independently of account usernames; they are bound only to server identifiers. Hence, a user with multiple accounts cannot determine which account a recovery credential belongs to. Practically, this allows a server to determine if any two registered accounts belong to the same user. During the recovery process for one account, the server provides a recovery credential identifier for another account. The user will respond with a valid signature for the recovery public key associated with the other account if and only if the two accounts both belong to the user. This breaks the unlinkability of

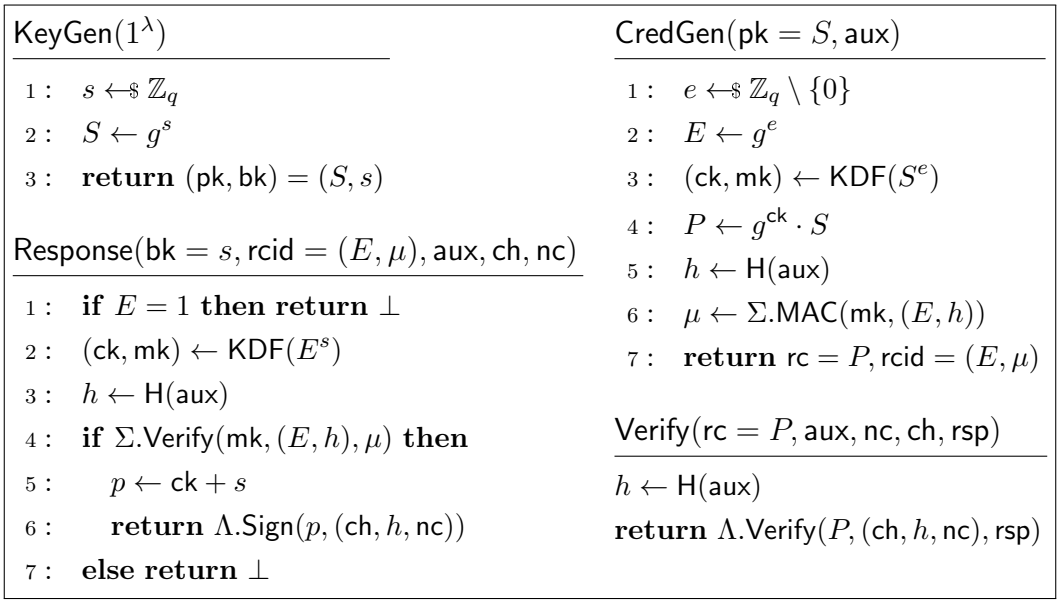

Figure 5.1: Yubico's credential-based recovery scheme

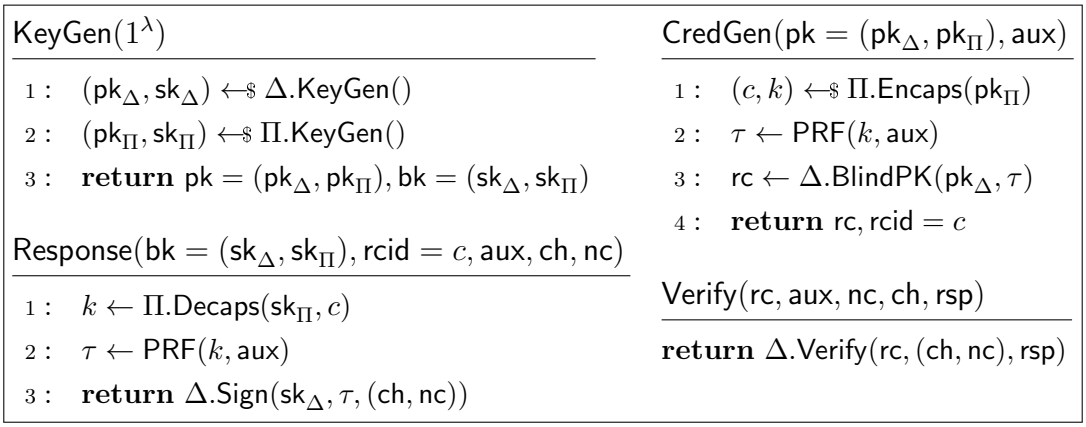

Figure 5.2: Our post-quantum credential-based recovery scheme

Figure 5.3: A generic credential-based recovery protocol

Yubico's scheme. An identical approach leads to an attack on recovery authentication in which a user can "recover" one of their accounts when actually attempting to recover the other. Fortunately, both of these weaknesses are relatively minor, and the protocol can be proven secure in our model (albeit under a non-standard assumption) as long as users do not have multiple accounts at the same server.

We have conveyed these concerns to the authors of Yubico's proposal. They brought our attention to the fact that WebAuthn already admits a similar attack on unlinkability, so the recovery protocol does not introduce a new attack vector. This weakness in WebAuthn is discussed further in [14]. Regardless, the group-based recovery protocol should be handled with care if used in conjunction with a non-WebAuthn authentication protocol which does not admit the same attack.

Chapter 6

Security Analysis

We now prove recovery authentication security and unlinkability of both gCBR and pqCBR.

6.1 Recovery Authentication

Theorem 6.1. *Let* gCBR *be the credential-based recovery protocol described in Figures 5.3 and 5.1. For any efficient adversary $\mathcal{A}$ making at most n_{Register} queries to* ORegister*, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ such that*

$$\mathsf{Adv}^{\mathrm{1rec}}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathcal{B}_0}(\lambda) + \binom{n_{\mathsf{Register}}}{2} \cdot \frac{1}{q-1} + n_{\mathsf{user}} \cdot \mathsf{Adv}^{\mathrm{sg\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + n_{\mathsf{Register}} \cdot \left(\mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\Lambda,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathrm{SUF\text{-}CMA}}_{\Sigma,\mathcal{B}_3}(\lambda)\right).$$

Proof. We begin by arguing that the first winning condition, on line 3 of the recovery authentication experiment in Figure 4.1, is never met. Suppose that $(P_1, i_1) \neq (P_2, i_2)$ and $\pi^{i_1}_{P_1}.\mathsf{sid} = \pi^{i_2}_{P_2}.\mathsf{sid} \neq \perp$. Since the session identifier is only set to a non-$\perp$ value when a session accepts, and a session in the accept state cannot be modified, $\pi^{i_1}_{P_1}.\mathsf{status} = \mathsf{accept} = \pi^{i_2}_{P_2}$. Session identifiers are also only set on sessions with set roles, so both $\pi^{i_1}_{P_1}$ and $\pi^{i_2}_{P_2}$ have set roles.

If $\pi^{i_1}_{P_1}.\mathsf{role} = \mathsf{server} = \pi^{i_2}_{P_2}$, then $\pi^{i_1}_{P_1}.\mathsf{selfid} = \pi^{i_2}_{P_2}.\mathsf{selfid}$, $\pi^{i_1}_{P_1}.\mathsf{peerid} = \pi^{i_2}_{P_2}.\mathsf{peerid}$ by session identifier equality. Since server identifiers are unique, $P_1 = P_2$. But servers never issue the same ch value for the same username in two different sessions, so the ch portions of the two session identifiers must differ. Since the session identifiers are the same, it follows that at

least one of $\pi_{P_1}^{i_1}$ and $\pi_{P_2}^{i_2}$ is a user session. Since users never issue the same nc value for the same server identifier in two different sessions, a similar argument shows that at least one of $\pi_{P_1}^{i_1}$ and $\pi_{P_2}^{i_2}$ is a server session. Therefore $\pi_{P_1}^{i_1}.\mathsf{role} \neq \pi_{P_2}^{i_2}.\mathsf{role}$. It follows by equality of session identifiers that $\pi_{P_1}^{i_1}.\mathsf{selfid} = \pi_{P_2}^{i_2}.\mathsf{peerid}$ and $\pi_{P_1}^{i_1}.\mathsf{peerid} = \pi_{P_2}^{i_2}.\mathsf{selfid}$. This shows that $\pi_{P_1}^{i_1}$ and $\pi_{P_2}^{i_2}$ do in fact match.

We now bound the adversary's advantage in triggering the second winning condition, proceeding by a sequence of games.

