Quantum-Safe Account Recovery for WebAuthn

Douglas Stebila
University of Waterloo
Waterloo, Ontario, Canada
dstebila@uwaterloo.ca

Spencer Wilson
University of Waterloo
Waterloo, Ontario, Canada
spencer.wilson@uwaterloo.ca

ABSTRACT

WebAuthn is a passwordless authentication protocol which allows users to authenticate to online services using public-key cryptography. Users prove their identity by signing a challenge with 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.

WebAuthn's reliance on proof-of-possession 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. To solve this problem, 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). With this solution, users authenticate with a single token, only relying on their backup token if necessary for account recovery. However, Yubico's protocol relies on the hardness of the discrete logarithm problem for its security and hence is vulnerable to an attacker with a powerful enough quantum computer.

We present a WebAuthn recovery protocol which can be instantiated with quantum-safe primitives. We also critique the security model used in previous analysis of Yubico's protocol and propose a new framework which we use to evaluate the security of both the group-based and the quantum-safe 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 require the cryptographic primitives underlying the protocols to satisfy a number of novel security properties such as KEM unlinkability, which we formalize. We prove that well-known quantum-safe algorithms, including CRYSTALS-Kyber, satisfy the properties required for analysis of our quantum-safe protocol.

CCS CONCEPTS

• **Security and privacy → Public key (asymmetric) techniques**; **Authentication**; *Pseudonymity, anonymity and untraceability*.

KEYWORDS

account recovery, FIDO2, post-quantum, quantum-safe, WebAuthn

ACM Reference Format:
Douglas Stebila and Spencer Wilson. 2024. Quantum-Safe Account Recovery for WebAuthn. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3634737.3661138

ASIA CCS '24, July 1–5, 2024, Singapore, Singapore

1 INTRODUCTION

1.1 Motivation

Passwords have provided the primary method of user authentication on the Internet for decades. Password-based authentication is easy for users to understand, does not require significant supporting 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. Safely storing and verifying passwords is a non-trivial task prone to developer error. Moreover, password-based authentication is, without additional tools such as password managers, inherently user-unfriendly: strong, unique passwords are difficult to remember.

Passwordless authentication attempts to avoid problems with password-based authentication by simply getting rid of passwords. This approach 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. Among the authentication solutions proposed by the FIDO Alliance is the FIDO2 passwordless authentication protocol. FIDO2 is the composition of two subprotocols, WebAuthn and CTAP. WebAuthn provides an API by which authenticators (for example, USB security keys) use public key credentials to authenticate to servers (referred to as "relying parties") via an intermediary client (typically a web browser). Client–authenticator communication is specified by CTAP. Both protocols have been subjected to academic security analysis and are beginning to see widespread deployment.

WebAuthn introduces its own problems, however. One of these centres on account recovery. An individual who loses the device they use to authenticate has no built-in method for recovering access to their accounts. This is a major obstacle preventing users from embracing FIDO2 [18]. Current advice to users is to purchase two tokens and register them both at each site: if one is lost, then the other can be still be used to log in [13]. This, however, doubles the amount of work that users are required to perform at registration.

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. A user can link their two tokens, store the backup in a safe place, and use the primary authenticator for day-to-day logins. 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 [11]. The authors of [11] 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 based on key-blinding signature schemes. We also highlight a number of weaknesses in the security analysis of Yubico's protocol. Notably, we describe a simple attack which allows a malicious server to determine whether two accounts belong to the same user, violating unlinkability. This attack escaped detection under the model in [11]. To address these weaknesses, we propose a novel session-based 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 quantum-safe protocol under this new model, with the simplifying assumption that each user has a single primary token and a single backup token. We leave analysis of multiple-backup functionality to future work. Notably, our model makes no reference to specific details of WebAuthn. This means that recovery protocols proven secure under our model 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 cryptographic primitives to satisfy a number of non-standard security properties. Most notably, we require KEM decapsulation to be collision resistant and, in some proofs, pseudorandom. We provide formal definitions of these security properties and prove that they are satisfied by CRYSTALS-Kyber. We additionally introduce a new Diffie–Hellman-like assumption in order to establish the security of Yubico's existing quantum-unsafe protocol under our new model; however, we are unable to reduce this new assumption to any standard ones.

1.3 Related Work

Our contributions are adjacent to a number of recent efforts. Several papers have examined the provable security of WebAuthn, beginning with [3]. This work was expanded upon by [15], which was the first to analyze the protocol's privacy properties; [15] also advances a protocol for revocation which is similar to Yubico's recovery proposal. Of particular relevance to our work is [4], which proposed a provably secure quantum-safe version of WebAuthn. Previous work on WebAuthn account recovery includes [1], which proposes a solution based on group signatures, and [11], on which our work builds directly.

Our work makes prominent use of signature schemes with key blinding, which [10] and [9] define and discuss at length and [7] presents as a class of signature schemes with randomizable keys. We discuss a novel anonymity property of key encapsulation mechanisms; related analysis is done in [14], [22], and [19].

After our research was completed, we became aware of [5] and [12], two recent efforts to solve the quantum-safe WebAuthn recovery problem. Our work was done independently from these and takes a different approach. The key ideas of the three constructions are based on different primitives: [12] on split KEMs, [5] on deterministically generated keypairs, and ours on key-blinding signature schemes. Additionally, all three constructions use different security models: [12] follows the original model from [11], [5] keeps this model with minor tweaks, and our work introduces a novel, session-based model. Our model is the only one which detects the attack on unlinkability mentioned above.

2 PRELIMINARIES

In this section, we lay out the cryptographic primitives on which Yubico's recovery protocol and our post-quantum version rely. We omit standard definitions and instead focus on reintroducing specialized concepts on which we build and describing several novel security properties. We reserve proofs of these novel properties—when applicable—for Section 5.

2.1 Novel PRF Security Properties

2.1.1 Collision Resistance. In the security analysis of our post-quantum recovery protocol, we will require a non-standard property of a pseudo-random function (PRF): collision resistance, where the key is included in the input. Although this property is not strictly implied by the standard PRF security definition, it is in practice a reasonable assumption, as we discuss in more detail in Section 5.

2.1.2 Shifted Group PRF. To analyze the security of Yubico's protocol under our 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, we say that a function $F\colon \mathbb{G} \to \mathbb{Z}_q$ satisfies the sgPRF property if the function $F'\colon \mathbb{Z}_q \times \mathbb{G} \setminus \{1\} \to \mathbb{Z}_q$ defined by $F'(s, E) = F(E^s) + s$ 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 Novel KEM Security Properties

For analysis of our post-quantum protocol, we require two novel security properties of KEMs. In Section 5, we show that these properties are satisfied by CRYSTALS-Kyber, which was recently selected for standardization by NIST.

2.2.1 Collision Resistance. The first property we require is per-key collision resistance: for a given random keypair, it should be difficult for a CCA adversary to produce two ciphertexts which decapsulate to the same value. We refer to this property as CR-CCA.

