Revisiting FIDO2 Privacy: Toward Stronger Unlinkability with Global Revocation

***Abstract*—The FIDO2 standard, which replaces passwords with cryptographic credentials and enables hardware-token-based logins, is now widely deployed. Two properties are essential for privacy and key management: unlinkability, which prevents services from linking the same token across accounts, and global revocation, which allows users to invalidate all credentials from a lost or compromised token. Hanzlik et al. (S&P'23) were the first to formalize these jointly as *unlinkability with global revocation* and to realize them in their bip32PA scheme. However, in practice, bip32PA falls short: 1) Its per-service deterministic credential identifiers let an adversary link a token by observing multiple registrations – a common scenario when users enroll the same token under different accounts. 2) Its security model prohibits submitting a challenged registration credential identifier to the authentication oracle, a restriction that rules out realistic attack vectors such as physical token access or client-side probing.**

In this paper, we address the above limitations with three main contributions: 1) We introduce a *chain-based recoverable nonce mechanism* that enables unique per-registration credentials while supporting global revocation, yielding two new schemes: bip32PA-MU and bip32PA-SU, which achieve medium and strong unlinkability, respectively, under existing models. 2) We *refine the unlinkability definition* to better reflect realistic attack surfaces in FIDO2 deployments, and introduce three enhanced variants – bip32PA$^+$, bip32PA-MU$^+$, and bip32PA-SU$^+$ – each achieving weak, medium, and strong unlinkability, under the refined model. 3) We implement all five variants in a modified FIDO2 codebase (abut 8,000 LOC), demonstrating seamless compatibility and efficiency. For instance, bip32PA-MU$^+$ reduces online authentication latency by 3.3× (from 43.4 ms to 13.2 ms) compared to bip32PA, with no additional communication overhead.

1. Introduction

Online authentication remains a critical security challenge because it directly affects account protection and service provider trust. To mitigate the risks of password-based authentication, the Fast IDentity Online (FIDO) Alliance introduced FIDO2, an authentication framework that leverages public-key cryptography to enhance both security and usability [1]. FIDO2 consists of two key components: i) *WebAuthn* (W3C's Web Authentication), which enables users to register a credential – essentially a public verification key – with a server (Relying Party), binding it to their account. Authentication follows a challenge-response model, where a token (Authenticator) proves possession of the corresponding secret key. WebAuthn also supports attestation, which provides proof that the credential originated from a trusted device. ii) *CTAP2* (Client-to-Authenticator Protocol version 2.x), which facilitates secure communication between a client (e.g., browser or OS) and a token. CTAP2 requires the user to verify their identity – by entering a PIN, scanning a fingerprint, or pressing a button – before the token produces any signature. By combining WebAuthn and CTAP2, FIDO2 prevents phishing and credential theft and substantially strengthens authentication security [2].

Despite the widespread adoption of FIDO2, its privacy and key management mechanisms remain an area of ongoing research. Two crucial properties – *unlinkability* and *global revocation* – were only recently formalized by Hanzlik et al. [3]. *Unlinkability* ensures that interactions involving the same token cannot be linked across different sessions or registrations, thereby preserving user privacy. Meanwhile, *global revocation* addresses the challenge of key management: in traditional FIDO2 deployments, if a user loses their authentication token, they must manually revoke each key with individual servers, which is impractical. Global revocation enables users to revoke all associated keys simultaneously, which significantly reduces the overhead of managing lost or compromised credentials.

Unlinkability with Global Revocation. To capture the interplay between privacy and key management in FIDO2, Hanzlik et al. [3] introduced the notion of *unlinkability with global revocation*. Their model evaluates whether an adversary $\mathcal{A}$ can distinguish between two tokens (T_0 and T_1) by interacting with different system components. Specifically, the model grants $\mathcal{A}$ access to four main oracles: 1) Challenge, which generates registration responses – including a credential identifier cid, a credential pk, and a registration signature σ_r – or authentication responses, which contain an authentication signature σ_a. 2) Revoke, which generates revocation keys for all tokens except T_0 and T_1. 3) Left, which simulates the behavior of token T_b (where $b \in \{0, 1\}$) by querying Challenge. 4) Right, which simulates the other token, T_{1-b}, in a similar manner.

The adversary's goal is to determine which token a given registration or authentication response originates from. To reflect different attacker capabilities, Hanzlik et al. define three levels of unlinkability with progressively stricter restrictions: 1) **Strong Unlinkability** prevents *trivial linkage attacks* [4], [5], where an adversary links a registration to a later authentication by reusing the credential identifier cid seen in the registration. Specifically, if $\mathcal{A}$ obtains a registration response (cid, pk, σ_r) from Left, it is prohibited from querying Challenge with that same cid to obtain an

authentication response from the same token. 2) **Medium Unlinkability** enforces a stricter condition by prohibiting $\mathcal{A}$ from querying any cid to Challenge for authentication responses. 3) **Weak Unlinkability** imposes the strongest restriction that prevents $\mathcal{A}$ from querying any cid to Challenge for both registration and authentication responses. To achieve unlinkability in the global revocation setting, Hanzlik et al. propose bip32PA [3], a FIDO2 variant that employs deterministic key derivation (e.g., BIP32, widely used in Bitcoin [6]). However, their scheme still has unresolved concerns.

Insufficient unlinkability in practice. Although bip32PA is proven to achieve weak unlinkability under its security model, it remains vulnerable in real-world deployments. If an adversary registers the same token with a server more than once, they can trivially link the registrations and break unlinkability. This vulnerability is realistic in practical FIDO2 deployments, where users often register the same token under multiple accounts on the same service. These practical considerations motivate the need for FIDO2 variants that offer stronger unlinkability – preventing token linkage across multiple registrations – while still preserving support for global revocation.

Limitations of existing definitions. As noted above, to avoid trivial linkage, existing unlinkability definitions [4] [5] prohibit the adversary from querying the challenged registration credential identifier cid to the authentication oracle. However, this restriction excludes realistic attack vectors observed in practical FIDO2 deployments, such as: 1) **Physical Token Access**: In enterprise environments, shared workstations, or scenarios involving temporarily lost tokens, an adversary may gain brief access to a token and attempt authentication using a known cid [7], and 2) **Client-side Probing or Malware**: In practice, tokens are often plugged into shared or multipurpose client devices (e.g., public terminals or enterprise desktops). A malicious application on such a device (e.g., a browser extension or background process) can communicate with the token via USB, NFC, or BLE, reuse a known cid from a legitimate registration, and attempt to probe the token by sending custom authentication challenges [8]. This gap between theoretical models and practical threats motivates a refined unlinkability definition – one that more accurately captures these real-world attack surfaces without over-restricting adversarial capabilities.

Our Contribution. The central challenge lies in strengthening unlinkability under global revocation, while preserving full compatibility with existing FIDO2 deployments. Although prior works [1], [5] have improved unlinkability by incorporating the CTAP2 component, they overlook two critical factors: the inherent limitations of WebAuthn (as discussed above) and the necessity of supporting global revocation. Our work builds upon [3], which studies unlinkability for non-resident keys – where tokens store only a symmetric key and derive signing keys on demand, without attestation – under global revocation. We extend this line of work by refining the unlinkability model and designing enhanced schemes that achieve stronger, more practical unlinkability guarantees. Our main contributions are as follows:

- We introduce a *chain-based recoverable nonce mechanism*, where each nonce is derived from its predecessor and can be recovered in reverse. By integrating this mechanism into key derivation, we ensure that each registration yields a unique public key, thereby addressing weak unlinkability. The latest nonce enables the recovery of previously derived public keys, which supports global revocation. We construct credential identifiers in two ways, resulting in two bip32PA variants – bip32PA-MU and bip32PA-SU – that achieve medium and strong unlinkability, respectively, under the existing security model [3]. Both variants remain fully compatible with current FIDO2 server implementations, with only minimal additional storage for the latest nonce within the token.
- We refine unlinkability to better reflect real-world adversaries by permitting queries of the challenged credential identifier cid to Challenge, under the strict condition that the authentication message M_a overlaps with the registration response only in cid. This models realistic attacks in which an adversary may temporarily access a user's token or interact through a compromised client, yet still lacks the secret material needed to generate valid responses. Crucially, only a legitimate token owner, possessing the correct secret-derived data, can produce valid authentication responses for the challenged cid, which makes this refinement both realistic and secure.
- To support our refined model, we introduce a *verifiable pre-signature mechanism*, enabling tokens to generate pre-signatures for future authentication rounds. These pre-signatures can be securely stored on the client and later verified by the token to reconstruct valid signatures. Using this mechanism, we propose three enhanced variants – bip32PA$^+$, bip32PA-MU$^+$, and bip32PA-SU$^+$ – each achieving weak, medium, and strong unlinkability under the refined security model. While bip32PA$^+$ improves upon bip32PA, it remains vulnerable to multiple-registration attacks; we include it for completeness to cover all three unlinkability levels in the refined model.
- We integrated all five proposed variants into an existing FIDO2 codebase modifying 8,000 LOC - to demonstrate their practical feasibility and seamless compatibility with deployed FIDO2 systems. Our benchmarks show that, relative to the original bip32PA, the bip32PA-MU$^+$ variant alone reduces online authentication cost by over 3.3× (from 43.4 ms down to 13.2 ms) while preserving identical communication overhead between the client and server. The sole performance trade-off is that, aside from bip32PA$^+$, the chain-based revocation variants incur somewhat higher revocation-check latency. This overhead is justified by their substantially stronger unlinkability guarantees.

Related Work. We briefly review related work on the unlinkability and revocation of FIDO2.

Unlinkability in FIDO2. The first formal model for FIDO2 was introduced by Barbosa et al. [2], which analyzed passwordless authentication using locally keys stored in the token. Their work highlighted weaknesses in PIN-based

access control and proposed a password-authenticated key agreement (PAKE) as an alternative. Bindel, Cremers, and Zhao [1] extended this model to align with CTAP 2.1, which emphasizes token binding and user verification. However, both models assume permanent key storage on tokens and lack formal privacy definitions for cross-server unlinkability of the token-generated credentials.

Privacy concerns in FIDO2 were first identified by Guirat et al. [9], who showed that attestation keys could link tokens across services, and by Kepkowski et al. [4], who demonstrated timing attacks enabling malicious servers to track users across different relying parties. These findings underscored the need for stronger unlinkability guarantees. Hanzlik et al. [10] addressed this by formalizing unlinkability for WebAuthn in the context of non-resident keys, where tokens derive signing keys on demand rather than storing them. Their model ensures that registrations do not reveal whether they originate from the same token, even under global revocation. However, their scheme only achieves weak unlinkability in this setting. Moreover, their definition assumes that adversaries cannot query the challenged registration credential identifier for authentication responses – an assumption that does not align with real-world FIDO2 deployments and excludes attack vectors such as physical token access and client-side probing or malware.

Recently, Barbosa et al. [5] revisited the unlinkability and security analysis of FIDO2, incorporating both WebAuthn and CTAP2. Their work introduced refined security models to address previous gaps and established the first formal privacy guarantees for FIDO2 as a whole. However, their model retains the insufficiency of Hanzlik et al.'s unlinkability definition within the WebAuthn component and fails to incorporate global revocation. This leaves open challenges in achieving stronger unlinkability against practical adversarial conditions.

Revocation in FIDO2. Unlike traditional authentication systems, FIDO2 lacks a centralized revocation mechanism and relies instead on users to manually delete credentials. Grassi et al. [11] criticized this approach for its scalability challenges, particularly when users lose their authentication devices. While cryptographic revocation techniques such as Camenisch-Lysyanskaya accumulators [12] and Boneh-Shacham verifier-local revocation [13] provide both privacy protection and efficient revocation, they require substantial modifications to FIDO2's architecture. Hanzlik et al. [3] addressed this limitation by introducing a formal model for global revocation in the context of non-resident keys. Their approach ensures that all credentials derived from a compromised token can be revoked without compromising unlinkability. However, their solution still suffers from weak unlinkability guarantees in the global revocation setting. Unlike blocklisting-based methods (e.g., [14]), which risk exposing token identities, their model maintains user privacy during revocation checks. Their work complements other privacy-preserving identity schemes, such as Dauterman et al.'s VIF-based identities [15], which lack built-in revocation, and Frymann et al.'s token backup protocols [16], which focus on credential recovery rather than revocation. By integrating unlinkability-preserving revocation mechanisms into WebAuthn, these efforts contribute to the broader goal of enhancing security and privacy in FIDO2 authentication.

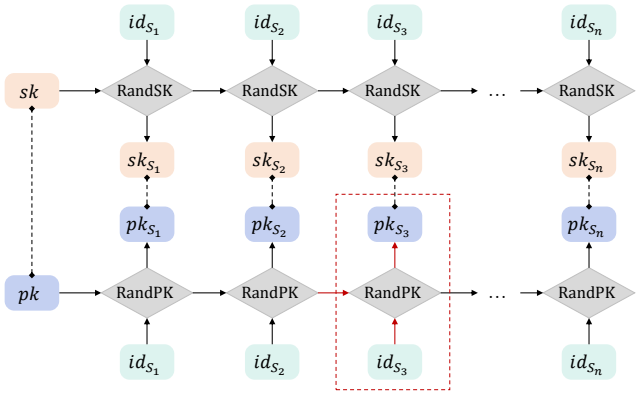

Figure 1: Brief review of key derivation and revocation in bip32PA [3]. Each registration of a token on the same server id_{S_i} is associated with the same secret/public key pair (sk_{S_i}, pk_{S_i}). The procedure highlighted by the red arrow and red dotted box shows how the server id_{S_3} regenerates and revokes the public key pk_{S_3} using the token's published pk and $chain$. Note that the private $chain$ is required for invoking RandSK and RandPK; for simplicity, it is omitted here.

2. Solution Overview

Review of bip32PA [3]. bip32PA uses deterministic key derivation (i.e., the BIP32 standard) to support global revocation. As shown in Figure 1, it starts with an ECDSA master key pair – consisting of a master secret key sk and a master public key $pk = g^{sk}$ (where g is the base point of an elliptic curve) – and a chaincode $chain$ (a random seed) to derive new key pairs. For a specific identity id_{S_i}, BIP32 defines two functions: RandSK, which computes $\rho := \mathsf{H}(id_{S_i}, chain, pk)$ and derives the secret key as $sk_{S_i} := sk + \rho$ (where H is a secure hash function); and RandPK, which computes $\rho := \mathsf{H}(id_{S_i}, chain, pk)$ and derives the public key as $pk_{S_i} := pk \cdot g^{\rho}$. This mechanism permits efficient global revocation by issuing pk and $chain$, and all derived public keys can be recomputed and revoked. However, since each registration on the same server results in the same public key, bip32PA achieves only weak unlinkability. In particular, given a challenged registration for token T_b, an adversary can re-register both tokens and compare the public keys to determine the correct bit b.

Stronger Unlinkability with Global Revocation. A natural way to improve unlinkability is to ensure that each registration produces a unique public key. This allows the token to hold distinct public keys even when registering with the same server. Introducing a random nonce into the invocations of RandSK and RandPK can ensure that each registration yields a fresh key pair. The nonce can be organized as a credential identifier cid, later used to regenerate the secret key during authentication. Nonetheless, directly adding a nonce requires modifications to the token protocol and

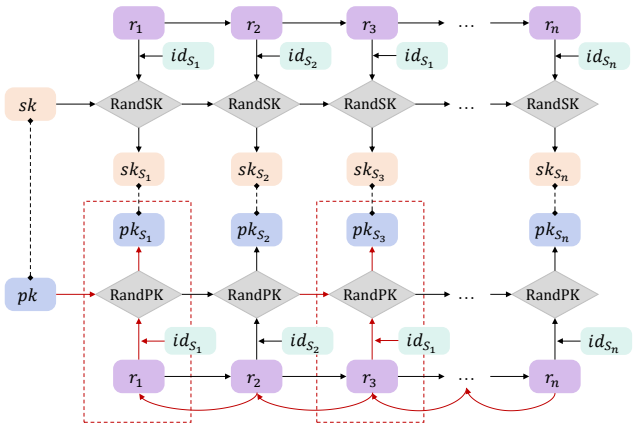

Figure 2: Overview of improved key derivation and revocation. Our design introduces a chain-based recoverable nonce design, where r_i is generated from r_{i-1} via encryption and recovered via decryption. In the revocation procedure (highlighted by the red arrow and red dotted box), the server id_{S_1} regenerates and revokes the token's registered public keys pk_{S_1} and pk_{S_3} using the published pk, $chain$, and the latest nonce r_n. Here, we also omit $chain$ during invoking RandSK and RandPK.

may disturb existing FIDO2 server implementations. Moreover, supporting global revocation necessitates a significant amount of nonces. To address these challenges, we draw inspiration from the output-feedback (OFB) mode in block ciphers [17] and propose a chain-based recoverable nonce design, where each nonce is derived from its predecessor (e.g., $r_i := \mathsf{Enc}(chain, r_{i-1})$), allowing the token to store only the latest nonce. During global revocation, disclosing the latest nonce r_n together with pk and $chain$ allows recovery of the previous nonces via $r_{i-1} := \mathsf{Dec}(chain, r_i)$; from these, the corresponding public keys (e.g., pk_{S_1} and pk_{S_3} in Figure 2) can be derived and revoked.

In our designs, cid is constructed in two ways to support global revocation, where $r_i := \mathsf{Enc}(chain, r_{i-1})$, and $seed$ forms part of the master secret key:

- bip32PA-MU achieves medium unlinkability by directly setting the nonce as cid, namely, $cid := r_i$.
- bip32PA-SU achieves strong unlinkability by additionally linking cid to id_S, namely, $cid := \mathsf{Enc}(seed, (id_S, r_i))$.

Revisiting Unlinkability. FIDO2 is vulnerable to a trivial linkage attack, formalized by Hanzlik et al. [3]. In this attack, an adversary $\mathcal{A}$ obtains a challenged registration response $(cid, R_r := (pk, \sigma_r))$ for token T_b and server id_{S_L}, then queries the authentication oracle with the same cid and any message M_a to receive $R_a := \sigma_a$. By checking σ_a against pk, $\mathcal{A}$ links the sessions, which violates unlinkability. To prevent this, Hanzlik et al. [3] disallow any authentication query involving the challenged cid, effectively blocking reuse of cid after registration. However, this restriction excludes realistic attack vectors – such as physical token access and client-side probing or malware – that occur in real-world FIDO2 deployments.

To better reflect practical usage and realistic adversaries, we relax this restriction. Our refined unlinkability model now permits authentication queries on the challenged cid, provided the registration and authentication messages share no information beyond cid. In practice, only a party that possesses both the token and the correct authentication material (e.g., the honest client's pre-signature in our design) can produce a valid response for a given cid. This accurately captures scenarios where an attacker temporarily gains access to a token but lacks the client's secrets – for if both token and client are compromised, unlinkability is already lost. To support this model, we revise the structure of credential lists and introduce a refined notion of credential separation. Formal definitions appear in Section 5.

Our Proposals with Redefined Unlinkability. One potential solution is requiring the adversary to submit a correct authentication message for a valid signature. Our first attempt involves pre-selecting the next authentication message in both registration and authentication phases. In this design, the registration message M_r is replaced with $\widehat{M}_r := (M_r, M_a)$, where M_r and M_a are the current registration message and next-round authentication message, respectively and the authentication message becomes $\widehat{M}_a := (M_a, M'_a)$, where M_a and M'_a represent the current and next-round authentication messages, respectively. The token generates and stores the signature for the next authentication round, allowing it to return the signature if the queried authentication message is correct. This modification permits the adversary to query the challenged cid in the authentication oracle, with the restriction that the queried message M_a must differ from the challenged one. As a result, our model more closely captures real-world threat scenarios without unduly constraining the adversary.

Nonetheless, this approach requires modifications to the token protocol, as it mandates storing the signature for the next-round authentication for each server. This could be infeasible given the token's limited storage capacity, especially when it must support multiple servers. An alternative is to offload the storage of pre-computed signatures to the client. In this approach, the token computes a ciphertext $\widehat{\sigma}_a := \widehat{\mathsf{H}}(lk) \oplus \sigma_a$ as the pre-signature, where the locking key lk is derived from msk, cid, id_S, and M_a. The client then stores this pre-signature and submits $(M_a, \widehat{\sigma}_a, M'_a)$ during authentication. To recover σ_a, the token derives lk from these values and concurrently pre-computes the pre-signature for the next round. This mechanism is similar to the password management features in browsers like Google Chrome and Microsoft Edge, with the only additional cost being the storage of a pre-signature, which, while slightly larger than a password, remains an acceptable trade-off.

A remaining challenge is that the current approach does not yet satisfy the redefined unlinkability. An adversary might submit cid with a crafted authentication message $\widehat{M}_a := (M_a, \sigma^*_a, M'_a)$ to the authentication oracle, where σ^*_a is randomly chosen value serving as a pre-signature. Even if the adversary cannot obtain the valid signature for M_a, it will receive a correct next-round pre-signature $\widehat{\sigma}'_a$ and then further query the authentication oracle to derive a valid signature σ'_a for M'_a, thereby linking the interactions. To

prevent this, we bind the authentication message with the pre-signature to obtain a *verifiable pre-signature*, namely, setting $\widehat{\sigma}_a := \widehat{\mathsf{H}}(lk) \oplus (M_a||\sigma_a)$. Consequently, if an adversary submits an invalid pre-signature (i.e., one that does not correctly incorporate M_a), the token will discard the request. With these mechanisms, we construct three variants: bip32PA$^+$, bip32PA-MU$^+$, and bip32PA-SU$^+$.

3. Preliminaries

Notations. We denote $y \leftarrow \mathsf{A}(x)$ as executing an algorithm A via inputting x to obtain the output y, and $y := \mathsf{A}(x;\gamma)$ as the algorithm A with an explicit randomness γ. We also let $y \in \mathsf{A}(x)$ denote that y is a possible output of A on input x. Let $x \in S$ be choosing a randomness x uniformly from the set S, $[n]$ be the set $\{1,\ldots,n\}$, and λ be the system security parameter. In the whole paper, we suppose that pp is an implicit input public parameter of all algorithms. The digital signatures are involved in terms of standard and rerandomizable, and we let $\mathbb{M}$ and $\mathbb{S}$ be the message and signature set, respectively. The symmetric-key encryption is assumed to be authenticated and anonymous by default.

We consider three types of parties $\mathcal{P} = \mathcal{T} \cup \mathcal{C} \cup \mathcal{S}$, where $(\mathcal{T},\mathcal{C},\mathcal{S})$ represents the set of tokens, clients and servers, respectively. Each client is typically equipped with a token. If required, each server $S \in \mathcal{S}$ and each client $C \in \mathcal{C}$ maintain a registration context $\mathbf{rcs}_S$ and a pre-signature context $\mathbf{pcs}_C$ as internal state respectively, both of them are initially empty key-value tables. In $\mathbf{rcs}_S$, a mapping from the credential identifier cid to the public key pk and current authentication challenge rs_a, while in $\mathbf{pcs}_C$, a mapping from the credential identifier cid to the current pre-signature. Each token $T \in \mathcal{T}$ maintains a fixed state, initialized once with a key msk_T, which remains unchanged. Like [3], we also assume each server has a unique identifier id_S, corresponding to a URL in practice. In all defined experiments, these identifiers are assumed to be known by all parties. The initial interaction between users and servers is not explicitly modeled, as it may vary by use case. Consequently, we assume the server already has the user's account information, including their cid.