Game 0. This game is identical to the security experiment $\mathsf{Exp}^{\mathrm{1rec}}_{\mathsf{gCBR},\mathcal{A}}$. Therefore

$$\mathsf{Adv}^{\mathrm{1rec}}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0 except that the game aborts if the challenger observes a collision for H. With $\mathcal{B}_0$ defined to be the adversary which challenges $\mathcal{A}$ and returns this observed collision,

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 2. This game is identical to Game 1 except that the game aborts if two E-values generated by calls to $\mathsf{CredGen}$ collide. Since $\mathsf{CredGen}$ is only called by $\mathsf{ORegister}$, the number of E-values generated is at most n_{Register}. Since these values are sampled uniformly from a set of size $q-1$, the probability of a pair colliding is $1/(q-1)$. It follows that the probability of a collision is bounded above by

$$\binom{n_{\mathsf{Register}}}{2} \cdot \frac{1}{q-1},$$

whence

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq \binom{n_{\mathsf{Register}}}{2} \cdot \frac{1}{q-1} + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 3. This game is identical to Game 2 except for the following changes:

- The derived value mk in $\mathsf{Response}$ and $\mathsf{CredGen}$ is replaced by randomly sampling a key for Σ.
- The derived value p in $\mathsf{Response}$ is replaced by randomly sampling an element of $\mathbb{Z}_q$.
- The computation of P in $\mathsf{CredGen}$ is replaced by randomly sampling $p \leftarrow\!\!\$\ \mathbb{Z}_q$ and setting $P \leftarrow g^p$.

This sampling is done consistently, so that the same mk- and p-values are used if the input E^s to KDF is repeated.

This game can be viewed as the terminal game in a sequence of hybrid games $\mathcal{H}_i$, where $\mathcal{H}_0 = \mathcal{G}_2$ and $\mathcal{H}_i$ incorporates these changes for users 1 through i. At each step, the increase in advantage is bounded by the sg-prf security of KDF; hence,

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{sg-prf}}_{\mathsf{KDF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 4. This game is identical to Game 3 except that the game aborts if the adversary wins in any way except by a mismatched MAC tag μ. Formally, the game aborts if there exists (S, i) such that $\pi^i_S.\mathsf{role} = \mathsf{server}$, $\pi^i_S.\mathsf{status} = \mathsf{accept}$ and for which there is no (U, j) such that π^j_U matches with π^i_S except for the μ-portion of the session identifier. We show that this abort condition is only triggered if $\mathcal{A}$ forges a signature.

Let $\mathcal{B}_2$ be an EUF-CMA adversary for Λ. From its challenger, $\mathcal{B}_2$ receives a public key P^* and a signing oracle for the corresponding private key. Then $\mathcal{B}_2$ acts as the challenger for Game 2 with $\mathcal{A}$, choosing some call to ORegister and inserting its public key P^* in place of the freshly generated P. Let the E-value for this call to ORegister be E^*; note that by the previous abort condition this uniquely determines the call to ORegister. Whenever $\mathcal{B}_2$ needs to produce a signature for the corresponding private key, it uses its signing oracle.

If $\mathcal{A}$ wins Game 4, then the abort condition has not triggered and there exists some (S^*, i) which satisfies the winning condition on line 7 of the recovery authentication experiment. Hence, the session π^i_{S*} must have successfully completed on some input $(\mathsf{nc}^*, \mathsf{rsp}^*)$ with previously generated values rc^* and ch^*. The value rc^* must have been generated by a call to ORegister. If this call used the value E^*, then $\mathsf{rc}^* = P^*$. In that case, $\mathcal{B}_2$ submits $m^* = (\mathsf{ch}^*, h^*, \mathsf{nc}^*)$ and $\sigma^* = \mathsf{rsp}^*$ to its EUF-CMA challenger. If $\mathcal{B}_2$ uses its oracle for the correct user, then σ^* is clearly a valid signature on m^* under P^*. It remains to show only that m^* was not previously queried to the signing oracle.

Assume that $\mathcal{B}_2$ uses its oracle for the correct user U^*, and suppose that m^* was previously queried to the signing oracle. This can only occur via a call to UserComplete for (U^*, j), which calls Response to produce a signature. The signing oracle is only used if E^* is input to Response, so E^* must have been provided as input to UserComplete for (U^*, j).

Note that $\pi^i_{S^*}.\mathsf{sid}$ is equal to the concatenation of

1. the identifer of S^*,
2. the username under which U^* registered rc^* at S^*,

3. the identifier $\mathsf{rcid}^* = (E^*, \mu^*)$ which U^* registered alongside rc^*,

4. ch^*, and

5. nc^*.

We have just argued that the session identifiers for (S^*, i) and (U^*, j) must agree on E^*. Since the message $(\mathsf{ch}^*, h^*, \mathsf{nc}^*)$, where h^* is the hash of the identifier of S^*, was input to the signing oracle, it follows that they must also agree on the first component and the last two components. Finally, they must agree on the username because U^* has only one account, and hence only one username, at S^*. It follows that the sessions (S^*, i) and (U^*, j) match except for the MAC tag μ. But this means that the abort condition was triggered; hence, m^* was never queried to the signing oracle.

This shows that $\mathcal{B}_2$ wins its EUF-CMA game as long as it correctly guesses the call to $\mathsf{ORegister}$ and $\mathcal{A}$ wins Game 4, implying that

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{Register}} \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\Lambda,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Finally, we reduce Game 4 to the SUF-CMA security of Σ. Let $\mathcal{B}_3$ be an SUF-CMA challenger for Σ. From its challenger, $\mathcal{B}_3$ receives a tag oracle and a verification oracle. The $\mathcal{B}_3$ acts as the challenger for Game 4 with $\mathcal{A}$, choosing some call to $\mathsf{ORegister}$ and using its tag oracle to produce the value μ^* for this registration. As before, let the E-value for this registration be E^*. If $\mathcal{B}_3$ needs to perform a verification in some call to $\mathsf{UserComplete}$ with E^* as input, it uses its verification oracle. Note, however, that since E^* values are unique for each call to $\mathsf{ORegister}$, the tag oracle is used only once.

The adversary $\mathcal{A}$ only wins Game 4 if there is some accepting server session π^i_S for which there is no matching user session (U, j). Furthermore, the only possible mismatch can be on the tag μ in the session identifier. The adversary can pick any almost-matching user session and submit its differing μ-value as a forged tag on the value (E^*, h^*), the unique message tagged by its oracle. Since the user session successfully completed and must have verified its μ-value for (E^*, h^*), and the tag oracle is used only once, this indeed a valid forgery. Hence,

$$\mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{Register}} \cdot \mathsf{Adv}^{\text{SUF-CMA}}_{\Sigma,\mathcal{B}_3}(\lambda).$$

The desired security statement is given by combining these bounds. □

The proof of security for pqCBR follows a similar structure.

Theorem 6.2. *Let* pqCBR *be the credential-based recovery protocol described in Figures 5.3 and 5.2. For any efficient adversary $\mathcal{A}$ making at most n_{user} queries to* NewUser*, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathrm{rec}}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\mathrm{cr}}_{\Pi,\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathrm{cr}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{EUF-CMEA}}_{\Delta,\mathcal{B}_2}(\lambda)$$

Proof. An identical argument to that given in the proof of Theorem 6.1 shows that the winning condition on line 3 of the recovery authentication experiment in Figure 4.1 is never met. Hence, we similarly bound the adversary's advantage in triggering the second winning condition, proceeding by a sequence of games.

Game 0. This game is identical to the security experiment $\mathsf{Exp}^{\mathrm{rec}}_{\mathsf{pqCBR},\mathcal{A}}$. Therefore

$$\mathsf{Adv}^{\mathrm{rec}}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0 except that the game aborts if there are two calls to Response with the same sk_Π-value and different rcid-values which result in the same k-value being used on line 2. This abort condition is only triggered if the distinct rcid-values result in a collision for $\Pi.\mathsf{Decaps}(\mathsf{sk}_\Pi, \cdot)$. Since there are n_{user} distinct KEM keys sk_Π in the game, it follows that

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\mathrm{cr}}_{\Pi,\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

Game 2. This game is identical to Game 1 except that the game aborts if there are two calls to Response with the same sk_Π-value and different rcid-values which result in the same τ-value being used on line 3. The previous abort condition ensures that different rcid-values result in different k-values being input to PRF; hence, this condition is only triggered if the challenger observes a collision for PRF. With $\mathcal{B}_1$ defined to be the adversary which challenges $\mathcal{A}$ and returns this collision, it follows that

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cr}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

We now reduce winning Game 2 to producing a forgery for Δ. Let $\mathcal{B}_2$ be a EUF-CMEA adversary for Δ. From the EUF-CMEA challenger, $\mathcal{B}_2$ receives an oracle which produces signatures and public keys for any blinding factor τ; the adversary $\mathcal{B}_2$ is challenged to produce (m^*, σ^*, τ^*) such that σ^* verifies on m^* under the public key produced by the blinding factor τ^*. Now, $\mathcal{B}_2$ faithfully simulates Game 2 with $\mathcal{A}$, choosing a user at random and using its oracle to answer BlindPK and Sign queries for that user's Δ-keypair.