2.2.2 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 by observing encapsulations and decapsulations. Concretely, it should be infeasible to distinguish between encapsulations (respectively, decapsulations) and random sampling from the ciphertext (respectively, key) spaces. We refer to this property as KEM unlinkability. The corresponding security experiment is described in Figure 1. We define the adversary's advantage to be

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

2.3 Key-Blinding Signature Scheme

As defined in [10], a *key-blinding signature scheme* Δ consists of four algorithms defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: generates an *identity* or *seed* keypair $(\mathsf{pk}, \mathsf{sk})$ from which blinded keys will be derived.

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

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$
2 : $\mathcal{L} \leftarrow \emptyset$
3 : $b \leftarrow\$ \{0,1\}$
4 : $b' \leftarrow\$ \mathcal{A}^{O^E_b, O^D_b}()$
5 : **return** $[\![b' = b]\!]$

Oracle $O^E_1()$

1 : $c \leftarrow\$ \mathcal{C}$
2 : $k \leftarrow\$ \mathcal{K}$
3 : $\mathcal{L} \leftarrow \mathcal{L} \cup (c, k)$
4 : **return** (c, k)

Oracle $O^D_1(c)$

1 : **if** $(c, k') \in \mathcal{L}$ **then return** k'
2 : $k \leftarrow\$ \mathcal{K}$
3 : $\mathcal{L} \leftarrow \mathcal{L} \cup (c, k)$
4 : **return** (c, k)

Figure 1: The KEM-UL **experiment for a KEM** Π**. The oracles** O^E_0 **and** O^D_0 **are defined by** $\mathsf{Encaps}(\mathsf{pk})$ **and** $\mathsf{Decaps}(\mathsf{sk}, \cdot)$**, respectively.**

- $\mathsf{BlindPK}(\mathsf{pk}, \tau)$: 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 fresh keypairs and blindings with the fixed keypair. We refer to [10] for formalizations of these properties.

Key-blinding signature schemes are typically used in applications such as the Tor network where users wish to retain a single public key but also remain anonymous. Similar properties are desirable for account recovery: users should not have to maintain a large database of recovery keys and may wish their various accounts to be unlinkable.

2.4 Asynchronous Remote Key Generation

Previous security analysis of Yubico's WebAuthn recovery extension in [11] 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 consists of five algorithms:

[1]The definition in [10] 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.

- $\mathsf{Setup}(1^\lambda)$: deterministically outputs the parameters pp for the scheme.
- $\mathsf{KeyGen}(\mathsf{pp})$: outputs a seed keypair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}, \mathsf{aux})$: probabilistically outputs a public key pk' and a corresponding credential cred bound to 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.

In Yubico's protocol, the backup authenticator generates a seed keypair 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. This usage is depicted in Figure 2.

The ARKG scheme defined in [11] based on Yubico's proposal in [17] is implemented using elliptic curve group arithmetic. Its details are given in Figure 3. It is parameterized by a finite group $\mathbb{G}$ of order q with generator g, a signature scheme Σ, and key derivation functions KDF_1 and KDF_2.

The security of an ARKG scheme as defined in [11] 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 desirable in the context of WebAuthn, where private key proof-of-posession is used for authentication but user credentials should not be correlatable.

2.4.1 Public-Key Unlinkability. An adversary for public-key unlinkability is challenged to distinguish between a fixed distribution $\mathcal{D}$ (in [11], 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 4. The adversary's advantage is defined to be

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

2.4.2 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 5. The "weak" security variants remove the DeriveSK oracle, and the "honest" variants require that the adversary output a credential obtained from its DerivePK oracle. The adversary's advantage for the malicious strong key variant is defined to be the probability that the msKS experiment returns 1.

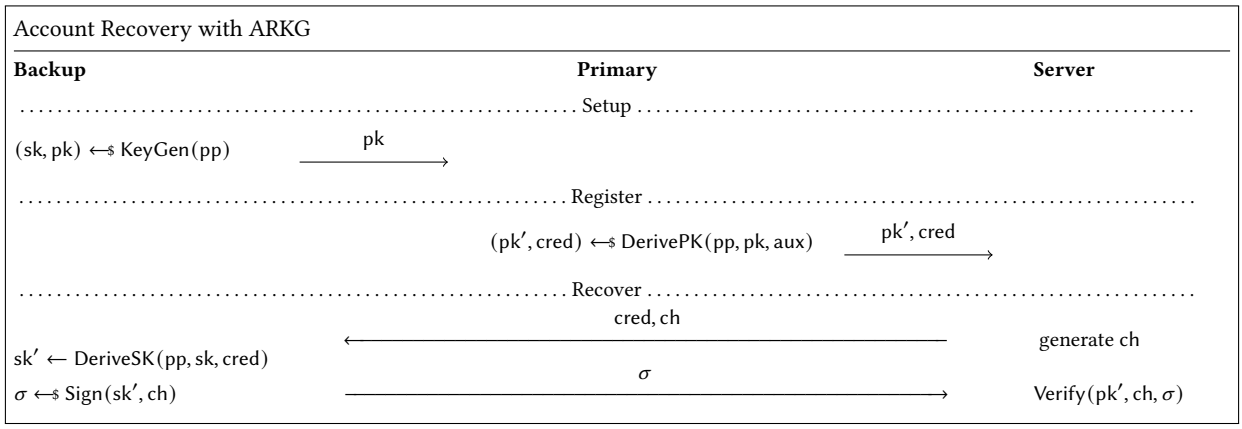

Figure 2: Using an ARKG scheme for account recovery, as in Yubico's WebAuthn extension. The aux **data is the server ID. The challenge string** ch **follows a format defined by WebAuthn and includes a random string.**

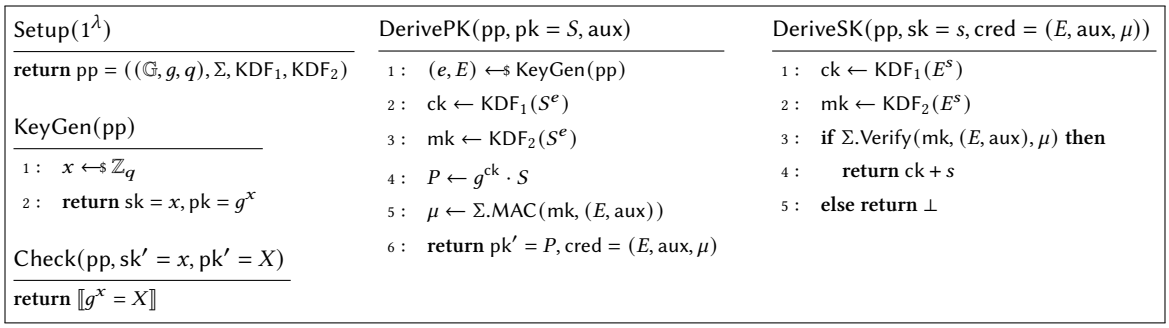

Figure 3: The group-based ARKG construction from [11], based on Yubico's WebAuthn recovery extension [17]

