To attest or not to attest, this is the question – Provable attestation in FIDO2

Nina Bindel[1], Nicolas Gama[1],
Sandra Guasch[1], and Eyal Ronen[2]

[1] SandboxAQ, Palo Alto, CA, USA, {first.second}@sandboxaq.com
[2] Tel Aviv University, Tel Aviv, Israel, eyal.ronen@cs.tau.ac.il

Abstract. FIDO2 is currently the main initiative for passwordless authentication in web servers. It mandates the use of secure hardware authenticators to protect the authentication protocol's secrets from compromise. However, to ensure that only secure authenticators are being used, web servers need a method to attest their properties. The FIDO2 specifications allow for authenticators and web servers to choose between different attestation modes to prove the characteristics of an authenticator, however the properties of most these modes have not been analysed in the context of FIDO2. In this work, we analyse the security and privacy properties of FIDO2 when different attestation modes included in the standard are used, and show that they lack good balance between security, privacy and revocation of corrupted devices. For example, the basic attestation mode prevents remote servers from tracing user's actions across different services while requiring reduced trust assumptions. However in case one device is compromised, all the devices from the same batch (e.g., of the same brand or model) need to be recalled, which can be quite complex (and arguably impractical) in consumer scenarios. As a consequence we suggest a new attestation mode based on the recently proposed TokenWeaver, which provides more convenient mechanisms for revoking a single token while maintaining user privacy.

Table of Contents

1 Introduction

For many years, using passwords as a single authentication mechanism has been one of the biggest cause of security problems in the Internet. Weak passwords, phishing, credential stuffing, and many other attack vectors can have devastating results such as full compromise of accounts and sensitive data. To help prevent

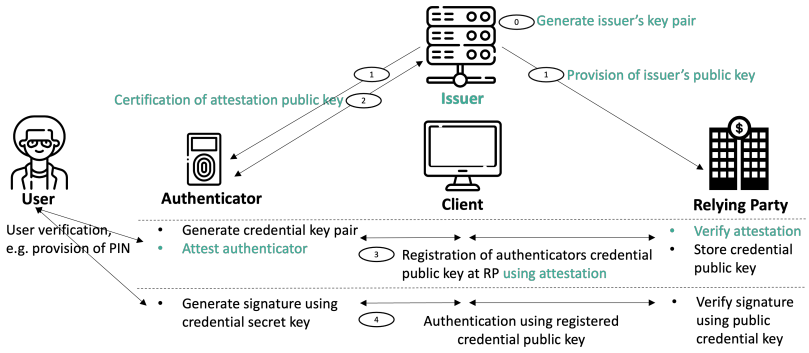

Fig. 1: Message of flow of Fido2 with attestation (in teal).

such attacks, the Fast Identity Online (FIDO) Alliance proposes their FIDO protocol for passwordless authentication [Allb].

The protocol uses a secure hardware device called an authenticator (e.g., security keys or certain smart phones) for authentication to a server. The authenticators are able to securely generate and store secret credentials. The FIDO protocol consists of two phases: a registration phase in which the authenticator generates a credential key pair, and the public key is stored at the server, also called Relying Party (RP) (see (3) in Figure 1); and authentication of the authenticator on the RP using the registered credentials (see (4) in Figure 1). In both phases the user verifies registration/authentication by entering a gesture (e.g., entering a PIN or pressing a button).

The FIDO2 protocol is implemented using two sub-protocols: WebAuthn — a protocol between authenticator, client (e.g., a browser), and RP in which the RP sends a challenge, the authenticator responds with a signature, and the client acts as an intermediary, and CTAP — a protocol between authenticator and client that binds the client to the authenticator to restrict access to the authenticator. FIDO2, and in particular WebAuthn, offers a plethora of different modes such as enterprise mode (which allows identifying information), key storage modes (on the authenticator, encrypted on the server, shared between different devices or not, etc.), or different attestation modes.

1.1 Attestation in FIDO2

FIDO2 supports several attestation modes, that can enforce the use of approved (or of 'different levels certified') secure authenticators. The goal of attestation is to prevent users from using weak or uncertified authenticators that might put the user, the user's company, or the RP at risk.

At the same time, attestation should preserve the privacy of the user. More concretely, a coalition of web servers should not be able to trace FIDO2 actions to a particular authenticator, or to guess if the same authenticator is used for two different accounts (except in enterprise mode). The different attestation modes

present in the standard achieve different notions of security and privacy that imply a variety of nuances in the trust assumptions.

For instance, in the *attestation CA* (attCA) mode, the protocol relies on a certificate authority (CA) that issues certificates for different attestation credentials (ideally one per registered credential key per RP) in a way that the attestation keys cannot be linked to each other or the authenticator identifier. However, we need to trust the CA to not reveal the link between the attestation credential and authenticator, to not issue certificates for rogue authenticators, and to be online and available whenever needed.

Another example is the attestation mode basic, where the issuer/manufacturer hardcodes the same attestation key in a batch of at least 100,000 devices (i.e., from the same model or brand). As long as the private attestation key is not leaked and accepted in the trust store of client and servers, this mode provides security against impersonation and unlinkability of user actions among all the devices from the same batch. However, even secure authenticators may be compromised. For example, [SRW22] presents an attack on FIDO2 authenticator implemented inside Samsung's TrustZone Trusted Execution Enviroment (TEE). In basic mode, once one of the keys is leaked, security for all keys in the same batch is lost. Which means that the attestation certificate would need to be revoked on all the servers that trust it, and hardware authenticators would need to be physically replaced. In summary, in this work, we ask the following questions:

Can we provide a unified model that captures the security and privacy properties of all attestation modes? Are there significant limitations to the current FIDO2 attestation modes and can they be overcome?

1.2 Contributions

We can answer the questions above in the affirmative:

- We define a new class of *Extended Passwordless Authentication Protocols with Attestation* (ePIAA) that allows to unify and prove the security and privacy properties for all FIDO2 attestation modes.
- Based on our analysis of current modes, we suggest a new mode called smpTW that improves upon the current basic mode by allowing it to recover security even after an authenticator was compromised.

We analyse the publicly available FIDO2 attestation modes present in the specification. To this end we unify and extend the models from [Bar+21; BCZ23; HLW23] with the new class ePIAA. We then continue to formally define and prove the different security and privacy definitions for each attestation mode and compare them. We summarize the different properties and considered modes in our and previous work in Table 1 and provide more details in Section 2.1.

In our unified analysis, we find that only WebAuthn with attestation mode basic satisfies our desired definitions for authentication security and unlinkability (none and self only satisfy weaker authentication security and attCA only weaker

Table 1: WebAuthn properties and considered attestation modes analysed in the literature and this work.