If $\mathcal{A}$ wins Game 2, then the abort conditions have not triggered and there exists some (S^*, i) which satisfies the winning condition on line 7 of the recovery authentication experiment. Hence, the session $\pi^i_{S^*}$ must have successfully completed on some input $(\mathsf{nc}^*, \mathsf{rsp}^*)$ with previously generated values rc^* and ch^*. The value rc^* must have been generated during the registration process with some blinding factor τ^*, which can be determined by $\mathcal{B}_2$. If rc^* was generated via $\mathcal{B}_2$'s oracle, then $\mathcal{B}_2$ submits $m^* = (\mathsf{ch}^*, \mathsf{nc}^*)$, $\sigma^* = \mathsf{rsp}^*$, and τ^* to its EUF-CMEA challenger. If $\mathcal{B}_2$ uses its oracle for the correct user, then σ^* is clearly a valid signature on m^* under rc^*, the public key produced by the blinding factor τ^*. It remains to show only that (m^*, τ^*) was not previously queried to the signing oracle.

Assume that $\mathcal{B}_2$ uses its oracle for the correct user U^*, and suppose that (m^*, τ^*) were previously queried to the signing oracle. This can only occur via a call to $\mathsf{UserComplete}$ for (U^*, j), which calls $\mathsf{Response}$ to produce a signature. By the second abort condition, τ^* is only produced in $\mathsf{Response}$ if the input rcid- and aux-values are the same as the ones input when rc^* was registered. This implies that $\pi^j_{U^*}.\mathsf{peerid}$ must be the identifier of the server S^*. Similarly, $\pi^j_{U^*}.\mathsf{selfid}$ must be the username under which U^* registered at S^*.

Note that $\pi^i_{S^*}.\mathsf{sid}$ is equal to the concatenation of

1. the identifer of S^*,
2. the username under which U^* registered rc^* at S^*,
3. the identifier rcid^* which U^* registered alongside rc^*,
4. ch^*, and
5. nc^*.

We have just argued that the first three components must match with $\pi^j_{U^*}.\mathsf{sid}$. Since the message $(\mathsf{ch}^*, \mathsf{nc}^*)$ is signed for (U^*, j), $\pi^j_{U^*}.\mathsf{sid}$ also agrees with $\pi^i_{S^*}.\mathsf{sid}$ on the final two components. Therefore $\pi^j_{U^*}.\mathsf{sid} = \pi^i_{S^*}.\mathsf{sid}$. This implies that $\pi^i_{S^*}$ and $\pi^j_{U^*}$ match. Moreover, because $\pi^i_S.\mathsf{peerid}$ is the username under which U^* registered rc^* at S^*, it follows that $(U^*, S^*, \pi^i_{S^*}.\mathsf{peerid}) \in \mathcal{L}_{\mathsf{Register}}$. Hence, $\mathcal{A}$ does not win Game 2 via (S^*, i), which is a contradiction. It follows that

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{EUF-CMEA}}_{\Delta,\mathcal{B}_2}(\lambda).$$

The desired security statement is given by combining these bounds. □

6.2 Unlinkability

Theorem 6.3. *Let* gCBR *be the credential-based recovery protocol described in Figures 5.1 and 5.3. For any efficient adversary $\mathcal{A}$ making at most n_{user} queries to* NewUser*, n_{Register} total queries to* ORegister *and* $\mathsf{ORegister}_b$*, and n_{Action} total queries to* Action *and* Action_b*, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, such that*

$$\mathsf{Adv}^{\text{I-UL}}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq \binom{n_{\mathsf{Register}}}{2} \cdot \frac{1}{q-1} + \mathsf{Adv}^{\text{cr}}_{\mathsf{H},\mathcal{B}_0}(\lambda) + n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{sg-prf}}_{\mathsf{KDF},\mathcal{B}_1}(\lambda) + (n_{\mathsf{Register}} + n_{\mathsf{Action}}) \cdot \mathsf{Adv}^{\text{SUF-CMA}}_{\Sigma,\mathcal{B}_2}(\lambda).$$

Proof. First, note that since the fail bit is set based on information available to the adversary, the adversary "knows" when it will trigger the failing conditions. Hence, for every adversary $\mathcal{A}$ which triggers the failing conditions, there exists another adversary with exactly the same advantage and running time which never triggers the failing conditions; it behaves identically to $\mathcal{A}$ except that when $\mathcal{A}$ would issue a failure-triggering query, it returns a random bit. Hence, we may assume that $\mathcal{A}$ never causes fail to be set to 1.

We proceed by a sequence of games.

Game 0. This game is identical to the original security experiment $\mathsf{Exp}^{\text{I-UL}}_{\mathsf{gCBR},\mathcal{A}}$, so

$$\mathsf{Adv}^{\text{I-UL}}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0, except that the game aborts if the challenger observes a collision for H or between E-values at registration. By a similar argument to the proof of Theorem 6.1,

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq \binom{n_{\mathsf{Register}}}{2} \cdot \frac{1}{q-1} + \mathsf{Adv}^{\text{cr}}_{\mathsf{H},\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{gCBR},\mathcal{A}}(\lambda)$$

Game 2. This game is identical to Game 1 except for the following changes:

- In CredGen, the computation $(\mathsf{ck}, \mathsf{mk}) \leftarrow \mathsf{KDF}(E^s); P \leftarrow g^{\mathsf{ck}} \cdot S$ is replaced by $p \leftarrow\!\!\$\ \mathbb{Z}_q; \mathsf{mk} \leftarrow\!\!\$\ \{0,1\}^\lambda; P \leftarrow g^p$.
- In Response, the computation $(\mathsf{ck}, \mathsf{mk}) \leftarrow \mathsf{KDF}(E^s); p \leftarrow \mathsf{ck} + s$ is replaced by $p \leftarrow\!\!\$\ \mathbb{Z}_q; \mathsf{mk} \leftarrow\!\!\$\ \{0,1\}^\lambda$.

- A map $E \mapsto (p, \mathsf{mk})$ is maintained in st_U for each user U. On a call to $\mathsf{CredGen}$ or $\mathsf{Response}$ via $\mathsf{Register}(U, \cdot)$, $\mathsf{Register}_b(U, \cdot)$, $\mathsf{Action}_b(U, \cdot)$, or $\mathsf{Action}(U, \cdot)$, this map is consulted before sampling to ensure that responses are consistent.

This game can be viewed as the terminal game in a sequence of hybrid games $\mathcal{H}_i$, where $\mathcal{H}_0 = \mathcal{G}_1$ and $\mathcal{H}_i$ incorporates these changes for users 1 through i. At each step, the increase in advantage is bounded by the sg-prf security of KDF; hence,

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{sg-prf}}_{\mathsf{KDF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{gCBR},\mathcal{A}}(\lambda)$$

Game 3. This game is identical to Game 2 except that the game aborts, with a random bit being returned, if there is some (U, i) for which $(\pi^i_U.\mathsf{peerid}, \pi^i_U.\mathsf{selfid}, \pi^i_U.\mathsf{sid.rcid})$ does not correspond to some call to $\mathsf{ORegister}$ for either

- U, if $U \notin \{U_0, U_1, U_0^*, U_1^*\}$;
- U, U_0^*, or U_1^*, if $U \in \{U_0, U_1\}$; or
- U, U_0, or U_1, if $U \in \{U_0^*, U_1^*\}$.

That is, $\pi^i_U.\mathsf{sid.rcid}$ must have been returned by a call to $\mathsf{ORegister}$ which registered U (or another other user in the special cases) at the server with identifier $\pi^i_U.\mathsf{peerid}$ under the username $\pi^i_U.\mathsf{selfid}$.