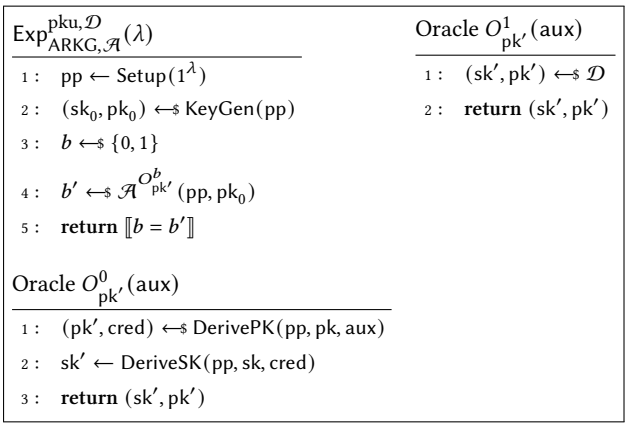

Figure 4: The PK-unlinkability experiment for ARKG

3 PROTOCOL MODEL

3.1 Problems with the ARKG model

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 requirements of the protocol. 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 of the model seem out of step with real-world threats. In particular, the ARKG adversary is given at once too much power and not enough.

Consider the msKS security model, presented in Figure 5. 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 does not inherently constitute a weakness in analysis. However, this is the only way in which the adversary can interact with the derived secret keys. Notably, the adversary cannot obtain signatures made

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

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

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

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

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

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

Figure 5: The msKS **experiment for ARKG**

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 [11]: an ARKG scheme which satisfies PK-unlinkability with some distribution (see Figure 4) 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 is 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 with either the primary token or the backup token in any other way.[2] The recovery credentials are computed and sent to servers alongside derived public keys; hence, a realistic security model must reveal them to the adversary.

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 [10], 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 Section 4 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 can be claimed to be secure. We now attempt to provide such a security model.

3.2 Credential-based Recovery

We propose an abstraction which captures both the core cryptographic details and the logistics of the associated protocol, which we call *credential-based recovery*. To simplify security analysis, we assume that each user has exactly one primary token and exactly one backup token, and that these tokens are not shared among users. In reality, 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 authenticator 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 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 assumptions about channel security during the recovery process.

We 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.

3.2.1 Protocol Abstraction. A credential-based recovery protocol defines an interaction between a user U and a server S. The protocol has two components: an 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

[2] There is some room for doubt about the PK-unlinkability security definition in [11]. 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.

some long-term state as well as short-term state associated with individual sessions.

The stateless interface consists of the following algorithms:

- KeyGen(1^λ): outputs a seed keypair (pk, 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.
- CredGen(pk, aux): inputs a primary key pk and some auxiliary information aux, outputs a recovery credential rc (bound to aux) and its identifier rcid.
- Response(bk, rcid, aux, ch, nc): inputs a secret key bk, a recovery credential identifier rcid, auxiliary information aux, a challenge ch, and a new credential nc; outputs a response rsp.
- Verify(rc, aux, nc, ch, 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 listed below in the order in which they are intended to be performed:

- Register: The user generates a recovery credential which the server stores.
- 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 (either user or server),
- status: the session status (either recover, accept, or reject),
- sid: the session identifier, and
- st: additional state for the session, to be used as defined by specific protocols.

We place no restrictions on variables defined in long-term state, but we do assume that all state is initialized to $\perp$ or $\emptyset$.

3.2.2 Recovery Authentication. We model the recovery authentication security of a credential-based recovery protocol CBR with the experiment $\mathsf{Exp}^{\mathsf{rec}}_{\mathsf{CBR}}$, formally defined in Figure 6. 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], [15], and [4], 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 all of the following conditions hold:

- one of π_1.role and π_2.role is user and the other is server;
- π_1.status = accept = π_2.status;
- π_1.selfid = π_2.peerid and π_2.selfid = π_2.peerid; and
- π_1.sid = π_2.sid.

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

- Two distinct non-matching sessions have the same session identifier. In practice, this means that the protocol is vulnerable to a replay attack.
- A server session π accepts without a matching session for a user registered at S under the username π.peerid. In practice, 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 to be the probability that the rec experiment returns 1.

3.2.3 Single-Account Recovery Authentication. Unfortunately, Yubico's proposed standard does not satisfy recovery authentication security, as we will show in Section 4. It does, however, satisfy a slightly weaker 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 for 1rec is identical to the one for rec except that the adversary fails if any user is registered twice at any server.

3.2.4 Unlinkability. In the unlinkability security experiment, denoted $\mathsf{Exp}^{\mathsf{UL}}_{\mathsf{CBR}}$ and depicted in Figure 7, the adversary is challenged

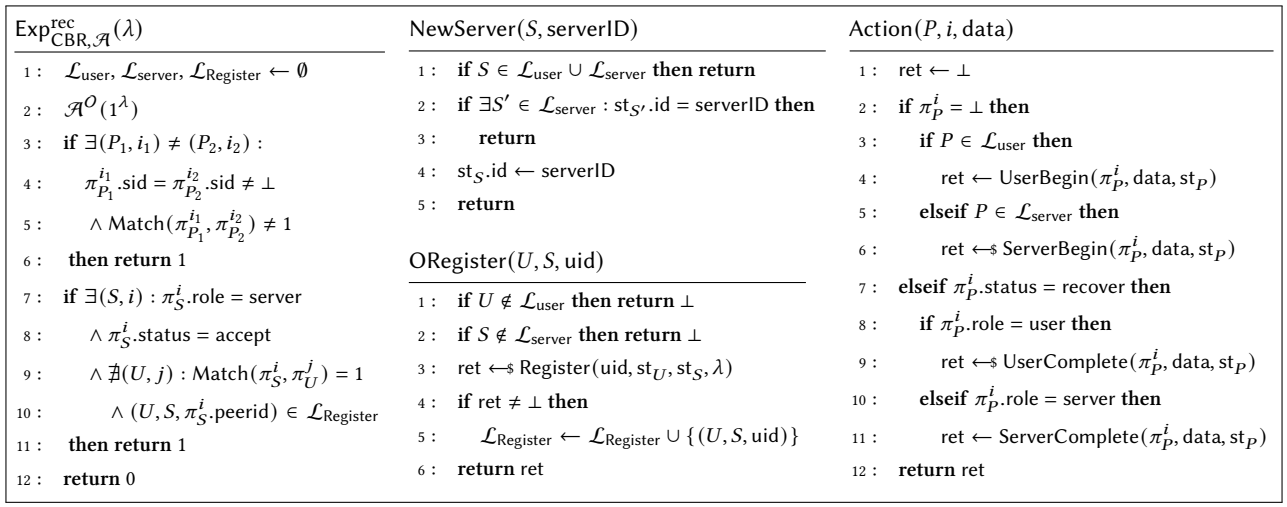

Figure 6: The rec **experiment for** CBR**. The** Match **and** NewUser **functions are described in 3.2.2.**

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 NewUser, NewServer, $\mathsf{ORegister}_b$, and Action_b, with the latter two defined as follows:

- The oracle $\mathsf{ORegister}_b$ is identical to ORegister, except that on a query with $U = U_0^*$, it will use st_{U_b} when calling 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

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

3.2.5 Inter-Domain Unlinkability. Yubico's proposed standard fails to meet our definition of unlinkability due to the same weakness that prevents it from satisfying our notion of recovery authentication, outlined in Section 4. 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 ORegister and $\mathsf{ORegister}_b$ if the given user has already registered at the given server.
- The fail flag is set in 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.