We defer the formal definition of Passwordless Authentication (PLA) to Appendix A, and provide a brief overview here. Figure 3 also illustrates the syntax for reference. A PLA scheme with global revocation is a tuple PLA $=$ $(\mathsf{Gen},\mathsf{Reg},\mathsf{Auth},\mathsf{GRev})$, where $\mathsf{Gen}(pp)$ generates a master secret key msk stored on the token. The registration protocol Reg proceeds as follows: 1) The server runs $(c, st) \leftarrow \mathsf{rchall}(id_S)$. 2) The client computes $M_r \leftarrow \mathsf{rcomm}(id_S, c)$. 3) The token computes $(cid, R_r) \leftarrow \mathsf{rresp}(msk, id_S, M_r)$, generating the credential identifier cid. 4) The server verifies it by computing $(b_r, cred) \leftarrow \mathsf{rcheck}(st, cid, R_r)$ and stores the credential as $\mathbf{rcs}[cid] := cred$. The authentication protocol Auth follows a similar structure, except the token algorithm aresp takes cid as input rather than generating it, and the server algorithm acheck outputs only a decision bit. The global revocation protocol GRev consists of: 1) The token runs $rk \leftarrow \mathsf{Revoke}(msk)$. 2) The server checks a revoking credential by running $b \leftarrow \mathsf{CheckCred}(id_S, cred, rk)$.

4. Our Proposals with Stronger Unlinkability

This section introduces our two variants, bip32PA-MU and bip32PA-SU, both of which adopt the BIP32 design from bip32PA [3]. We illustrate our construction using ECDSA as an example, though any signature scheme compatible with BIP32 can be used. We first describe the core mechanisms of each variant; the complete workflows are shown in Figure 3. For the security proofs, we model two hash functions as random oracles: $\mathsf{H}_0 : \{0,1\}^* \to \{0,1\}^{2\lambda}$ and $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$.

Key Generation. The key consists of an ECDSA key pair (sk_0, pk_0), a chaincode ch, a seed $seed$, and a latest revocation information $lrev$. The chaincode ch, public key pk_0 and latest revocation information $lrev$ enable key revocation, while $seed$ is used to derive randomness for signing in both bip32PA-MU and bip32PA-SU (also for the encryption/decryption of cid in bip32PA-SU). Specifically, the components are generated as $sk_0 \leftarrow \mathbb{Z}_p$, $pk_0 := g^{sk_0}$, $ch \leftarrow \{0,1\}^{\lambda}$, $seed \leftarrow \{0,1\}^{\lambda}$, and $lrev$ is initialized as a bitstring of $\lambda - 1$ zeros followed by a single 1 (i.e., $\{0\}^{\lambda-1}\|1$). Finally, the master secret key is defined as $msk := (sk_0,\ pk_0, ch, seed, lrev)$.

Registration. This protocol follows the WebAuthn specification and closely aligns with bip32PA reviewed in Appendix B. The critical distinction in both bip32PA-MU and bip32PA-SU lies in how keys are derived. The token organizes their credential identifier cid in the different variants.

- For bip32PA-MU, the token computes $cid := \mathsf{Enc}(ch, lrev)$, and lets $r := lrev := cid$.
- For bip32PA-SU, the token first computes $r := \mathsf{Enc}(ch, lrev)$, $lrev := r$, and then generates $cid := \mathsf{Enc}(seed, (id_S, r))$.

Then, in both bip32PA-MU and bip32PA-SU, the token uses $\rho := \mathsf{H}_1(pk_0, ch, r, id_S)$ to rerandomize the initial key pair (sk_0, pk_0) into (sk, pk) via $sk := \mathsf{RandSK}(sk_0, \rho)$ and $pk := \mathsf{RandPK}(pk_0, \rho)$. The message and random coin are prepared as $m := (\mathsf{H}_0(id_S), cid, pk, M_r)$ and $coins := \mathsf{H}_1(seed, m)$ respectively. Here, $coins$ is also used to derandomize the signing process, which is for the security in the BIP32 setting [18]. Finally, the token generates the signature $\sigma := \mathsf{Sig}(sk, m; coins)$.

Authentication. In this phase, the server is supposed to know the credential identifier cid of interest. One example to find cid is that after a user inputs its username, the server could find the cid from the mapping between usernames to credential identifiers. The token first recovers the randomness r. For bip32PA-MU, $r := cid$. For bip32PA-SU, the token invokes $(id, r) := \mathsf{Dec}(seed, cid)$ and aborts if $id \neq id_S$. Then the token uses r to regenerate the signing key $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, r, id_S))$, and signs the authentication challenge $\sigma := \mathsf{Sig}(sk, m; coins)$ where $m := (\mathsf{H}_0(id_S), M_a), coins := \mathsf{H}_1(seed, m)$.

Global Revocation. Both bip32PA-MU and bip32PA-SU can support the global revocation. Both of them in-

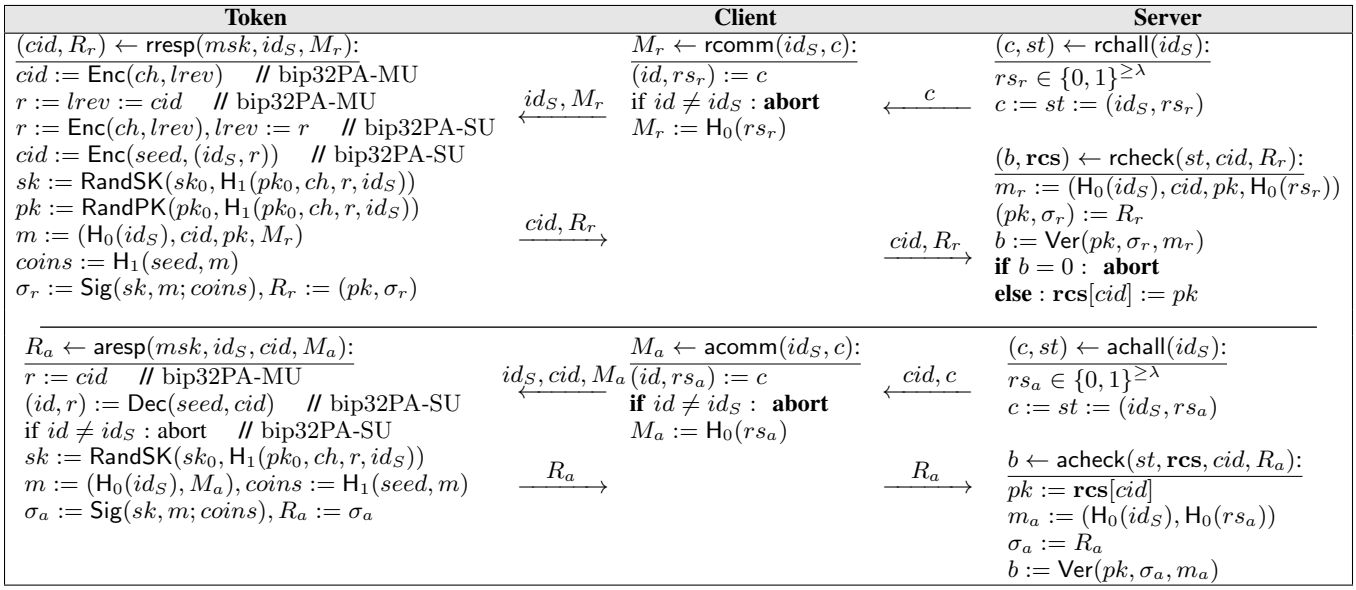

Figure 3: The optimized WebAuthn registration (top) and authentication (bottom) protocol for bip32PA-MU and bip32PA-SU. Annotated statements are executed only in the specified variant. Functions V_t and V_s (cf., Section 3) are given as $(\mathsf{H}_0(id_S), \mathsf{H}_0(rs), cid)$.

volve algorithms $\mathsf{Revoke}(msk)$ and $\mathsf{CheckCred}(id_S, cred, rk)$. The former is defined as follows: 1) Parse $(sk_0, pk_0, ch, seed, lrev) := msk$; 2) Return $rk := (pk_0, ch, lrev)$. The later is defined as follows:

1) Parse rk as $(pk_0, ch, lrev)$ and $cred$ as pk. Initialize $i := 0$.
2) While $lrev \neq 0^{\lambda-1}||1$, do the following:
   - Compute $r[i] := \mathsf{Dec}(ch, lrev)$.
   - Update $lrev := r[i]$.
   - Increment $i := i + 1$.
3) For j from 0 to $i - 1$:
   - Compute $pk' := \mathsf{RandPK}(pk_0, \mathsf{H}_1(pk_0, ch, r[j], id_S))$.
   - If $pk = pk'$, return 1.
4) If no match is found after the loop, return 0.

Remark 1. We embed the chain-based recoverable nonce mechanism (introduced in Section 2) directly within our scheme's key derivation and revocation logic, rather than presenting it as a separate building block. This integration is both natural and intentional: the mechanism is conceptually akin to the OFB mode in block ciphers, where a secure primitive (e.g., a PRF or block cipher) is iteratively applied to produce a chain of values. Similarly, our scheme derives each nonce deterministically from its predecessor, supporting backward recoverability and per-registration uniqueness – key properties for achieving unlinkability and revocation. As with OFB, the simplicity and modularity of this feedback-based construction make it more appropriate to incorporate directly. Consequently, our security analysis treats the design holistically; specifically, we model the chain-based recoverable nonce mechanism as a random mapping and reduce its security to the anonymous authentication security of SKE.

Unlinkability with Global Revocation. We show unlinkability of both bip32-MU and bip32-SU in presence of global revocation. We provide a proof sketch here and defer the formal proofs to Appendix C.

Lemma 1 *Let $\mathcal{A}$ be an adversary in the medium unlinkability with global revocation game of* bip32PA-MU. *Suppose $\mathcal{A}$ makes at most Q_{H_0} and Q_{H_1} queries to the random oracles H_0 and H_1, respectively. Then, there exists an algorithm $\mathcal{B}$, running in time comparable to $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{mUnl\text{-}GR},\text{bip32PA-MU}} \leq \frac{4 \cdot Q_{\mathsf{H}_1} + 3 \cdot |\mathcal{T}|^2}{2^{\lambda}} + \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Lemma 2 *Let $\mathcal{A}$ be an adversary in the strong unlinkability with global revocation game of* bip32PA-SU. *Suppose $\mathcal{A}$ makes at most Q_C queries to* Challenge. *Then, there exist two algorithms $\mathcal{B}$ and $\mathcal{B}'$, running in time comparable to $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathsf{sUnl\text{-}GR},\text{bip32PA-SU}}_{\mathcal{A}} \leq |\mathcal{T}|^2 \cdot (\mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\mathrm{PRF}} + \frac{2Q_C}{2^{v^*}}).$$

Proof Sketch. Each challenged token T_j ($j \in \{0,1\}$) derives its signing keys and nonces using two prefix-based random oracles: $G_1(x) = \mathsf{H}_1(pk_{T_j,0}, ch_{T_j}, \cdot, \cdot)$ and $G_2(y) = \mathsf{H}_1(seed_{T_j}, \cdot)$. Since ECDSA public keys are perfectly rerandomizable, each derived key $pk := \mathsf{RandPK}(pk_{T_j,0}, G_1(\cdot))$ is statistically independent across the Left, Right, and Challenge oracles. Thus, the adversary cannot link outputs via random-oracle queries. We then replace all encryption and PRF-based derivations with uniformly random values stored in small lookup tables. Specifically, we simulate: credential identifiers $cid := \mathsf{Enc}(ch, lrev)$ via $R[T, cid] := lrev$ or $cid := \mathsf{Enc}(seed, (id_S, r))$ via $R_1[T, r] := lrev$ and

$R_2[T, cid] := (id_S, r)$; signing randomness $\mathsf{H}_1(seed, M_a)$ via $L_L[T, M_a] := coins$; and rerandomized signing keys $sk := \mathsf{RandSK}(sk_0, G_1(\cdot))$ via $SK[T, (r, id_S)] := sk$. By standard hybrid arguments – relying on the anonymity of SKE and the PRF security of key derivations – these simulations are indistinguishable from the real protocol. Finally, we partition token-specific tables to enforce credential separation. For bip32PA-MU (medium unlinkability), we use two disjoint tables: one for Challenge, and one shared by Left and Right, ensuring that no cid overlaps between them. For bip32PA-SU (strong unlinkability), we pre-select the two challenge tokens and split into three disjoint tables (one per oracle), aborting on any reuse or unregistered cid. In both cases, this structure ensures the adversary's view is independent of the hidden bit, establishing unlinkability.

To see why bip32PA-MU does not generally achieve strong unlinkability, consider the following attack. The adversary selects two challenge tokens, T_0 and T_1, and a server with $S_L = S_R$, then submits them to the experiment. It performs honest registration interactions with Left, Right, and Challenge using token T_0. While the use of random nonces during registration ensures distinct cid values and independent keys, the authentication phase reuses cid directly as the nonce r_{T_j} in the secret key derivation: $sk = \mathsf{RandSK}(sk_{T_j,0}, \mathsf{H}_1(pk_{T_j,0}, ch_{T_j}, r_{T_j}, \cdot))$. The adversary can exploit this by selecting a random cid and using it during authentication with all three oracles on the same challenge message M. Each oracle returns a valid signature, and since signature computation is deterministic and reveals the corresponding public key, the adversary can compare these keys to link tokens. This attack would be prevented if cid values were authenticated (e.g., using a MAC). Therefore, bip32PA-MU guarantees only medium unlinkability, not strong unlinkability.

Revocation Soundness. We show that in both bip32-MU and bip32-SU, only the legitimate token owner can revoke its keys. We provide a proof sketch here and defer the formal proofs to Appendix C.

Lemma 3 *Let $\mathcal{A}$ be an adversary in the revocation soundness game of* bip32PA-MU*, making at most Q_{H_1} queries to H_1 and Q_C queries to* Challenge*. Then, there exists an algorithm $\mathcal{B}$, running in time comparable to $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound,bip32PA-MU}} \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + 1}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + \frac{Q_C \cdot Q_{\mathsf{H}_1}}{2^{\lambda}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth,SKE}}.$$

Lemma 4 *Let $\mathcal{A}$ be an adversary in the revocation soundness game of* bip32PA-SU*, making at most Q_{H_1} queries to H_1 and Q_C queries to* Challenge*. Then, there exists an algorithm $\mathcal{B}$, running in time comparable to $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound,bip32PA-MU}} \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + 1}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + \frac{Q_C \cdot Q_{\mathsf{H}_1}}{2^{\lambda}} + 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth,SKE}}.$$

Proof Sketch. The original experiment proceeds as follows. $\mathcal{A}$ interacts with a token T^* (holding a master key $msk_T := (sk_{T^*,0}, pk_{T^*,0}, ch_{T^*}, seed_{T^*}, lrev_{T^*})$) and a server S^* during registration. After obtaining $cred^* := pk$, $\mathcal{A}$ outputs a revocation key $rk^* := (pk_0^*, ch^*, lrev^*)$. $\mathcal{A}$ succeeds if $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*) := 1$, namely, if there exists an index j such that $pk := \mathsf{RandPK}(pk_0^*, \mathsf{H}_1(pk_0^*, ch^*, r[j], id_{S^*}))$, where the array $r[\cdot]$ records all previous nonces corresponding to $lrev^*$.

The security of both bip32PA-MU and bip32PA-SU relies on the following arguments. First, the entropy of ch_{T^*} and $lrev_{T^*}$ prevents $\mathcal{A}$ from producing $rk^* = (pk_{T^*,0}, ch_{T^*}, lrev_{T^*})$. Second, in bip32PA-MU, the values of cid returned by the Challenge oracle are replaced with uniformly random values. Consistency is maintained by internally mapping cid to its corresponding $lrev$. Similarly, in bip32PA-SU, two internal mappings are maintained: one linking $lrev$ to r, and another linking (id_S, r) to cid. During a guessed registration interaction, we embed an independent public key pk by appropriately programming the random oracle H_1, which ensures that $\mathcal{A}$ cannot distinguish real interactions from simulated ones. Security then reduces to solving the following problem: Given (sk, pk) and access to $\widetilde{\mathsf{H}}$, find a pair (pk', ch, r, id) such that $pk := \mathsf{RandPK}(pk', \widetilde{\mathsf{H}}(pk', ch, r, id))$. As shown in Lemma 16, this problem is statistically hard, which completes the proof.

Impersonation with Global Revocation. We show that both bip32-MU and bip32-SU achieve impersonation security in the presence of global revocation. A proof sketch is provided below, with formal proofs deferred to Appendix C.

Lemma 5 *Let $\mathcal{A}$ be an adversary in the impersonation with global revocation game of* bip32PA-MU*. Suppose $\mathcal{A}$ makes at most Q_{H_0} and Q_{H_1} queries to the random oracles H_0 and H_1, respectively, at most Q_{S} queries to* Start*, and at most Q_{C} queries to* Challenge*. Then, there exist two algorithms $\mathcal{B}$ and $\mathcal{B}'$, running in time comparable to $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{Imp-GR,bip32PA-MU}} \leq \frac{Q_{\mathsf{H}_0}^2 + |\mathcal{S}|^2}{2^{2\lambda}} + \frac{Q_S^2 + Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + |\mathcal{T}| \cdot (\mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth,SKE}} + \mathbf{Adv}^{\mathcal{B}'}_{\text{uf-hrk-idx1,SIG}_{\widetilde{\mathsf{H}}}}).$$

Lemma 6 *Let $\mathcal{A}$ be an adversary in the impersonation with global revocation game of* bip32PA-SU*. Suppose $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to random oracles $\mathsf{H}_0, \mathsf{H}_1$, respectively, at most Q_{S} queries to* Start*, and at most Q_{C} queries to* Challenge*. Then there exist two algorithms $\mathcal{B}$ and $\mathcal{B}'$ with the same running time as $\mathcal{A}$ such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{Imp-GR,bip32PA-SU}} \leq \frac{Q_{\mathsf{H}_0}^2 + |\mathcal{S}|^2}{2^{2\lambda}} + \frac{Q_S^2 + Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + |\mathcal{T}|(2 \cdot \mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth,SKE}} + \mathbf{Adv}^{\mathcal{B}'}_{\text{uf-hrk-idx1,SIG}_{\widetilde{\mathsf{H}}}}).$$

Proof Sketch. We adapt techniques from [3] to reduce impersonation security with global revocation for both bip32PA-MU and bip32PA-SU to the unforgeability under honestly rerandomized keys with indexed randomness (uf-hrk-idx1) of a rerandomizable signature scheme. This notion extends standard EUF-CMA security by: (1) providing a randomness oracle $\mathsf{RandO}(idx)$ that returns a fixed randomness ρ_{idx}

for each index idx, ensuring consistent outputs on repeated queries; and (2) offering a signing oracle $\mathsf{SignO}(idx, m)$ that signs messages under keys rerandomized by ρ_{idx}, forbidding repeated queries on the same pair (idx, m). A forgery is any (idx^*, m^*, σ^*) not previously output by SignO.

We embed our schemes into this model by: (1) setting $idx = \mathsf{H}_0(id_S)$ to bind all server operations to a unique index; and (2) deriving $\rho_{idx} = \mathsf{RandO}(idx)$ for each idx. Although both variants generate fresh cid values per registration for the same idx, they enforce invariants compatible with uf-hrk-idx1: bip32PA-MU sets $r := cid$ and caches ρ_{idx} for (id_S, r) at each registration; bip32PA-SU maps (idx, r) to a unique cid and associates ρ_{idx} with (id_S, r). Thus, each cid under id_S corresponds to the same ρ_{idx}, preserving key derivation consistency. By tracking signed pairs (idx, m), we ensure at most one signing query per message. This guarantees that any successful impersonation immediately yields a fresh forgery (idx^*, m^*, σ^*), violating the uf-hrk-idx1 security of the underlying signature scheme.

5. Revisiting Unlinkability

In the original unlinkability definition [3], all three experiments – **sUnl**, **mUnl**, and **wUnl** – prohibit the adversary from submitting the challenged cid (obtained from Left or Right during registration) to Challenge. While this prevents trivial linking, it is overly restrictive to ban all related queries involving cid. This section relaxes this by allowing adversaries to query the challenged cid to Challenge, except with a specific message (e.g., the challenged registration message). The main differences in this revised unlinkability arise only in the output phase – namely, in the definition of credential lists and credential separation conditions. For clarity, we focus solely on these changes, as all other phases remain unchanged from the original definition.

Credential Lists. In addition to recording the full input (cid, M) and output (cid, R_r) of Challenge, Left and Right queries, we introduce $\mathcal{L}^{ra}$ to track cid values queried to both $\mathcal{L}^r_{ch}$ and $\mathcal{L}^a_{lr}$, and $\mathcal{L}^{ar}$ for those queried to both $\mathcal{L}^a_{ch}$ and $\mathcal{L}^r_{lr}$). These two lists are used to capture cases where both registration and authentication are queried simultaneously, with the common term in R_r and M satisfies $R_r \cap M \neq \varnothing$. The redefined credential lists are as follows:

- $\mathcal{L}^r_{ch}$: All two-tuples (cid, R_r) returned by $\mathsf{Challenge}(\pi^{i,0}_T, id_S, \cdot, \cdot)$ queries not made via Left or Right, for $T \in \{T_0, T_1\}$ and $S \in \{S_L, S_R\}$.
- $\mathcal{L}^a_{ch}$: All two-tuples (cid, M) in inputs of $\mathsf{Challenge}(\pi^{i,j}_T, id_S, \cdot, \cdot)$ queries not made via Left or Right, for $j > 0$.
- $\mathcal{L}^r_{lr}$: All two-tuples (cid, R_r) returned by Left or Right queries when $j_b = 0$ or $j_{1-b} = 0$, respectively.
- $\mathcal{L}^a_{lr}$: All two-tuples (cid, M) in inputs of Left or Right queries when $j_b > 0$ or $j_{1-b} > 0$, respectively.
- $\mathcal{L}^{ra}$: All cid values such that $((cid, R_r) \in L^r_{ch}) \cap ((cid, M) \in L^a_{lr}) \cap (R_r \cap M \neq \varnothing)$.
- $\mathcal{L}^{ar}$: All cid values such that $((cid, M) \in L^a_{ch}) \cap ((cid, R_r) \in L^r_{lr}) \cap (R_r \cap M \neq \varnothing)$.