	BBCW [Bar+21]	HLW [HLW23]	BCZ [BCZ23]	This work
Properties				
Authentication Security	✓	✓	✓	✓
Unlinkability	✗	✓	✗	✓
PQ-readiness	✗	✗	✓	✓
Post-compromise Security	✗	✗	✗	(✓)
Attestation modes				
none	✗	✗	✓	✓
self	✗	✓	✗	✓
basic	(✓)	✗	✗	✓
attCA	✗	✗	✗	✓
smpTW	✗	✗	✗	✓
Adversary type during the protocol phases				
Certification	-	-	-	Active
Registration	Active	Active	Passive	Active
Authentication	Active	Active	Active	Active

unlinkability). Since basic attestation poses the risk of batch corruptions that might lead to the replacement of the entire authenticator batch, we propose a new attestation mode smpTW based on the (simple) TokenWeaver general attestation scheme proposed in [CJR22]. smpTW is very similar to basic mode and satisfies our desired definitions. However, in constrast to basic mode, it allows to recover security even after an authenticator in the same batch was compromised.

Since we are considering potential future improvements to the FIDO2 standard, we analyse the existing schemes from a quantum-resistant perspective: our security reductions between the authentication security and unlinkability of WebAuthn and the collision resistance of hash functions, forgery resistance of signatures, or semantic security of encryption primitives are valid against classical (PPT) and quantum (QPT) polynomial-time adversaries, so that any instantiation of the protocol using post-quantum primitive yields a post-quantum secure passwordless authentication protocol. As efficient instantiations of PQ blind signatures are not yet standardised, we opted for an attestation mode based on *simple TokenWeaver* instead of the full *TokenWeaver*. Together with the work in [BCZ23], our work contributes to a full-fledged post-quantum solution including privacy-preserving secure attestation modes.

1.3 Overview of the paper

We first present background on WebAuthn with attestation, TokenWeaver, and PCS in Section 2. In Section 3, we define *extended Passwordless Authentication Protocols with Attestation (ePlAA)* which is based on BCZ's [BCZ23] ePlA

class— we add an initiation and certification phase to the existing registration and authentication. We continue with the detailed definition of authentication security and unlinkability in Section 4. Section 5 presents the analysis of existing FIDO2 attestation modes. We present our new attestation mode smpTW in Section 6 and conclude in Section 7.

2 Background

We introduce needed background on WebAuthn, such as on its key and attestation modes, and prior security models, as well as on the attestation framework TokenWeaver and post-compromis security.

Notation. We write $x \leftarrow_{\$} X$ for x being the returned value of a probabilistic algorithm X; we write $x \leftarrow X$ if X is deterministic. Moreover, we denote the security parameter by λ and the number of oracle queries to oracle $\mathcal{O}$ by $q_{\mathcal{O}}$.

In addition, we denote the signature scheme used for assertion signatures (during the authentication phase) by $\Sigma = (\Sigma.\mathsf{KG}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Vfy})$ and use $\Sigma_A = (\Sigma_A.\mathsf{KG}, \Sigma_A.\mathsf{Sign}, \Sigma_A.\mathsf{Vfy})$ to denote the attestation signature scheme. In addition, symmetric and asymmetric encryption schemes are denoted as $\mathcal{E}_s = (\mathcal{E}_s.\mathsf{KG}, \mathcal{E}_s.\mathsf{Encrypt}, \mathcal{E}_s.\mathsf{Decrypt})$ and $\mathcal{E}_a = (\mathcal{E}_a.\mathsf{KG}, \mathcal{E}_a.\mathsf{Encrypt}, \mathcal{E}_a.\mathsf{Decrypt})$, respectively. We denote $\Lambda = (\mathsf{CG}, \mathsf{CVfy})$ the scheme for generating and verifying digital certificates from issuer's public keys. In addition to signature generation and verification, these operations also consist of other operations such as adding/checking an expiration time or revocation mechanisms. We make use of the standard definitions for existential unforgability under chosen-message attack (euf-cma), indistinguishably under chosen-plaintext attack (ind-cpa), and collision resistance (coll-res).

2.1 WebAuthn

The FIDO2 protocol aims at passwordless authentication of users and is run between a Relying Party (RP), a client, and an authenticator owned by a user. In addition, an issuer of attestation certificates might be involved, depending on the attestation type used. A high-level message flow is depicted in Figure 1. FIDO2 specifies two subprotocols: WebAuthn and CTAP. As this paper is concerned with attestation, we only focus on the WebAuthn protocol and omit the CTAP protocol. The current stable version is WebAuthn 2 [W3C21] with an editor's draft available for WebAuthn 3 [W3C23]. Our results apply for both versions and as such we will refer to WebAuthn for the remainder of the paper.

Key Modes. WebAuthn offers two key modes: resident and non-resident keys, where the latter means that the keys are stored (symmetrically encrypted) at the server as analysed in [HLW23]. Resident keys, in contrast, are stored either locally or as an encrypted list at a trusted third party to enable key sharing between different devices such as different phones that can be used as authenticators.

Recently, passkeys [Alla] have emerged as a new concept of key management. They are resident keys, which can be used from multiple devices as they can be synchronised across devices registered by the same user in a given platform, such as Google, Apple or Microsoft, or they can be used from a nearby device (e.g., using Bluetooth). Passkeys that are migrated across devices are encrypted end-to-end. To cover this new concept, this work focuses on resident keys.

Attestation Modes. During registration, the server sends a preferred attestation conveyance mode (called attestation preferences) together with its challenge. This preference indicates whether the server is in fact not interested in getting an attestation from the authenticator (preference none); if the client is allowed to alter the authenticator's attestation statement, e.g. to remove uniquely identifying information (preference indirect); if the client should forward the authenticator attestation without any changes (preference direct); or if the server is allowed to request uniquely identifying information from authenticators (preference enterprise).

During registration, the server sends one of the following preferred attestation conveyance mode (called attestation preferences) together with its challenge:

none: This preferences indicates that the server is not interested in getting an attestation from the authenticator. Moreover, the client should remove potentially identifying information. The authenticator should use attestation modes none or self (see below) in this case.

indirect means that the server is interested in attestation but does not need to come *directly* from the authenticator and instead *indirectly* from the client. That means the client is allowed to alter the authenticator's attestation statement, e.g., remove uniquely identifying information.

direct indicates that the server is interested in authenticator attestation and that it should be provided by the authenticator and forwarded by the client without any changes. All attestation modes are allowed in this preferences, in particular attestation mode none is still an option in case the authenticator cannot provide attestation and if the server's attestation policy allows it.

enterprise permits an RP to request uniquely identifying information from authenticators during a registration.