We claim that the abort condition only occurs if $\mathcal{A}$ has forged a MAC tag. Consider an adversary $\mathcal{B}_2$ for the SUF-CMA security of Σ. This adversary acts as the challenger for Game 3, randomly choosing some call to either $\mathsf{Register}$ or $\mathsf{UserComplete}$ which samples a fresh MAC key mk. Instead of using the fresh key, $\mathcal{B}_2$ answers the query, and subsequent queries using the same E-value, using its MAC and Verify oracles. If the abort condition is triggered for (U, i), $\mathcal{B}_2$ submits $(E, \mathsf{H}(\pi^i_U.\mathsf{peerid})), \mu$ to its SUF-CMA challenger, where $(E, \mu) = \pi^i_U.\mathsf{sid.rcid}$. Since the session identifier was set, the tag μ must have verified on $(E, \mathsf{H}(\pi^i_U.\mathsf{peerid}))$; hence, if $\mathcal{B}_2$ chooses the correct call to $\mathsf{Register}$ or $\mathsf{UserComplete}$ in which to insert its MAC oracle, this will be a valid tag for the key used in the SUF-CMA game. Since the MAC oracle is used only in registration, and E-values for registrations do not collide, the MAC oracle is queried at most once; the abort condition ensures that it was not queried on the message-tag pair submitted by $\mathcal{B}_2$. It follows that $\mathcal{B}_2$ wins its game whenever the abort condition is triggered and it guesses the correct call to $\mathsf{Register}$ or $\mathsf{UserComplete}$. It follows that

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{gCBR},\mathcal{A}}(\lambda) \leq (n_{\mathsf{Register}} + n_{\mathsf{Action}}) \cdot \mathsf{Adv}^{\text{SUF-CMA}}_{\Sigma,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

The adversary's advantage in Game 3 is already 0, as we will show. The rest of the proof is devoted to showing that the behaviour of oracle calls for U_0 (and U_1) is independent of the behaviour of oracle calls for U_b^* (and U_{1-b}^*), despite their shared state. Our goal is to eventually partition the shared state such that the non-starred user only accesses values in one partition and the starred user only access values in the other. At that point, we can provide the starred users with fresh state variables, making them truly indistinguishable. The following games achieve this with some careful bookkeeping around the adversary's failure conditions.

Game 4. This game is identical to Game 3 except that the game additionally aborts if there is some i such that $(\pi^i_{U_0^*}.\mathsf{peerid}, \pi^i_{U_0^*}.\mathsf{selfid}, \pi^i_{U_0^*}.\mathsf{sid.rcid})$ does not correspond to some call to $\mathsf{ORegister}$ for U_0^*.

We claim that this abort condition is never triggered. By the previous abort condition, this can only occur if $(\pi^i_{U_0^*}.\mathsf{peerid}, \pi^i_{U_0^*}.\mathsf{selfid}, \pi^i_{U_0^*}.\mathsf{sid.rcid})$ corresponds to a call to $\mathsf{ORegister}$ for either U_0 or U_1; without loss of generality, say U_0. It follows that $(U_0, S, \pi_{U_0^*}.\mathsf{selfid}) \in \mathcal{L}_{\mathsf{Register}}$, where S is the server identified by $\pi^i_{U_0^*}.\mathsf{peerid}$. Since servers do not register the same username to different accounts, it follows that $(U_0^*, S, \pi_{U_0^*}.\mathsf{selfid}) \notin \mathcal{L}_{\mathsf{Register}}$. It follows that the failing condition on line 3 of $\mathsf{CheckFail}$ is triggered, which contradicts our assumption that $\mathcal{A}$ never triggers a failing condition. Therefore

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 5. This game is identical to Game 4 except that the game additionally aborts if there is some i such that either

- $(\pi^i_{U_1^*}.\mathsf{peerid}, \pi^i_{U_1^*}.\mathsf{selfid}, \pi^i_{U_1^*}.\mathsf{sid.rcid})$ does not correspond to some call to $\mathsf{ORegister}$ for U_1^*,
- $(\pi^i_{U_0}.\mathsf{peerid}, \pi^i_{U_0}.\mathsf{selfid}, \pi^i_{U_0}.\mathsf{sid.rcid})$ does not correspond to some call to $\mathsf{ORegister}$ for U_0, or
- $(\pi^i_{U_1}.\mathsf{peerid}, \pi^i_{U_1}.\mathsf{selfid}, \pi^i_{U_1}.\mathsf{sid.rcid})$ does not correspond to some call to $\mathsf{ORegister}$ for U_1.

A similar argument to the one given for the previous game shows that these abort conditions are never met, giving

$$\mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Since abort conditions from Games 3, 4, and 5 are triggered based on public information, we may assume that $\mathcal{A}$ never makes a query that would trigger one of them; when $\mathcal{A}$ would make such a query, it simply returns a random bit.

Game 6. This game is identical to Game 5, except that $\mathsf{st}_U.\mathsf{uid}$ is no longer updated or accessed. Instead, $\mathsf{UserBegin}(\pi_U^i, \mathsf{uid}, \mathsf{serverID}, \mathsf{st}_U)$ rejects if $(U, S, \mathsf{uid}) \notin \mathcal{L}_{\mathsf{Register}}$, where S is the server with identifier $\mathsf{serverID}$. The previous abort condition ensures that there is no difference in behaviour from Game 5, so

$$\mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 7. This game is identical to Game 6 except for the following changes:

- Data stored in long-term state in calls to any of $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}_b(U_0, \cdot)$ is flagged.
- The game aborts if a call to any of $\mathsf{ORegister}(U_0^*, \cdot)$, $\mathsf{Register}_b(U_0^*, \cdot)$, $\mathsf{Action}(U_0^*, \cdot)$, or $\mathsf{Action}_b(U_0^*, \cdot)$ accesses flagged data.
- The game aborts if a call to any of $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}_b(U_0, \cdot)$ accesses non-flagged data.

We claim that these abort conditions are never triggered. Long-term state st_U contains two types of data:

- a mapping $E \mapsto (p, \mathsf{mk})$ for values of E which are either generated in $\mathsf{Register}$ or queried to $\mathsf{UserComplete}$ for U,
- a mapping $\mathsf{serverID} \mapsto \{\mathsf{nc}\}$, the set of nc-values previously issued for $\mathsf{serverID}$.

We begin by showing that access of flagged data for U_0^* never occurs. Both mappings are accessed only in $\mathsf{UserComplete}$.

If E is provided as input to $\mathsf{UserComplete}$ for U_0^*, then by the previous abort conditions, it must have been generated in some call to $\mathsf{ORegister}$ for U_0^*. Since E-values do not collide at registration, it cannot have been generated in some call to $\mathsf{ORegister}$ for U_0. Hence, it also cannot have been queried to $\mathsf{UserComplete}$ for U_0. Therefore E is not flagged. similarly, if $\mathsf{st}_U.\mathsf{nc}[\mathsf{serverID}]$ is looked up in $\mathsf{UserComplete}$ for U_0^*, then U_0^* must be registered at the server with identifier $\mathsf{serverID}$. Since U_0^* and U_0 are never registered at the same server, this value also cannot be flagged.

An identical argument shows that U_0 never accesses non-flagged data. Therefore

$$\mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 8. This game is identical to Game 7 except for the following changes:

- Data stored in long-term state in calls to any of $\mathsf{ORegister}(U_1, \cdot)$, $\mathsf{Register}_b(U_1, \cdot)$, $\mathsf{Action}(U_1, \cdot)$, or $\mathsf{Action}(U_1, \cdot)$ is also flagged.
- The game aborts if a call to any of $\mathsf{ORegister}(U_d^*, \cdot)$, $\mathsf{Register}_b(U_d^*, \cdot)$, $\mathsf{Action}(U_d^*, \cdot)$, or $\mathsf{Action}(U_d^*, \cdot)$ for $d \in \{0, 1\}$ accesses flagged data.
- The game aborts if a call to any of $\mathsf{ORegister}(U_d, \cdot)$, $\mathsf{Register}_b(U_d, \cdot)$, $\mathsf{Action}(U_d, \cdot)$, or $\mathsf{Action}(U_d, \cdot)$ for $d \in \{0, 1\}$ accesses non-flagged data.

A similar argument to the previous game shows that

$$\mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_8}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 9. This game is identical to Game 8 except that fresh long-term state variables are used for U_0^* and U_1^* instead of st_{U_b} and $\mathsf{st}_{U_{1-b}}$. Since only flagged data are stored in st_{U_b} and $\mathsf{st}_{U_{1-b}}$ and only non-flagged data is stored in $\mathsf{st}_{U_0^*}$ and $\mathsf{st}_{U_1^*}$, there is no detectable difference between Games 8 and Games 9. Therefore

$$\mathsf{Adv}^{\mathcal{G}_8}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_9}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Moreover, the behaviour of $\mathsf{ORegister}_b$ and Action_b is independent of the value of b, so

$$\mathsf{Adv}^{\mathcal{G}_9}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = 0.$$

The desired security statement follows from combining these bounds. □

We now turn our attention to the unlinkability of pqCBR, following a similar proof structure.

Theorem 6.4. *Let* pqCBR *be the credential-based recovery protocol described in Figures 5.2 and 5.3. For any efficient adversary $\mathcal{A}$ making at most n_{user} queries to* NewUser *and n_O total queries to* ORegister, $\mathsf{ORegister}_b$, OAction, *and* $\mathsf{OAction}_b$ *there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathrm{UL}}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq \binom{n_O}{2} \cdot 2^{-\lambda} + n_{\mathsf{user}}\mathsf{Adv}^{\text{KEM-UL}}_{\Pi,\mathcal{B}_0}(\lambda) + n_O\Big(\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{UL-CMEA}}_{\Delta,\mathcal{B}_3}(\lambda)\Big).$$

Proof. As in Theorem 6.3, it suffices to bound the advantage of an adversary which never triggers the failing conditions. We proceed by a sequence of games.