Credential Separation: To account for the adversary's ability to query the challenged cid for registration (i.e., Left or Right for $j = 0$) to the Challenge oracle ($j > 0$), or to use a corrupted registration cid from Challenge ($j = 0$) in a challenged authentication query (i.e., Left or Right for $j > 0$). The only restriction is that the authentication message contained in the registration response cannot be queried. We below modify the definition of **sUnl** and require these sets to be empty.

$$\begin{aligned}
\textbf{sUnl}: &\quad \mathcal{S}_{\mathsf{sUnl}} := \mathcal{L}^{ra} \cup \mathcal{L}^{ar},\\
\textbf{mUnl}: &\quad \mathcal{S}_{\mathsf{mUnl}} := \mathcal{S}_{\mathsf{sUnl}} \cup (\mathcal{L}^a_{ch} \cup \mathcal{L}^a_{lr}),\\
\textbf{wUnl}: &\quad \mathcal{S}_{\mathsf{wUnl}} := \mathcal{S}_{\mathsf{mUnl}} \cup (\mathcal{L}^r_{ch} \cup \mathcal{L}^r_{lr}).
\end{aligned}$$

Analysis of Existing Schemes. We now analyze the linkability of bip32PA, bip32PA-MU, and bip32PA-SU under the adversary's enhanced capabilities. As shown in [3], bip32PA can be linked by querying the challenged cid for registration to the authentication oracle. Consequently, bip32PA trivially fails to satisfy the refined unlinkability property. Following the reasoning in [3], the analysis for the last two schemes is similarly straightforward. Specifically, given the challenged two-tuple (cid, R_r) for registration, the adversary extracts $R_r := (pk, \sigma)$ and constructs an independent authentication message M^*_a from R_r. By querying (cid, M^*_a) to the Challenge oracle, the adversary obtains a valid signature $R_a := \sigma$ in both bip32PA-MU and bip32PA-SU, as cid enables the derivation of the secret key required for signing (see Figure 3). The adversary can then use pk to verify the signature and identify the correct token. Therefore, bip32PA-MU and bip32PA-SU also fail to satisfy the redefined unlinkability property.

Modified Syntax. As noted in the solution overview, the verifiable pre-signature mechanism strengthens unlinkability by allowing a token to pre-select its next authentication message and compute a corresponding "pre-signature" in advance. To support this, we extend the standard PLA tuple to $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth}, \mathsf{GRev})$, where Gen and GRev remain unchanged. Registration now includes an extra decapsulation step rdecap, and authentication an update step aupd. Concretely (see Figure 4): During registration, the interactions are: 1) The server issues a challenge $(c, st) \leftarrow \mathsf{rchall}(\mathrm{id}_S)$. 2) The client formats this into $\widehat{M}_r \leftarrow \mathsf{rcomm}(id_S, c)$. 3) The token generates $(cid, \widehat{R}_r) \leftarrow \mathsf{rresp}(msk, id_S, \widehat{M}_r)$. 4) The client decapsulates to recover $(R_r, \mathbf{pcs}) \leftarrow \mathsf{rdecap}(cid, \widehat{R}_r)$. (5) Finally, the server checks $(b_r, cred) \leftarrow \mathsf{rcheck}(st, cid, R_r)$, and stores $\mathbf{rcs}[cid] := (pk, rs_a)$. The authentication flow is analogous, with three key differences: 1) Both achall and acheck now take the server's context **rcs** to retrieve the credential and current challenge. 2) acomm uses **pcs** to incorporate the stored pre-signature. 3) aresp consumes the existing cid rather than producing a new one. Moreover, instead of a decapsulation step, the client performs an offline update $\mathrm{pcs} \leftarrow \mathsf{aupd}(cid, \mathbf{pcs}, \widehat{R}_a)$ to refresh its pre-signature context for the next authentication.

Optimized Protocols. We now present our three "+" variants to achieve the refined unlinkability with global revo-

cation. The overview of our adjusted syntax of bip32PA$^+$ can be found in Figure 4, and that of bip32PA-MU$^+$ and bip32PA-SU$^+$ is shown in Figure 5. Three secure hash functions $\mathsf{H}_0 : \{0,1\}^* \to \{0,1\}^{2\lambda}$, $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$, $\widehat{\mathsf{H}} : \{0,1\}^{l_2} \to \mathbb{M}||\mathbb{S}$ are involved and modeled as random oracles in the security proofs.

Key Generation. Initializing a token begins with invoking Gen to obtain a master secret key msk. The key generation for bip32PA$^+$ is identical to that of bip32PA [3], which generates $msk := (sk_0, pk_0, ch, seed)$ (see Appendix B). Similarly, the key generation for bip32PA-MU$^+$ and bip32PA-SU$^+$ (both generating $msk := (sk_0, pk_0, ch, seed, lrev)$) follows the same process as bip32PA-MU and bip32PA-SU (see Section 4).

Registration. Let $\mathbf{rcs}_S$ be the registration context held by every server S and $\mathbf{pcs}$ be the pre-signature context held by every client, where $\mathbf{rcs}$ stores clients' credentials and next-round nonces, $\mathbf{pcs}$ stores clients' pre-signatures for the next-round authentication. In this phase, the server S randomly chooses $rs_r, rs_a \in \{0,1\}^{\lambda}$ and sends the challenge $c := (id_S, rs_r, rs_a)$ to the client, where id_S is S's identifier. The client checks the validity of id_S, and computes $M_r := \mathsf{H}_0(rs_r), M_a := \mathsf{H}_0(rs_a)$, and then sends id_S and $\widehat{M_r} := (M_r, M_a)$ to the token. Below, the token generates the secret/public key pair for different variants.

- In bip32PA$^+$, the token computes $cid := \mathsf{H}_1(seed, id_S)$ and it derives the secret/public key pair as $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, id_S)), pk := \mathsf{RandPK}(pk_0, \mathsf{H}_1(pk_0, ch, id_S))$.
- In bip32PA-MU$^+$, the token computes $cid := \mathsf{Enc}(ch, lrev)$ and sets $r := lrev := cid$. Then, it derives the secret/public key pair as $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, r, id_S)), pk := \mathsf{RandPK}(pk_0, \mathsf{H}_1(pk_0, ch, r, id_S))$.
- In bip32PA-SU$^+$, the token computes $r := \mathsf{Enc}(ch, lrev), cid := \mathsf{Enc}(seed, (id_S, r))$, and sets $lrev := r$. Then, it derives the secret/public key pair as $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, id_S))$, $pk := \mathsf{RandPK}(pk_0, \mathsf{H}_1(pk_0, ch, r, id_S))$.

Then, the token parses $(M_r, M_a) := \widehat{M_r}$ to prepare $m_r := (\mathsf{H}_0(id_S), cid, pk, M_r), m_a := (\mathsf{H}_0(id_S), M_a)$, and it uses the above sk to generate signatures. For all the three schemes, derandomization of the signature is necessary according to the security of ECDSA in the BIP32 setting. Thus, the token derives the random coins $coins_r := \mathsf{H}_1(seed, m_r)$ and $coins_a := \mathsf{H}_1(seed, m_a)$ to compute $\sigma_r := \mathsf{Sig}(sk, m_r; coins_r)$ and $\sigma_a := \mathsf{Sig}(sk, m_a; coins_a)$. Next, the token generates the locking key $lk := \mathsf{PRF}(msk, (cid, id_S, M_a))$ and creates the pre-signature $\widehat{\sigma}_a := \widehat{\mathsf{H}}(lk) \oplus (M_a||\sigma_a)$. Finally, the token sends a response message $\widehat{R}_r := (pk, \sigma_r, \widehat{\sigma}_a)$ to the client. The client updates the pre-signature context as $\mathbf{pcs}[cid] := \widehat{\sigma}_a$, and forwards $R_r := (pk, \sigma_r)$ to the server. The server checks the validity of the client's response as follows. It aborts and does not update the $\mathbf{rcs}$ if $\mathsf{Ver}(pk, \sigma_r, m_r) := 0$, where $m_r := (\mathsf{H}_0(id_S), cid, pk, \mathsf{H}_0(rs_r))$. Otherwise, it accepts and updates pk via $\mathbf{rcs}[cid] := (pk, rs_a)$.

Authentication. To launch the authentication, the server retrieves $(pk, rs_a) := \mathbf{rcs}[cid]$ for the current authentication, and it randomly generates a new challenge $rs'_a \in \{0,1\}^{\geq\lambda}$ for the next-round authentication. After receiving $(cid, (id_S, rs_a, rs'_a))$ from the server, the client checks the validity of id_S and then computes $M_a := \mathsf{H}_0(rs_a), M'_a := \mathsf{H}_0(rs'_a)$. The client also retrieves $\widehat{\sigma}_a := \mathbf{pcs}[cid]$, and sends the message $\widehat{M_a} := (M_a, \widehat{\sigma}_a, M'_a)$, the server identifier id_S and credential identifier cid to the token.

Before recovering the signature, the token first deals with cid. For bip32PA$^+$, the token aborts if $cid \neq \mathsf{H}_1(seed, id_S)$. For bip32PA-MU$^+$, the token directly sets $r := cid$, and for bip32PA-SU$^+$, the token first computes $(id, r) := \mathsf{Dec}(seed, cid)$ and aborts if $id \neq id_S$. To recover the signature, the token in all the three schemes, recreates the locking key $lk := \mathsf{PRF}(msk, (cid, id_S, M_a))$ and computes the signature $(M||\sigma_a) := \widehat{\mathsf{H}}(lk) \oplus \widehat{\sigma}_a$. It aborts if $M \neq M_a$; Otherwise, the token sends $R_a := \sigma_a$ to the client, which then forwards it to the server. The server checks the validity of response as follows. It first retrieves the registration context $\mathbf{rcs}$ to obtain the token's public and current challenge $(pk, rs_a) := \mathbf{rcs}[cid]$. Then, it sets $m_a := (\mathsf{H}_0(id_S), \mathsf{H}_0(rs_a))$ and accepts the response and updates $\mathbf{rcs}[cid] := (pk, rs'_a)$ if $\mathsf{Ver}(pk, \sigma_a, m_a) := 1$.

In addition to the above online authentication, the token will continue to generate the pre-signature for the next challenge, which is performed offline and hence improves the efficiency. The token first recovers the signing key sk (i.e., $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, id_S))$ for bip32PA$^+$, and $sk := \mathsf{RandSK}(sk_0, \mathsf{H}_1(pk_0, ch, r, id_S))$ for bip32PA-MU$^+$ and bip32PA-SU$^+$) that it generated in the registration phase. Then for all the variants, the tokens signs the next challenge. Specifically, the token prepares $m'_a := (\mathsf{H}_0(id_S), M'_a), coins'_a := \mathsf{H}_1(seed, m'_a)$. Then it signs the message $\sigma'_a := \mathsf{Sig}(sk, m'_a; coins'_a)$. Finally, the token generates the locking key $lk' := \mathsf{PRF}(msk, (cid, id_S, M'_a))$ and pre-signature $\widehat{\sigma}_a := \widehat{H}(lk') \oplus (M'_a||\sigma'_a)$. After the generation, the token will send $\widehat{R}_a := \widehat{\sigma}_a$ to the client, and the client updates $\mathbf{pcd}[cid] := \widehat{\sigma}_a$.

Global Revocation. All the three "+" variants can support the global revocation. The derivation of secret/public key in these schemes is inherent from their respective previous versions (i.e., bip32PA, bip32PA-MU and bip32PA-SU), so we refer their global revocation functionality to Appendix B and Section 4 respectively.

Remark 2. The verifiable pre-signature mechanism, defined as $\widehat{\sigma}_a := \widehat{\mathsf{H}}(lk) \oplus (M_a||\sigma_a)$, serves as an access control mechanism during authentication. In the authentication phase, given $(id_S, cid, \widehat{M_a})$, the token parses $\widehat{M_a}$ as $(M_a, \widehat{\sigma}_a, M'_a)$ and computes the locking key as $lk := \mathsf{PRF}(msk, (cid, id_S, M_a))$. It then recovers $(M||\sigma_a) := \widehat{\mathsf{H}}(lk) \oplus \widehat{\sigma}_a$ and verifies that $M = M_a$. This process ensures that a valid authentication signature can only be generated if a valid pre-signature $\widehat{\sigma}_a$ is provided. Crucially, an adversary cannot forge a valid $\widehat{\sigma}_a$ without access to the locking key lk, which is deterministically derived from the secret master key msk stored within the token. Even if the adversary

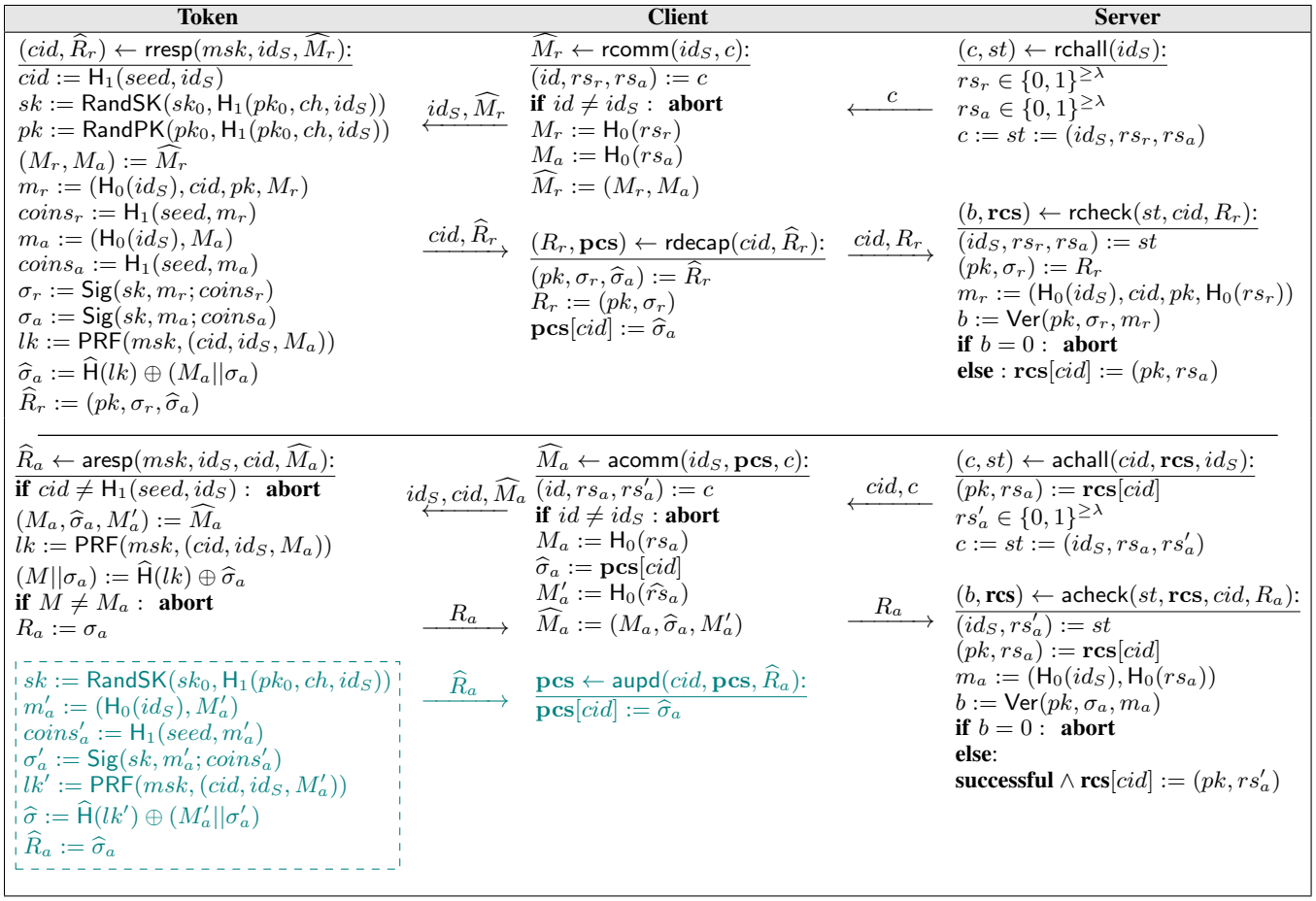

Figure 4: The WebAuthn registration (top) and authentication (bottom) protocol for our variation bip32PA$^+$. Annotated statements are executed only in the specified variant. Functions V_t and V_s are shown as $(\mathsf{H}_0(id_S), \mathsf{H}_0(rs), cid)$. Procedures in the dotted box run offline and do not affect online authentication.

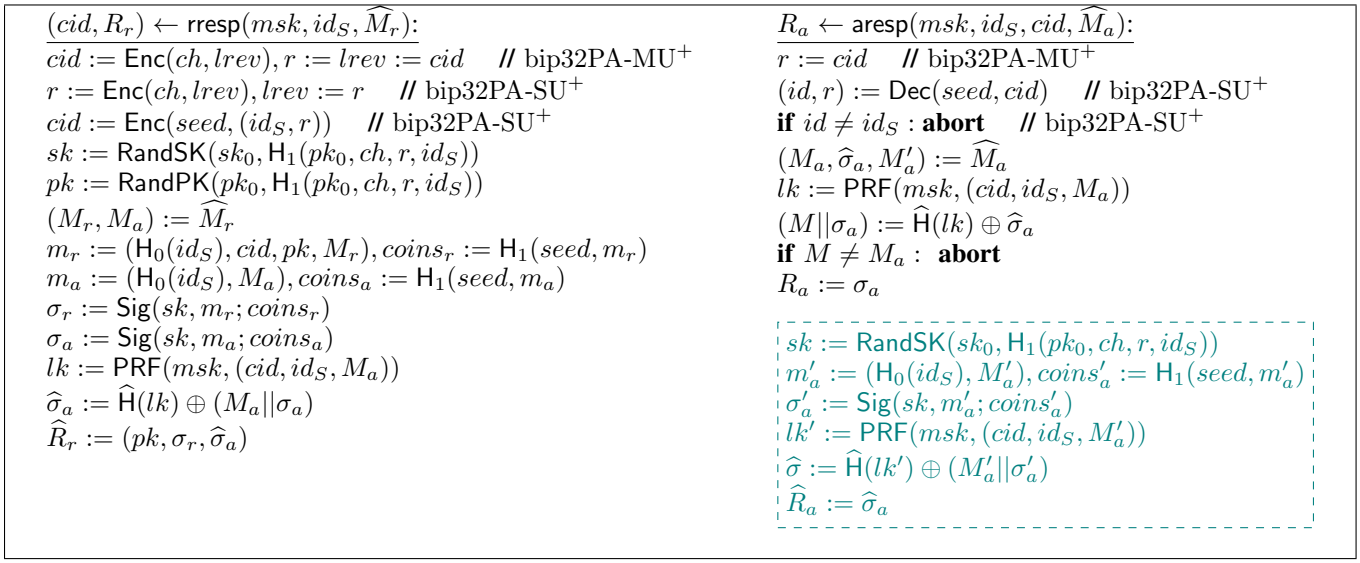

Figure 5: The WebAuthn registration (left) and authentication (right) protocol for our variations bip32PA-MU$^+$ and bip32PA-SU$^+$. Function V_t, V_s, and algorithms rcomm, acomm, rchall, achall, rdecap, aupd, rcheck, acheck are presented as in Figure 4.

knows the authentication challenge M_a, it cannot compute $\widehat{\sigma}_a$ without lk, thus preventing unauthorized access.

This verifiable pre-signature mechanism enables our refined unlinkability model, in which the adversary is permitted to query the challenged credential identifier cid during authentication. However, to succeed, it must present a pre-signature that was not issued by the challenged registration oracle. This more accurately reflects real-world FIDO2 threats – such as an adversary who temporarily accesses a token or interacts via a compromised client – yet cannot forge the target user's pre-signatures. For the sake of security proofs, we replace the PRF-based computation of locking keys with fresh, uniformly random values stored in a lookup map. This abstraction allows us to reduce the security of our construction to the underlying PRF security.

Unlinkability with Global Revocation. We show unlinkability of both bip32-MU and bip32-SU in presence of global revocation. We provide a proof sketch here and defer the formal proofs to Appendix C.

Lemma 7 *Let $\mathcal{A}$ be an adversary in the weak unlinkability with global revocation game of* $\mathrm{bip32PA}^+$. *Assume that $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to $\mathsf{H}_0, \mathsf{H}_1$, respectively. Then, there exists an algorithm $\mathcal{B}$ with the same running time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathsf{wUnl\text{-}GR},\mathrm{bip32PA}^+}_{\mathcal{A}} \leq \frac{4 \cdot Q_{\mathsf{H}_1} + 3 \cdot |\mathcal{T}|^2}{2^\lambda} + \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{PRF}}.$$

Lemma 8 *Let $\mathcal{A}$ be an adversary in the medium unlinkability with global revocation game of* $\mathrm{bip32PA\text{-}MU}^+$. *Assume that $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to random oracles $\mathsf{H}_0, \mathsf{H}_1$, respectively. Then, there exists two algorithms $\mathcal{B}$ and $\mathcal{B}'$, running in the same time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathsf{mUnl\text{-}GR},\mathrm{bip32PA\text{-}MU}^+}_{\mathcal{A}} \leq \frac{4 \cdot Q_{\mathsf{H}_1} + 3 \cdot |\mathcal{T}|^2}{2^\lambda} + \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}} + |\mathcal{T}| \cdot (\mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}} + 2 \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\mathrm{PRF}}).$$

Lemma 9 *Let $\mathcal{A}$ be an adversary in the strong unlinkability with global revocation game of* $\mathrm{bip32PA\text{-}SU}^+$. *Assume that $\mathcal{A}$ makes at most Q_C queries to* Challenge. *Then, there exists two algorithms $\mathcal{B}$ and $\mathcal{B}'$ with the same running time as $\mathcal{A}$:*

$$\mathbf{Adv}^{\mathsf{sUnl\text{-}GR},\mathrm{bip32PA\text{-}SU}^+}_{\mathcal{A}} \leq |\mathcal{T}|^2 \cdot (\mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}} + 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\mathrm{PRF}} + \frac{2Q_C}{2^{v^*}}).$$

Proof Sketch. In all three variants, $\mathcal{A}$ can only learn about the two challenge tokens T_0, T_1 is via the two prefixed random oracles $\mathsf{G}_1(\cdot) = \mathsf{H}_1(pk_{T_j,0}, ch_{T_j}, \cdot, \cdot)$, and $\mathsf{G}_2(\cdot) = \mathsf{H}_1(seed_{T_j}, \cdot)$, for $j \in \{0,1\}$, accessed only through Left, Right, and Challenge. Since public keys are perfectly rerandomizable, each derived $pk := \mathsf{RandPK}(pk_{T_j,0}, \rho)$ with $\rho := \mathsf{G}_1(\cdot)$ is statistically independent of all others. Thus no public-key outputs can link any two oracle invocations.

Next, we replace all encryptions and PRF-based derivations with fresh uniform values stored in small lookup tables. Specifically, we map credential identifier $cid := \mathsf{Enc}(ch, lrev)$ with $R[T, cid] := lrev$ or $cid := \mathsf{Enc}(seed, (id_S, r))$ with $R_1[T, r] := lrev, R_2[T, cid] := (id_S, r)$, signing randomness $coins := \mathsf{H}_1(seed, M_a)$ with $L_L[T, M_a] := coins$, locking-key $lk := \mathsf{PRF}(msk, (cid, id_S, M_a))$ with $LK[T, (\cdot)] := lk$, and rerandomized signing-key $sk := \mathsf{RandSK}(sk_0, \mathsf{G}_1(\cdot))$ as $SK[T, (r, id_S)] := sk$. A standard sequence of hybrids-using SKE security for the encryption-to-table step and PRF security for the locking- and signing-key steps – shows this replacement is indistinguishable.