We rule out none because we want to analyse different attestation modes, indirect because we want to be able to consider a malicious client, and enterprise because this work is concerned with the unlinkability of authenticators that is clearly not given when uniquely identifying information is provided. Therefore, we consider only the direct mode (without making this explicit in our abstraction in Figure 7) in this work, including the analysis from Section 5. In mode direct, WebAuthn allows the following attestations modes:

None: The default mode in the specifications is to use no attestation, i.e., no signature is included in the authenticator's response during registration.

Self: The authenticator's response is signed using the secret credential key during registration (and authentication).

Basic: Attestation key pairs are shared by "batches" of at least 100,000 authenticators for privacy reasons.

AttCA: An authenticator can generate multiple Attestation Identity Keys (AIKs) and request certificates for each from one or more attestation CAs. For our analysis we assume authenticators use a fresh AIK when registering to a server in order to prevent trivial privacy attacks. In the 'Attestation CA' mode, an authenticator is based on a Trusted Platform Module (TPM) and holds an authenticator-specific Endorsement Key (EK) which is used to securely communicate with a trusted third party, the Attestation CA. The authenticator can generate multiple Attestation Identity Keys (AIKs), requesting certificates for each from the Attestation CA to limit the exposure of the EK (which is a global correlation handle) to Attestation CA(s). For our analysis, we assume authenticators use can be requested for each authenticator-generated public key credential individually, and conveyed to Relying Parties as attestation certificates. this is the case, as otherwise unlinkability can be broken trivially. From a privacy perspective, the Attestation CA receives data that enables it to track user behavior over several RPs.

AnonCA: This mode is similar to AttCA with the difference that certificates are dynamically generated per-credential by a cloud-operated CA owned by the (trusted) manufacturer. We do not consider this mode in this work as for the little public information available it is the same setup as AttCA with the assumption of a fresh AIK per server registration.

In the previous Webauthn 1, another attestation mode using (Elliptic Curve) Direct Anonymous Attestation (DAA), based on [CDL16], was allowed which seemed to offer an interesting compromise between security, privacy, and availability of the CA. It has been deprecated in WebAuthn 2 though.

WebAuthn as Passwordless Authentication Protocol. Recently, several works have been set up to model security and privacy properties of WebAuthn (also of CTAP and of the combination of the two protocols in FIDO2) and to reduce them provably to the security of the involved cryptographic primitives. Barbosa, Boldyreva, Chen, and Warinschi (BBCW) [Bar+21] started this line of research, defining the cryptographic class of *Passwordless Authentication (PlA)* and *authentication security* consisting of two subprotocols $\mathsf{Register} = (\mathsf{rChall}, \mathsf{rCom}, \mathsf{rRsp}, \mathsf{rVrfy})$ and $\mathsf{Authenticate} = (\mathsf{aChall}, \mathsf{aCom}, \mathsf{aRsp}, \mathsf{aVrfy})$.

BBCW proved WebAuthn with attestation mode basic to be secure against active adversaries during Register and Authenticate. However, they assume each authenticator has a different attestation key (batch size equals 1). Hence, no privacy is preserved. Bindel, Cremers, and Zhao (BCZ) [BCZ23] extended BBCW's framework modeling various additional properties of FIDO2 (calling the resulting class extended PlA (ePlA)) with attestation mode none and analysed the protocol's readiness for the post-quantum transition. In parallel, Hanzlik, Loss, and Wagner (HLW) [HLW23] extended BBCW's framework in a different way, adding non-resident keys under attestation mode self to the analysis. Moreover,

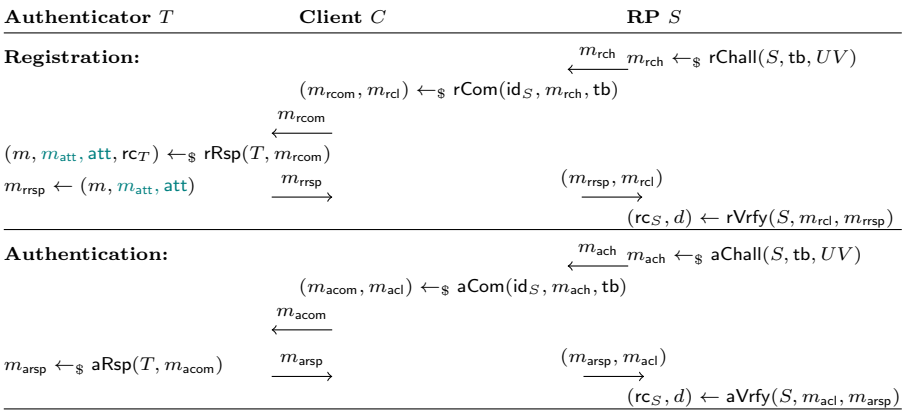

Fig. 2: WebAuthn flow [BCZ23]; attestation values are highlighted in teal.

they are the first to define privacy formally (i.e., *unlinkability* of authenticators) for FIDO2.

Returning to the details of WebAuthn as ePlA, during registration, the RP first samples a challenge (in rChall) and sends it together with other information (as m_{rch}) to the client. The client then computes a command and a client message (m_{rcom} and m_{rcl}, respectively, in rCom) and sends m_{rcom} to the authenticator. The authenticator samples a credential key pair and chooses the attestation mode (states in att) and generates the attestation statement m_{att} (potentially including an attestation signature) depending on the attestation mode, packs the information in m_{rrsp} and sends it to the client. The client forwards it, together with m_{rcl} to the RP. The RP verifies the attestation statement (after checking that att is in the set of its accepted attestation modes) and makes a decision d whether it accepts the registration. We depict the flow of the WebAuthn protocol between the authenticator, the client, and the relying party (RP) (i.e. without the issuer) in Figure 2.

The authentication message flow is very similar to the registration one, the biggest difference being the assertion signature is generated using the credential secret key. We recall the formal abstraction of $\mathsf{Authenticate}$ as given in [BCZ23] in Figure 3; we define $\mathsf{Register}$ *including attestation* in Figure 7, Section 5.

2.2 TokenWeaver and Post-Compromise Security

TokenWeaver [CJR22] is an attestation framework that is able to provide Post-Compromise Security (PCS) [CGCG16] for Trusted Execution Environments (TEEs) while still preserving the users' privacy. The goal of PCS [CGCG16] is to recover from a compromise of an honest party and regain lost security. If the honest party performs a "healing" step, the attacker is "locked out" and the compromised secrets are replaced with new ones unknown to the attacker.