We define advantage as in the UL experiment. 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.

4 GROUP-BASED AND QUANTUM-SAFE 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.

4.1 Protocol Descriptions

In Figure 8, we describe the credential-based recovery scheme defined by Yubico's standard. Readers will note the similarities with the ARKG scheme described in [11]. Notably, our 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

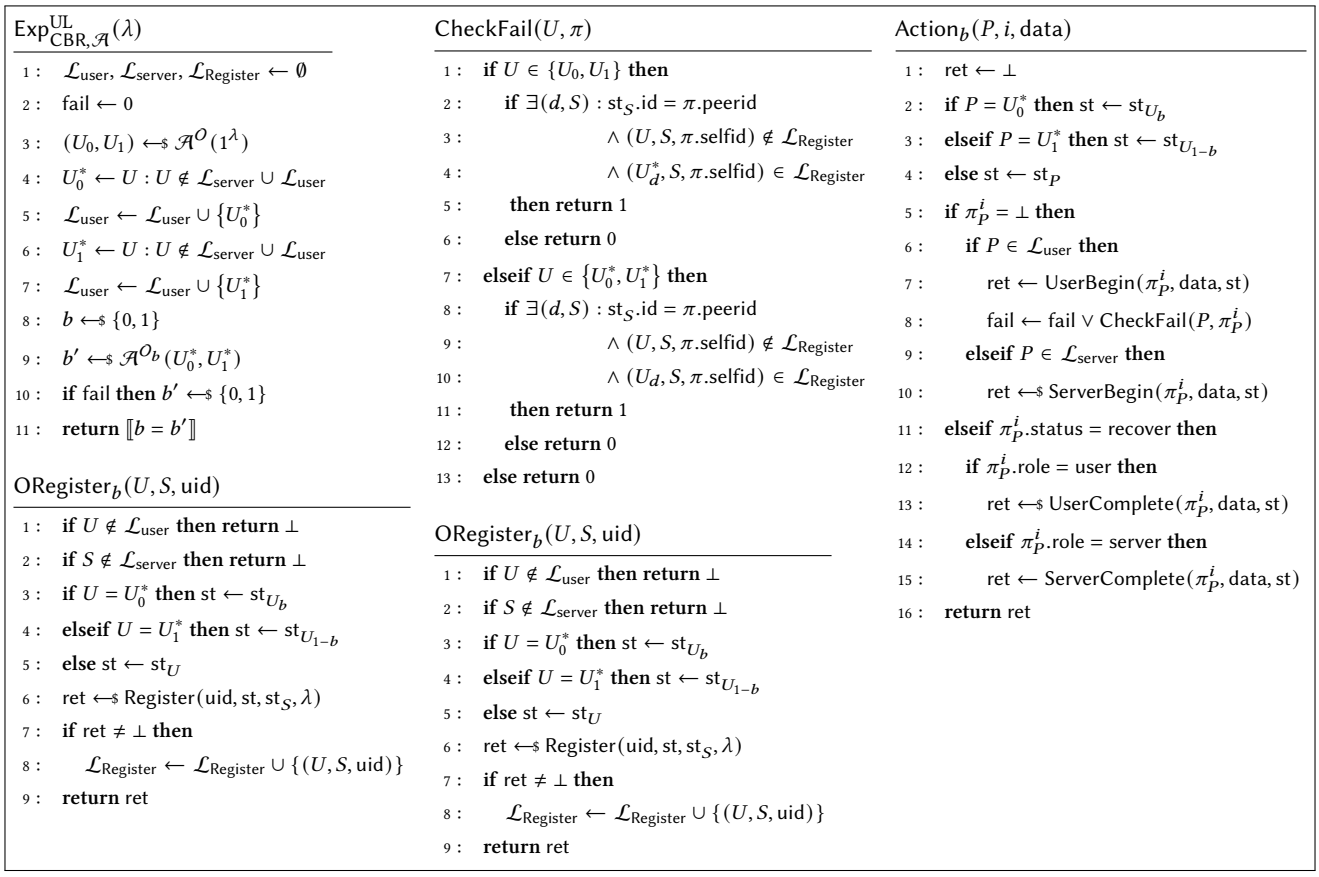

Figure 7: The UL **experiment for** CBR**. For a description of the oracles** NewUser**,** NewServer**,** ORegister**, and** Action**, see Figure 6.**

the ARKG description; these elements are present in Yubico's proposal but omitted from the analysis in [11]. Our post-quantum credential-based recovery scheme is described in Figure 9.

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 10, 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 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.

4.2 Weaknesses in Yubico's 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.

KeyGen(1^λ)

1 : $s \leftarrow\$ \mathbb{Z}_q$
2 : $S \leftarrow g^s$
3 : **return** (pk, bk) = (S, s)

Verify(rc = P, aux, nc, ch, rsp)

1 : $h \leftarrow \mathsf{H}(\text{aux})$
2 : **return** $\Lambda.\mathsf{Verify}(P, (\text{ch}, h, \text{nc}), \text{rsp})$

Response(bk = s, rcid = (E, μ), aux, ch, nc)

1 : **if** $E = 1$ **then return** $\perp$
2 : $(\text{ck}, \text{mk}) \leftarrow \mathsf{KDF}(E^s)$
3 : $h \leftarrow \mathsf{H}(\text{aux})$
4 : **if** $\Sigma.\mathsf{Verify}(\text{mk}, (E, h), \mu)$ **then**
5 : $\quad p \leftarrow \text{ck} + s$
6 : $\quad$ **return** $\Lambda.\mathsf{Sign}(p, (\text{ch}, h, \text{nc}))$
7 : **else return** $\perp$

CredGen(pk = S, aux)

1 : $e \leftarrow\$ \mathbb{Z}_q \setminus \{0\}$
2 : $E \leftarrow g^e$
3 : $(\text{ck}, \text{mk}) \leftarrow \mathsf{KDF}(S^e)$
4 : $P \leftarrow g^{\text{ck}} \cdot S$
5 : $h \leftarrow \mathsf{H}(\text{aux})$
6 : $\mu \leftarrow \Sigma.\mathsf{MAC}(\text{mk}, (E, h))$
7 : **return** rc = P, rcid = (E, μ)

Figure 8: Yubico's credential-based recovery scheme

KeyGen(1^λ)