Game 0. This game is identical to the original security experiment $\mathsf{Exp}^{\mathrm{UL}}_{\mathsf{pqCBR},\mathcal{A}}$, so

$$\mathsf{Adv}^{\mathrm{UL}}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0 except for the following changes:

- Users do not generate or store Π keypairs.
- All calls to $\Pi.\mathsf{Encaps}$ are replaced by randomly sampling a ciphertext c and a key k from the message space and key space of Π, respectively. A mapping $c \mapsto k$ is maintained in the long-term state from which the public key would have been retrieved.
- All calls to $\Pi.\mathsf{Decaps}$ are replaced by first looking up the provided c-value in the mapping in the long-term state from which the private key would have been retrieved. If no match is found, a random key k is sampled and returned, and the value (c, k) is stored in the list.

This game can be viewed as the final game in a sequence of hybrid games $\mathcal{H}_i$, with $\mathcal{H}_0$ and in which $\mathcal{H}_i$ uses its oracles to answer encapsulation and decapsulation queries for public key i. The loss of advantage at each step is bounded by the KEM-UL security of Π. Thus

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_{\mathsf{user}} \cdot \mathsf{Adv}^{\text{KEM-UL}}_{\Pi,\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

Game 2. This game is identical to Game 1 except for the following changes:

- All calls to $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$ where k was freshly sampled for a "ciphertext" c are replaced by sampling τ at random. A mapping $(c, \mathsf{aux}) \mapsto \tau$ is maintained in long-term state, as in the previous game.
- All calls to $\tau \leftarrow \mathsf{PRF}(k, \mathsf{aux})$ where k was retrieved from long-term state for a "ciphertext" c are replaced by
  - looking up the value of τ, if $(c, \mathsf{aux}) \mapsto \tau$ is in long-term state, or
  - randomly sampling τ and updating the mapping otherwise.

This game can be viewed as the final game in a sequence of hybrid games $\mathcal{H}'_i$, with $\mathcal{H}'_0 = \mathcal{G}_1$ and in which calls to $\mathsf{PRF}(k, \cdot)$ are successively replaced by random sampling. The loss of advantage at each step is bounded by $\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}^i}(\lambda)$, where $\mathcal{B}^i$ uses its PRF oracle to answer queries for the ith sample of k. Since the number of samples of k is bounded by n_O, it follows that

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_O \cdot \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqCBR},\mathcal{A}}(\lambda),$$

where $\mathcal{B}_1$ is the most successful of the adversaries $\mathcal{B}^i$.

Game 3. This game is identical to Game 2 except that the game aborts if two samples of τ collide. Since PRF outputs binary strings of length at least λ, it follows by a similar argument to the proof of Theorem 6.1 that

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq \binom{n_O}{2} \cdot 2^{-\lambda} + \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{pqCBR},\mathcal{A}}(\lambda).$$

Game 4. This game is identical to Game 3 except for the following changes:

- All calls to $\Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$ or $\Delta.\mathsf{Sign}(\mathsf{sk}_\Delta, \tau, \cdot)$ where τ was freshly sampled for (c, aux) are replaced by calling $\Delta.\mathsf{KeyGen}$ and using the resulting fresh keypair to perform the operation. A mapping $(c, \mathsf{aux}) \mapsto (\mathsf{pk}_\Delta, \mathsf{sk}_\Delta)$ is maintained in long-term state, as in the previous game.
- All calls to $\Delta.\mathsf{BlindPK}(\mathsf{pk}_\Delta, \tau)$ or $\Delta.\mathsf{Sign}(\mathsf{sk}_\Delta, \tau, \cdot)$ where τ was retrieved from long-term state for (c, aux) are replaced by looking up the corresponding Δ-keypair and using that keypair to perform the operation.

This game can be viewed as the final game in a sequence of hybrid games $\mathcal{H}^j_i$, with $\mathcal{H}_0 = \mathcal{G}_3$ and where $\mathcal{H}^j_i$ replaces the Δ-keypair used to perform operations for the ith user's jth τ-value with a freshly sampled keypair. The loss of advantage at each step is bounded by $\mathsf{Adv}^{\text{UL-CMEA}}_{\mathcal{B},\mathcal{D}^j_i}(\lambda)$, where $\mathcal{B}^j_i$ uses its $\mathsf{BlindPK}$ and Sign oracles to answer queries for the ith user, issuing its challenge for the jth τ-value. Since the total number of Δ-operations is bounded by n_O, it follows that

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{pqCBR},\mathcal{A}}(\lambda) \leq n_O \cdot \mathsf{Adv}^{\text{UL-CMEA}}_{\Delta,\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{pqCBR},\mathcal{A}}(\lambda),$$

where $\mathcal{B}_2$ is the most successful of the adversaries $\mathcal{B}^j_i$.

As in the proof of Theorem 6.3, the adversary's advantage at this point is already 0. We proceed in a similar fashion, partitioning the starred and non-starred users' shared state so that we can eventually provide the starred users with fresh state variables.

Game 5. This game is identical to Game 4, except that $\mathsf{st}_U.\mathsf{uid}$ is no longer updated or accessed. Instead, $\mathsf{UserBegin}(\pi_U^i, \mathsf{uid}, \mathsf{serverID}, \mathsf{st}_U)$ rejects if $(U, S, \mathsf{uid}) \notin \mathcal{L}_{\mathsf{Register}}$, where S is the server with identifier $\mathsf{serverID}$.

For users other than U_d or U_d^* for $d \in \{0, 1\}$, there is no difference in behaviour. Note, however, that $\mathsf{UserBegin}$ for U_0 will accept on a username-server identifier pair which corresponds to an account for U_0^* (similarly for U_1, U_0^*, and U_1^*). This would result in the fail bit being set to 1, however. Hence, no such query to $\mathsf{UserBegin}$ is made, and there is no detectable difference in behaviour. Then

$$\mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 6. This game is identical to Game 5 except for the following changes:

- Data stored in long-term state in calls to any of $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}(U_0, \cdot)$ is flagged.
- The game aborts if a call to any of $\mathsf{ORegister}(U_0^*, \cdot)$, $\mathsf{Register}_b(U_0^*, \cdot)$, $\mathsf{Action}(U_0^*, \cdot)$, or $\mathsf{Action}(U_0^*, \cdot)$ accesses flagged data.
- The game aborts if a call to any of $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}(U_0, \cdot)$ accesses non-flagged data.

We claim that these abort conditions are never triggered. Long-term state st_U contains two types of data:

- a mapping $(c, \mathsf{aux}) \mapsto (\mathsf{pk}_\Delta, \mathsf{sk}_\Delta)$ for values of c generated in $\mathsf{Register}$ or queried to $\mathsf{UserComplete}$ for U,
- a mapping $\mathsf{aux} \mapsto \{\mathsf{nc}\}$, the set of nc-values previously issued for $\mathsf{serverID}$.

We begin by showing that access of flagged data for U_0^* never occurs. Both mappings are accessed only in $\mathsf{UserComplete}$.

If (c, aux) is looked up in the mapping in $\mathsf{UserComplete}$ for U_0^*, then $\mathsf{aux} = (\mathsf{serverID}, \mathsf{uid})$, where U_0^* is registered at the server with identifier $\mathsf{serverID}$ under the username uid. Since U_0 and U_0^* are never registered at the same server under the same username, (c, aux) must not be flagged in the first mapping. Similarly, aux cannot be flagged in the second mapping.

An identical argument shows that U_0 never accesses non-flagged data. Therefore

$$\mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 7. This game is identical to Game 6 except for the following changes:

- Data stored in long-term state in calls to any of $\mathsf{ORegister}(U_1, \cdot)$, $\mathsf{Register}_b(U_1, \cdot)$, $\mathsf{Action}(U_1, \cdot)$, or $\mathsf{Action}(U_1, \cdot)$ is also flagged.
- The game aborts if a call to any of $\mathsf{ORegister}(U_d^*, \cdot)$, $\mathsf{Register}_b(U_d^*, \cdot)$, $\mathsf{Action}(U_d^*, \cdot)$, or $\mathsf{Action}(U_d^*, \cdot)$ for $d \in \{0, 1\}$ accesses flagged data.
- The game aborts if a call to any of $\mathsf{ORegister}(U_d, \cdot)$, $\mathsf{Register}_b(U_d, \cdot)$, $\mathsf{Action}(U_d, \cdot)$, or $\mathsf{Action}(U_d, \cdot)$ for $d \in \{0, 1\}$ accesses non-flagged data.