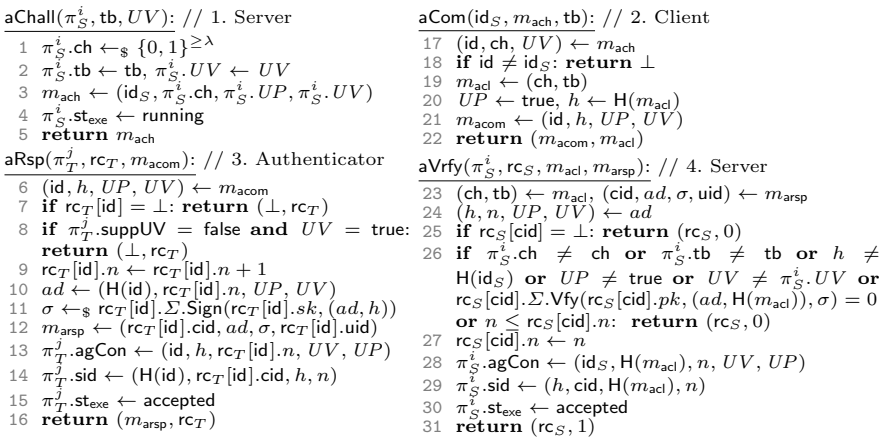

Fig. 3: Authentication of WebAuthn as described in [BCZ23] as instantiation of ePIA = (Register, Authenticate).

In the context of FIDO2, the TEE is the authenticator. For example, in basic mode, if an authenticator is compromised, we lose all security claims that rely on the compromised attestation secret key. In TokenWeaver's "healing" step, the authenticator can be provisioned with a new attestation key. Upon such a provision request, the authenticator either recovers or detects a compromise, thus achieving PCS.

The full TokenWeaver solution is based on using ephemeral attestation signing keys, provisioned in a privacy preserving protocol based on blind signatures. The authors also discussed a simplified *Global Attestation Key* variant. In the FIDO2 context, this smpTW mode, is very similar to the basic mode, but periodically rotates a short-lived "batch" attestation key. PCS is achieved by the provisioning protocol that is used by the device to get the next epoch's attestation key. smpTW allows the system to recover from or detect a compromise when the previous global attestation key expires.

3 Definition of Extended Passwordless Authentication Protocols with Attestation

To analyse the security (i.e., passwordless authentication and PCS) and privacy (i.e., unlinkability) properties of WebAuthn with different attestation modes, we follow the line of work started with [Bar+21]. Namely, they defined the cryptographic primitive *Passwordless Authentication* (PlA) protocols and view WebAuthn as an instantiation of PlA. [BCZ23] defined *extended* PlA (ePlA) to give a more fine-grained analysis. In this section we present our new class of ePIA *with Attestation* (ePIAA) that 1) unifies the classes used in [BCZ23; HLW23] to the extend possible and 2) extends it by allowing different attestations modes.

We define ePIAA for four entities — authenticators T, clients C, servers S, and issuers I each having a state to store information — by four phases, Initiate, Certification, Register and Authenticate described as follows.

We assume that if an algorithm gets a specific entity as input (e.g., an authenticator T) it also gets its state (e.g., st_T) as input and potentially updates it during the algorithm without making this explicit. The initial state of a server for attestation type basic, for example, is empty and later updated (e.g., with the initialisation context that includes the issuers public key pk_I to verify the authenticator's attestation signature, see Figure 7).

Moreover, we associate every party with an ID, e.g., id_T of authenticator T that uniquely identifies the authenticator.

Initiate: is the initialisation of authenticators, servers, and issuers, resulting in initialisation contexts ic_T, ic_S, and ic_I. In addition, the authenticators' attestation material att_m is defined (see Section 4.3 for more details).

Certification: is the certification of an authenticator's attestation key run between an authenticator T, a client C, and an issuer I and after a successful run of Initiate. At the end, both T and I hold certification contexts, which are relevant for subsequent registrations and authentications. Certification can be decomposed into the following algorithms

$m_{\mathsf{cgen}} \leftarrow_{\$} \mathsf{cGen}(T)$**:** The *attestation key generation* takes as input an authenticator T, and outputs a generation[3] message m_{cgen}.

$(m_{\mathsf{crsp}}, \mathsf{cc}_I) \leftarrow_{\$} \mathsf{cRsp}(I, \mathsf{id}_T, m_{\mathsf{cgen}})$**:** The *certification response* takes as input an issuer I, a authenticator ID id_T, and a generation message m_{cgen}, and outputs a response message m_{crsp} and an issuer's certification context cc_I.

$(\mathsf{cc}_T, d) \leftarrow \mathsf{cVrfy}(T, \mathsf{id}_I, m_{\mathsf{crsp}})$**:** The *certification verification* takes as input an authenticator T, a server identity id_S, and a response message m_{crsp}, and outputs a authenticator-associated certification context cc_T and a decision bit $d \in \{0, 1\}$.

Register: is a two-pass challenge-response protocol run between an authenticator T, a client C, and a server S, and at most once per tuple (T, S) (i.e., not for additional clients) and after a successful run of Certification between T and an issuer. At the end, both T and S hold registration contexts, which are relevant for subsequent authentications. Register can be decomposed as follows:

$m_{\mathsf{rch}} \leftarrow_{\$} \mathsf{rChall}(S, \mathsf{tb}, UV)$**:** The *challenge generation* takes as input a server S, an authenticator binding state tb[4], and a user verification condition $UV \in \{\mathsf{true}, \mathsf{false}\}$ (which indicates whether user verification is required), and outputs a challenge message m_{rch}.

$(m_{\mathsf{rcom}}, m_{\mathsf{rcl}}) \leftarrow \mathsf{rCom}(\mathsf{id}_S, m_{\mathsf{rch}}, \mathsf{tb})$**:** The *client command algorithm* takes as input the intended server identity id_S, a challenge message m_{rch}, and an authenticator binding state tb, and outputs a client message m_{rcl} and a command message m_{rcom}.

[3] The name is chosen since during many of the attestation modes, the attestation key is generated on the authenticator during this step.

[4] WebAuthn [W3C21, Sec 5.8] optionally uses token binding [Pop+18] to cryptographically bind the information provided by the authenticator to the TLS layer, as modelled in [BCZ23].

$(m_{\mathsf{rrsp}}, \mathsf{rc}_T, \mathsf{sid}, \mathsf{agCon}) \leftarrow_{\$} \mathsf{rRsp}(T, m_{\mathsf{rcom}})$**:** The *registration response* takes as inputs an authenticator T (and implicitly its initialisation and certification context ic_T and cc_t that are included in st_T) and a command message m_{rcom}, and outputs a response message m_{rrsp}, an authenticator-associated registration context rc_T, a session id sid, and the agreed content from the authenticators's perspective agCon.