Finally, we simulate the Left, Right and Challenge oracles with fresh keys and randomness, relying on the various credential-separation conditions to forbid any cross-oracle replay or collision: 1) For $\mathrm{bip32PA\text{-}SU}^+$, the strong unlinkability forbids the reuse of a registration cid in any authentication query that overlaps in more than the cid field. We therefore pre-select the two challenge tokens and then split each token's tables into three disjoint maps – one for Challenge, one for Left, and one for Right. Any cross-use beyond cid aborts, so the adversary's view is independent of the hidden bit. 2) For $\mathrm{bip32PA\text{-}MU}^+$, the medium unlinkability forbids any authentication-side reuse of a cid. We split each token's tables once – between Challenge (which never sees cid in authentication) and the union {Left,Right}. Credential separation prevents collisions, yielding unlinkability. 3) For $\mathrm{bip32PA}^+$, the weak unlinkability forbids any registration-side reuse of a cid. Challenge thus never observes a cid. We simply simulate Left and Right with fresh key-material, making the two oracles trivially independent.

Revocation Soundness and Impersonation with Global Revocation. The revocation soundness for the three "+" variants follows the same core structure as the original schemes, with a critical modification in how locking keys are managed. In the "+" variants, instead of deriving locking keys $lk := \mathsf{PRF}(msk, (cid, id_S, M_a))$, we sample lk uniformly at random and store them in a table $LK[T, (\cdot)] := lk$. This adjustment necessitates an additional reduction to PRF security. For impersonation with global revocation, the proof structure remains largely unchanged, except for two key adaptations: 1) Locking keys are handled via the same table-lookup approach. 2) We must also account for the two-tuple randomness c (either (rs_r, rs_a) during registration or (rs_a, rs'_a) during authentication) when simulating Start. Because no other novel challenges arise, we state the main security lemmas here and refer the reader to Appendix D for the complete proofs.

Lemma 10 *Let $\mathcal{A}$ be an adversary in the revocation soundness game of* $\mathrm{bip32PA}^+$ *with Q_{H_1} queries to H_1 and Q_C queries to* Challenge. *Then there exists an algorithm $\mathcal{B}$ with the same running time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{rev\text{-}sound},\mathrm{bip32PA}^+} \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + Q^2_{\mathsf{H}_1} + 1 + Q_C \cdot Q_{\mathsf{H}_1}}{2^\lambda} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{prf},\mathrm{PRF}}.$$

Lemma 11 *Let $\mathcal{A}$ be an adversary in the revocation soundness game of* bip32PA-MU$^+$ *with at most Q_{H_1} queries to H_1 and Q_C queries to* Challenge. *Then there exists algorithms $\mathcal{B}$ and $\mathcal{B}'$ with the same running time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound},\text{bip32PA-MU}^+} \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + 1}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + \frac{Q_C \cdot Q_{\mathsf{H}_1}}{2^{\lambda}} + |\mathcal{T}| \cdot (\mathbf{Adv}^{\mathcal{B}}_{\text{prf},\text{PRF}} + \mathbf{Adv}^{\mathcal{B}'}_{\text{anon-auth},\text{SKE}}).$$

Lemma 12 *Let $\mathcal{A}$ be an adversary in the revocation soundness game of* bip32PA-SU$^+$ *with at most Q_{H_1} queries to H_1 and Q_C queries to* Challenge. *Then there exists algorithms $\mathcal{B}$ and $\mathcal{B}'$ with the same running time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound},\text{bip32PA-SU}^+} \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + 1}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}} + \frac{Q_C \cdot Q_{\mathsf{H}_1}}{2^{\lambda}} + |\mathcal{T}| \cdot (\mathbf{Adv}^{\mathcal{B}}_{\text{prf},\text{PRF}} + 2\mathbf{Adv}^{\mathcal{B}'}_{\text{anon-auth},\text{SKE}}).$$

Lemma 13 *Let $\mathcal{A}$ be an adversary in the impersonation with global revocation game of* bip32PA$^+$. *Assume that $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to $\mathsf{H}_0, \mathsf{H}_1$, respectively, at most Q_{S} queries to oracle* Start, *and at most Q_{C} queries to oracle* Challenge. *Then there exist two algorithms $\mathcal{B}$ and $\mathcal{B}'$ with the same running time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{\mathcal{A}}_{\text{Imp-GR},\text{bip32PA}^+} \leq \frac{Q_{\mathsf{H}_0}^2 + Q_S^2 + |\mathcal{S}|^2}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}} + |\mathcal{T}| \cdot \left(\mathbf{Adv}^{\mathcal{B}}_{\text{prf},\text{PRF}} + \mathbf{Adv}^{\mathcal{B}'}_{\text{uf-hrk-idx1},\text{SIG}_{\tilde{\mathsf{H}}}}\right).$$

Lemma 14 *Let $\mathcal{A}$ be an adversary in the impersonation with global revocation game of* bip32PA-MU$^+$. *Assume that $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to random oracles $\mathsf{H}_0, \mathsf{H}_1$, respectively, at most Q_{S} queries to oracle* Start, *and at most Q_{C} queries to oracle* Challenge. *Then there exists three algorithms $\mathcal{B}$, $\mathcal{B}'$ and $\mathcal{B}^*$ with the same running time as $\mathcal{A}$ such that $\mathcal{A}$'s advantage can be upper bounded by:*

$$\frac{Q_{\mathsf{H}_0}^2 + Q_S^2 + |\mathcal{S}|^2}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + Q_{\mathsf{H}_1}^2}{2^{\lambda}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\text{prf},\text{PRF}} |\mathcal{T}| \cdot (\mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth},\text{SKE}} + \mathbf{Adv}^{\mathcal{B}^*}_{\text{uf-hrk-idx1},\text{SIG}_{\tilde{\mathsf{H}}}}).$$

Lemma 15 *Let $\mathcal{A}$ be an adversary in the impersonation with global revocation game of* bip32PA-MU. *Assume that $\mathcal{A}$ makes at most $Q_{\mathsf{H}_0}, Q_{\mathsf{H}_1}$ queries to random oracles $\mathsf{H}_0, \mathsf{H}_1$, respectively, at most Q_{S} queries to* Start, *and at most Q_{C} queries to* Challenge. *Then there exists three algorithms $\mathcal{B}$, $\mathcal{B}'$ and $\mathcal{B}^*$ with the same running time as $\mathcal{A}$ such that $\mathcal{A}$'s advantage can be upper bounded by:*

$$\frac{Q_{\mathsf{H}_0}^2 + Q_S^2 + |\mathcal{S}|^2}{2^{2\lambda}} + \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}| + Q_{\mathsf{H}_1}^2}{2^{\lambda}} + |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\text{prf},\text{PRF}} + |\mathcal{T}| \cdot (2\mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth},\text{SKE}} + \mathbf{Adv}^{\mathcal{B}^*}_{\text{uf-hrk-idx1},\text{SIG}_{\tilde{\mathsf{H}}}}).$$

6. Performance Evaluation

We implemented five variants of bip32PA – bip32PA$^+$, bip32PA-MU, bip32PA-SU, bip32PA-MU$^+$, and bip32PA-SU$^+$ – alongside the original bip32PA [3] over the secp256k1[1] elliptic curve on a 16-thread client-server testbed (Server: Windows 11/Python 3.8 on Intel Core Ultra 5 125H 1.20 GHz, 32 GB RAM, 1 TB SSD; Client: Android 15/JDK 17/Gradle 8.6 on Snapdragon 8 Gen 3, 16 GB RAM, 512 GB storage). In online authentication, bip32PA-MU$^+$ achieved the lowest latency (13.21 ms vs. 43.39 ms for the original bip32PA). Per-credential storage is 32 B on the client and 96 B on the server for bip32PA, bip32PA-MU and bip32PA-SU; the "$^+$" variants increase this to 96 B and 128 B, respectively. Revoking 2^{20} tokens (about 1 million) took 172 s for single-step schemes (bip32PA, bip32PA$^+$) and 391 s for chain-based variants (MU/SU styles with depth 5), reflecting the additional chain-decoding work. All variants integrate seamlessly with FIDO2, scale on multi-core hardware, and total about 8,000 LOC of crypto, protocol, and revocation code (source: https://anonymous.4open.science/r/bip32PA-72DC/).

Execution Time and Storage Cost. We evaluate the performance and storage efficiency of the five bip32PA variants against the original scheme, focusing on registration time, authentication time, and storage costs (see Table 1). Registration occurs once per credential, while authentication happens repeatedly, making online authentication time the most critical efficiency factor. The $^+$ variants (i.e., bip32PA$^+$, bip32PA-MU$^+$, bip32PA-SU$^+$) support offline precomputation, which significantly reduces online authentication time at the cost of slightly higher registration time and storage. Notably, bip32PA-MU$^+$ achieves the lowest online authentication time (13.21 ms), more than three times faster than the original scheme (43.39 ms), while still providing medium unlinkability under stronger adversarial ability.

Regarding storage, the client in bip32PA, bip32PA-MU, and bip32PA-SU needs to store only the server's id_S (32 B). The $^+$ variants involve an additional pre-signature for the next-round authentication, increasing client storage to 96 B. For the server, the base schemes store the identifier cid and public key pk (96 B), whereas the $^+$ variants also store a random string rs for the next-round authentication, raising server storage to 128 B. For the token, bip32PA and bip32PA$^+$ only store $(sk_0, pk_0, ch, seed)$ (129 B), while others – bip32PA-MU, bip32PA-SU, bip32PA-MU$^+$, and bip32PA-SU$^+$ – additionally store revocation material $lrev$, increasing token storage to 161 B.

In terms of security, bip32PA, bip32PA-MU, and bip32PA-SU offer weak, medium, and strong unlinkability, respectively, under the original unlinkability model. The $^+$ variants redefine the adversarial model, providing stronger privacy guarantees. In summary, bip32PA-MU$^+$ strikes the best balance, offering a 3.3× reduction in online authentication time compared to the original scheme, medium unlinkability under stronger adversarial assumptions, and moderate storage overhead. This makes our enhanced schemes highly practical, especially in scenarios requiring both high authentication efficiency and robust privacy protections.

1. https://en.bitcoin.it/wiki/Secp256k1

TABLE 1: Comparison of execution time and storage cost. Execution time for these comparing schemes were averaged across 100 runs. Reg. denotes Registration and Auth. denotes Authentication. Storage costs (token, client, server) represent a single credential entry. To eliminate external latency, user touch verification was disabled on the token during testing, isolating protocol-specific overhead.

Protocol	Reg. Time [ms]	Auth. Time [ms]	Token's Storage [Byte]	Client's Storage [Byte]	Server's Storage [Byte]
bip32PA [3]	92.94	43.39	129	32	96
bip32PA-MU	97.32	46.13	161	32	96
bip32PA-SU	99.11	47.84	161	32	96
bip32PA$^+$	109.03	21.21 (online) 49.77 (offline)	129	96	128
bip32PA-MU$^+$	113.12	13.21 (online) 49.96 (offline)	161	96	128
bip32PA-SU$^+$	113.54	14.78 (online) 51.91 (offline)	161	96	128

Global Revocation Efficiency. We first analyze the efficiency of our five bip32PA variants under a certificate revocation list (CRL)-style framework [3]. For bip32PA$^+$ and the original bip32PA, the computational cost of revocation checks scales as $B \cdot N \cdot C_{\mathsf{CheckCred}}$, where B is the revocation list size, N is the number of stored credentials, and $C_{\mathsf{CheckCred}}$ is the cost of a single credential check. For variants bip32PA-MU, bip32PA-SU, bip32PA-MU$^+$, and bip32PA-SU$^+$, revocation involves traversing a chain of depth D (the number of steps required to decode $lrev = 0^{\lambda-1}||1$), which increases the cost to $B{\cdot}N{\cdot}D{\cdot}C_{\mathsf{RandPK}}$. Here, we denote C_{RandPK} as the cost of a public key derivation upon $lrev$.

The optimization of precomputation and binary search also applies to all our variants: 1) For each revoked token with state $lrev$, the server uses the revocation key returned by Revoke to decode the chain $r[0], \ldots, r[D-1]$ (i.e., decoding until $lrev = 0^{\lambda-1}||1$) and applies RandPK iteratively to produce all D derived public keys. These $B \cdot D$ keys are cached once. 2) The server keeps its N registered public keys in sorted order. Each of the $B{\cdot}D$ revoked keys is then checked in $O(\log N)$ time instead of $O(N)$. Thus a full sweep costs $O(B{\cdot}D)C_{\mathsf{RandPK}} + O(B{\cdot}D{\cdot}\log N)C_{BigInteger}$ ($C_{BigInteger}$ is the cost of invoking BigInteger comparison algorithm to compare public keys), i.e., $O(B \cdot D \cdot \log N)$. 3) When a new token registers, its public key is compared via binary search against the $B \cdot D$ precomputed revoked keys in $O(\log(B \cdot D))$ time. 4) As tokens are revoked over time, the server updates its cache of revoked-key chains by adding or removing the corresponding D derived keys, retaining efficient $O(D \log N)$ per-token update.

We implemented the revocation sweep to measure real-world performance. Our test harness executes a single revocation-key check by 1) deriving the candidate public key pk' via RandPK and 2) performing a binary search – $\log N$ comparisons – against a random credential. We repeat this B times and average over 100 runs, parallellizing across 16 threads (as in [3]). We varied $N \in \{2^{30}, 2^{32}, \ldots, 2^{40}\}$ and $B \in \{2^{18}, 2^{20}, 2^{22}, 2^{24}\}$. Binary-search overhead remained under one second even for $N = 2^{40}$, showing that credential database size is not a bottleneck. Table 2 reports results for $N = 2^{40}$. Single-step revocation ($D = 1$) – bip32PA and bip32PA$^+$ – processes $B = 2^{20}$ entries in 172 s. Chain-based variants ($D = 5$) incur about 2.3× overhead (391 s vs. 172 s at $B = 2^{20}$), matching the $O(B \cdot D \cdot \log N)$ cost. Even at $B = 2^{24}$, all schemes complete in under two hours. Incremental updates and off-peak batching make the mechanism practical at Internet scale.

TABLE 2: Revocation time (s) vs. blocklist size B ($N = 2^{40}$).

Protocol	2^{18}	2^{20}	2^{22}	2^{24}
bip32PA / bip32PA$^+$	42	172	689	2693
MU / SU / MU$^+$ / SU$^+$	111	391	1797	6684

7. Conclusion and Future Work

We improve FIDO2 unlinkability under global revocation by addressing practical gaps in existing models and constructions. First, we introduce a chain-based recoverable nonce mechanism that yields two schemes – bip32PA-MU and bip32PA-SU – achieving medium and strong unlinkability, respectively, under the original model. Next, we refine the unlinkability definition itself and present three additional variants – bip32PA$^+$, bip32PA-MU$^+$, and bip32PA-SU$^+$ – each offering weak, medium, or strong unlinkability under the refined model. Rigorous security proofs confirm the soundness of all five schemes, and our implementation demonstrates their efficiency and seamless compatibility with existing deployments.

Despite capturing more realistic threat scenarios, our refined unlinkability model still assumes that an adversary cannot obtain valid authentication signatures for the challenged registration credential identifier. If a malicious party – such as a compromised server or someone who breaks TLS – did acquire those signatures, trivial linkage attacks would succeed. In other words, our model does not defend against insiders or attackers who directly see authentication signatures under the same identifier. Addressing this gap is an important direction for future work: for example, one might issue a fresh public key at authentication that the server can still verify as registered, yet without breaking compatibility with existing FIDO2 deployments.

References

[1] N. Bindel, C. Cremers, and M. Zhao, "Fido2, CTAP 2.1, and webauthn 2: Provable security and post-quantum instantiation," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1471–1490. [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179454

[2] M. Barbosa, A. Boldyreva, S. Chen, and B. Warinschi, "Provable security analysis of FIDO2," in *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, ser. Lecture Notes in Computer Science, T. Malkin and C. Peikert, Eds., vol. 12827. Springer, 2021, pp. 125–156. [Online]. Available: https://doi.org/10.1007/978-3-030-84252-9_5

[3] J. L. Lucjan Hanzlik and B. Wagner, "Token meets wallet: Formalizing privacy and revocation for FIDO2," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1491–1508. [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179373

[4] M. Kepkowski, L. Hanzlik, I. D. Wood, and M. A. Kâafar, "How not to handle keys: Timing attacks on FIDO authenticator privacy," *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 4, pp. 705–726, 2022. [Online]. Available: https://doi.org/10.56553/popets-2022-0129

[5] M. Barbosa, A. Boldyreva, S. Chen, K. Cheng, and L. Esquível, "Revisiting the security and privacy of fido2," *Cryptology ePrint Archive*, 2025.

[6] P. Das, S. Faust, and J. Loss, "A formal treatment of deterministic wallets," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 651–668. [Online]. Available: https://doi.org/10.1145/3319535.3354236

[7] A. Shakevsky, E. Ronen, and A. Wool, "Trust dies in darkness: Shedding light on samsung's trustzone keymaster design," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 251–268. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/shakevsky

[8] D. Kuchhal, M. Saad, A. Oest, and F. Li, "Evaluating the security posture of real-world FIDO2 deployments," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 2381–2395. [Online]. Available: https://doi.org/10.1145/3576915.3623063

[9] I. B. Guirat and H. Halpin, "Formal verification of the W3C web authentication protocol," in *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, HoTSoS 2018, Raleigh, North Carolina, USA, April 10-11, 2018*, M. P. Singh, L. A. Williams, R. Kuhn, and T. Xie, Eds. ACM, 2018, pp. 6:1–6:10. [Online]. Available: https://doi.org/10.1145/3190619.3190640

[10] N. Frymann, D. Gardham, F. Kiefer, E. Lundberg, M. Manulis, and D. Nilsson, "Asynchronous remote key generation: An analysis of yubico's proposal for W3C webauthn," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 939–954. [Online]. Available: https://doi.org/10.1145/3372297.3417292

[11] Yubico, "Yubikey u2f key generation," 2022. [Online]. Available: https://developers.yubico.com/U2F/Protocol_details/Key_generation.html

[12] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, ser. Lecture Notes in Computer Science, M. Yung, Ed., vol. 2442. Springer, 2002, pp. 61–76. [Online]. Available: https://doi.org/10.1007/3-540-45708-9_5

[13] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, V. Atluri, B. Pfitzmann, and P. D. McDaniel, Eds. ACM, 2004, pp. 168–177. [Online]. Available: https://doi.org/10.1145/1030083.1030106

[14] J. Camenisch, M. Drijvers, and J. Hajny, "Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, WPES@CCS 2016, Vienna, Austria, October 24 - 28, 2016*, E. R. Weippl, S. Katzenbeisser, and S. D. C. di Vimercati, Eds. ACM, 2016, pp. 123–133. [Online]. Available: http://dl.acm.org/citation.cfm?id=2994625

[15] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo, "True2f: Backdoor-resistant authentication tokens," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 398–416. [Online]. Available: https://doi.org/10.1109/SP.2019.00048

[16] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991. [Online]. Available: https://doi.org/10.1007/BF00196725

[17] M. Dworkin, "Recommendation for block cipher modes of operation," *NIST special publication*, vol. 800, p. 38B, 2001.

[18] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi, "The exact security of BIP32 wallets," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1020–1042. [Online]. Available: https://doi.org/10.1145/3460120.3484807

Appendix A. Formal Definition of PLA

Passwordless Authentication Scheme (PLA). A PLA scheme consists of the following components:

- Gen: This key generation algorithm takes as input a public parameters pp and outputs a master secret key msk.
- Reg: This registration protocol consists of the following four algorithms:
  - A randomized registration challenge generation algorithm $(c, st) \leftarrow \mathsf{rchall}(id_S)$ that takes a server identity id_S as input and outputs a challenge c and a state st.
  - A deterministic registration command creation algorithm $M_r \leftarrow \mathsf{rcomm}(id_S, c)$ that takes id_S and c as input and outputs a message M_r.
  - A randomized registration response algorithm $(cid, R_r) \leftarrow \mathsf{rresp}(msk, id_S, M_r)$ that takes msk, id_S, and M_r as input and outputs a credential identifier cid and a response R_r.
  - A deterministic registration check algorithm $(b, cred) \leftarrow \mathsf{rcheck}(st, cid, R_r)$ that takes st, cid, and R_r as input and outputs a bit $b \in \{0, 1\}$ and a credential $cred$.
- This authentication protocol consists of the following four algorithms:
  - A randomized authentication challenge generation algorithm $(c, st) \leftarrow \mathsf{achall}(id_S)$ that takes id_S as input and outputs a challenge c and a state st.
  - A deterministic authentication command creation algorithm $M_a \leftarrow \mathsf{acomm}(id_S, c)$ that takes id_S and c as input and outputs a message M_a.

– A randomized authentication response algorithm $R_a \leftarrow \mathsf{aresp}(msk, id_S, cid, M_a)$ that takes msk, id_S, cid, and M_a as input and outputs a response R_a.
– A deterministic authentication check algorithm $b \leftarrow \mathsf{acheck}(st, \mathbf{rcs}, cid, R_a)$ that takes st, a registration context **rcs**, cid, and R_a as input and outputs a bit $b \in \{0, 1\}$.

Completeness of PLA. A PLA scheme is complete, if for all $msk \in \mathsf{Gen}(par)$, parties T and S, and sets $\mathbf{R}_{init}$, $\mathbf{R}_{betw}$ of tuples $(cid, cred)$, the probability that the following experiment outputs 0 is negligible where we assume that cid derived in Step 2 is not in the list $\mathbf{R}_{betw}$: 1) Initialize a key-value table **rcs**, and set $\mathbf{rcs}[cid] := cred$ for each $(cid, cred) \in \mathbf{R}_{init}$. 2) Run the registration protocol Reg of T at S: $(c, st) \leftarrow \mathsf{rchall}(id_S)$, $M_r \leftarrow \mathsf{rcomm}(id_S, c)$, $(cid, R_r) \leftarrow \mathsf{rresp}(msk, id_S, M_r)$, and $(b_r, cred) \leftarrow \mathsf{rcheck}(st, cid, R_r)$. If $b_r = 0$, output 0. Otherwise, update $\mathbf{rcs}[cid] := cred$. 3) For each $(cid, cred) \in \mathbf{R}_{betw}$, set $\mathbf{rcs}[cid] := cred$. 4) Run the authentication protocol Auth of T at S: $(c, st) \leftarrow \mathsf{achall}(id_S)$, $M_a \leftarrow \mathsf{acomm}(id_S, c)$, $R_a \leftarrow \mathsf{aresp}(msk, id_S, cid, M_a)$, and $b_a \leftarrow \mathsf{acheck}(st, \mathbf{rcs}, cid, R_a)$. 5) Return b_a.