A similar argument to the previous game shows that

$$\mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Game 8. This game is identical to Game 8 except that fresh long-term state variables are used for U_0^* and U_1^* instead of st_{U_b} and $\mathsf{st}_{U_{1-b}}$. Since only flagged data are stored in st_{U_b} and $\mathsf{st}_{U_{1-b}}$ and only non-flagged data is stored in $\mathsf{st}_{U_0^*}$ and $\mathsf{st}_{U_1^*}$, there is no detectable difference between Games 8 and Games 9. Therefore

$$\mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_8}_{\mathsf{gCBR},\mathcal{A}}(\lambda).$$

Moreover, the behaviour of $\mathsf{ORegister}_b$ and Action_b is independent of the value of b, so

$$\mathsf{Adv}^{\mathcal{G}_8}_{\mathsf{gCBR},\mathcal{A}}(\lambda) = 0.$$

The desired security statement follows from combining these bounds. □

Chapter 7

Instantiation

We now discuss the instantiation of the primitives used to construct the post-quantum CBR protocol. In particular, we show that the novel properties which we used to analyze the protocol's security are met by existing quantum-safe algorithms.

7.1 Pseudorandom Function

In the proof of Theorem 6.2, we relied on the PRF used in pqCBR being globally collision resistant. The extendable-output functions SHAKE-128 and SHAKE-256, often used as PRFs in post-quantum algorithms, are designed to provide collision resistance [20]. Two other notable candidates, HMAC and HKDF, are constructed from hash functions in such a way that a collision for the PRF is also a collision for the underlying hash function.

7.2 Key Blinding Scheme

Although we made no additional security assumptions on the key-blinding signature scheme in our security analysis of pqCBR, we required four extra properties to prove that our approach yielded a secure ARKG scheme, as outlined in 2.6. These properties were key-recovery security, unique blinding, private key unblinding, and strong independent blinding. We now show that the isogeny-based scheme blCSI-FiSh satisfies these four properties. For a detailed description of blCSI-FiSh and its parent scheme, CSI-FiSh, see [9] and [4].

Unique blinding is immediate: blinded public keys are obtained from blinded secret keys by a free and transitive group action, so there is a one-to-one correspondence between them. Private key unblinding is also trivial. Blinded secret keys are obtained by adding a blinding factor to the secret key, so UnblindSK simply subtracts the blinding factor from the blinded key to obtain the secret key. As for key-recovery security, an adversary who can retrieve the secret key from the public key can forge a CSI-FiSh signature, giving the following theorem:

Theorem 7.1. *For any efficient adversary $\mathcal{A}$, there exists an efficient algorithm $\mathcal{B}$ such that.*

$$\mathsf{Adv}^{\mathrm{KR}}_{\mathsf{blCSI\text{-}FiSh},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathsf{CSI\text{-}FiSh},\mathcal{B}}(\lambda)$$

The proof of S-Ind-Blind security is similarly straightforward.

Theorem 7.2. *Let* KDF *be the key derivation function used in* blCSI-FiSh*. For any efficient adversary $\mathcal{A}$ making at most n oracle queries, there exists an efficient algorithm $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{S-Ind-Blind}}_{\mathsf{blCSI\text{-}FiSh},\mathcal{A}}(\lambda) \leq n \cdot \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{KDF},\mathcal{B}}(\lambda).$$

Proof. When τ is uniformly sampled, the PRF security of KDF ensures that the derived blinding factor is indistinguishable from a randomly sampled element of $\mathbb{Z}_N^L$. Since derived secret keys are obtained by adding the blinding factor to the secret key, it follows that derived key pairs are uniformly distributed. The factor of n in the bound arises because an independent PRF key is sampled for each oracle call. □

Remark 7.3. Although blCSI-FiSh is the only one of the key blinding schemes from [9] which satisfies these four properties, this does not mean that it is the only one which is safe to be integrated with post-quantum WebAuthn. As discussed in Chapter 4, we do not believe that the ARKG model accurately captures the security properties required of a recovery solution.

7.3 Key Encapsulation Mechanism

We required two non-standard properties of the key encapsulation mechanism in pqCBR: collision resistance and unlinkability, defined in sections 2.4.1 and 2.4.2 respectively. We focus our attention on proving that these properties are satisfied by CRYSTALS-Kyber, which has been selected by NIST for standardization. For a detailed description of the

algorithm, see [2]. CRYSTALS-Kyber encompasses both a public-key encryption scheme and a key encapsulation mechanism; we will denote the former by KyberPKE and the latter by KyberKEM.

We will refer to the following CRYSTALS-Kyber parameters:

- q: the prime 3329.
- n: the bit-length of encapsulated keys. Equal to 256 for all security levels.
- H: a hash function with digest bit-length n, instantiated with SHA3-256,
- G: a hash function with digest bit-length $2n$, instantiated with SHA3-512,
- R_q: the ring $\mathbb{Z}_q[X]/(X^n + 1)$. In particular, R_q has size q^n.
- k: the dimension of the public key matrix $\mathbf{A}$. Equal to 2 for NIST Level 1 security, 3 for NIST Level 3 security, and 4 for NIST Level 5 security.

The collision resistance of KyberKEM is immediate: decapsulation outputs on input c are of the form $\mathsf{KDF}(K\|\mathsf{H}(c), n)$, where $\mathsf{KDF} = \text{SHAKE-256}$ and $\mathsf{H} = \text{SHA3-256}$ and K is a fixed-length variable string. This gives the following result.

Theorem 7.4. *For any efficient adversary $\mathcal{A}$, there exists an efficient algorithm $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cr}}_{\text{SHA3-256}}(\lambda) + \mathsf{Adv}^{\mathrm{cr}}_{\text{SHAKE-256}(\cdot,n),\mathcal{B}}(\lambda).$$

The proof that Kyber satisfies unlinkability is much more involved. We begin by introducing a simpler problem, which we refer to as "ciphertext guessing", and showing that it is difficult for KyberKEM. We then reduce breaking the unlinkability of KyberKEM to ciphertext guessing. The ciphertext guessing game challenges an adversary given no information about the public key (not even encapsulation or decapsulation oracles) to output a ciphertext c^* which is not implicitly rejected by Kyber decapsulation. This is described formally in Figure 7.1. Although it might seem bizarre to keep a public key private, this situation arises naturally in the context of unlinkability, where no guarantees can be made against an active adversary with access to the public key.

We make use of the fact that KyberPKE encryption with truly random coins is pseudorandom under the Module-LWE assumption. As in [2], we denote this pseudorandomness property by pr.

$\mathsf{Exp}^{\text{cguess}}_{\mathsf{KyberPKE},\mathcal{A}}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$
2 : $h \leftarrow \mathsf{H}(\mathsf{pk})$
3 : $c^* \leftarrow \mathcal{A}()$
4 : $m^* \leftarrow \mathsf{Dec}(\mathsf{sk}, c^*)$
5 : $r^* \leftarrow \mathsf{G}(m^*, h)$
6 : **return** $[\![c^* = \mathsf{Enc}(\mathsf{pk}, m^*, r^*)]\!]$

Figure 7.1: The ciphertext guessing game for KyberPKE

Lemma 7.5. *Suppose that* H *and* G *are random oracles. For any efficient adversary* $\mathcal{A}$ *making at most* n_H *queries to* H *and* n_G *queries to* G*, there exist efficient algorithms* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *such that*

$$\mathsf{Adv}^{\text{cguess}}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) \leq n_\mathsf{H} \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KyberPKE},\mathcal{B}_0}(\lambda) + \frac{n_\mathsf{G}}{2^n} + \mathsf{Adv}^{\text{pr}}_{\mathsf{KyberPKE},\mathcal{B}_1}(\lambda) + \frac{1}{q^{nk}}.$$

Proof. We proceed by a sequence of games.