$(\mathsf{rc}_S, d, \mathsf{sid}, \mathsf{agCon}) \leftarrow \mathsf{rVrfy}(S, m_{\mathsf{rcl}}, m_{\mathsf{rrsp}})$**:** The *registration verification* takes as inputs a server S, a client message m_{rcl}, and a response message m_{rrsp}, and outputs a server-associated registration context rc_S, a decision bit d, a session id sid, and the agreed content from the server's perspective to indicate whether the registration request was accepted.

Authenticate**:** is a two-pass challenge-response protocol run between an authenticator T, a client C, and a server S after a successful run of Register, in which both T and S generated their registration contexts. S either accepts or rejects the authentication attempt. Authenticate can be decomposed as follows:

$m_{\mathsf{ach}} \leftarrow_{\$} \mathsf{aChall}(S, \mathsf{tb}, UV)$**:** The *challenge generation* takes as input a server S, an authenticator binding state tb, and a user verification condition $UV \in \{\mathsf{true}, \mathsf{false}\}$, and outputs a challenge message m_{ach}.

$(m_{\mathsf{acl}}, m_{\mathsf{acom}}) \leftarrow \mathsf{aCom}(\mathsf{id}_S, m_{\mathsf{ach}}, \mathsf{tb})$**:** the *client command algorithm* inputs the intended server identity id_S, a challenge message m_{ach}, and a binding state tb, and outputs a client message m_{acl} and a command message m_{acom}.

$(m_{\mathsf{arsp}}, \mathsf{rc}_T) \leftarrow_{\$} \mathsf{aRsp}(T, m_{\mathsf{acom}})$**:** The *authentication response* takes as inputs an authenticator T (implicitly along with its associated contexts ic_T, cc_T, and rc_T stored in the state st_T) and a command message m_{acom}, and outputs a response message m_{arsp} and the updated registration context rc_T.

$(\mathsf{rc}_S, d) \leftarrow \mathsf{aVrfy}(S, m_{\mathsf{acl}}, m_{\mathsf{arsp}})$**:** The *authentication verification* takes as inputs a server S (implicitly along with its associated registration contexts ic_S and rc_S stored st_S), a client message m_{acl}, and a response message m_{arsp}, and outputs the updated registration context rc_S and a decision bit d indicating whether the authentication request was accepted.

We assume all initialisation, certification, and registration contexts to be initialised with the empty set without making this explicit in the pseudo-codes.

Similar to [BCZ23; Bar+21], we model concurrent or sequential sessions of a server S or issuer I, and sequential sessions of an authenticator T, using the following notation. π_S^i is the i-th instance of server sessions, i.e., $S = \{\pi_S^i\}_i$; π_I^k is the k-th instance of issuer sessions, i.e., $I = \{\pi_I^k\}_k$; and π_T^j is the j-th instance of either server-authenticator or issuer-authenticator sessions, i.e., $T = \{\pi_T^j\}_j$.

Extending the session variables defined in [BCZ23], the following variables are used by our ePIAA protocol.

$\pi_I^k, \pi_T^j, \pi_S^i$**:** k-th issuer, j-th authenticator, i-th server session.

$\pi_I^k.\mathsf{sid}$**,** $\pi_T^j.\mathsf{sid}$**,** $\pi_S^i.\mathsf{sid}$**:** Issuer, authenticator, and server session identifiers. The session identifiers of two distinct sessions (between server/issuer and authenticator) are expected to be the same.

$\pi_S^i.\mathsf{ch}$**:** Registration/authentication challenge, generated in rChall/aChall.

$\pi_T^j.\mathsf{st}_{\mathsf{exe}}, \pi_S^i.\mathsf{st}_{\mathsf{exe}}, \pi_I^k.\mathsf{st}_{\mathsf{exe}}$**:** Execution statuses being either $\{\perp$, running, accepted$\}$ and updated in respective algorithms.
$\pi_S^i.\mathsf{pkCP}$**:** List of accepted assertion signature schemes by S.
$\pi_S^i.\mathsf{attPol}$**:** Attestation policy of the server, e.g., which attestation modes and algorithms are accepted.
$\pi_T^j.\mathsf{att}$**:** Attestation mode chosen by the authenticator. Implicitly, this includes also the algorithm that is used for the attestation signature (if attestation signatures are generated).
$\pi_T^j.\mathsf{suppUV}$**:** Variable indicating whether T supports user verification.
$\pi_I^k.\mathsf{agCon}$**,** $\pi_S^i.\mathsf{agCon}$**,** $\pi_T^j.\mathsf{agCon}$**:** The contents that are expected to be agreed in session with the same session ID.

4 Security and Privacy Definition of ePIAA

We continue refining the security and privacy properties from [BCZ23; HLW23] definitions to cover attestation for our new class ePIAA defined in the previous section. We first describe our threat model(s) for the passwordless authentication and unlinkability of ePIAA. Next we define the oracles that are allowed to be accessed during the two experiments, authenticator groups, and session partnering, before we then formally define passwordless authentication security (PAuth) and unlinkability (Unl) of ePIAA.

4.1 Threat Model

First of all, as common in this line of research, we do not model the user. Following [BCZ23], we assume that the users always provide the user presence or user verification confirmation when it is required and leave the users implicit in the authentication and unlinkability experiments. In addition, we assume that there is only one set of credentials per server to be active at a time, as otherwise the user needs to make a choice which credential to use, as implicitly assumed in [BCZ23].

We continue describing assumptions on the four entities: authenticators, servers, clients, and issuers. First, we assume the identifier id_S of each server S is unique and we consider that some selected *authenticators* can be corrupted both in the authentication and unlinkability experiment. This is defined in more detail in the experiments and in the different instantiations. In particular, we do not assume authenticators to be "tamper-proof", i.e., the adversary is allowed to read locally stored contexts of authenticators, which means that they will have access to the private keys corresponding to the credentials used to register into a specific service, and to the attestation private key. Moreover, we assume *servers* to be malicious in both experiments (with the exception of the winning server during the authentication security). In addition, we allow malicious *clients* for both experiments (with the exception of the two clients interacting with the target authenticators in the unlinkability experiment). These exceptions correspond to the asterisks in Table 2 which summarizes our assumptions

Table 2: Assumptions for the passwordless authentication and unlinkability experiments in each phase, with adversary types in the channels between the different participants. Entities which can be controlled by the adversary are highlighted in teal. (*) denotes some natural restrictions on the challenge entities that decide on a win in the security and privacy experiments, which are further specified in this section.