Server and Token Oracles. Before review the security and privacy models for WebAuthn, we present the server and token oracles that an adversary may access. These oracles enable the adversary to freely interact with tokens and servers, forming the basis for all the subsequent security definitions. For a server S, we define $\mathbf{rcs}_S$ as the registration context storing credentials, st_S as the state transferred between the algorithms rchall, achall, and acheck, and C_S as a mapping used to associate registration interactions with authentication interactions. Let $\mathcal{A}$ be an adversary and $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ a PLA. Each party $P \in \mathcal{T} \cup \mathcal{S}$ is associated with a set of handles $\pi_P^{i,j}$, representing two types of instances: 1) $j = 0$: The ith registration instance of P. 2) $j \geq 1$: The jth authentication instance corresponding to the ith registration. For each token $T \in \mathcal{T}$, a secret key $msk_T \leftarrow \mathsf{Gen}(par)$ is given. For each server $S \in \mathcal{S}$, $\mathbf{rcs}_S$, C_S, and st_S are initialized as empty. The adversary $\mathcal{A}$ interacts with the following oracles: 1) $\mathsf{Start}(\pi_S^{i,j})$: If $j = 0$: $(c, st) \leftarrow \mathsf{rchall}(id_S)$. If $j > 0$: $(c, st) \leftarrow \mathsf{achall}(id_S)$. The oracle sets $st_S[i, j] := st$ and returns c to $\mathcal{A}$. 2) $\mathsf{Challenge}(\pi_T^{i,j}, id_S, cid, M)$: If $j = 0$: $(cid, R_r) \leftarrow \mathsf{rresp}(msk_T, id_S, M)$. If $j > 0$: $R_a \leftarrow \mathsf{aresp}(msk_T, id_S, \mathsf{cid}, M)$. The result ($cid, R_r$ or R_a) is returned to $\mathcal{A}$. Note: cid is ignored for $j = 0$. 3) $\mathsf{Complete}(\pi_S^{i,j}, cid, R)$: Abort if $\mathsf{Start}(\pi_S^{i,j})$ has not been queried. If $j = 0$: $(b, cred) \leftarrow \mathsf{rcheck}(st_S[i, j], cid, R)$, then set $C_S[i] := cid$ and $\mathbf{rcs}_S[cid] := cred$. If $j > 0$: Abort if $cid \neq C_S[i]$; otherwise, $b \leftarrow \mathsf{acheck}(st_S[i, j], \mathbf{rcs}_S, cid, R)$. In all cases, b is returned to $\mathcal{A}$.

Here, for each $(i, j, T, S) \in \mathbb{N} \times \mathbb{N} \times \mathcal{T} \times \mathcal{S}$, the oracles $\mathsf{Start}(\pi_S^{i,j})$, $\mathsf{Challenge}(\pi_T^{i,j}, \cdot, \cdot, \cdot)$, and $\mathsf{Complete}(\pi_S^{i,j}, \cdot, \cdot)$ are executed only once. In addition, partnering of handles is defined based on a shared session identifier. Two handles, $\pi_S^{i,j}$ (on the server) and $\pi_T^{i',j'}$ (on the token), are considered partnered if they share the same session identifier. Formally, let V_T and V_S be functions specified by the PLA scheme. The function V_T takes as input the transcript $tr_T^{i,j} = (id_S, cid, M, R)$ observed by a token $T \in \mathcal{T}$ during an oracle call to $\mathsf{Challenge}(\pi_T^{i,j}, \cdot, \cdot, \cdot)$, and outputs a bitstring $V_T(tr_T^{i,j})$. Similarly, V_S takes as input the transcript $tr_S^{i,j} = (c, cid, R)$ observed by a server $S \in \mathcal{S}$ during oracle calls to $\mathsf{Start}(\pi_S^{i,j})$ and $\mathsf{Complete}(\pi_S^{i,j}, \cdot, \cdot)$, and outputs a bitstring $V_S(tr_S^{i,j})$. Handles $\pi_T^{i,j}$ and $\pi_S^{i',j'}$ are partnered if and only if $(j = 0 \iff j' = 0)$ and $V_T(tr_T^{i,j}) = V_S(tr_S^{i',j'})$.

Impersonation Security. We now review the definition of impersonation security for PLA scheme. Informally, a scheme is secure against impersonation if only the token that registered can authenticate with the server, and a single interaction cannot be reused for multiple authentications.

Definition 1 (Impersonation Security) *For a PLA scheme* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ *and adversary* $\mathcal{A}$*, the experiment* $\mathsf{Imp}_{\mathrm{PLA}}^{\mathcal{A}}$ *proceeds as follows.*

- ***Setup.*** *For each token* $T \in \mathcal{T}$*, a key is created by invoking* $msk_T \leftarrow \mathsf{Gen}(par)$.
- ***Online Phase.*** $\mathcal{A}$ *interacts with the oracles* Start, Challenge*, and* Complete *defined above.*
- ***Output Phase.*** $\mathcal{A}$ *terminates, and the experiment outputs 1 iff there exists a server handle* $\pi_S^{i,j}$ *with* $j > 0$ *such that: 1)* $\pi_S^{i,0}$ *is partnered with a token handle* $\pi_T^{k,0}$*. 2)* $\pi_S^{i,j}$ *accepted, namely,* $\mathsf{acheck}(st_S[i, j], \mathbf{rcs}_S, cid, R) = 1$ *in the call* $\mathsf{Complete}(\pi_S^{i,j}, cid, R)$*. 3)* $\pi_S^{i,j}$ *is not partnered with any token handle* $\pi_T^{i',j'}$ *or is partnered with a token handle that is itself partnered with a different server handle* $\pi_S^{i'',j''}$*.*

The adversary's advantage is defined as:

$$\mathbf{Adv}_{\mathsf{Imp},\mathrm{PLA}}^{\mathcal{A}} := \Pr[\mathsf{Imp}_{\mathrm{PLA}}^{\mathcal{A}} = 1].$$

The winning conditions imply that: 1) the token T registered at server S, and 2) the adversary authenticated at S for that registration, and 3) the authentication was either unlinked to any token or linked to a different registration context. This captures the *trust-on-first-use* model, where servers trust the user during registration as the legitimate one.

Unlinkability. This property ensures that interactions with the same token and different registrations of the same token cannot be linked. Three levels of unlinkability are defined in a PLA scheme: 1) **Strong Unlinkability** guarantees that the adversary cannot link token interactions, even if they obtain the credential identifier (cid) during the registration phase. Specifically, the adversary cannot submit the cid to the Challenge oracle to determine which token was used in the Left or Right oracles, preventing trivial attacks where the adversary tries to match responses based on the cid. 2) **Medium Unlinkability** also restricts the adversary from querying the same cid at both the Challenge oracle and the Left/Right oracles, even if the cid was not part of the registration response. 3) **Weak Unlinkability** further ensures that the same cid is not returned during registration at both Challenge and Left/Right oracles.

Definition 2 (Unlinkability) *For a PLA scheme* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ *and adversary* $\mathcal{A}$*, the experiments* ***sUnl****,* ***mUnl****, and* ***wUnl*** *proceed as follows.*

- ***Setup.*** *For each token* $T \in \mathcal{T}$*, a key is created by invoking* $msk_T \leftarrow \mathsf{Gen}(par)$.
- ***Phase 1.*** $\mathcal{A}$ *interacts with the oracles* Start*,* Challenge*, and* Complete *defined above.*
- ***Phase 2.*** $\mathcal{A}$ *outputs two token identifiers* T_0, T_1 *and two server identifiers* $S_L, S_R \in \mathcal{S}$*. Let* i_0 *and* i_1 *be the smallest identifiers such that* $\pi_{T_0}^{i_0,0}$ *and* $\pi_{T_1}^{i_1,0}$ *have not been queried to the* Challenge *oracle in Phase 1. A bit* $b \in \{0,1\}$ *is chosen uniformly at random, and* $j_0 := 0$*,* $j_1 := 0$*. Initialize oracles* Left *and* Right *as follows:*
  - $\mathsf{Left}(cid, M)$*: Return* $\mathsf{Challenge}(\pi_{T_b}^{i_b,j_b}, id_{S_L}, cid, M)$ *and set* $j_b \leftarrow j_b + 1$.
  - $\mathsf{Right}(cid, M)$*: Return* $\mathsf{Challenge}(\pi_{T_{1-b}}^{i_{1-b},j_{1-b}}, id_{S_R}, cid, M)$ *and set* $j_{1-b} \leftarrow j_{1-b} + 1$.
- ***Phase 3.*** $\mathcal{A}$ *interacts with all oracles in Phases 1 and 2.*
- ***Output Phase.*** $\mathcal{A}$ *outputs a bit* $\hat{b}$*. Define the following credential lists:*
  - $\mathcal{L}_{ch}^r$*: All cid values returned by* $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot, \cdot)$ *queries not made via* Left *or* Right*, for* $T \in \{T_0, T_1\}$ *and* $S \in \{S_L, S_R\}$.
  - $\mathcal{L}_{ch}^a$*: All cid values in inputs of* $\mathsf{Challenge}(\pi_T^{i,j}, id_S, \cdot, \cdot)$ *queries not made via* Left *or* Right*, for* $j > 0$.
  - $\mathcal{L}_{lr}^r$*: All cid values returned by* Left *or* Right *queries when* $j_b = 0$ *or* $j_{1-b} = 0$*, respectively.*
  - $\mathcal{L}_{lr}^a$*: All cid values in inputs of* Left *or* Right *queries when* $j_b > 0$ *or* $j_{1-b} > 0$*, respectively.*

  The experiment outputs 1 if and only if: $\hat{b} = b$*, and*
  - ***Instance Freshness:*** *No* Challenge *query was made using both* $\pi_{T_0}^{i_0,k_0}$ *and* $\pi_{T_1}^{i_1,k_1}$ *for any* k_0, k_1*, and*
  - ***Credential Separation:*** *The following sets are empty:*

$$\begin{aligned} \textbf{\textit{sUnl}}: &\quad \mathcal{S}_{\mathsf{sUnl}} := (\mathcal{L}_{ch}^r \cap \mathcal{L}_{lr}^a) \cup (\mathcal{L}_{lr}^r \cap \mathcal{L}_{ch}^a), \\ \textbf{\textit{mUnl}}: &\quad \mathcal{S}_{\mathsf{mUnl}} := \mathcal{S}_{\mathsf{sUnl}} \cup (\mathcal{L}_{ch}^a \cup \mathcal{L}_{lr}^a), \\ \textbf{\textit{wUnl}}: &\quad \mathcal{S}_{\mathsf{wUnl}} := \mathcal{S}_{\mathsf{mUnl}} \cup (\mathcal{L}_{ch}^r \cup \mathcal{L}_{lr}^r). \end{aligned}$$

For $x \in \{w, m, s\}$*, the adversary's advantage is defined as:*

$$\mathbf{Adv}_{x\mathsf{Unl},\mathrm{PLA}}^{\mathcal{A}} := \big|\Pr[x\mathsf{Unl}_{\mathrm{PLA}}^{\mathcal{A}} = 1] - \frac{1}{2}\big|.$$

Global Revocation. This property addresses the need for users to revoke their cryptographic keys across all servers to which their token is registered, without having to access each server individually.

Definition 3 (Global Revocation) *A PLA scheme* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ *satisfies global revocation if the following two algorithms exist:*

- Revoke *: Takes a master secret key* msk *as input and outputs a revocation key* rk*.*
- CheckCred*: Takes a server identity* id_S*, a credential* $cred$*, and a revocation key* rk *as input, and outputs a bit* $b \in \{0,1\}$*. If* $b = 1$*, the credential is revoked; otherwise, it remains valid.*

The completeness of this definition ensures that when msk *is generated and* Revoke *and* CheckCred *are used as described, the revocation process works correctly.*

Revocation Soundness. This property ensures that only the legitimate owner of the master secret key can revoke keys that are stored on an honest server.

Definition 4 (Revocation Soundness) *For a PLA scheme* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$*, and adversary* $\mathcal{A}$*, the experiment* $\mathsf{rev\text{-}sound}_{\mathrm{PLA}}^{\mathcal{A}}$ *proceeds as follows.*

- ***Setup****: For each token* $T \in \mathcal{T}$*, a master key* msk_T *is created by invoking* $\mathsf{Gen}(par)$.
- ***Online Phase****:* $\mathcal{A}$ *has access to the* Start*,* Challenge*, and* Complete *oracles defined above.*
- ***Output Phase****:* $\mathcal{A}$ *chooses two tuples* (T^*, i_{T^*}) *and* (S^*, i_{S^*})*. The experiment returns* $cred^*$ *to* $\mathcal{A}$ *if* $\mathsf{Complete}(\pi_{S^*}^{i_{S^*},0})$ *has been queried, and outputs 0 otherwise. Then the adversary outputs a revocation key* rk^**.*
- *The experiment returns* 1 *iff* $\pi_{T^*}^{i_{T^*},0}$ *and* $\pi_{S^*}^{i_{S^*},0}$ *are partnered and* $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*) = 1$.

The adversary's advantage in this experiment is defined by the probability that it can output a valid rk^* *that successfully revokes an honest credential.*

$$\mathbf{Adv}_{\mathsf{rev\text{-}sound},\mathrm{PLA}}^{\mathcal{A}} := \Pr[\mathsf{rev\text{-}sound}_{\mathrm{PLA}}^{\mathcal{A}} = 1].$$

Impersonation Security with Global Revocation. Security against impersonation must remain effective even when cryptographic keys are revoked globally and servers experience delayed state updates. The following is the notion of impersonation security with global revocation.

Definition 5 (Impersonation Security-GR) *Let* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ *be a PLA. This security notion is similar to the experiment defined in Definition 1, with the following modification. In the experiment, denoted as* $\mathsf{Imp\text{-}GR}\mathcal{A}^{\mathrm{PLA}}$*, after generating* msk_T *for each* $T \in \mathcal{T}$*, the experiment also computes the revocation key* $rk_T := \mathsf{Revoke}(msk_T)$ *for each* $T \in \mathcal{T}$*. The set* $\{rk_T\}_{T \in \mathcal{T}}$ *is then provided to the adversary* $\mathcal{A}$ *as additional input. The adversary's advantage in this experiment is defined as:*

$$\mathbf{Adv}_{\mathsf{Imp\text{-}GR},\mathrm{PLA}}^{\mathcal{A}} := \Pr[\mathsf{Imp\text{-}GR}_{\mathrm{PLA}}^{\mathcal{A}} = 1].$$

Unlinkability with Global Revocation. In the presence of global revocation, all keys associated with a token become linkable once the corresponding revocation key is published. To account for this, the unlinkability experiment under the global revocation setting grants the adversary access to all revocation keys except those for the challenge tokens. There are also three variants of unlinkability in this setting, denoted as $\mathsf{sUnl\text{-}GR}_{\mathrm{PLA}}^{\mathcal{A}}$, $\mathsf{mUnl\text{-}GR}_{\mathrm{PLA}}^{\mathcal{A}}$, and $\mathsf{wUnl\text{-}GR}_{\mathrm{PLA}}^{\mathcal{A}}$.

Definition 6 (Unlinkability-GR) *Let* $\mathrm{PLA} = (\mathsf{Gen}, \mathsf{Reg}, \mathsf{Auth})$ *be a PLA. This security notion is similar to the experiment defined in Definition 2, with the following modification. In the experiment, denoted as* $x\mathsf{Unl\text{-}GR}$ *(*$\forall x \in \{w, m, s\}$*), after generating* msk_T *for each* $T \in \mathcal{T}$*, the experiment also computes the revocation key* $rk_T := \mathsf{Revoke}(msk_T)$ *for each* $T \in \mathcal{T}$*. In Phase 2, when* $\mathcal{A}$ *outputs* (T_0, T_1)

and (S_L, S_R), the experiment provides $\mathcal{A}$ with the set of all revocation keys except those for the challenge tokens, i.e., $\{rk_T\}_{T\in\mathcal{T}\setminus\{T_0,T_1\}}$. The rest of the experiment follows the procedure of $x\mathsf{Unl}$. For each $x \in \{w, m, s\}$, we define the adversary's advantage as:

$$\mathbf{Adv}^{\mathcal{A}}_{x\mathsf{Unl\text{-}GR},\mathrm{PLA}} := |\Pr[x\mathsf{Unl\text{-}GR}^{\mathcal{A}}_{\mathrm{PLA}} = 1] - \frac{1}{2}|.$$

Appendix B. Overview of bip32PA

bip32PA [3] builds upon rerandomizable ECDSA. Let $\mathbb{G}$ be a group of prime order p with generator g, and let $\mathsf{H}' : \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle. The core algorithms are as follows:

- KeyGen: Sample $sk \leftarrow \mathbb{Z}_p$, and set $pk := g^{sk}$.
- $\mathsf{RandSK}(sk, \rho)$: Return $sk' := sk + \rho$.
- $\mathsf{RandPK}(pk, \rho)$: Return $pk' := pk \cdot g^{\rho}$.
- $\mathsf{Sig}(sk, m)$: Compute $z := \mathsf{H}'(m)$, then sign z using ECDSA.
- $\mathsf{Ver}(pk, \sigma, m)$: Compute $z := \mathsf{H}'(m)$ and verify σ.

Unlike key-prefixed variants that sign $\mathsf{H}'(pk, m)$, bip32PA maintains compatibility with existing FIDO2/WebAuthn deployments by leveraging the injectivity of the server identifier id_S. Specifically, it prefixes each message with $\mathsf{H}_0(id_S)$, where $\mathsf{H}_0 : \{0,1\}^* \to \{0,1\}^{2\lambda}$ is a secure hash function. To ensure deterministic signatures over identical messages, bip32PA derandomizes the signing process by computing signing coins as $\mathsf{H}_1(seed, m)$, with $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{\lambda}$ also modeled as a secure hash.

Main components of bip32PA are presented as follows.

Key Generation. The master secret key msk comprises an ECDSA key pair (sk_0, pk_0), a chaincode ch, and a seed $seed$. These are sampled as follows: $sk_0 \leftarrow \mathbb{Z}_p, pk_0 := g^{sk_0}, ch \leftarrow \{0,1\}^{\lambda}, seed \leftarrow \{0,1\}^{\lambda}$. The chaincode and public key facilitate global revocation, while the seed supports deterministic per-server key derivation.

Registration and Authentication. The protocol follows the standard WebAuthn flow with modified key derivation. For a given server identifier id_S, the token computes: $cid := \mathsf{H}_1(seed, \mathsf{id}_S), \rho := \mathsf{H}_1(pk_0, ch, \mathsf{id}_S)$, and derives the credential key: $sk := sk_0 + \rho, pk := pk_0 \cdot g^{\rho}$. To sign the registration challenge M_r, the token computes: $m := (\mathsf{H}_0(\mathsf{id}_S), cid, pk, M_r), coins := \mathsf{H}_1(seed, m), \sigma_r := \mathsf{Sig}(sk, m; coins)$, and returns the registration response $R_r := (pk, \sigma_r)$. To sign the authentication challenge M_a, it computes: $m := (\mathsf{H}_0(\mathsf{id}_S), M_a), coins := \mathsf{H}_1(seed, m), \sigma_a := \mathsf{Sig}(sk, m; coins)$, and returns the authentication response $R_a := \sigma_a$.

Global Revocation. bip32PA supports efficient global revocation via a revocation key $rk := (pk_0, ch)$. To check whether a credential pk is valid for a given id_S, a verifier recomputes: $pk' := \mathsf{RandPK}(pk_0, \mathsf{H}_1(pk_0, ch, \mathsf{id}_S))$, and accepts if $pk = pk'$. This allows revocation without additional interaction, preserving user privacy and compatibility.

Appendix C. Omitted Proofs for bip32PA-MU and -SU

Lemma 16 *Let $\mathrm{SIG}_{\widetilde{\mathsf{H}}} = (\mathsf{KGen}, \mathsf{RandSK}, \mathsf{RandPK}, \mathsf{Sig}, \mathsf{Ver})$ be the ECDSA signature scheme, and let $\widetilde{\mathsf{H}} : \{0,1\}^* \to \{0,1\}^{\lambda}$ be a random oracle. For any adversary $\mathcal{A}$ making at most $Q_{\widetilde{\mathsf{H}}}$ queries to $\widetilde{\mathsf{H}}$, consider the following game:*

1) *Run $\mathcal{A}^{\widetilde{\mathsf{H}}}$ on input pp to obtain an identifier $id \in \{0,1\}^*$.*
2) *Generate $(sk, pk) \leftarrow \mathsf{KGen}(pp)$ and give (sk, pk) to $\mathcal{A}^{\widetilde{\mathsf{H}}}$.*
3) *When $\mathcal{A}^{\widetilde{\mathsf{H}}}$ outputs (pk', ch, r), output 1 if and only if*

$$\mathsf{RandPK}(pk', \widetilde{\mathsf{H}}(pk, ch, r, id)) = pk.$$

Then, the probability that the game outputs 1 is at most $\frac{Q_{\widetilde{\mathsf{H}}}}{2^{\lambda}}$.