1 : $(\text{pk}_\Delta, \text{sk}_\Delta) \leftarrow\$ \Delta.\mathsf{KeyGen}()$
2 : $(\text{pk}_\Pi, \text{sk}_\Pi) \leftarrow\$ \Pi.\mathsf{KeyGen}()$
3 : **return** pk = $(\text{pk}_\Delta, \text{pk}_\Pi)$, bk = $(\text{sk}_\Delta, \text{sk}_\Pi)$

Response(bk = $(\text{sk}_\Delta, \text{sk}_\Pi)$, rcid = c, aux, ch, nc)

1 : $k \leftarrow \Pi.\mathsf{Decaps}(\text{sk}_\Pi, c)$
2 : $\tau \leftarrow \mathsf{PRF}(k, \text{aux})$
3 : **return** $\Delta.\mathsf{Sign}(\text{sk}_\Delta, \tau, (\text{ch}, \text{nc}))$

CredGen(pk = $(\text{pk}_\Delta, \text{pk}_\Pi)$, aux)

1 : $(c, k) \leftarrow\$ \Pi.\mathsf{Encaps}(\text{pk}_\Pi)$
2 : $\tau \leftarrow \mathsf{PRF}(k, \text{aux})$
3 : $\text{rc} \leftarrow \Delta.\mathsf{BlindPK}(\text{pk}_\Delta, \tau)$
4 : **return** rc, rcid = c

Figure 9: Our post-quantum credential-based recovery scheme. The Verify **function is omitted as it simply calls** Δ.Verify**.**

Register(uid, st_U, st_S, λ)

1 : **if** $\text{st}_S.\text{rc}[\text{uid}] \neq \perp$ **then return** $\perp$
2 : **if** $(\text{st}_U.\text{pk}, \text{st}_U.\text{bk}) = \perp$
3 : $\quad$ **then** $(\text{st}_U.\text{pk}, \text{st}_U.\text{bk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$
4 : aux $\leftarrow (\text{st}_S.\text{id}, \text{uid})$ / pqCBR
5 : aux $\leftarrow \text{st}_S.\text{id}$ / gCBR
6 : $(\text{rc}, \text{rcid}) \leftarrow\$ \mathsf{CredGen}(\text{st}_U.\text{pk}, \text{aux})$
7 : $\text{st}_U.\text{uid}[\text{st}_S.\text{id}] \leftarrow \text{st}_U.\text{uid}[\text{st}_S.\text{id}] \cup \{\text{uid}\}$
8 : $\text{st}_S.\text{rc}[\text{uid}] \leftarrow (\text{rc}, \text{rcid})$
9 : **return** rc, rcid

ServerBegin(π_S^i, data = (uid, ch), st_S)

1 : **if** $\text{st}_S.\text{rc}[\text{uid}] = \perp$ **then return**
2 : **if** $\text{ch} \in \text{st}_S.\text{ch}[\text{uid}]$ **then return**
3 : $\pi_S^i.\text{selfid} \leftarrow \text{st}_S.\text{id}$
4 : $\pi_S^i.\text{peerid} \leftarrow \text{uid}$
5 : $\pi_S^i.\text{role} \leftarrow \text{server}$
6 : $\pi_S^i.\text{status} \leftarrow \text{recover}$
7 : $\pi_S^i.\text{st} \leftarrow \text{ch}$
8 : $\text{st}_S.\text{ch}[\text{uid}] \leftarrow \text{st}_S.\text{ch}[\text{uid}] \cup \{\text{ch}\}$
9 : **return**

UserBegin(π_U^i, data = (uid, serverID), st_U)

1 : $\pi_U^i.\text{selfid} = \text{uid}$
2 : $\pi_U^i.\text{peerid} = \text{serverID}$
3 : $\pi_U^i.\text{role} = \text{user}$
4 : **if** $\text{uid} \notin \text{st}_U.\text{uid}[\text{serverID}]$
5 : $\quad$ **then** $\pi_U^i.\text{status} = \text{reject}$
6 : **else** $\pi_U^i.\text{status} = \text{recover}$
7 : **return**

UserComplete(π_U^i, data = (rcid, nc, ch), st_U)

1 : aux $\leftarrow (\pi_U^i.\text{peerid}, \pi_U^i.\text{selfid})$ / pqCBR
2 : aux $\leftarrow \pi_U^i.\text{peerid}$ / gCBR
3 : **if** $\text{nc} \in \text{st}_U.\text{nc}[\text{aux}]$ **then return** $\perp$
4 : $\text{rsp} \leftarrow\$ \mathsf{Response}(\text{st}_U.\text{bk}, \text{rcid}, \text{aux}, \text{nc}, \text{ch})$
5 : **if** $\text{rsp} \neq \perp$ **then**
6 : $\quad \pi_U^i.\text{status} \leftarrow \text{accept}$
7 : $\quad \pi_U^i.\text{sid} \leftarrow (\pi_U^i.\text{peerid}, \pi_U^i.\text{selfid}, \text{ch}, \text{rcid}, \text{nc})$
8 : $\quad \text{st}_U.\text{nc}[\text{aux}] \leftarrow \text{st}_U.\text{nc}[\text{aux}] \cup \{\text{nc}\}$
9 : **else** $\pi_U^i.\text{status} \leftarrow \text{reject}$
10 : **return** rsp

Figure 10: A generic credential-based recovery protocol

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 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 [15]. 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. Yubico's draft specification has been updated to reflect our work.

Due to space limitations, we state the security bounds for gCBR and pqCBR in the main body and refer readers who wish to examine the details to Appendices B–E.

4.3 Recovery Authentication

Theorem 4.1 (gCBR satisfies single-account recovery authentication). *Let* gCBR *be the credential-based recovery protocol described in Figures 10 and 8, instantiated with a group of order q. 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*

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

Theorem 4.2 (pqCBR satisfies recovery authentication). *Let* pqCBR *be the credential-based recovery protocol described in Figures 10 and 9. 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*

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

4.4 Unlinkability

Theorem 4.3 (gCBR satisfies inter-domain unlinkability). *Let* gCBR *be the credential-based recovery protocol described in Figures 8 and 10, instantiated with a group of order q. For any efficient adversary $\mathcal{A}$ making at most n_{user} queries to* NewUser*, n_{Register} total queries to* ORegister *and* 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*

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

Theorem 4.4 (pqCBR satisfies unlinkability). *Let* pqCBR *be the credential-based recovery protocol described in Figures 9 and 10. For any efficient adversary $\mathcal{A}$ making at most n_{user} queries to* NewUser *and n_O total queries to the oracles* ORegister*,* ORegister$_b$*,* OAction*, and* OAction$_b$ *there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, and $\mathcal{B}_2$ such that*

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

5 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.

5.1 Pseudorandom Function

In the proof of Theorem 4.2, we relied on the PRF used in pqCBR being globally collision resistant. The extendable-output functions (XOFs) SHAKE-128 and SHAKE-256, often used as PRFs in post-quantum algorithms, are designed to provide collision resistance [8]. 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.