Phase	Passwordless Authentication		Unlinkability	
	$\mathcal{A}$ type	Entities	$\mathcal{A}$ type	Entities
Initialisation I-T	None	I, T	Active	I, T
Initialisation I-S	Passive	I, S	Active	I, S
Certification	Active	I, T^*, C	Active	I, T^*, C^*
Registration	Active	T^*, C, S^*	Active	T^*, C^*, S
Authentication	Active	T^*, C, S^*	Active	T^*, C^*, S

modeled. Regarding issuers, for the authentication experiment we assume the issuer behaves properly, meaning it initializes authenticators honestly and certifies them according to their attestation material. For unlinkability we assume malicious issuers and that existing issuers can be corrupted by an adversary. This includes an issuer revealing information about the certificates issued to the different authenticators in order to break their unlinkability.

After describing the threat model of the different involved entities, we turn now to additional assumptions during the respective phases. The *initialisation* phase does not need to be trusted during the unlinkability experiment. Therefore, we allow the adversary to initialise authenticators and servers with any issuer. In the case of the authentication experiment, we assume the communication channel between issuer and authenticator to be authenticated and confidential. Hence, it is considered trusted and we do not allow active or passive adversaries between these two entities. Such a direct connection could be possible during manufacturing time. We do allow, however, the adversary to access the issuer's public key—denoted by giving access to ic_S. We assume further, an active adversary in the *certification phase* (for both unlinkability and authentication experiments). We make no security assumptions on the communication channels between authenticator, client, and server in the *authentication* or *registration phases* for both authentication and unlinkability.

4.2 Oracles

We describe all oracles (except RLeft, RRight, ALeft, ARight) used in the unlinkability and authentication experiments in Figure 4.

During the game executions the adversary can create new servers and (initialised) authenticators through the oracles NewS and NewT, respectively. In these two oracles, initiate functions $\mathsf{Initiate}(S)$ and $\mathsf{Initiate}(T)$ are called, respectively. These functions run the respective initialisation operations for servers and

NewT(T, suppUV, (I, k)):
32 **if** $\mathsf{suppUV}_T \neq \bot$: **return** $\bot$
33 **if** $I \notin \mathcal{L}_I$: **return** $\bot$ // only if in $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}()$
34 Initiate (T, π^k_I)
35 $\mathsf{suppUV}_T \leftarrow \mathsf{suppUV}$
36 **return**

NewS(S, pkCP, attPol, (I, k)):
37 **if** $\mathsf{pkCP}_S \neq \bot$ **and** attPol $\neq \bot$: **return** $\bot$
38 **if** $I \notin \mathcal{L}_I$: **return** $\bot$ // only if in $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}()$
39 Initiate (S, π^k_I)
40 $\mathsf{pkCP}_S \leftarrow$ pkCP, $\mathsf{attPol}_S \leftarrow$ attPol
41 **return**

Corrupt(T, G):
42 **if** $T \in G$: **return** $\bot$
43 $\forall i$: $\mathcal{L}_{\mathsf{frsh}} \leftarrow \mathcal{L}_{\mathsf{frsh}} \setminus \{(S_i, T)\}$
44 **return** $\{\mathsf{rc}_T[S_i]\}_i, \mathsf{ic}_T, \mathsf{cc}_T$

CorruptI(I):
45 **return** $\mathsf{ic}_I, \mathsf{cc}_I$

cGen(T, h):
46 **if** $\pi^k_I \neq \bot$ **or** $\pi^h_T \neq \bot$ **or** $\mathsf{ic}_T = \bot$: **return** $\bot$
47 $m_{\mathsf{cgen}} \leftarrow_{\$} \mathsf{cGen}(\pi^h_T)$
48 **return** m_{cgen}

cResp($(I, k), m_{\mathsf{cgen}}, \mathsf{id}_T$):
49 **if** $\pi^k_I.\mathsf{st}_{\mathsf{exe}} \neq$ running: **return** $\bot$
50 $(m_{\mathsf{crsp}}, \mathsf{cc}_I) \leftarrow_{\$} \mathsf{cRsp}(\pi^k_I, \mathsf{id}_T, m_{\mathsf{cgen}})$
51 **return** m_{crsp}

cCompl($(T, h), m_{\mathsf{crsp}}$):
52 $\pi^h_T.\mathsf{st}_{\mathsf{exe}} \neq$ running: **return** $\bot$
53 $(\mathsf{cc}_T, d) \leftarrow \mathsf{cVrfy}(T, \mathsf{id}_I, m_{\mathsf{crsp}})$
54 **return** d

rChall((S, i), tb, UV):
55 **if** $\mathsf{pkCP}_S = \bot$ **or** $\mathsf{attInfo}_S = \bot$ **or** $\pi^i_S \neq \bot$: **return** $\bot$
56 $\pi^i_S.\mathsf{pkCP} \leftarrow \mathsf{pkCP}_S$, $\pi^i_S.\mathsf{attInfo} \leftarrow \mathsf{attInfo}_S$
57 $m_{\mathsf{rch}} \leftarrow_{\$} \mathsf{rChall}(\pi^i_S, \mathsf{tb}, UV)$
58 **return** m_{rch}

rResp($(T, j), m_{\mathsf{rcom}}$):
59 **if** $\mathsf{suppUV}_T = \bot$ **or** $\pi^j_T \neq \bot$ **or** $\mathsf{ic}_T = \bot$ **or** $\mathsf{cc}_T = \bot$:
60 **return** $\bot$
61 $(m_{\mathsf{rrsp}}, \mathsf{rc}_T, \mathsf{sid}, \mathsf{agCon}) \leftarrow_{\$} \mathsf{rRsp}(\pi^j_T, m_{\mathsf{rcom}})$
62 // set $\pi^j_T.\mathsf{sid} \leftarrow$ sid, $\pi^j_T.\mathsf{agCon} \leftarrow$ agCon in rRsp
63 **if** $\pi^j_T.\mathsf{sid} \in \{S_L, S_R\}$: $\mathcal{L}^{\mathsf{r}}_{\mathsf{ch}} \leftarrow \mathcal{L}^{\mathsf{r}}_{\mathsf{ch}} \cup (T)$ // only if in $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}()$
64 $\mathrm{Ar}_c[\pi^j_T.\mathsf{sid}] \leftarrow T$
65 **return** m_{rrsp}