Proof 1 *Let $sk' \in \mathbb{Z}_p$ be such that $g^{sk'} = pk'$. The winning condition then becomes $\widetilde{\mathsf{H}}(pk', ch, r, id) = sk - sk'$. Since pk' uniquely determines sk', each query to $\widetilde{\mathsf{H}}$ has at most a $1/2^{\lambda}$ chance to satisfy the condition. Applying a union bound over $Q_{\widetilde{\mathsf{H}}}$ queries completes the proof.*

Proof 2 *We prove Lemma 1 via a sequence of games. Let p_i denote the probability that Game G_i outputs 1.*

Game *G_0. This is the standard unlinkability game. A master secret key is generated as $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$, and a revocation key is derived for each token $T \in \mathcal{T}$: $rk_T \leftarrow \mathsf{Revoke}(msk_T)$. The adversary $\mathcal{A}$ interacts with the oracles* Start, Challenge, *and* Complete. *It then outputs challenge tokens T_0, T_1 and servers S_L, S_R, receives $\{rk_T\}_{T\in\mathcal{T}\setminus\{T_0,T_1\}}$, and gains access to oracles* Left *and* Right. *These invoke $\mathsf{Challenge}(\pi^{i_b,j_b}_{T_b}, id_{S_L}, \cdot, \cdot)$ and $\mathsf{Challenge}(\pi^{i_{1-b},j_{1-b}}_{T_{1-b}}, id_{S_R}, \cdot, \cdot)$, respectively. $\mathcal{A}$ also has access to random oracles H_0 and H_1. By definition:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{mUnl\text{-}GR},\text{bip32PA-MU}} = |p_0 - \frac{1}{2}|.$$

Game *G_1. Identical to G_0 except it aborts if $\mathcal{A}$ queries $\mathsf{H}_1(pk_{T_j,0}, ch_{T_j}, \cdot, \cdot)$ or $\mathsf{H}_1(seed_{T_j}, \cdot)$ for any challenge token $j \in \{0,1\}$. As $\mathcal{A}$ only sees ch_{T_j} and $seed_{T_j}$ via hash outputs, we use a union bound:*

$$|p_0 - p_1| \le \frac{4Q_{\mathsf{H}_1}}{2^{\lambda}}.$$

Game *G_2. Builds on G_1 and aborts if any two distinct tokens share ch_T, $lrev_T$, or $seed_T$. These are sampled uniformly from $\{0,1\}^{\lambda}$, so by the union bound:*

$$|p_1 - p_2| \le \frac{3|\mathcal{T}|^2}{2^{\lambda}}.$$

Game *G_3. Extends G_2 by aborting if H_0 produces a collision, i.e., $\mathsf{H}_0(x) = \mathsf{H}_0(x')$ for some $x \neq x'$. By the birthday bound:*

$$|p_2 - p_3| \le \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game *G_4. Instead of computing $cid := \mathsf{Enc}(ch, lrev)$ and setting $r := lrev := cid$, we now sample cid uniformly*

from $\{0,1\}^{v^*}$ *on each* $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot)$ *query. We store* $R[T, cid] = lrev$ *and use this mapping to derive* $r = lrev := cid$*. For authentication queries (*$j > 0$*), the same* cid *is reused. By a hybrid argument over tokens, invoking the anonymous authentication security of* SKE*:*

$$|p_3 - p_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth,SKE}}.$$

Game G_5*. We modify* Left*: Instead of computing* $\rho := \mathsf{H}_1(pk_{T_b,0}, ch_{T_b}, r_{T_b}, id_{S_L})$ *and rerandomizing the keys, we generate a fresh key pair* $(sk, pk) \leftarrow \mathsf{Gen}(par)$ *and use it throughout. A lookup table* L_L *stores per-message signing randomness. If a message* m *is new,* $L_L[m]$ *is sampled uniformly. Since the game aborts on collisions,* $\mathcal{A}$*'s view is preserved:*

$$p_4 = p_5.$$

Game G_6*. The oracle* Right *is modified analogously to* Left *in* G_5*. Again, this preserves the adversary's view:*

$$p_5 = p_6.$$

In G_6*, both* Left *and* Right *are independent of the hidden bit* b*, so:*

$$p_6 = \frac{1}{2}.$$

Proof 3 *We prove Lemma 2 using a sequence of games. Let* p_i *denote the probability that Game* G_i *outputs 1.*

Game G_0***.*** *As in the strong unlinkability experiment, each token* $T \in \mathcal{T}$ *has master secret key* $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ *and revocation key* $rk_T \leftarrow \mathsf{Revoke}(msk_T)$*. The adversary* $\mathcal{A}$ *interacts with oracles* Start*,* Challenge*, and* Complete*, picks challenge tokens* T_0, T_1 *and servers* S_L, S_R*, obtains the revocation keys* $\{rk_T\}_{T \in \mathcal{T} \setminus \{T_0, T_1\}}$*, and then uses oracles* Left *and* Right *(both specializations of* Challenge*) with random oracles* H_0 *and* H_1*. By definition:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{sUnl\text{-}GR},\text{bip32PA-SU}} = |p_0 - \frac{1}{2}|.$$

Game G_1***.*** *In this game, at start we pick* $T_0^*, T_1^* \leftarrow \mathcal{T}$*. If the adversary later selects* $(T_0, T_1) \neq (T_0^*, T_1^*)$*, the game abotrs with a random bit. Since* $\mathcal{A}$ *learns nothing about our guess until abort:*

$$|p_1 - \frac{1}{2}| = \frac{1}{|\mathcal{T}|^2} \cdot |p_0 - \frac{1}{2}|.$$

Game G_2***.*** *This game modifies how encryption and decryption work for tokens* T_0^* *and* T_1^**. In* G_1*, each registration with token* T_b *(via* Challenge*,* Left*, or* Right*) produces* $r_{T_b} := \mathsf{Enc}(ch_{T_b}, lrev_{T_b}), cid_{T_b} := \mathsf{Enc}(seed_{T_b}, (id_S, r_{T_b}))$*. Each authentication uses* cid_{T_b} *to retrieve* $(id, r_{T_b}) := \mathsf{Dec}(seed_{T_b}, cid_{T_b})$*, aborting on failure.*

In G_2*, for tokens* T_0^*, T_1^* *only, we replace encryption and decryption with lookup tables: 1) Initialize maps:* $R_0[\cdot], R_1[\cdot], L_0[\cdot], L_1[\cdot]$*. 2) In each registration involving* T_i^**, sample random* $r_i, cid_i \in \{0,1\}^{v^*}$ *and set:* $R_i[r_i] := (ch_{T_i^*}, lrev_{T_i^*}), L_i[cid_i] := (id_S, r_i)$*. 3) In authentication with* T_i^**, if* cid_T *appears in* $L_i[\cdot]$*, use its value. Otherwise, abort. By the anonymous authentication security of* SKE*:*

$$|p_1 - p_2| \leq \mathbf{Adv}^{\mathcal{B}}_{\text{anon-auth, SKE}}.$$

Game G_3***.*** *This game changes how signing keys for tokens* T_0^*, T_1^* *are derived. In* G_2*:* $sk_T := \mathsf{RandSK}(sk_{T,0}, \mathsf{H}_1(pk_{T,0}, ch_T, r_T, id_S))$*. We now introduce a map* $SK_{T_i^*}[\cdot]$*. If undefined, we sample keys uniformly and randomly:* $sk_{T_i^*} := SK_{T_i^*}[r_{T_i^*}, id_S]$*. By the pseudorandomness of* PRF *in* $|\mathcal{T}|$ *hybrid steps:*

$$|p_2 - p_3| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\text{PRF}}.$$

Now the only dependency on bit b *is via the shared maps* $SK_{T_0^*}, SK_{T_1^*}$ *used in challenge oracles.*

Game G_4***.*** *We separate each* $SK_{T_b^*}$ *into two independent maps:* $SK_{T_b^*}$ *for* Challenge*, and* SK_{L_b} *for* Left*. These maps are disjoint;* Challenge *never accesses* SK_{L_b}*, and vice versa. The adversary would notice this only if the same* cid *is returned in registration twice (once in* Left*,* Right*, and once in* Challenge*), or it sends* cid *to one oracle in authentication (e.g.,* Challenge*), which was returned by the other oracle (e.g.,* Left*) in registration. The former happens with probability at most* $2Q_C/|\{0,1\}^{v^*}|$*, and the latter is forbidden by strong credential separation. Hence, we have:*

$$|p_3 - p_4| \leq \frac{2Q_C}{2^{v^*}}.$$

Game G_5***.*** *We apply the same table-splitting to* Right *for token* T_{1-b}^**, namely* $SK'_{T_{1-b}^*}$ *for* Challenge*, and* SK_R *for* Right*. As before, these maps are disjoint and independently sampled. Thus, we have:*

$$p_4 = p_5.$$

In G_5*, the adversary's view is independent of bit* b*, so:*

$$p_5 = \frac{1}{2}.$$

Proof 4 *We prove the revocation soundness property of* bip32PA-MU *(Lemma 3) through a sequence of games. Let* $\mathbf{Adv}_i$ *denote the adversary's advantage in Game* G_i *(i.e., the probability that the game outputs 1).*

Game G_0***:*** *This is the real revocation soundness game. For each token* $T \in \mathcal{T}$*, a key* $msk_T := (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ *is generated. The adversary interacts with the oracles* Start*,* Challenge*, and* Complete*, eventually outputting tuples* (T^*, i_{T^*}) *and* (S^*, i_{S^*})*. It then receives the credential* $cred^* := pk$ *and outputs a revocation key* rk^**. The game outputs 1 if and only if the oracles* $\pi_{T^*}^{i_{T^*},0}$ *and* $\pi_{S^*}^{i_{S^*},0}$ *are partnered and* $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*)$ *returns true. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{rev\text{-}sound},\text{bip32PA-MU}}.$$

Game G_1***:*** *We abort if* $\mathcal{A}$ *queries* $\mathsf{H}_1(pk_{T,0}, ch_T, r, \cdot)$ *for any* $T \in \mathcal{T}$*. Since all knowledge of* ch_T *and* r *is obtained only through the random oracle, applying a union bound over all queries and tokens gives:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{2\lambda}}.$$

Game G_2***:*** *We abort if a collision occurs in* H_1*, i.e., if there exist distinct inputs* $x \neq x'$ *such that* $\mathsf{H}_1(x) = \mathsf{H}_1(x')$*. Since* H_0 *outputs uniformly random values* $\{0,1\}^\lambda$*, and hence:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_{\mathsf{H}_1}^2}{2^\lambda}.$$

Game G_3***:*** *We abort if* $\mathcal{A}$ *outputs* $rk^* = (pk_{T^*,0}, ch_{T^*}, lrev_{T^*})$*. As all information about* ch_{T^*} *and* $lrev_{T^*}$ *is accessible only via random oracle queries, the probability of guessing this correctly is at most* $1/2^{2\lambda}$*:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{1}{2^{2\lambda}}.$$

Game G_4***:*** *We change the generation of credential identifiers and nonces. Instead of computing* $cid := \mathsf{Enc}(ch, lrev)$ *and setting* $r := lrev := cid$*, we sample* cid *uniformly at random from* $\{0,1\}^{v^*}$ *for each registration query* $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot)$*, store* $R[T, cid] = lrev$*, and let* $r = lrev := cid$*. For authentication queries with* $j > 0$*, the same* cid *is reused to derive the secret key. Applying a hybrid argument over the tokens and using the anonymous authentication security of* SKE *yields:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game G_5***:*** *We sample a random index* $I \leftarrow [Q_C]$ *at the beginning and abort if the* I*th* Challenge *query is not the first registration query for the pair* (T^*, id_{S^*})*. Since such a query must exist for* $\mathcal{A}$ *to win, it follows:*

$$\mathbf{Adv}_4 = Q_C \cdot \mathbf{Adv}_5.$$

Game G_6***:*** *We modify how the* I*th* Challenge *query is handled. Normally, a key pair* (sk, pk) *is derived via rerandomization:* $\rho := \mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*})$*, then* $sk := \mathsf{RandSK}(sk_{T^*,0}, \rho)$ *and* $pk := \mathsf{RandPK}(pk_{T^*,0}, \rho)$*.*

Instead, we now generate $(sk, pk) \leftarrow \mathsf{KGen}(par)$ *freshly and program the random oracle as:* $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*}) := sk - sk_{T^*,0}$*. Since rerandomized keys are statistically indistinguishable from fresh keys when* ρ *is uniform, we have:*

$$\mathbf{Adv}_5 = \mathbf{Adv}_6.$$

Finally, by Lemma 16, any adversary succeeding in G_6 *implies a break of the underlying rerandomization assumption. A reduction simulating* G_6 *forwards* H_1 *queries to* H*:*

$$\mathbf{Adv}_6 \leq \frac{Q_{\mathsf{H}_1}}{2^\lambda}.$$

Proof 5 *We prove the revocation soundness of* bip32PA-SU *(Lemma 4) using a sequence of games. Let* $\mathbf{Adv}_i$ *denote the adversary's advantage in Game* G_i*, namely, the probability that the game outputs 1.*

Game G_0*. This is the standard revocation soundness game. For each token* $T \in \mathcal{T}$*, a master secret key is generated as* $msk_T := (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$*. The adversary interacts with oracles* Start*,* Challenge*, and* Complete*, then outputs tuples* (T^*, i_{T^*}) *and* (S^*, i_{S^*})*. Upon receiving a credential* $cred^* := pk$*, it outputs a revocation key* rk^**. The game outputs 1 if: The oracles* $\pi_{T^*}^{i_{T^*},0}$ *and* $\pi_{S^*}^{i_{S^*},0}$ *are partnered, and* $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*)$ *returns true. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{rev\text{-}sound},\mathrm{bip32PA\text{-}SU}}.$$

Game G_1*. We abort if* $\mathcal{A}$ *queries* $\mathsf{H}_1(pk_{T,0}, ch_T, r, \cdot)$ *for any* $T \in \mathcal{T}$*. As* ch_T *and* r *are only obtainable through random oracle queries, a union bound gives:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{2\lambda}}.$$

Game G_2*. We now abort if a collision occurs in* H_1*, i.e., there exist distinct inputs* $x \neq x'$ *such that* $\mathsf{H}_1(x) = \mathsf{H}_1(x')$*. This event is bounded by:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_{\mathsf{H}_1}^2}{2^\lambda}.$$

Game G_3*. We abort if* $\mathcal{A}$ *outputs the revocation key* $rk^* = (pk_{T^*,0}, ch_{T^*}, lrev_{T^*})$*. As* ch_{T^*} *and* $lrev_{T^*}$ *are revealed only via random oracle queries:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{1}{2^{2\lambda}}.$$

Game G_4*. We now modify the generation of* r*,* cid*, and* $lrev$*. For each registration query* $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot)$*: 1) Sample* r *and* cid *uniformly from* $\{0,1\}^{v^*}$*; 2) Store* $R_1[T, r] = lrev$ *and* $R_2[T, cid] = (id_S, r)$*; 3) Set* $lrev := r$*. For authentication queries (*$j > 0$*), retrieve* $(id, r) := R_2[T, cid]$ *instead of decrypting* cid*. A hybrid argument over all tokens yields:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game G_5*. We randomly choose an index* $I \leftarrow [Q_C]$ *and abort if the* I*th* Challenge *query is not the first registration query for* (T^*, id_{S^*})*. Since a successful adversary must issue such a query:*

$$\mathbf{Adv}_4 = Q_C \cdot \mathbf{Adv}_5.$$

Game G_6*. We modify the* I*th* Challenge *query: instead of using rerandomization* $\rho := \mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*})$ *to derive a key, we generate* $(sk, pk) \leftarrow \mathsf{KGen}(par)$ *freshly and program the random oracle as:* $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*}) := sk - sk_{T^*,0}$*. Since rerandomized and freshly generated keys are statistically indistinguishable, we have:*

$$\mathbf{Adv}_5 = \mathbf{Adv}_6.$$

Finally, by Lemma 16, we can bound the adversary's success in G_6 *via a reduction that forwards* H_1 *queries to a simulated oracle* $\widetilde{\mathsf{H}}$*:*

$$\mathbf{Adv}_6 \leq \frac{Q_{\mathsf{H}_1}}{2^\lambda}.$$

Proof 6 *We prove Lemma 5 via a sequence of games. We denote by* G_i *the* i*-th game in the sequence, and* $\mathbf{Adv}_i$ *the probability that the game outputs 1.*

Game G_0***:*** *This is the original impersonation game with global revocation. For each token* $T \in \mathcal{T}$*, a master secret*

key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ is generated. The adversary is given the global revocation keys $rk_T := (pk_{T,0}, ch_T, lrev_T)$ and access to oracles Start, Challenge, *and* Complete. *By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Imp\text{-}GR},\mathrm{bip32PA\text{-}MU}}.$$

Game G_1***:*** *Abort if there exists a collision in the hash function* H_0*, i.e.,* $\mathsf{H}_0(x) = \mathsf{H}_0(x')$ *for* $x \neq x'$*. Since* H_0 *outputs uniformly in* $\{0,1\}^{2\lambda}$*, the difference is bounded by:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game G_2***:*** *Abort if the same random string rs is sampled in two* Start *oracle queries. Since $rs \in \{0,1\}^{\leq\lambda}$, the collision probability satisfies:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q^2_S}{2^{\lambda}}.$$

Game G_3***:*** *Abort if the adversary queries $\mathsf{H}_1(seed_T, \cdot)$ for any $T \in \mathcal{T}$. As each $seed_T$ is uniformly random:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}}.$$

Game G_4***:*** *We modify how credential identifiers and nonces are generated. For each* Challenge *query during registration ($j = 0$), a random $cid \leftarrow \{0,1\}^{v^*}$ is sampled and stored as $R[T, cid] := lrev$. Then, we set $r := lrev := cid$. For authentication queries ($j > 0$), the same cid is reused. By a hybrid argument over the tokens and the anonymous authentication security of* SKE *at each step:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game G_5***:*** *Abort if two distinct revocation nonces $r_1 \neq r_2$ yield the same value under* H_1*, i.e.,* $\mathsf{H}_1(\cdot,\cdot,r_1,\cdot) = \mathsf{H}_1(\cdot,\cdot,r_2,\cdot)$*, which may allow multiple signatures under the same derived key. Since the image of* H_1 *is uniformly distributed in* $\{0,1\}^{\lambda}$*, we get:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq \frac{Q^2_{\mathsf{H}_1}}{2^{\lambda}}.$$

Game G_6***:*** *Abort if there is a collision in server identifiers under* H_0*, i.e.,* $\mathsf{H}_0(id_S) = \mathsf{H}_0(id_{S'})$ *for* $S \neq S' \in \mathcal{S}$*. Since each id_S is unique:*

$$|\mathbf{Adv}_5 - \mathbf{Adv}_6| \leq \frac{|\mathcal{S}|^2}{2^{2\lambda}}.$$

Before proceeding, define a successful impersonation as one where: 1) No aborts occur; 2) The adversary completes an authentication via $\mathsf{Complete}(\pi^{i,j}_S, \cdot)$ *for $j > 0$; 3) $\pi^{i,j}_S$ has no partner; and 4) $\pi^{i,0}_S$ is partnered with some $\pi^{i',0}_T$. This is referred to as a* forged authentication *and the related* target registration*.*

Game G_7***:*** *Select a random token $T^* \notin \mathcal{T}$ at the start. Abort unless the target registration uses T^*. Since T^* is chosen uniformly:*

$$\mathbf{Adv}_6 = |\mathcal{T}| \cdot \mathbf{Adv}_7.$$

Game G_8***:*** *For* Challenge *queries involving T^*, signatures are generated using pre-sampled coins (rather than derived from $\mathsf{H}_1(seed_{T^*}, m)$). This transformation is conceptual, justified by the uniformity introduced in G_3:*

$$\mathbf{Adv}_7 = \mathbf{Adv}_8.$$

We now reduce to the uf-hrk-idx1 *security of* $\mathrm{SIG}_{\widetilde{\mathsf{H}}}$ *via an adversary $\mathcal{B}'$ with access to a signing oracle* SignO *and a randomness oracle* RandO*:*

- *$\mathcal{B}'$ receives a challenge public key pk^* and selects T^* randomly. It sets $pk_{T^*,0} := pk^*$, samples $ch_{T^*}, seed_{T^*}$, and defines $rk_{T^*} := (pk_{T^*,0}, ch_{T^*})$. For all other tokens, keys are generated honestly. All revocation keys rk_T are provided to $\mathcal{A}$.*
- *$\mathcal{B}'$ simulates oracles for $\mathcal{A}$:*
  1) *For* H_1*: if queried on $(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_S)$, return* $\mathsf{RandO}(\mathsf{H}_0(id_S))$*; otherwise, use lazy sampling.*
  2) *For* $\mathsf{Challenge}(\pi^{i,j}_T, \cdot, \cdot)$*: If $T \neq T^*$, behave as in G_8; if $T = T^*$ and $L_\sigma[id_S, m]$ is undefined, compute $idx := \mathsf{H}_0(id_S)$, query* $\mathsf{RandO}(idx)$*, and obtain a signature on $\widehat{m} := (\mathsf{H}_0(id_S), m)$ using* SignO*.*
- *Upon receiving $\mathcal{A}$'s forgery, $\mathcal{B}'$ outputs (m^*, σ^*) with $m^* = \mathsf{H}_0(rs)$, $idx^* = \mathsf{H}_0(id_S)$.*

Since $\mathcal{B}'$ perfectly simulates G_8 and extracts a valid forgery for a fresh message:

$$\mathbf{Adv}_8 \leq \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{uf\text{-}hrk\text{-}idx1},\mathrm{SIG}_{\widetilde{\mathsf{H}}}}.$$

Proof 7 *We prove Lemma 6 via a sequence of games. Let G_i denote the i-th game in the sequence, and $\mathbf{Adv}_i$ denote the probability that Game G_i outputs 1.*

Game G_0***:*** *This is the original impersonation game with global revocation. A master secret key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ is generated for each token $T \in \mathcal{T}$, and the global revocation key $rk_T := (pk_{T,0}, ch_T, lrev_T)$ is given to the adversary. The adversary $\mathcal{A}$ can query the oracles* Start, Challenge, *and* Complete*. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Imp\text{-}GR},\mathrm{bip32PA\text{-}SU}}.$$

Game G_1***:*** *Abort if a collision occurs in* H_0*, i.e.,* $\mathsf{H}_0(x) = \mathsf{H}_0(x')$ *for* $x \neq x'$*. Since* H_0 *outputs uniformly in* $\{0,1\}^{2\lambda}$*:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game G_2***:*** *Abort if the same nonce $rs \in \{0,1\}^{\leq\lambda}$ is sampled in two* Start *queries:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q^2_S}{2^{\lambda}}.$$

Game G_3***:*** *Abort if $\mathcal{A}$ queries $\mathsf{H}_1(seed_T, \cdot)$ for any $T \in \mathcal{T}$. As $seed_T$ is uniform, this occurs with probability:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}}.$$

Game G_4***:*** *Replace encryption computations with uniformly random sampling. For any* Challenge *query on T, generate*

r, cid uniformly from $\{0,1\}^{v^*}$, store $R_1[T,r] = lrev$, $R_2[T,cid] = (id_S, r)$, and set $lrev := r$. During authentication queries (with $j > 0$), recover $(id_S, r) := R_2[T, cid]$. By a hybrid argument:

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game *G_5: Abort if two distinct revocation nonces $r_1 \neq r_2$ cause a collision in H_1. This ensures no derived key collision from distinct inputs:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq \frac{Q^2_{\mathsf{H}_1}}{2^\lambda}.$$

Game *G_6: Abort if server ID collisions occur: $\mathsf{H}_0(id_S) = \mathsf{H}_0(id_{S'})$ for $S \neq S'$. As H_0 is uniform:*

$$|\mathbf{Adv}_5 - \mathbf{Adv}_6| \leq \frac{|\mathcal{S}|^2}{2^{2\lambda}}.$$

Before proceeding, we define success conditions: the game outputs 1 if 1) no abort occurs, 2) the adversary completes an authentication via $\mathsf{Complete}(\pi^{i,j}_S, \cdot)$ with $j > 0$, 3) the run $\pi^{i,j}_S$ has no partner on the token side, and 4) its pre-run $\pi^{i,0}_S$ partners with some token session $\pi^{i',0}_T$. We call this a forged authentication *with* target registration*.*

Game *G_7: A random token $T^* \notin \mathcal{T}$ is selected. Abort if T^* is not used in the target registration. Since T^* is hidden from $\mathcal{A}$, we get:*

$$\mathbf{Adv}_6 = |\mathcal{T}| \cdot \mathbf{Adv}_7.$$

Game *G_8: Maintain a map $L_\sigma[\cdot,\cdot]$. For $\mathsf{Challenge}$ queries on T^*, generate signatures using pre-sampled coins instead of computing $\mathsf{H}_1(seed_{T^*}, m)$. Since G_3 already ensures uniformity of H_1, this change is conceptual:*

$$\mathbf{Adv}_7 = \mathbf{Adv}_8.$$

We now reduce to the $\mathsf{uf\text{-}hrk\text{-}idx1}$ security of $\mathrm{SIG}_{\widetilde{\mathsf{H}}}$. Construct adversary $\mathcal{B}'$ with access to a signing oracle SignO and randomness oracle RandO:

- *$\mathcal{B}'$ receives challenge public key pk^*, selects random target token T^*, and sets $pk_{T^*,0} := pk^*$, samples $ch_{T^*}, seed_{T^*}$, and sets $rk_{T^*} := (pk_{T^*,0}, ch_{T^*})$. Keys for $T \in \mathcal{T} \setminus \{T^*\}$ are generated honestly. All revocation keys rk_T are given to $\mathcal{A}$.*
- *Oracles are simulated as follows:*
  1) *H_1: On input $(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_S)$, returns $\mathsf{RandO}(\mathsf{H}_0(id_S))$. Other queries use lazy sampling.*
  2) *$\mathsf{Challenge}(\pi^{i,j}_T, \cdot, \cdot)$: If $T \neq T^*$, proceed as in G_8. If $T = T^*$ and $L_\sigma[id_S, m]$ is undefined, compute $idx := \mathsf{H}_0(id_S)$, query $\mathsf{RandO}(idx)$, then use SignO to sign $\widehat{m} := (\mathsf{H}_0(id_S), m)$ at index idx. Store signature in $L_\sigma[id_S, m]$.*
- *Upon output, $\mathcal{B}'$ extracts a forgery (m^*, σ^*), with $m^* = \mathsf{H}_0(rs)$ and index $idx^* = \mathsf{H}_0(id_S)$.*