Game 0. This game is precisely the cguess game for KyberPKE. Therefore

$$\mathsf{Adv}^{\text{cguess}}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0, except that the return value is 1 if and only if the first component c_1^* matches the first component of $\mathsf{Enc}(\mathsf{pk}, m^*, r^*)$. Since this is a less restrictive winning condition, it is clear that

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 2. This game is identical to Game 1, except that the game aborts if the adversary queries pk to the random oracle H. Since the adversary receives no information about pk, the probability that it guesses pk correctly in a single query is bounded by 2^{-h_pk}, where h_pk is the min-entropy of the public key. We note, as in Corollary 1 of [9], that this quantity is bounded by the advantage $\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KyberPKE},\mathcal{B}_0}(\lambda)$. Therefore

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) \leq n_\mathsf{H} \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KyberPKE},\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 3. This game is identical to Game 2, except that h is sampled uniformly at random instead of being computed as $\mathsf{H}(\mathsf{pk})$. Since the adversary does not query pk to the H random oracle, there is no change in advantage:

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 4. This game is identical to Game 3, except that the game aborts if the adversary queries (m^*, H) to the random oracle. Since the adversary must correctly guess the randomly sampled value H in order to make such a query,

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) \leq \frac{n_{\mathsf{G}}}{2^n} + \mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 5. This game is identical to Game 4, except that r^* is sampled uniformly at random instead of being computed as $\mathsf{G}(m^*, h)$. Since the adversary does not query (m^*, h) to the H random oracle, there is no change in advantage:

$$\mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 6. This game is identical to Game 5, except that the winning condition is changed to $c_1^* = \mathsf{Enc}(\mathsf{pk}, m', r^*)_1$, where m' is a randomly sampled message. Since the first component of $\mathsf{Enc}(\mathsf{pk}, m^*, r^*)$ depends only on pk and r^*, there is no change in advantage:

$$\mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

Game 7. This game is identical to Game 6, except that the value $\mathsf{Enc}(\mathsf{pk}, \cdot, r^*)$ is replaced by a random sample from the ciphertext space. In particular, the first component is a random sample from R_q^k. The loss in advantage is bounded by the advantage of an adversary $\mathcal{B}_1$ in distinguishing a random Kyber ciphertext from random samples from the ciphertext space. It follows that

$$\mathsf{Adv}^{\mathcal{G}_6}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{pr}}_{\mathsf{KyberPKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda).$$

In order to win this game, the adversary must correctly guess a uniformly sampled value from R_q^k. Since $|R_q| = q^n$,

$$\mathsf{Adv}^{\mathcal{G}_7}_{\mathsf{KyberPKE},\mathcal{A}}(\lambda) = \frac{1}{q^{nk}}.$$

The desired security statement follows from combining these bounds. □

We now reduce the KEM-UL experiment for $\mathsf{KyberKEM}$ to ciphertext guessing.

Theorem 7.6. *For any efficient adversary $\mathcal{A}$ making at most n_E encapsulation queries and n_D decapsulation queries, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\text{KEM-UL}}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) \leq +n_E \cdot \mathsf{Adv}^{\text{SPR-CCA}}_{\mathsf{KyberKEM},\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{SHAKE\text{-}256},\mathcal{B}_1}(\lambda) + n_D \cdot \mathsf{Adv}^{\text{cguess}}_{\mathsf{KyberPKE},\mathcal{B}_2}(\lambda).$$

Proof. We proceed by a sequence of games.

Game 0. This game is identical to the experiment $\mathsf{Exp}^{\text{KEM-UL}}_{\mathsf{KyberKEM},\mathcal{A}}$. Therefore

$$\mathsf{Adv}^{\text{KEM-UL}}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda).$$

Game 1. This game is identical to Game 0, except that the encapsulation oracle is replaced by an oracle which samples and outputs a uniformly random ciphertext and a uniformly random key, and the decapsulation oracle responds consistently with these outputs. This game can be viewed as the final game in a sequence of hybrid games $\mathcal{H}_i$, defined such that $\mathcal{H}_i$ is identical to $\mathcal{G}_0$ except that encapsulation queries 1 through i are answered with such a pair of random samples. At each step, the loss of advantage is bounded by the SPR-CCA security of $\mathsf{KyberKEM}$: an SPR-CCA adversary $\mathcal{B}^i$ who substitutes its challenge ciphertext-key pair for the ith encapsulation response plays $\mathcal{H}_{i-1}$ if the pair was honestly output and $\mathcal{H}_i$ if the pair was uniformly sampled. Hence,

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) \leq n_E \cdot \mathsf{Adv}^{\text{SPR-CCA}}_{\mathsf{KyberKEM},\mathcal{B}_0}(\lambda) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda),$$

where $\mathcal{B}_0$ is the most successful of the adversaries $\mathcal{B}^i$s.

Game 2. This game is identical to Game 1, except that all implicit rejection outputs $\mathsf{SHAKE\text{-}256}(z, \mathsf{H}(c))$ are replaced by random samples (up to consistency, so that the same value is output for c if queried multiple times). The loss in advantage is bounded by the PRF-security of $\mathsf{SHAKE\text{-}256}$, where we regard the random prefix z as the key and $\mathsf{H}(c)$ as the label. Therefore

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{prf}}_{\mathsf{SHAKE\text{-}256},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda).$$

Game 3. This game is identical to Game 2, except that the game aborts (returning a random bit) if $\mathcal{A}$ queries a value c^* to the decapsulation oracle which is *not* implicitly rejected.

We show that if $\mathcal{A}$ triggers the abort condition, then we can construct a winning adversary $\mathcal{B}_2$ for the cguess game for $\mathsf{KyberPKE}$. The adversary $\mathcal{B}_2$ randomly chooses some

index $1 \leq i \leq n_D$. It challenges $\mathcal{A}$ to Game 3, answering $\mathcal{A}$'s first $i-1$ decapsulation queries by returning random keys (up to consistency if the same value is queried). Upon receiving the ith decapsulation query c^*, it halts and returns c^* to its challenger. (If $i = n_D + 1$, then $\mathcal{B}$ submits $\mathcal{A}$'s return value.) Note that $\mathcal{B}$ wins the cguess game whenever whenever it guesses correctly the index where $\mathcal{A}$ first submits a non-rejecting query. It follows that

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) \leq n_D \cdot \mathsf{Adv}^{\mathrm{cguess}}_{\mathsf{KyberPKE},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda).$$

Since implicitly rejected decapsulation queries simply return a random key, the adversary $\mathcal{A}$'s oracles are now independent of the value of b. Therefore

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{KyberKEM},\mathcal{A}}(\lambda) = 0.$$

The desired security statement follows from combining these bounds. □

Remark 7.7. The security bound given by combining Theorem 7.6 with Lemma 7.5 is disturbingly non-tight. However, this is not likely to pose an issue in the context of WebAuthn recovery. Encapsulations are only performed when the user creates a new recovery credential, and decapsulations are only performed when the user attempts to recover an account. Hence, the values n_E and n_D are likely to be quite small in practice. Although the value of n_{H} could be significantly larger, the proof of Lemma 7.5 shows that it is multiplied by the min-entropy of a Kyber public key in the security bound. Not only are these public keys pseudorandom under the MLWE assumption, they also contain a pseudorandom 256-bit suffix, obtained via a call to SHAKE-256 on a truly random 256-bit value [2], meaning that their min-entropy is likely very close to a constant 256 bits.

Chapter 8

Evaluation

In Table 8.1, we list the sizes of values which must be communicated between parties in the recovery protocol. These numbers are based on [17], [2], and [9]. In particular, they assume that ECDSA on the P-256 curve and HMAC-SHA256 (with output truncated to 16 bytes) are used to instantiate gCBR, and the Level 3 parameter set for KyberKEM is used to instantiate pqCBR. We do not include blPicnic in the table, as it has not been implemented and detailed information about signature size is not available. In Table 8.2, we provide the time costs of blinding, signing, and verifying for blinded signature schemes and compare them with analogous costs for the associated (non-blinded) signature schemes. This data is copied directly from [9], whose implementations of the blinded signature schemes are available at http://github.com/tedeaton/pq-key-blinding. Information about the platform on which the runtimes were collected was not available. The schemes blLegRoast and blCSI-FiSh were implemented in C, while blDilithium-QROM was implemented in Sage.