5.2 Key Encapsulation Mechanism

We required two non-standard properties of the key encapsulation mechanism in pqCBR: collision resistance and unlinkability, defined in Sections 2.2.1 and 2.2.2 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 Kyber.

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.

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

Theorem 5.1. *For any efficient adversary $\mathcal{A}$, there exists an efficient algorithm $\mathcal{B}$ such that*

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

The proof that Kyber satisfies unlinkability is more technical and relies on the random oracle model, as do the proofs of security in the Kyber specification [2]. It makes use of the strong pseudorandomness (SPR-CCA) property of Kyber (shown in in [19]) and the pseudorandomness of KyberPKE as well as algorithmic details such as implicit rejection. In the interest of space, we will state only the security result, leaving the details for Appendix A.

Theorem 5.2. *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*

$$\begin{aligned}\mathsf{Adv}^{\text{KEM-UL}}_{\mathsf{Kyber},\mathcal{A}}(\lambda) \leq\ & n_E \cdot \mathsf{Adv}^{\text{SPR-CCA}}_{\mathsf{Kyber},\mathcal{B}_0}(\lambda) \\ &+ \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{SHAKE\text{-}256},\mathcal{B}_1}(\lambda) \\ &+ n_D n_{\mathsf{H}} \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{KyberPKE},\mathcal{B}_0}(\lambda) + \frac{n_D n_{\mathsf{G}}}{2^n} \\ &+ n_D \cdot \mathsf{Adv}^{\mathrm{pr}}_{\mathsf{KyberPKE},\mathcal{B}_1}(\lambda) + \frac{n_D}{q^{nk}}.\end{aligned}$$

Remark 5.3. The security bound given by Theorem 5.2 is unfortunately 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 Theorem 5.2 shows that it is reduced exponentially 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 at least 256 bits.

6 EVALUATION

In Table 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 [10]. 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 Kyber 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 3, 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 [10], 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 appear less drastic when compared to an instantiation of WebAuthn using the base post-quantum signature scheme. A more noticeable efficiency loss is seen when comparing the signing and verifying times of the blinded schemes with their base schemes, as given in Table 3. 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. 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 sum would not provide meaningful data, as the implementation of Kyber is highly optimized, while the implementations of the blinded signature schemes are non-optimized proofs of concept.

A notable difference between Dilithium-QROM and its blinded counterpart involves the matrix $\mathbf{A}$, which forms part of public keys.

Table 1: Credential, credential identifier, and response sizes for gCBR and pqCBR

Protocol Instantiation	|rc|	|rcid|	|rsp|
ECDSA-P256, HMAC-SHA256	64 B	80 B	64 B
Kyber, blCSI-FiSh	16 kB	1.06 kB	0.45 kB
Kyber, blDilithium-QROM	10 kB	1.06 kB	5.7 kB
Kyber, blLegRoast	0.5 kB	1.06 kB	11.22 kB

Table 2: Public key and signature sizes for post-quantum blinded and non-blinded signature schemes, from [10]

Scheme	|pk|	$\|\sigma\|$
CSI-FiSh	16 kB	0.45 kB
blCSI-FiSh	16 kB	0.45 kB
Dilithium-QROM	7.7 kB	5.7 kB
blDilithium-QROM	10 kB	5.7 kB
LegRoast	0.50 kB	7.94 kB
blLegRoast	0.50 kB	11.22 kB

Table 3: Performance of post-quantum blinded and non-blinded signature schemes, from [10]

.

Scheme	KeyGen / BlindPK	Sign	Verify
CSI-FiSh	10800 ms	554 ms	553 ms
blCSI-FiSh	10600 ms	546 ms	540 ms
Dilithium-QROM	3810 ms	9360 ms	2890 ms
blDilithium-QROM	1650 ms	28300 ms	717 ms
LegRoast	0.9 ms	12.4 ms	11.7 ms
blLegRoast	0.9 ms	18.6 ms	17.8 ms

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 [11]. 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 [10] 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 [20].

6.1 Stronger Security Notions

Although the definition of EUF-CMEA security given in [10] 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 keypairs 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.

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 recovery standard. We also devised a novel security model for account recovery and analyzed both Yubico's protocol and our own under it, introducing several new 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.

We are unable to prove the security of Yubico's recovery extension without relying on a bespoke computational assumption. In particular, we are unable to reduce its security to any well-studied Diffie–Hellman-like problem, including PRF-ODH in its various flavours. We recommend that the protocol be subjected to further rigorous analysis and modified if necessary .

6.2 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 account recovery, which places no such restriction on the user. Incorporating this feature would increase the complexity of our security model . 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.

6.3 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. Our security analysis could also be hardened by using an automated verification tool. 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 practical use cases. 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 classical and quantum-safe—would provide insight into how tightly bound our proposal is to a single algorithm. Also of interest would be a comparison of our approach with those taken by [5] and [12] with regards to security, usability, and ease of implementation. This could involve evaluating the other constructions under our security model and vice versa.

Another direction for future work would be to implement and benchmark our protocol. This would involve implementing extensions to the CTAP and WebAuthn APIs as well as crytographic code to create and use recovery credentials. The API extensions proposed by Yubico in [17] would suffice for our protocol. For the cryptographic code, one could leverage the blinded signature implementations from [10] and SandboxAQ's implementation of post-quantum FIDO2 using Firefox and a Nitrokey token [21]. A single-threaded Python implementation would also be of interest to compare with the existing proof-of-concept implementation of Yubico's group-based protocol [17].

ACKNOWLEDGMENTS

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2022-03187 and NSERC Alliance grant ALLRP 578463-22.

REFERENCES

[1] Sunpreet S. Arora, Saikrishna Badrinarayanan, Srinivasan Raghuraman, Maliheh Shirvanian, Kim Wagner, and Gaven Watson. 2022. Avoiding Lock Outs: Proactive