As $\mathcal{B}'$ perfectly simulates Game G_8 and produces a valid forgery for a fresh message, it holds that:

$$\mathbf{Adv}_8 \leq \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{uf\text{-}hrk\text{-}idx1},\mathrm{SIG}_{\widetilde{\mathsf{H}}}}.$$

Appendix D.
Omitted Proofs for bip32PA$^+$, bip32PA-MU$^+$ and bip32PA-SU$^+$

Proof 8 *We prove Lemma 7 using a sequence of games. Let G_i denote the i-th game in the sequence, and let p_i denote the probability that G_i outputs 1.*

Game *G_0. This is the real weak unlinkability game. A master secret key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T)$ is generated. For each token $T \in \mathcal{T}$, a revocation key $rk_T \leftarrow \mathsf{Revoke}(msk_T)$ is derived. The adversary $\mathcal{A}$ has access to oracles Start, $\mathsf{Challenge}$, and $\mathsf{Complete}$, and outputs challenge tokens T_0, T_1 and servers S_L, S_R. It is then given all revocation keys $\{rk_T\}_{T \in \mathcal{T} \setminus \{T_0, T_1\}}$ and access to two special oracles, Left and Right, which internally run $\mathsf{Challenge}(\pi^{i_b, j_b}_{T_b}, id_{S_L}, \cdot)$, and $\mathsf{Challenge}(\pi^{i_{1-b}, j_{1-b}}_{T_{1-b}}, id_{S_R}, \cdot)$, respectively.*

Unlike standard unlinkability games, we allow $\mathcal{A}$ to submit authentication queries involving the challenge credential identifiers *cid_{T_0} or cid_{T_1} to the $\mathsf{Challenge}$ oracle. However, the authentication message $\widehat{M}_a$ is only allowed to overlap with the corresponding registration response $\widehat{M}_r$ in the cid field. Any additional overlap causes the game to abort. This models a refined notion of unlinkability that reflects realistic server behavior and excludes trivial linkability. By definition, the adversary's advantage is:*

$$\mathbf{Adv}^{\mathsf{wUnl\text{-}GR},\mathrm{bip32PA}^+}_{\mathcal{A}} = |p_0 - \frac{1}{2}|.$$

Game *G_1. Same as G_0, except the game aborts if $\mathcal{A}$ queries the random oracle H_1 on inputs $(pk_{T_j;0}, ch_{T_j}, \cdot)$ or $(seed_{T_j}, \cdot)$ for $j \in \{0,1\}$. Since $\mathcal{A}$ does not directly observe ch_{T_j} or $seed_{T_j}$, such queries may reveal information about b. Applying the union bound over Q_{H_1} total queries and two challenge tokens:*

$$|p_0 - p_1| \leq \frac{4 \cdot Q_{\mathsf{H}_1}}{2^\lambda}.$$

Game *G_2. Same as G_1, but the game now aborts if any two tokens $T \neq T'$ have identical values in any of the fields ch_T, $lrev_T$, or $seed_T$. These values are sampled uniformly and independently, so the probability of a collision is bounded by:*

$$|p_1 - p_2| \leq \frac{3 \cdot |\mathcal{T}|^2}{2^\lambda}.$$

Game *G_3. Same as G_2, but G_3 also aborts if a collision occurs in hash function H_0, namely, if there exist $x \neq x'$ such that $\mathsf{H}_0(x) = \mathsf{H}_0(x')$. By the standard birthday bound:*

$$|p_2 - p_3| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game *G_4. Same as G_3, but we change the derivation of locking keys in the oracles. Previously, the locking key was computed as $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$. Now, we introduce a map $LK[\cdot,\cdot,\cdot]$. For each registration query $\mathsf{Challenge}(\pi^{i,0}_T, id_S, \cdot)$, we sample $lk \leftarrow \{0,1\}^{l_2}$ uniformly at random and store $LK[T, cid, M_a] := (id_S, sk)$. For*

each authentication query $\mathsf{Challenge}(\pi_T^{i,j}, id_S, cid, \cdot)$ *with* $j > 0$*, the simulator uses* $(id_S, sk) := LK[T, cid, M_a]$ *if defined, and aborts otherwise. This change can be simulated through a sequence of at most* $|\mathcal{T}|$ *hybrid games. By the pseudorandomness of* PRF*:*

$$|p_3 - p_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{PRF}}.$$

Game G_5**.** *Same as* G_4*, but we now simulate oracle* Left*. We generate a fresh credential identifier* cid_L *and key pair* $(sk_L, pk_L) \leftarrow \mathsf{Gen}(par)$*, and maintain a map* L_L *for signature randomness. Upon the first query,* Left *returns* (cid_L, pk_L) *and signs messages* m_r*,* m_a *using* sk_L *with randomness* $L_L[m_r]$*,* $L_L[m_a]$*, sampled uniformly if undefined. On subsequent queries, it aborts if* $cid \neq cid_L$*, and otherwise signs as above. Since* cid_L *is fresh and randomness is uniform, this simulation is indistinguishable to* $\mathcal{A}$ *even if it queries* cid_L *in an authentication message (as long as overlap with registration is limited to* cid*). Hence:*

$$p_4 = p_5.$$

Game G_6**.** *Same as* G_5*, but we simulate oracle* Right *similarly. A fresh* cid_R *and key pair* $(sk_R, pk_R) \leftarrow \mathsf{Gen}(par)$ *are generated, and a randomness map* L_R *is maintained. On queries involving* cid_R*, the simulator returns signatures under* sk_R *with independently sampled randomness. As with* Left*, the simulation is indistinguishable and yields:*

$$p_5 = p_6.$$

In Game G_6*, both* Left *and* Right *are simulated using fresh keys and randomness independent of the challenge bit* b*. Even though* $\mathcal{A}$ *may query* cid_{T_0} *or* cid_{T_1} *in authentication messages, as long as the overlap with the registration message is restricted to* cid*, responses remain independent of* b*. Therefore, the adversary's view is independent of the challenge bit, and:*

$$p_6 = \frac{1}{2}.$$

Proof 9 *We prove Lemma 8 via a sequence of games. For each game* G_i*, let* p_i *denote the probability that* G_i *outputs* 1*.*

Game G_0**.** *This is the real medium-unlinkability game. For each token* $T \in \mathcal{T}$*, the challenger generates* $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ *and derives the revocation key* $rk_T \leftarrow \mathsf{Revoke}(msk_T)$*. The adversary* $\mathcal{A}$ *may query* Start*,* Challenge*, and* Complete*, and selects two challenge tokens* T_0, T_1 *and two servers* S_L, S_R*. It also receives all* rk_T *for* $T \neq T_0, T_1$*, and can invoke:* Left : $\mathsf{Challenge}(\pi_{T_b}^{i_b,j_b}, id_{S_L}, \cdot)$*,* Right : $\mathsf{Challenge}(\pi_{T_{1-b}}^{i_{1-b},j_{1-b}}, id_{S_R}, \cdot)$*.*

In the refined medium-unlinkability game, $\mathcal{A}$ *is allowed to use the challenge credential identifier (*cid_{T_0} *or* cid_{T_1}*) in authentication queries, provided that the authentication message* $\widehat{M_a}$ *overlaps the corresponding registration response* $\widehat{M_r}$ *only in the* cid *field; any further overlap causes the game to abort. By definition, the adversary's advantage is:*

$$\mathbf{Adv}^{\mathsf{mUnl\text{-}GR},\mathrm{bip32PA\text{-}MU}^+}_{\mathcal{A}} = |p_0 - \frac{1}{2}|.$$

Game G_1**.** *Same as* G_0*, but the game aborts if* $\mathcal{A}$ *ever queries the random oracle* H_1 *on* $(pk_{T_j;0}, ch_{T_j}, \cdot)$ *or* $(seed_{T_j}, \cdot)$ *for* $j \in \{0, 1\}$*. Since* ch_{T_j} *and* $seed_{T_j}$ *are not revealed, such a query leaks information. By a union bound over both challenge tokens and at most* Q_{H_1} *queries:*

$$|p_0 - p_1| \leq \frac{4 \cdot Q_{\mathsf{H}_1}}{2^\lambda}.$$

Game G_2**.** *Same as* G_1*, but the game now aborts if any two distinct tokens* $T \neq T'$ *satisfy* $ch_T = ch_{T'}$*,* $lrev_T = lrev_{T'}$*, or* $seed_T = seed_{T'}$*. Since all these values are sampled uniformly at random and independently:*

$$|p_1 - p_2| \leq \frac{3 \cdot |\mathcal{T}|^2}{2^\lambda}.$$

Game G_3**.** *Same as* G_2*, but we now abort if a collision occurs in the hash function* H_0*, i.e., if there exist* $x \neq x'$ *such that* $\mathsf{H}_0(x) = \mathsf{H}_0(x')$*. Then:*

$$|p_2 - p_3| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game G_4**.** *We now replace the construction of* $cid_T := \mathsf{Enc}(ch_T, lrev_T)$ *with a uniformly random sample from* $\{0,1\}^{v^*}$ *for each call to* $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot)$*. We store* $R[T, cid_T] := lrev_T$ *and set* $r_T := lrev_T := cid_T$*. Authentication queries (with* $j > 0$*) reuse the same* cid_T *to derive secret keys. A hybrid argument over all tokens using the anonymous authentication security of* SKE *gives:*

$$|p_3 - p_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game G_5**.** *Same as* G_4*, but we now alter the derivation of secret signing and locking keys. Instead of computing:* $sk_T := \mathsf{RandSK}(sk_{T,0}, \mathsf{H}_1(pk_{T,0}, ch_T, r_T, id_S))$*,* $lk_T := \mathsf{PRF}(msk_T, (cid_T, id_S, M_a))$*, we introduce two maps* $SK_T[\cdot]$ *and* $LK_T[\cdot]$ *for each token* T*. If* $SK_T[r_T, id_S]$ *or* $LK_T[cid_T, id_S, M_a]$ *is undefined, we sample it uniformly at random. Then we use these values as the secret keys. By pseudorandomness of* PRF *in a hybrid over all tokens:*

$$|p_4 - p_5| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\mathrm{PRF}}.$$

Game G_6**.** *Same as* G_5*, but we conceptually split the maps* SK_T^b *and* LK_T^b *used in both* $\mathsf{Challenge}(\pi_{T_b}^{i,j}, \cdot)$ *and oracle* Left*. Now,* SK_T^b *and* LK_T^b *are used only for the challenge token, while independent maps* $SK_L[\cdot]$ *and* $LK_L[\cdot]$ *are used in* Left*. Since the refined medium-unlinkability game enforces credential separation (i.e., any shared* (cid_T, id_S) *implies* $S = S_L$ *and distinct* $M_r \neq M_a$*), we get:*

$$p_5 = p_6.$$

Game G_7**.** *Similar to* G_6*, but now we also split the maps for token* T_{1-b} *and oracle* Right*. Maps* SK_T^{1-b} *and* LK_T^{1-b} *are used only in* Challenge*, while* $SK_R[\cdot]$ *and* $LK_R[\cdot]$ *are used in* Right*. Again, credential separation ensures that:*

$$p_6 = p_7.$$

We note that the transition from G_5 *to* G_7 *would not hold in a strong unlinkability game, as the adversary could*

use a new cid_T^* in both Left *and* Challenge*, potentially observing inconsistent behavior.*

In the final game G_7, the behavior of oracles Left *and* Right *is independent of the bit b. Therefore:*

$$p_7 = \frac{1}{2}.$$

Proof 10 *We prove Lemma 9 via a sequence of games. Let p_i denote the probability that Game G_i outputs 1.*

Game *G_0. This is the real strong-unlinkability game. For each token $T \in \mathcal{T}$, the challenger generates $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$, and derives its revocation key $rk_T \leftarrow \mathsf{Revoke}(msk_T)$. The adversary $\mathcal{A}$ may interact via the* Start*,* Challenge*, and* Complete *oracles. It then selects two challenge tokens T_0, T_1 and servers S_L, S_R. For all $T \neq T_0, T_1$, it receives rk_T and may invoke:* Left : $\mathsf{Challenge}(\pi_{T_b}^{i_b,j_b}, id_{S_L}, \cdot)$, Right : $\mathsf{Challenge}(\pi_{T_{1-b}}^{i_{1-b},j_{1-b}}, id_{S_R}, \cdot)$.

In the refined strong-unlinkability game, $\mathcal{A}$ may reuse the challenge credential identifiers (cid_{T_0} or cid_{T_1}) in authentication queries, provided the message $\widehat{M}_a$ overlaps with the registration response $\widehat{M}_r$ only in its cid – any further overlap causes an abort. By definition, the advantage is:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{sUnl\text{-}GR},\,\mathrm{bip32PA\text{-}SU}^+} = |p_0 - \frac{1}{2}|.$$

Game *G_1. We modify G_0 as follows. At the beginning, G_1 samples $T_0^*, T_1^* \leftarrow \mathcal{T}$. If the adversary later selects $(T_0, T_1) \neq (T_0^*, T_1^*)$, the game returns a random bit and aborts. Otherwise, it continues as G_0. Since $\mathcal{A}$ learns nothing about T_0^*, T_1^* before the abort, we get:*

$$|p_1 - \frac{1}{2}| = \frac{1}{|\mathcal{T}|^2} \cdot |p_0 - \frac{1}{2}|.$$

Game *G_2. This game modifies how encryption and decryption work for tokens T_0^*, T_1^*. In G_1, each registration with token T_b (via* Challenge*,* Left*, or* Right*) produces ciphertexts $r_{T_b} := \mathsf{Enc}(ch_{T_b}, lrev_{T_b}), cid_{T_b} := \mathsf{Enc}(seed_{T_b}, (id_S, r_{T_b}))$. Each authentication uses cid_{T_b} to retrieve $(id, r_{T_b}) := \mathsf{Dec}(seed_{T_b}, cid_{T_b})$, aborting on failure.*

In G_2, for tokens T_0^*, T_1^* only, we replace encryption and decryption with lookup tables: 1) Initialize maps: $R_0[\cdot], R_1[\cdot], L_0[\cdot], L_1[\cdot]$. 2) In each registration involving T_i^*, sample random $r_i, cid_i \in \{0,1\}^{v^*}$ and set: $R_i[r_i] := (ch_{T_i^*}, lrev_{T_i^*}), L_i[cid_i] := (id_S, r_i)$. 3) In authentication with T_i^*, if cid_T appears in $L_i[\cdot]$, use its value. Otherwise, abort. By the anonymous authentication security of SKE*:*

$$|p_1 - p_2| \leq \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\,\mathrm{SKE}}.$$

Game *G_3. We change how signing and locking keys for tokens T_0^*, T_1^* are derived. In G_2: $sk_T := \mathsf{RandSK}(sk_{T,0}, \mathsf{H}_1(pk_{T,0}, ch_T, r_T, id_S)), lk_T := \mathsf{PRF}(msk_T, (cid_T, id_S, M_a))$. Now, introduce maps $SK_{T_i^*}[\cdot]$ and $LK_{T_i^*}[\cdot]$. If undefined, sample keys uniformly at random: $sk_{T_i^*} := SK_{T_i^*}[r_{T_i^*}, id_S], lk_{T_i^*} := LK_{T_i^*}[cid_{T_i^*}, id_S, M_a]$. Using pseudorandomness of* PRF *in $|\mathcal{T}|$ hybrid steps:*

$$|p_2 - p_3| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\,\mathrm{PRF}}.$$

The only dependency on bit b now lies in the shared maps $SK_{T_0^*}, SK_{T_1^*}, LK_{T_0^*}, LK_{T_1^*}$ used in challenge oracles.

Game *G_4. We separate the maps used in* Challenge *and* Left*. Specifically,* Challenge *uses: $SK_{T_b^*}, LK_{T_b^*}$, and* Left *uses: SK_{L_b}, LK_{L_b}. These maps are disjoint;* Challenge *never accesses SK_{L_b}, LK_{L_b}, and vice versa. The adversary can detect this only if the same cid is returned in registration twice (once in* Left*,* Right*, and once in* Challenge*), or it sends $(cid, \widehat{M}_a := (M_a, \widehat{\sigma}_a, M'_a))$ to one oracle in authentication (e.g.,* Challenge*), where $\widehat{\sigma}_a$ was returned by the other oracle (e.g.,* Left*) in registration. The former only occurs with probability at most $2Q_C/|\{0,1\}^{v^*}|$, and the latter is forbidden due to the refined strong credential separation. It follows that:*

$$|p_3 - p_4| \leq \frac{2Q_C}{2^{v^*}}.$$

Game *G_5. We apply the same map-splitting to oracle* Right *for token T_{1-b}^*:* Challenge *uses:$SK'_{T_{1-b}^*}, LK'_{T_{1-b}^*}$, and* Right *uses: SK_R, LK_R. As before, these maps are disjoint and independently sampled. Thus:*

$$p_4 = p_5.$$

In G_5, the adversary's view is independent of bit b, so $p_5 = \frac{1}{2}$.

Proof 11 *We prove the revocation soundness of* $\mathrm{bip32PA}^+$ *(Lemma 10) using a sequence of games. Let $\mathbf{Adv}_i$ denote the adversary's success probability in Game G_i.*

Game *G_0: This is the real revocation soundness game. For each token $T \in \mathcal{T}$, a master secret key tuple is generated as $msk_T := (sk_{T,0}, pk_{T,0}, ch_T, seed_T)$. The adversary interacts with the oracles* Start*,* Challenge*, and* Complete*, and eventually outputs a tuple (T^*, i_{T^*}) and (S^*, i_{S^*}). After receiving the credential $cred^* := pk$, it outputs a revocation key rk^*. The game outputs 1 if and only if the sessions $\pi_{T^*}^{i_{T^*},0}$ and $\pi_{S^*}^{i_{S^*},0}$ are partnered and* $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*)$ *returns true. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{rev\text{-}sound},\mathrm{bip32PA}^+}.$$

Game *G_1: This game is the same as G_0, but we introduce an abort condition: if the adversary $\mathcal{A}$ queries $\mathsf{H}_1(pk_{T,0}, ch_T, \cdot)$ for any token $T \in \mathcal{T}$, the game aborts. Since ch_T is secret and uniformly random, the only way $\mathcal{A}$ could guess them is by chance. By the union bound over all Q_{H_1} queries and $|\mathcal{T}|$ tokens, we have:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^\lambda}.$$

Game *G_2: In this game, we further abort if a collision occurs in H_1, i.e., if there exist distinct inputs $x \neq x'$ such that $\mathsf{H}_1(x) = \mathsf{H}_1(x')$. Since H_0 outputs uniformly random values in $\{0,1\}^\lambda$, the probability of such a collision among Q_{H_1} queries is bounded by the birthday bound:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_{\mathsf{H}_1}^2}{2^\lambda}.$$

Game G_3***:*** *We now abort if $\mathcal{A}$ outputs the revocation key $rk^* = (pk_{T^*,0}, ch_{T^*})$, meaning they perfectly reconstructed the internal revocation tuple. Since ch_{T^*} is uniformly random and unknown to $\mathcal{A}$ except via oracle queries, the chance of correctly guessing both is at most $1/2^\lambda$:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{1}{2^\lambda}.$$

Game G_4***:*** *This game modifies how locking keys are generated. In Game G_3, they were derived deterministically as: $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$. Now, for each token $T \in \mathcal{T}$, we introduce a table $LK_T[\cdot]$. On input (cid, id_S, M_a), if the value is undefined, we sample $LK_T[cid, id_S, M_a]$ uniformly at random; otherwise, we reuse the stored value. We then define $lk := LK_T[cid, id_S, M_a]$.*

Replacing the PRF with a truly random function changes the distribution only if PRF *is not secure. By a standard PRF-hybrid argument over $|\mathcal{T}|$ tokens, we get:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{prf},\mathrm{PRF}}.$$

Game G_5***:*** *At the beginning of this game, we select a random index $I \in [Q_C]$, where Q_C is the number of* Challenge *queries made. We abort if the Ith* Challenge *query is not the first registration query for the pair (T^*, id_{S^*}). Since $\mathcal{A}$ must issue such a query in any successful attack, and I is selected uniformly, we obtain:*

$$\mathbf{Adv}_4 = Q_C \cdot \mathbf{Adv}_5.$$

Game G_6***:*** *This game alters the key generation in the Ith* Challenge *query. In Game G_5, the derived key pair was computed using: $\rho := \mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, id_{S^*}), sk := \mathsf{RandSK}(sk_{T^*,0}, \rho), pk := \mathsf{RandPK}(pk_{T^*,0}, \rho)$. Now, instead, we generate a fresh key pair $(sk, pk) \leftarrow \mathsf{KGen}(par)$, and program the random oracle as: $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, id_{S^*}) := sk - sk_{T^*,0}$.*

Because ρ is uniform and the rerandomized key pair is statistically indistinguishable from a freshly generated one, the behavior of the adversary remains unchanged:

$$\mathbf{Adv}_5 = \mathbf{Adv}_6.$$

Finally, we apply Lemma 16. A reduction simulates Game G_6 by forwarding H_1 queries to a challenger oracle $\widetilde{\mathsf{H}}$. If the adversary succeeds in Game G_6, then the reduction breaks the assumption in Lemma 16. Hence, we obtain:

$$\mathbf{Adv}_6 \leq \frac{Q_{\mathsf{H}_1}}{2^\lambda}.$$

Proof 12 *We prove the revocation soundness of* bip32PA-MU$^+$ *(Lemma 11) via a sequence of games. For each game G_i, we denote the adversary's advantage (i.e., the probability that the game outputs 1) by $\mathbf{Adv}_i$.*

Game G_0***:*** *This is the real revocation soundness game. For each token $T \in \mathcal{T}$, the long-term secret key $msk_T := (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ is generated. The adversary $\mathcal{A}$ interacts with the oracles* Start, Challenge, *and* Complete, *and eventually outputs two sessions (T^*, i_{T^*}), (S^*, i_{S^*}), along with a credential $cred^* := pk$ and a revocation key rk^*. The game outputs 1 if and only if the sessions $\pi^{i_{T^*},0}_{T^*}$ and $\pi^{i_{S^*},0}_{S^*}$ are partnered and $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*)$ holds. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound},\text{bip32PA-MU}^+}.$$

Game G_1***:*** *We abort if $\mathcal{A}$ queries $\mathsf{H}_1(pk_{T,0}, ch_T, r, \cdot)$ for any $T \in \mathcal{T}$. Since ch_T and r are unknown to $\mathcal{A}$ except via the random oracle, the probability of this event is bounded via a union bound:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{2\lambda}}.$$

Game G_2***:*** *We abort if a collision occurs in H_0, i.e., if $\mathsf{H}_1(x) = \mathsf{H}_1(x')$ for $x \neq x'$. Thus, we have:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q^2_{\mathsf{H}_1}}{2^\lambda}.$$

Game G_3***:*** *We abort if $\mathcal{A}$ outputs $rk^* = (pk_{T^*,0}, ch_{T^*}, lrev_{T^*})$. Since ch_{T^*} and $lrev_{T^*}$ are only accessible via random oracle queries, we have:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{1}{2^{2\lambda}}.$$

Game G_4***:*** *We change how locking keys lk are derived. In G_3, $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$. Now, for each $T \in \mathcal{T}$, we maintain a map $LK_T[\cdot]$, and sample lk uniformly at random upon first query. Using the pseudorandomness of* PRF *in $|\mathcal{T}|$ hybrids:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{prf},\mathrm{PRF}}.$$

Game G_5***:*** *We change the generation of cid and r. Instead of setting $cid := \mathsf{Enc}(ch, lrev)$ and $r := lrev := cid$, we sample cid uniformly from $\{0,1\}^{v^*}$ for each* Challenge *query and store $R[T, cid] := lrev$. For future sessions, we reuse the same cid to derive secret keys. Using a hybrid argument over tokens and the anonymity of* SKE*:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game G_6***:*** *We guess the first registration query involving (T^*, id_{S^*}) by selecting $I \leftarrow [Q_C]$. We abort if this guess is incorrect. Since $\mathcal{A}$ must register such a pair to win:*

$$\mathbf{Adv}_5 = Q_C \cdot \mathbf{Adv}_6.$$

Game G_7***:*** *We modify the I-th* Challenge *query. Instead of deriving (sk, pk) via $\rho := \mathsf{H}(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*})$, we generate $(sk, pk) \leftarrow \mathsf{KGen}(par)$ and program $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, id_{S^*}) := sk - sk_{T^*,0}$. Since rerandomized keys are statistically indistinguishable from fresh keys when ρ is uniform:*

$$\mathbf{Adv}_6 = \mathbf{Adv}_7.$$

By Lemma 16, we conclude:

$$\mathbf{Adv}_7 \leq \frac{Q_{\mathsf{H}_1}}{2^\lambda}.$$

Proof 13 *We prove the revocation soundness of* bip32PA-SU$^+$ *(Lemma 12) via a sequence of games. Let $\mathbf{Adv}_i$ denote*

the adversary's advantage in Game G_i, i.e., the probability that the game outputs 1.