rCompl($(S, i), m_{\mathsf{rcl}}, m_{\mathsf{rrsp}}$):
66 **if** $\mathsf{pkCP}_S = \bot$ **or** $\mathsf{attInfo}_S = \bot$ **or** $\pi^i_S.\mathsf{st}_{\mathsf{exe}} \neq$ running: **return** $\bot$
67 $(\mathsf{rc}_S, d, \mathsf{sid}, \mathsf{agCon}) \leftarrow_{\$} \mathsf{rVrfy}(\pi^i_S, m_{\mathsf{rcl}}, m_{\mathsf{rrsp}})$
68 # $\pi^i_S.\mathsf{sid} \leftarrow$ sid, $\pi^i_S.\mathsf{agCon} \leftarrow$ agCon in rVrfy
69 **if** $d = 1$ and $\mathrm{Ar}_c[\pi^i_S.\mathsf{sid}] \neq \bot$:
70 $\mathcal{L}_{\mathsf{frsh}} \leftarrow \mathcal{L}_{\mathsf{frsh}} \cup \{(S, \mathrm{Ar}_c[\pi^i_S.\mathsf{sid}])\}$
71 $\mathrm{Ar}_c[\pi^i_S.\mathsf{sid}] \leftarrow \bot$
72 **return** d

aChall((S, i), tb, UV):
73 **if** $\mathsf{pkCP}_S = \bot$ **or** $\mathsf{attInfo}_S = \bot$ **or** $\pi^i_S \neq \bot$: **return** $\bot$
74 $\pi^i_S.\mathsf{pkCP} \leftarrow \mathsf{pkCP}_S$, $\pi^i_S.\mathsf{attInfo} \leftarrow \mathsf{attInfo}_S$
75 $m_{\mathsf{ach}} \leftarrow_{\$} \mathsf{aChall}(\pi^i_S, \mathsf{tb}, UV)$
76 **return** m_{ach}

aResp($(T, j), m_{\mathsf{acom}}$):
77 **if** $\mathsf{suppUV}_T = \bot$ **or** $\pi^j_T \neq \bot$: **return** $\bot$
78 $\pi^j_T.\mathsf{suppUV} \leftarrow \mathsf{suppUV}_T$
79 $(m_{\mathsf{arsp}}, \mathsf{rc}_T) \leftarrow \mathsf{aRsp}(\pi^j_T, m_{\mathsf{acom}})$
80 **if** $\pi^j_T.\mathsf{sid} \in \{S_L, S_R\}$: $\mathcal{L}^{\mathsf{a}}_{\mathsf{ch}} \leftarrow \mathcal{L}^{\mathsf{a}}_{\mathsf{ch}} \cup (T)$ // only if in $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}()$
81 **return** m_{arsp}

aCompl($(S, i), m_{\mathsf{acl}}, m_{\mathsf{arsp}}$):
82 **if** $\pi^i_S = \bot$ **or** $\pi^i_S.\mathsf{st}_{\mathsf{exe}} \neq$ running : **return** $\bot$
83 $(\mathsf{rc}_S, d) \leftarrow \mathsf{aVrfy}(\pi^i_S, m_{\mathsf{acl}}, m_{\mathsf{arsp}})$
84 **if** $d = 1$ **and** $\mathsf{inGroup}(\mathsf{G}, m_{\mathsf{arsp}}, \mathsf{rc}_S) = 1$ **and** win-auth $= 0$:
85 win-auth $\leftarrow$ Win-auth(S, i)
86 **return** d

Fig. 4: Oracles for experiments defined in Figure 5 and 6; differences to the definitions of [BCZ23; HLW23] are highlighted in teal.

authenticators that we define in the instantiations in Section 5. By invoking the oracles rChall, rResp, and rCompl the adversary is able to actively interfere during the registration of authenticators. Moreover, via the oracles aChall, aResp and aCompl, it can actively interfere during authentication. It is important to note that we do not allow the adversary to initialize tokens and servers with information from an issuer created by the adversary itself during the authentication experiment as we assume the channel between issuer and authenticator to be secure as described above. However, the adversary is allowed to initialise authenticators and servers in the unlinkability experiment with information from an issuer created by such an adversary. Furthermore, the adversary can also query the Corrupt oracle to reveal an authenticator's registration,

certification, and initialisation contexts. It is important to emphasize that our CORRUPT definition differs from [BCZ23] in three aspects: 1) it is independent of the sever and as such it reveals *all* registration contexts of the authenticator, 2) it also reveals the initialisation and certification context of the authenticator as a natural extension to cover attestation modes, 3) and additionally it receives an authenticator group G which defines a set of authenticators that cannot be corrupted. The definition of the group is defined per instantiation. More information about authenticator groups is provided in Section 4.3. We additionally give the adversary access to the CORRUPTI oracle to reveal the internal state of an issuer contained in the initialization and certification contexts. Although the adversary can already read and modify the information exchanged between authenticator and issuer during the certification phase through the individual certification oracles, with this corrupt issuer oracle the adversary will also get access to any private keys or private configuration information an issuer may keep. This oracle will only be available in the unlinkability experiment since, as we mentioned in the previous section, we consider the issuer to be trusted in the authentication experiment.

In addition, in a particular stage of the unlinkability experiment, $\mathcal{A}$ will be given access to RLEFT, RRIGHT, ALEFT, ARIGHT oracles (defined in Figure 6), which will run the algorithms in the registration and authentication phases with two randomly assigned authenticators.

4.3 Authenticator Groups

We introduce a group of authenticators G as some set of authenticators that share the same attestation material att_m which is defined in $\mathsf{Initiate}$ and shared by the authenticators with the server during $\mathsf{Register}$. For example, a group may describe a set of authenticators which have attestation keys that have been issued by the same issuer, with the same validity period, or even the same attestation keys. For a meaningful definition of unlinkability, we assume that there are at least two authenticators per group.

We use a static group definition in the authentication security and unlinkability experiments, meaning that the adversary chooses the group of the target authenticator(s) in advance. However, this group could be defined dynamically given the actions of the adversary during the execution of the experiment. We claim that given that the execution of the protocol is independent of the group chosen by the adversary, the security results are the same.

We additionally define a method $\mathsf{inGroup}(\mathsf{G}, m_{\mathsf{arsp}}, \mathsf{rc}_S)$, which receives as input a group G, an authentication message response m_{arsp} and a server registration context rc_S. It outputs 1 if and only if m_{arsp} was created by an authenticator that is in G.

In the next two subsections, we give the notions of secure passwordless authentication and unlinkability in connection with a group of authenticators G. More concretely, when proving the property of passwordless authentication, the adversary is able to corrupt any authenticator but those in the same group as

the authenticator that is impersonated. In the unlinkability experiment, the adversary tries to distinguish two authenticators which are in the same group.