As is to be expected, the post-quantum protocol requires significantly more communication and storage space than its group-based counterpart. However, regardless of which blinded signature scheme is used, these increases are not as significant when compared to an instantiation of WebAuthn using the base post-quantum signature scheme. A more

Protocol	|rc|	|rcid|	|rsp|
gCBR[ECDSA-P256, HMAC-SHA256]	64 B	80 B	64 B
pqCBR[KyberKEM, blCSI-FiSh]	16 kB	1.06 kB	0.45 kB
pqCBR[KyberKEM, blDilithium-QROM]	10 kB	1.06 kB	5.7 kB
pqCBR[KyberKEM, blLegRoast]	0.50 kB	1.06 kB	11.22 kB

Table 8.1: Credential, credential identifier, and response sizes for gCBR and pqCBR

Scheme	$\lvert pk \rvert$	$\lvert \sigma \rvert$	KeyGen / BlindPK	Sign	Verify
CSI-FiSh	16 kB	0.45 kB	10800 ms	554 ms	553 ms
blCSI-FiSh	16 kB	0.45 kB	10600 ms	546 ms	540 ms
Dilithium-QROM	7.7 kB	5.7 kB	3810 ms	9360 ms	2890 ms
blDilithium-QROM	10 kB	5.7 kB	1650 ms	28300 ms	717 ms
LegRoast	0.50 kB	7.94 kB	0.9 ms	12.4 ms	11.7 ms
blLegRoast	0.50 kB	11.22 kB	0.9 ms	18.6 ms	17.8 ms

Table 8.2: Comparison of post-quantum blinded and non-blinded signature schemes using data from [9]

significant efficiency loss is seen when comparing the signing and verifying times of the blinded schemes with their base schemes, as given in Table 8.2. However, blinding a public key is always at least as fast as generating a fresh public key. Registration and recovery additionally require an encapsulation and a decapsulation, respectively, which will introduce an additional overhead over WebAuthn ceremonies which do not involve recovery credentials. However, we do not attempt to provide an estimate of the total cost of registration and recovery, as this would require adding the time of a KEM operation and the time of a blinded signature scheme operation. The implementation of KyberKEM is highly optimized, while the implementations of the blinded signature schemes are only proofs of concept—we would not be comparing apples to apples.

An important consideration for use of blDilithium-QROM is the matrix $\mathbf{A}$, which forms part of public keys. In the base signature scheme, this matrix is freshly generated for each public key. In the blinded version, the same $\mathbf{A}$ is used across all public keys derived from the same seed keypair. In order to provide unlinkability guarantees, the same $\mathbf{A}$ must be used by all tokens which cannot otherwise be distinguished. In practice, relying parties will require the backup authenticator's attestation identifier (AAGUID), which is shared among a large batch of tokens of the same model [10]. Hence, it suffices to ensure that all tokens with the same AAGUID also use the same $\mathbf{A}$. The security analysis of blDilithium-QROM in [9] required a new variant of the learning with errors assumption, "static $\mathbf{A}$ module LWE", in order to deal with the same matrix being used by multiple public keys.

The blCSI-FiSh signature scheme is especially attractive due to its small signature sizes. However, its security claim is contested, and there is no clear path to increasing its security parameters due to the intense class group computation required [21].

8.1 Stronger Security Notions

Although the definition of EUF-CMEA security given in [9] does not provide the adversary with the identity public key, the security proofs of both blDilithium-QROM and blCSI-FiSh show that they are in fact secure against an adversary who is given this extra information. This is because both are obtained via the Fiat–Shamir transform from an identification protocol whose public key includes the identity public key. This allows a stronger notion of recovery authentication security, in which the adversary is also allowed to obtain primary keys from lost primary tokens. Unlinkability, however, still requires the public keys to be kept secret.

As a final note, pqCBR need not be instantiated with quantum-safe primitives in order to be secure against a classical adversary. Indeed, we believe that two features of the post-quantum protocol could be incorporated into the group-based protocol with positive results. Modifying Yubico's protocol to use independent key pairs for shared secret establishment and signing could eliminate, or at least weaken, the required non-standard security assumption. The one-time additional overhead required for the extra keypair would be more than compensated for by the many-time reduction in recovery credential size by removing the MAC and incorporating the auxiliary data into the PRF label. We leave a detailed analysis of these modifications to future work.

Chapter 9

Conclusion

In this work, we proposed a quantum-safe protocol for account recovery with passwordless authentication. In particular, our construction gives a quantum-safe version of Yubico's proposed WebAuthn backup standard. We also introduced a novel security model for account recovery and analyzed both Yubico's protocol and our own under it, introducing several novel security properties along the way. This led us to discover a weakness in the former, which we conveyed to the authors of the proposal. Finally, we provided concrete instantiations (with proof) of primitives which meet the novel properties required for the analysis of our post-quantum protocol.

9.1 Limitations

Our security model does not account for users who have more than one backup authenticator. This constitutes a significant gap in our analysis of Yubico's proposal for WebAuthn backup, which places no such restriction on the user. Incorporating this feature would increase the complexity of our security model, in which we consider token, client, and human user to be a single entity with shared memory. In particular, it would make unlinkability more difficult to reason about. Our model also allows us to consider recovery from a more general point of view by abstracting away the CTAP subprotocol. However, this means that our results are conditional on CTAP providing secure communication.

9.2 Future Work

Our contributions could be extended or expanded on in several interesting directions. Perhaps the most important future work is a thorough evaluation of the novel security assumption on which we based the security proof for Yubico's protocol. A reduction of this assumption to one which is well studied—or a security proof that eliminates the need for a new assumption—would be highly desirable, given that the scheme is being proposed for standardization. Expanding our security model to account for users with multiple backup tokens, as Yubico's proposal allows, would provide security guarantees for a wider range of use cases in practice. Another interesting approach could be to examine unlinkability against a passive adversary with access to primary authenticators' keys. With regards to protocol instantiation, an examination of KEMs besides CRYSTALS-Kyber—both pre- and post-quantum—would provide insight into how tightly bound our proposal is to one particular algorithm. Also of interest would be a comparison of our approach with those taken by [7] and [11] with regards to security, usability, and ease of implementation. This could involve evaluating the other constructions under our novel security model and vice versa.

References

[1] Sunpreet S. Arora, Saikrishna Badrinarayanan, Srinivasan Raghuraman, Maliheh Shirvanian, Kim Wagner, and Gaven Watson. Avoiding lock outs: Proactive FIDO account recovery using managerless group signatures. Cryptology ePrint Archive, Paper 2022/1555, 2022. https://eprint.iacr.org/2022/1555.

[2] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.02), 2021. https://pq-crystals.org/kyber/resources.shtml.

[3] Manuel Barbosa, Alexandra Boldyreva, Shan Chen, and Bogdan Warinschi. Provable security analysis of FIDO2. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 125–156, Virtual Event, August 2021. Springer, Heidelberg.

[4] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[5] Nina Bindel, Cas Cremers, and Mang Zhao. FIDO2, CTAP 2.1, and WebAuthn 2: Provable security and post-quantum instantiation. Cryptology ePrint Archive, Paper 2022/1029, 2022. https://eprint.iacr.org/2022/1029.

[6] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 651–681. Springer, Heidelberg, August 2017.

[7] Sebastian A. Clermont. Post quantum asynchronous remote key generation. Master's thesis, Technische Universität Darmstadt, July 2022.

[8] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. Security analysis of signature schemes with key blinding. Cryptology ePrint Archive, Paper 2023/380, 2023. https://eprint.iacr.org/2023/380.

[9] Edward Eaton, Douglas Stebila, and Roy Stracovsky. Post-quantum key-blinding for authentication in anonymity networks. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT 2021*, volume 12912 of *LNCS*, pages 67–87. Springer, Heidelberg, October 2021.

[10] Nick Frymann, Daniel Gardham, Franziskus Kiefer, Emil Lundberg, Mark Manulis, and Dain Nilsson. Asynchronous remote key generation: An analysis of Yubico's proposal for W3C WebAuthn. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 939–954. ACM Press, November 2020.

[11] Nick Frymann, Daniel Gardham, and Mark Manulis. Asynchronous remote key generation for post-quantum cryptosystems from lattices. Cryptology ePrint Archive, Paper 2023/419, 2023. https://eprint.iacr.org/2023/419.

[12] Hidehito Gomi, Bill Leddy, and Dean H. Saxe. Recommended account recovery practices for FIDO relying parties, 2019. https://fidoalliance.org/recommended-account-recovery-practices.

[13] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 402–432. Springer, Heidelberg, May / June 2022.

[14] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Token meets wallet: Formalizing privacy and revocation for FIDO2. Cryptology ePrint Archive, Paper 2022/084, 2022. https://eprint.iacr.org/2022/084.

[15] Jeff Hodges. Recovering from device loss, 2018. https://github.com/w3c/webauthn/issues/931.

[16] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors,

CRYPTO 2012, volume 7417 of *LNCS*, pages 273–293. Springer, Heidelberg, August 2012.

[17] Emil Lundberg and Dain Nilsson. WebAuthn recovery extension, 2019. https://github.com/Yubico/webauthn-recovery-extension.

[18] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. Is FIDO2 the kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In *2020 IEEE Symposium on Security and Privacy*, pages 268–285. IEEE Computer Society Press, May 2020.

[19] Varun Maram and Keita Xagawa. Post-quantum anonymity of Kyber. Cryptology ePrint Archive, Paper 2022/1696, 2022. https://eprint.iacr.org/2022/1696.

[20] National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions, 2015.

[21] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[22] Keita Xagawa. Anonymity of NIST PQC round 3 KEMs. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 551–581. Springer, Heidelberg, May / June 2022.