Game *G_0. This is the standard revocation soundness game. Each token $T \in \mathcal{T}$ is initialized with a master secret key $msk_T := (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$. The adversary $\mathcal{A}$ interacts with the oracles* Start, Challenge, *and* Complete, *and eventually outputs two session identifiers (T^*, i_{T^*}) and (S^*, i_{S^*}), receives a credential $cred^* := pk$, and produces a revocation key rk^*. The game outputs 1 if: The sessions $\pi_{T^*}^{i_{T^*},0}$ and $\pi_{S^*}^{i_{S^*},0}$ are partnered, and $\mathsf{CheckCred}(id_{S^*}, cred^*, rk^*)$ holds. By definition, we have:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\text{rev-sound},\text{bip32PA-SU}^+}.$$

Game *G_1. We abort if $\mathcal{A}$ queries $\mathsf{H}_1(pk_{T,0}, ch_T, r, \cdot)$ for any $T \in \mathcal{T}$. Since ch_T and r are only accessible through random oracle queries, we apply a union bound:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{2\lambda}}.$$

Game *G_2. We abort if H_1 outputs a collision, i.e., if there exist $x \neq x'$ such that $\mathsf{H}_1(x) = \mathsf{H}_1(x')$. Hence:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}}.$$

Game *G_3. We further abort if $\mathcal{A}$ outputs a revocation key of the form $rk^* = (pk_{T^*,0}, ch_{T^*}, lrev_{T^*})$. Since ch_{T^*} and $lrev_{T^*}$ are not revealed directly, we have:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{1}{2^{2\lambda}}.$$

Game *G_4. This game changes how locking keys are derived. Instead of computing $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$, we initialize a map $LK_T[\cdot]$ for each token T. For every tuple (cid, id_S, M_a), if $LK_T[cid, id_S, M_a]$ is undefined, we sample it uniformly at random. The value $lk := LK_T[cid, id_S, M_a]$ is then used consistently. Using the pseudorandomness of* PRF *across $|\mathcal{T}|$ hybrid steps, we get:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{prf},\text{PRF}}.$$

Game *G_5. This game modifies how ciphertexts are generated during registration. Instead of computing $r := \mathsf{Enc}(ch, lrev), cid := \mathsf{Enc}(seed, (id_S, r)), lrev := cid$, we sample r and cid uniformly from $\{0,1\}^{v^*}$ for each $\mathsf{Challenge}(\pi_T^{i,0}, id_S, \cdot)$. We store mappings: $R_1[T, r] := lrev, R_2[T, cid] := (id_S, r)$, and set $lrev := r$. For authentication queries ($j > 0$), the value (id, r) is retrieved via $R_2[T, cid]$. By a hybrid argument over tokens:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\text{SKE}}.$$

Game *G_6. We guess the index $I \leftarrow [Q_C]$ corresponding to the first* Challenge *query involving (T^*, id_{S^*}). If this is not the correct query, we abort. Since such a query must exist when $\mathcal{A}$ wins:*

$$\mathbf{Adv}_5 = Q_C \cdot \mathbf{Adv}_6.$$

Game *G_7. We replace the key derivation in the Ith* Challenge *query. Instead of computing $\rho := \mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*})$, we sample a fresh key pair $(sk, pk) \leftarrow \mathsf{KGen}(par)$ and set $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r_{T^*}, id_{S^*}) := sk - sk_{T^*,0}$. By the statistical rerandomizability of rerandomized ECDSA keys:*

$$\mathbf{Adv}_6 = \mathbf{Adv}_7.$$

Finally, applying Lemma 16 via a reduction that programs H_1 through a simulated $\widetilde{\mathsf{H}}$, we obtain:

$$\mathbf{Adv}_7 \leq \frac{Q_{\mathsf{H}_1}}{2^{\lambda}}.$$

Proof 14 *We prove Lemma 13 via a sequence of games $G_0, G_1, \ldots, G_8$. Let $\mathbf{Adv}_i$ denote the probability that game G_i outputs 1.*

Game *G_0: This is the original impersonation game with global revocation. For each token $T \in \mathcal{T}$, a master secret key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T)$ is generated. The adversary is given the global revocation keys $rk_T := (pk_{T,0}, ch_T)$ and oracle access to* Start, Challenge, *and* Complete. *By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Imp\text{-}GR},\text{bip32PA}^+}.$$

Game *G_1: Abort if there is a collision in H_1, i.e., $\mathsf{H}_1(x) = \mathsf{H}_1(x')$ for $x \neq x'$. Since the output space is $\{0,1\}^{\lambda}$, we have:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q_{\mathsf{H}_1}^2}{2^{\lambda}}.$$

Game *G_2: Abort if the same randomness $c := (rs_r, rs_a)$ in registration or (rs_a, rs'_a) in authentication is sampled in two* Start *oracle calls. As each component is at most λ-bit:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_S^2}{2^{2\lambda}}.$$

Game *G_3: Abort if $\mathcal{A}$ queries $\mathsf{H}_1(seed_T, \cdot)$ for any token $T \in \mathcal{T}$. Since each $seed_T$ is sampled uniformly:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}}.$$

Game *G_4: Modify how locking keys are derived. Instead of computing $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$, we sample $lk \in \{0,1\}^{l_2}$ uniformly and store it in a map $LK[T, cid, M_a] := (id_S, lk)$. In authentication, if this map entry is missing, the oracle aborts. The change is justified by the pseudorandomness of the PRF:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{prf},\text{PRF}}.$$

Game *G_5: Abort if $\mathsf{H}_0(id_S) = \mathsf{H}_0(id_{S'})$ for distinct $id_S \neq id_{S'} \in \mathcal{S}$. Since H_0 outputs uniformly:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq \frac{|\mathcal{S}|^2}{2^{2\lambda}}.$$

We now define key terminology. A game outputs 1 (success) if: 1) No aborts occur; 2) $\mathcal{A}$ completes authentication via Complete *on session $\pi_S^{i,j}$ with $j > 0$; 3) $\pi_S^{i,j}$ has no partner session; 4) $\pi_S^{i,0}$ partners with some $\pi_T^{i',0}$ (target registration). We call this a* forged authentication *with a valid* target registration.

Game G_6***:*** *Randomly choose a token $T^* \notin \mathcal{T}$. Abort if T^* is not involved in the target registration. Since T^* is hidden from $\mathcal{A}$, the success probability scales:*

$$\mathbf{Adv}_5 = |\mathcal{T}| \cdot \mathbf{Adv}_6.$$

Game G_7***:*** *Modify signature generation for T^* by using pre-sampled randomness and signature oracle access. Maintain a map $L_\sigma[\cdot,\cdot]$. Due to the independence of outputs from H_1 in G_3, this is purely conceptual:*

$$\mathbf{Adv}_6 = \mathbf{Adv}_7.$$

We now construct an adversary $\mathcal{B}'$ that uses $\mathcal{A}$ to break the unforgeability of $\mathrm{SIG}_{\widetilde{\mathsf{H}}}$ under index-one hardness with random keys.

- *$\mathcal{B}'$ receives challenge public key pk^* and defines $pk_{T^*,0} := pk^*$, randomly samples $ch_{T^*}, seed_{T^*}$, and sets $rk_{T^*} := (pk_{T^*,0}, ch_{T^*})$. For all $T \neq T^*$, it generates keys honestly and gives all rk_T to $\mathcal{A}$.*
- *Oracle simulations:*
  - *For H_1, if queried on $(pk_{T^*,0}, ch_{T^*}, id_S)$, return $\mathsf{RandO}(\mathsf{H}_0(id_S))$; otherwise, respond via lazy sampling.*
  - *For $\mathsf{Challenge}(\pi_T^{i,j}, \cdot)$: If $T \neq T^*$, proceed as in G_8. Otherwise, compute $idx := \mathsf{H}_0(id_S)$, query $\mathsf{RandO}(idx)$, then request $\sigma := \mathsf{SignO}(idx, m)$ for $m := (\mathsf{H}_0(id_S), M_a)$. Set $L_\sigma[id_S, M_a] := \sigma$. Use it to compute the pre-signature $\widehat{\sigma} := \widehat{\mathsf{H}}(lk) \oplus (M_a \| \sigma)$.*
- *After $\mathcal{A}$ outputs a forged authentication, if T^* was not the registration token, $\mathcal{B}'$ aborts. Otherwise, $\mathcal{B}'$ extracts the forgery $(\sigma_a, \widehat{\sigma}_a)$, recovers $(M'_a \| \sigma'_a) := \widehat{\mathsf{H}}(lk) \oplus \widehat{\sigma}_a$, and outputs: $m^* := (\mathsf{H}_0(id_S), M'_a), \sigma^* := \sigma'_a, idx^* := \mathsf{H}_0(id_S)$.*

Since $\mathcal{B}'$ perfectly simulates G_7, and successfully extracts a valid forgery on a fresh message, we conclude:

$$\mathbf{Adv}_7 \leq \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{uf\text{-}hrk\text{-}idx1}, \mathrm{SIG}_{\widetilde{\mathsf{H}}}}.$$

Proof 15 *We prove Lemma 14 via a sequence of games G_i, where each game G_i modifies the previous game slightly. Let $\mathbf{Adv}_i$ denote the probability that game G_i outputs 1.*
Game G_0***.*** *This is the original impersonation with global revocation game. For each token $T \in \mathcal{T}$, a master secret key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ is generated. The adversary $\mathcal{A}$ receives the revocation keys $rk_T := (pk_{T,0}, ch_T, lrev_T)$, and interacts with the oracles* Start*,* Challenge*, and* Complete*. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Imp\text{-}GR}, \mathrm{bip32PA\text{-}MU}^+}.$$

Game G_1***.*** *Abort if there exists a hash collision in H_0: $\mathsf{H}_0(x) = \mathsf{H}_0(x')$ for $x \neq x'$. Since H_0 outputs uniformly in $\{0,1\}^{2\lambda}$:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game G_2***.*** *Abort if two* Start *queries sample the same randomness c, where $c := (rs_r, rs_a)$ during registration or $c := (rs_a, rs'_a)$ during authentication. As $rs_r, rs_a, rs'_a \in \{0,1\}^{\leq\lambda}$:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q_S^2}{2^{2\lambda}}.$$

Game G_3***.*** *Abort if $\mathcal{A}$ queries $\mathsf{H}_1(seed_T, \cdot)$ for any $T \in \mathcal{T}$. Since each $seed_T$ is uniformly random:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^\lambda}.$$

Game G_4***.*** *Modify the generation of credential identifiers: instead of computing $cid := \mathsf{Enc}(ch, lrev)$, we now sample $cid \leftarrow \{0,1\}^{v^*}$ uniformly at random. Store the mapping $R[T, cid] := lrev$, and reuse cid across queries involving the same token. By a hybrid argument over $|\mathcal{T}|$ tokens and the anonymous authentication security of* SKE*:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth}, \mathrm{SKE}}.$$

Game G_5***.*** *Change how locking keys are derived: instead of computing $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$, we sample $lk \leftarrow \{0,1\}^{l_2}$ and store it in a map $LK[T, cid, M_a] := (id_S, lk)$. Later queries use this map to retrieve the key. Using a hybrid over $|\mathcal{T}|$ tokens and the pseudorandomness of the PRF:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf}, \mathrm{PRF}}.$$

Game G_6***.*** *Abort if distinct nonces $r_1 \neq r_2$ yield a collision in H_1, i.e., $\mathsf{H}_1(\cdot,\cdot,r_1,\cdot) = \mathsf{H}_1(\cdot,\cdot,r_2,\cdot)$. Since H_1 outputs uniformly in $\{0,1\}^\lambda$:*

$$|\mathbf{Adv}_5 - \mathbf{Adv}_6| \leq \frac{Q^2_{\mathsf{H}_1}}{2^\lambda}.$$

Game G_7***.*** *Abort on a server identity collision: $\mathsf{H}_0(id_S) = \mathsf{H}_0(id_{S'})$ for $S \neq S'$. As server identifiers are unique:*

$$|\mathbf{Adv}_6 - \mathbf{Adv}_7| \leq \frac{|\mathcal{S}|^2}{2^{2\lambda}}.$$

From now on, we define the event of interest as a forged authentication*: the adversary successfully completes authentication via $\mathsf{Complete}(\pi_S^{i,j}, \cdot)$ for $j > 0$, with no partner to $\pi_T^{i',j'}$, and where the initial session $\pi_S^{i,0}$ is partnered with some $\pi_T^{i',0}$ (the* target registration*).*
Game G_8***.*** *Sample a random token $T^* \notin \mathcal{T}$ at the beginning. Abort unless T^* is used in the target registration. Since T^* is chosen uniformly, we have:*

$$\mathbf{Adv}_7 = |\mathcal{T}| \cdot \mathbf{Adv}_8.$$

Game G_9***.*** *In queries involving T^*, generate signatures using pre-sampled randomness instead of $\mathsf{H}_1(seed_{T^*}, \cdot)$. Due to uniformity from Game G_3, this change is conceptual:*

$$\mathbf{Adv}_8 = \mathbf{Adv}_9.$$

We now construct an adversary $\mathcal{B}^*$ against the uf-hrk-idx1 *security of $\mathrm{SIG}_{\widetilde{\mathsf{H}}}$, simulating Game G_9 using the signing and randomness oracles* SignO *and* RandO*. The details are:*

- ***Setup:*** *$\mathcal{B}^*$ receives challenge pk^* and assigns it to $pk_{T^*,0}$, sampling other token parameters honestly.*
- ***Hash Simulation:*** *For $\mathsf{H}_1(pk_{T^*,0}, ch_{T^*}, r, id_S)$, use $\mathsf{RandO}(\mathsf{H}_0(id_S))$; otherwise, respond lazily.*
- ***Signature Generation:*** *For T^*, use SignO to generate σ on $\widehat{m} := (\mathsf{H}_0(id_S), m)$ with index $\mathsf{H}_0(id_S)$.*
- ***Forgery Extraction:*** *Upon a successful forged authentication, extract the forgery (m^*, σ^*, idx^*) where:*

$$m^* := (\mathsf{H}_0(id_S), M'_a), \sigma^* := \sigma'_a, idx^* := \mathsf{H}_0(id_S).$$

Since $\mathcal{B}^*$ perfectly simulates G_9 and produces a valid forgery on a new message, we have:

$$\mathbf{Adv}_9 \leq \mathbf{Adv}^{\mathcal{B}^*}_{\mathsf{uf\text{-}hrk\text{-}idx1},\mathrm{SIG}_{\widetilde{\mathsf{H}}}}.$$

Proof 16 *We prove Lemma 15 by a sequence of games $G_0, G_1, \ldots, G_9$, where in each game G_i, the probability that the game outputs 1 is denoted by $\mathbf{Adv}_i$.*

Game *G_0. This is the original impersonation game with global revocation. For each token $T \in \mathcal{T}$, a master secret key $msk_T = (sk_{T,0}, pk_{T,0}, ch_T, seed_T, lrev_T)$ is generated. The adversary is given the global revocation key $rk_T := (pk_{T,0}, ch_T, lrev_T)$ and has access to oracles Start, $\mathsf{Challenge}$, and $\mathsf{Complete}$. By definition:*

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Imp\text{-}GR},\mathrm{bip32PA\text{-}SU}^+}.$$

Game *G_1. This game aborts if a hash collision occurs in H_0, i.e., if $\mathsf{H}_0(x) = \mathsf{H}_0(x')$ for $x \neq x'$. As the outputs are uniformly sampled from $\{0,1\}^{2\lambda}$, we have:*

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \frac{Q^2_{\mathsf{H}_0}}{2^{2\lambda}}.$$

Game *G_2. This game aborts if two Start queries sample the same randomness c, where $c := (rs_r, rs_a)$ during registration or $c := (rs_a, rs'_a)$ during authentication. As $rs_r, rs_a, rs'_a \in \{0,1\}^{\leq\lambda}$, we have:*

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \frac{Q^2_S}{2^{2\lambda}}.$$

Game *G_3. This game aborts if the adversary queries $\mathsf{H}_1(seed_T, \cdot)$ for any $T \in \mathcal{T}$. Since each $seed_T$ is sampled uniformly, we obtain:*

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq \frac{Q_{\mathsf{H}_1} \cdot |\mathcal{T}|}{2^{\lambda}}.$$

Game *G_4. We modify the registration process: instead of computing $r := \mathsf{Enc}(ch, lrev)$, $cid := \mathsf{Enc}(seed, (id_S, r))$, and setting $lrev := cid$, we now sample r, cid uniformly from $\{0,1\}^{v^*}$. We record mappings $R_1[T, r] := lrev$, $R_2[T, cid] := (id_S, r)$, and set $lrev := r$. For authentication queries, we retrieve $(id, r) := R_2[T, cid]$ instead of decrypting cid. Using a hybrid argument over all $T \in \mathcal{T}$:*

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| \leq 2|\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}}_{\mathsf{anon\text{-}auth},\mathrm{SKE}}.$$

Game *G_5. In G_4, $lk := \mathsf{PRF}(msk_T, (cid, id_S, M_a))$. Now we sample lk uniformly and store it as $LK[T, cid, M_a] := (id_S, lk)$. During authentication, if (id_S, lk) is defined, we use it; otherwise, the oracle aborts. By the pseudorandomness of* PRF*:*

$$|\mathbf{Adv}_4 - \mathbf{Adv}_5| \leq |\mathcal{T}| \cdot \mathbf{Adv}^{\mathcal{B}'}_{\mathsf{prf},\mathrm{PRF}}.$$

Game *G_6. This game aborts if distinct revocation nonces $r_1 \neq r_2$ result in a collision in H_1. Since H_1 maps to $\{0,1\}^{\lambda}$, the collision probability is:*

$$|\mathbf{Adv}_5 - \mathbf{Adv}_6| \leq \frac{Q^2_{\mathsf{H}_1}}{2^{\lambda}}.$$

Game *G_7. We abort if $\mathsf{H}_0(id_S) = \mathsf{H}_0(id_{S'})$ for distinct $S \neq S' \in \mathcal{S}$. This yields:*

$$|\mathbf{Adv}_6 - \mathbf{Adv}_7| \leq \frac{|\mathcal{S}|^2}{2^{2\lambda}}.$$

From this point, we restrict attention to executions where: 1) No aborts occur; 2) The adversary completes an authentication via $\mathsf{Complete}(\pi^{i,j}_S, \cdot)$ with $j > 0$; 3) $\pi^{i,j}_S$ has no partner; 4) The corresponding $\pi^{i,0}_S$ has a partner $\pi^{i',0}_T$. We call this a forged authentication *with* target registration*.*

Game *G_8. We sample a random token $T^* \notin \mathcal{T}$. The game aborts if the target registration does not involve T^*. Since the choice of T^* is hidden:*

$$\mathbf{Adv}_7 = |\mathcal{T}| \cdot \mathbf{Adv}_8.$$

Game *G_9. For T^*, we change how signatures are generated in the $\mathsf{Challenge}$ oracle. Rather than querying $\mathsf{H}_1(seed_{T^*}, \cdot)$, we use a randomness oracle and signing oracle to simulate the signature generation. Since G_3 already ensures the uniformity of H_1, this change is conceptual:*

$$\mathbf{Adv}_8 = \mathbf{Adv}_9.$$

We now build an adversary $\mathcal{B}^*$ against the $\mathsf{uf\text{-}hrk\text{-}idx1}$*-security of the indexed signature scheme $\mathrm{SIG}_{\widetilde{\mathsf{H}}}$. Adversary $\mathcal{B}^*$ receives a challenge public key pk^* and has access to oracles* SignO*,* RandO*. It proceeds as follows:*

- *$\mathcal{B}^*$ selects random $T^* \notin \mathcal{T}$, sets $pk_{T^*,0} := pk^*$, samples $ch_{T^*}, seed_{T^*}$, and constructs $rk_{T^*} := (pk^*, ch_{T^*})$. All other rk_T are honestly generated and given to $\mathcal{A}$.*
- *To simulate H_1 queries of the form $(pk_{T^*}, ch_{T^*}, r_{T^*}, id_S)$, $\mathcal{B}^*$ responds using $\mathsf{RandO}(\mathsf{H}_0(id_S))$. All other queries are answered via lazy sampling.*
- *For $\mathsf{Challenge}(\pi^{i,j}_T, \cdot, \cdot)$: If $T \neq T^*$, simulate honestly as in G_8. If $T = T^*$, extract index $idx := \mathsf{H}_0(id_S)$, query $\mathsf{RandO}(idx)$, and obtain signature $\sigma \leftarrow \mathsf{SignO}(idx, \widehat{m})$ where $\widehat{m} := (idx, m)$. Then compute $\widehat{\sigma} := \widehat{\mathsf{H}}(lk) \oplus (M_a \| \sigma)$.*

At the end, $\mathcal{B}^*$ extracts a valid forgery (m^*, σ^*, idx^*) using the forged authentication output. Since the forgery is valid and not queried before, we have:

$$\mathbf{Adv}_9 \leq \mathbf{Adv}^{\mathcal{B}^*}_{\mathsf{uf\text{-}hrk\text{-}idx1},\mathrm{SIG}_{\widetilde{\mathsf{H}}}}.$$