FIDO Account Recovery using Managerless Group Signatures. Cryptology ePrint Archive, Paper 2022/1555. 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. 2021. CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (version 3.02). https://pq-crystals.org/kyber/resources.shtml
[3] Manuel Barbosa, Alexandra Boldyreva, Shan Chen, and Bogdan Warinschi. 2021. Provable Security Analysis of FIDO2. In *CRYPTO 2021, Part III (LNCS, Vol. 12827)*, Tal Malkin and Chris Peikert (Eds.). Springer, Heidelberg, Virtual Event, 125–156. https://doi.org/10.1007/978-3-030-84252-9_5
[4] Nina Bindel, Cas Cremers, and Mang Zhao. 2023. FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1471–1490. https://doi.org/10.1109/SP46215.2023.10179454
[5] Jacqueline Brendel, Sebastian Clermont, and Marc Fischlin. 2023. Post-Quantum Asynchronous Remote Key Generation for FIDO2 Account Recovery. Cryptology ePrint Archive, Paper 2023/1275. https://eprint.iacr.org/2023/1275
[6] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. 2017. PRF-ODH: Relations, Instantiations, and Impossibility Results. In *CRYPTO 2017, Part III (LNCS, Vol. 10403)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, 651–681. https://doi.org/10.1007/978-3-319-63697-9_22
[7] Sofia Celi, Scott Griffy, Lucjan Hanzlik, Octavio Perez Kempner, and Daniel Slamanig. 2023. SoK: Signatures With Randomizable Keys. Cryptology ePrint Archive, Paper 2023/1524. https://eprint.iacr.org/2023/1524
[8] Morris Dworkin. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://doi.org/10.6028/NIST.FIPS.202
[9] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. 2023. Security Analysis of Signature Schemes with Key Blinding. Cryptology ePrint Archive, Paper 2023/380. https://eprint.iacr.org/2023/380
[10] Edward Eaton, Douglas Stebila, and Roy Stracovsky. 2021. Post-quantum Key-Blinding for Authentication in Anonymity Networks. In *LATINCRYPT 2021 (LNCS, Vol. 12912)*, Patrick Longa and Carla Ràfols (Eds.). Springer, Heidelberg, 67–87. https://doi.org/10.1007/978-3-030-88238-9_4
[11] Nick Frymann, Daniel Gardham, Franziskus Kiefer, Emil Lundberg, Mark Manulis, and Dain Nilsson. 2020. Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn. In *ACM CCS 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 939–954. https://doi.org/10.1145/3372297.3417292
[12] Nick Frymann, Daniel Gardham, and Mark Manulis. 2023. Asynchronous Remote Key Generation for Post-Quantum Cryptosystems from Lattices. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. 928–941. https://doi.org/10.1109/EuroSP57164.2023.00059
[13] Hidehito Gomi, Bill Leddy, and Dean H. Saxe (Eds.). 2019. *Recommended Account Recovery Practices for FIDO Relying Parties*. FIDO Alliance. https://fidoalliance.org/recommended-account-recovery-practices
[14] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. 2022. Anonymous, Robust Post-quantum Public Key Encryption. In *EUROCRYPT 2022, Part III (LNCS, Vol. 13277)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 402–432. https://doi.org/10.1007/978-3-031-07082-2_15
[15] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. 2023. Token meets Wallet: Formalizing Privacy and Revocation for FIDO2. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1491–1508. https://doi.org/10.1109/SP46215.2023.10179373
[16] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. 2012. On the Security of TLS-DHE in the Standard Model. In *CRYPTO 2012 (LNCS, Vol. 7417)*, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Springer, Heidelberg, 273–293. https://doi.org/10.1007/978-3-642-32009-5_17
[17] Emil Lundberg and Dain Nilsson. 2019. *WebAuthn Recovery Extension*. Yubico. https://github.com/Yubico/webauthn-recovery-extension
[18] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. 2020. Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. In *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 268–285. https://doi.org/10.1109/SP40000.2020.00047
[19] Varun Maram and Keita Xagawa. 2023. Post-quantum Anonymity of Kyber. In *PKC 2023, Part I (LNCS, Vol. 13940)*, Alexandra Boldyreva and Vladimir Kolesnikov (Eds.). Springer, Heidelberg, 3–35. https://doi.org/10.1007/978-3-031-31368-4_1
[20] Chris Peikert. 2020. He Gives C-Sieves on the CSIDH. In *EUROCRYPT 2020, Part II (LNCS, Vol. 12106)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, 463–492. https://doi.org/10.1007/978-3-030-45724-2_16
[21] SandboxAQ. 2024. PQC FIDO2. https://github.com/sandbox-quantum/pqc-fido2-impl
[22] Keita Xagawa. 2022. Anonymity of NIST PQC Round 3 KEMs. In *EUROCRYPT 2022, Part III (LNCS, Vol. 13277)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 551–581. https://doi.org/10.1007/978-3-031-07082-2_20

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

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 11: The ciphertext guessing game for KyberPKE

A PROOF OF THEOREM 5.2

To prove that Kyber satisfies KEM unlinkability, we begin by introducing a simpler problem, which we refer to as "ciphertext guessing", and showing that it is difficult for KyberPKE. We then reduce breaking the unlinkability of Kyber 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 11. 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.

Lemma A.1. *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*

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

Proof. We proceed by a sequence of games.

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

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

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

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

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 $2^{-h_{\mathsf{pk}}}$, where h_{pk} is the min-entropy of the public key. We note, as in Corollary 1 of [10], 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

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

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

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

The desired security statement follows from combining these bounds. □

We now reduce the KEM-UL experiment for Kyber to ciphertext guessing, proving Theorem 5.2. We additionally rely on the SPR-CCA security of Kyber, as evaluated in [19], and the pseudorandomness of SHAKE-256.

Proof of Theorem 5.2. We proceed by a sequence of games.

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

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

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 Kyber: 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,

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

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 SHAKE-256, where we regard the random prefix z as the key and $\mathsf{H}(c)$ as the label. Therefore

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

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 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

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

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

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

The desired security statement follows from combining these bounds. □

B PROOF OF THEOREM 4.1

Proof of Theorem 4.1. We begin by arguing that the first winning condition, on line 3 of the recovery authentication experiment in Figure 6, 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, $\pi_{P_1}^{i_1}$.status = accept = $\pi_{P_2}^{i_2}$. Session identifiers are also only set on sessions with set roles, so both $\pi_{P_1}^{i_1}$ and $\pi_{P_2}^{i_2}$ have set roles.

If $\pi_{P_1}^{i_1}$.role = server = $\pi_{P_2}^{i_2}$, then $\pi_{P_1}^{i_1}$.selfid = $\pi_{P_2}^{i_2}$.selfid and $\pi_{P_1}^{i_1}$.peerid = $\pi_{P_2}^{i_2}$.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 $\pi_{P_1}^{i_1}$.role $\neq$ $\pi_{P_2}^{i_2}$.role. It follows by equality of session identifiers that $\pi_{P_1}^{i_1}$.selfid = $\pi_{P_2}^{i_2}$.peerid and $\pi_{P_1}^{i_1}$.peerid = $\pi_{P_2}^{i_2}$.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 $\mathsf{Exp}^{\mathrm{1rec}}_{\mathrm{gCBR},\mathcal{A}}$. Therefore

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

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,

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

Game 2. This game is identical to Game 1 except that the game aborts if two E-values generated by calls to CredGen collide. Since CredGen is only called by 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

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

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

- The derived value mk in Response and CredGen is replaced by randomly sampling a key for Σ.
- The derived value p in Response is replaced by randomly sampling an element of $\mathbb{Z}_q$.
- The computation of P in 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 sgPRF security of KDF; hence,

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

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 π_S^i.role = server, π_S^i.status = accept and for which there is no (U, j) such that π_U^j matches with π_S^i 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 $\pi_{S^*}^i$ 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 $\pi_{S^*}^i$.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 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 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 UserComplete with E^* as input, it uses its verification oracle. Note, however, that since E^* values are unique for each call to 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. □

C PROOF OF THEOREM 4.2

Proof of Theorem 4.2. An identical argument to that given in the proof of Theorem 4.1 shows that the winning condition on line 3 of the recovery authentication experiment in Figure 6 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 $\mathsf{Exp}^{\mathsf{rec}}_{\mathsf{pqCBR},\mathcal{A}}$. Therefore

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

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

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

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

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

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 UserComplete for (U^*, j), which calls Response to produce a signature. By the second abort condition, τ^* is only produced in 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. □

D PROOF OF THEOREM 4.3

Proof of Theorem 4.3. 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 $\mathsf{Exp}^{\text{I-UL}}_{\text{gCBR},\mathcal{A}}$, so

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

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 4.1,

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

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 CredGen or Response via one of $\mathsf{Register}(U,\cdot)$, $\mathsf{Register}_b(U,\cdot)$, $\mathsf{Action}_b(U,\cdot)$, 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 sgPRF security of KDF; hence,

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

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 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 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 Register or 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 Register or 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 Register or UserComplete. It follows that

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

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 the tuple of identifiers $(\pi^i_{U_0^*}.\mathsf{peerid}, \pi^i_{U_0^*}.\mathsf{selfid}, \pi^i_{U_0^*}.\mathsf{sid.rcid})$ does not correspond to some call to ORegister for U_0^*.

We claim that this abort condition is never triggered. By the abort condition from the previous game, this can only occur if the tuple $(\pi^i_{U_0^*}.\mathsf{peerid}, \pi^i_{U_0^*}.\mathsf{selfid}, \pi^i_{U_0^*}.\mathsf{sid.rcid})$ corresponds to a call to ORegister for either U_0 or U_1; without loss of generality, say U_0. It follows that $(U_0, S, \pi_{U_0^*}.\mathsf{selfid}) \in \mathcal{L}_{\text{Register}}$, 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 $(U_0^*, S, \pi_{U_0^*}.\mathsf{selfid}) \notin \mathcal{L}_{\text{Register}}$. It follows that the failing condition on line 3 of CheckFail is triggered, which contradicts our assumption that $\mathcal{A}$ never triggers a failing condition. Therefore

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

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 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 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 ORegister for U_1.

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

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

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 the user ID st_U.uid is no longer updated or accessed. Instead, the protocol action UserBegin(π_U^i, uid, serverID, st_U) rejects if $(U, S, \mathsf{uid}) \notin \mathcal{L}_{\mathsf{Register}}$, where S is the server with identifier serverID. The previous abort condition ensures that there is no difference in behaviour from Game 5, so

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

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

- Data stored in long-term state in calls to $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}_b(U_0, \cdot)$ is flagged.
- The game aborts if a call to $\mathsf{ORegister}(U_0^*, \cdot)$, $\mathsf{Register}_b(U_0^*, \cdot)$, $\mathsf{Action}(U_0^*, \cdot)$, or $\mathsf{Action}_b(U_0^*, \cdot)$ accesses flagged data.
- The game aborts if a call to $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, 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 Register or queried to UserComplete for U,
- a mapping serverID $\mapsto \{\mathsf{nc}\}$, the set of nc-values previously issued for serverID.

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

If E is provided as input to UserComplete for U_0^*, then by the previous abort conditions, it must have been generated in some call to ORegister for U_0^*. Since E-values do not collide at registration, it cannot have been generated in some call to ORegister for U_0. Hence, it also cannot have been queried to UserComplete for U_0. Therefore E is not flagged. similarly, if st_U.nc[serverID] is looked up in UserComplete for U_0^*, then U_0^* must be registered at the server with identifier 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

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

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

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

A similar argument to the previous game shows that

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

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 $\mathrm{st}_{U_{1-b}}$. Since only flagged data are stored in st_{U_b} and $\mathrm{st}_{U_{1-b}}$ and only non-flagged data is stored in $\mathrm{st}_{U_0^*}$ and $\mathrm{st}_{U_1^*}$, there is no detectable difference between Games 8 and Games 9. Therefore

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

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

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

The desired security statement follows from combining these bounds. □

E PROOF OF THEOREM 4.4

Proof of Theorem 4.4. As in Theorem 4.3, 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 $\mathsf{Exp}^{\mathrm{UL}}_{\mathrm{pqCBR},\mathcal{A}}$, so

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

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 Π.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 Π.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

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

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 $\mathsf{Adv}^{\mathrm{prf}}_{\mathrm{PRF},\mathcal{B}^i}(\lambda)$, 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

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

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 4.1 that

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

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 the $\Delta.\mathsf{KeyGen}$ routine 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 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

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

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

As in the proof of Theorem 4.3, 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 the user ID $\mathsf{st}_U.\mathsf{uid}$ is no longer updated or accessed. Instead, the protocol action $\mathsf{UserBegin}(\pi^i_U, \mathsf{uid}, \mathsf{serverID}, \mathsf{st}_U)$ rejects if $(U, S, \mathsf{uid}) \notin \mathcal{L}_{\mathsf{Register}}$, where S is the server with identifier serverID.

For users other than U_d or U^*_d for $d \in \{0, 1\}$, there is no difference in behaviour. Note, however, that 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 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 $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, or $\mathsf{Action}(U_0, \cdot)$ is flagged.
- The game aborts if a call to $\mathsf{ORegister}(U^*_0, \cdot)$, $\mathsf{Register}_b(U^*_0, \cdot)$, $\mathsf{Action}(U^*_0, \cdot)$, or $\mathsf{Action}(U^*_0, \cdot)$ accesses flagged data.
- The game aborts if a call to $\mathsf{ORegister}(U_0, \cdot)$, $\mathsf{Register}_b(U_0, \cdot)$, $\mathsf{Action}(U_0, \cdot)$, 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 Register or queried to UserComplete for U,
- a mapping $\mathsf{aux} \mapsto \{\mathsf{nc}\}$, the set of nc-values previously issued for serverID.

We begin by showing that access of flagged data for U^*_0 never occurs. Both mappings are accessed only in UserComplete.

If (c, aux) is looked up in the mapping in UserComplete for U^*_0, then $\mathsf{aux} = (\mathsf{serverID}, \mathsf{uid})$, where U^*_0 is registered at the server with identifier 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 $\mathsf{ORegister}(U_1, \cdot)$, $\mathsf{Register}_b(U_1, \cdot)$, $\mathsf{Action}(U_1, \cdot)$, or $\mathsf{Action}(U_1, \cdot)$ is also flagged.
- The game aborts if a call to $\mathsf{ORegister}(U^*_d, \cdot)$, $\mathsf{Register}_b(U^*_d, \cdot)$, $\mathsf{Action}(U^*_d, \cdot)$, or $\mathsf{Action}(U^*_d, \cdot)$ for $d \in \{0, 1\}$ accesses flagged data.
- The game aborts if a call to $\mathsf{ORegister}(U_d, \cdot)$, $\mathsf{Register}_b(U_d, \cdot)$, $\mathsf{Action}(U_d, \cdot)$, 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. □