Intuitively, our definitions of different groups correspond to the following real-world scenarios. For example, an adversary may be restricted to corrupt authenticators of different brand / different models than the one it tries to impersonate. This corresponds to authenticators of different certification levels, of which some may be easier or harder to compromise. The different certification levels can be proven using attestation and as such, this oracle allows us to be more fine-grained regarding the adversary ability to corrupt entities for different attestation modes.

Section 5 includes a description of att_m, and therefore of G, for each instantiation using a different attestation mode.

4.4 Session Partnering

Partnering identifies authenticator, issuer, and server sessions that are successfully communicating with each other as expected, and is encoded through matching session identifiers. More precisely, we say a server (resp., issuer) session π_S^i (resp., π_I^k) *partners with an authenticator session* π_T^j if and only if $\pi_S^i.\mathsf{sid} = \pi_T^j.\mathsf{sid} \neq \bot$ (resp., $\pi_I^k.\mathsf{sid} = \pi_T^j.\mathsf{sid} \neq \bot$). We say a server (resp., issuer) session π_S^i (resp., π_I^k) *partners with an authenticator* T if it partners with one of T's sessions. We say an authenticator T is the *registration (resp., initialisation) partner* of a server S (resp., issuer I), if the registration context of T at S (resp., the initialisation or certification context at I) has been set, i.e., $\mathsf{rc}_T[\mathsf{id}_S] \neq \bot$ (resp., $\mathsf{ic}_T[\mathsf{id}_I] \neq \bot$ or $\mathsf{cc}_T[\mathsf{id}_I] \neq \bot$).

4.5 Passwordless Authentication Experiment for ePIAA

Aiming at similar security properties as [BCZ23], we say an ePIAA provides *passwordless authentication* (PAuth) if servers accept authentication responses if and only if they were generated by a unique honest partnered authentication session. The winning conditions are the same as in [BCZ23] except that we add that the adversary can also win by exploiting the certification (i.e., the attestation). Moreover, our definition depends on the choice of an authenticator group G. The resulting experiment is given in Figure 5.

We call a server session a *test session* if it accepts a response message coming from an authenticator in G. An ePIAA is secure if for every test session π_S^i (i.e., with $\pi_S^i.\mathsf{st}_{\mathsf{exe}} = \mathsf{accepted}$ and with $\mathsf{inGroup}(\mathsf{G}, m_{\mathsf{arsp}}, \mathsf{rc}_S) = 1$), such that none of the following four winning conditions hold:

1. the non-$\bot$ session identifiers of two authenticator sessions collide.
2. the non-$\bot$ session identifiers of two server sessions collide.
3. All the tokens that registered to the server S are fresh (not corrupted), yet, none of them has a partnering session[5]

[5] We have rephrased the respective condition in [BCZ23] which said "any registration partner of S" to avoid ambiguities. The meaning, however, has been maintained.

$\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G})$:

1 $\mathcal{L}_{\mathsf{frsh}} \leftarrow \emptyset$, win-auth $\leftarrow 0$, $\mathrm{Ar}_c \leftarrow \emptyset$
2 $N > 0$, for $k = 1..N$: $I_k \leftarrow \mathsf{Initiate}(I)$, $\mathcal{L}_I \leftarrow \{I_1..I_k\}$
3 $() \leftarrow_{\$} \mathcal{A}^{\mathcal{O}, I_1..I_k}(1^\lambda)$ //$\mathcal{A}$ can use different issuers to initialize authenticators and servers
4 **return** win-auth

$\mathsf{Win\text{-}auth}(S, i)$:

5 **if** $\exists (T_1, j_1), (T_2, j_2)$ such that $(T_1, j_1) \neq (T_2, j_2)$ **and** $\pi^{j_1}_{T_1}.\mathsf{sid} = \pi^{j_2}_{T_2}.\mathsf{sid} \neq \perp$: **return** 1
6 **if** $\exists (S_1, i_1), (S_2, i_2)$ such that $(S_1, i_1) \neq (S_2, i_2)$ **and** $\pi^{i_1}_{S_1}.\mathsf{sid} = \pi^{i_2}_{S_2}.\mathsf{sid} \neq \perp$: **return** 1
7 **if** $\forall T$ such that $\mathsf{rc}_T[\mathsf{id}_S] \neq \perp$, ($\neg \exists j$ such that $\pi^i_S.\mathsf{sid} = \pi^j_T.\mathsf{sid}$, **and** $(S, T) \in \mathcal{L}_{\mathsf{frsh}}$) : **return** 1
8 **if** $\exists (S', i'), (T', j')$ such that $\pi^{i'}_{S'}.\mathsf{sid} = \pi^{j'}_{T'}.\mathsf{sid} \neq \perp$ **and** $(S', T') \in \mathcal{L}_{\mathsf{frsh}}$ **and** $\pi^{i'}_{S'}.\mathsf{agCon} \neq \pi^{j'}_{T'}.\mathsf{agCon}$ **and** (**if** $\mathsf{G} \neq \{\}$ then $T' \in \mathsf{G}$): **return** 1
9 **return** 0

Fig. 5: Security experiment for ePIAA Protocols $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$, with oracles $\mathcal{O}$ defined in Figure 4; differences to [BCZ23] are highlighted in teal.

4. the agreed contents of a pair of partnered server session $\pi^{i'}_{S'}$ and authenticator session $\pi^{j'}_{T'}$ are distinct and $\textsc{Corrupt}(T', \mathsf{G})$ has not been queried.

Finally, we define authentication security for ePIAA.

Definition 1 (PAuth for ePIAA). *Let* $\mathsf{Compl} \in \{\mathsf{PPT}, \mathsf{QPT}\}$. *Let* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *be an Extended Passwordless Authentication Protocol with Attestation. We say that for any group* G *of authenticators sharing the same attestation material* att_m, ePIAA *provides* secure passwordless authentication*, or* PAuth *for short, if for all* Compl *adversaries* $\mathcal{A}$ *the advantage*

$$\mathsf{Adv}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) := \Pr\left[\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) = 1\right]$$

in winning the game $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}$ *defined in Figure 5 is negligible in the security parameter* λ*. During the game* $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}$*,* $\mathcal{A}$ *has access to the following oracles:* cGen, cResp, cCompl, rChall, rResp, rCompl, aChall, aResp, aCompl, NewT, NewS, *and* Corrupt *(see Figure 4).*

4.6 Unlinkability Experiment for ePIAA

The first formal definition of privacy in FIDO2, together with an analysis of attestation mode self has been given in [HLW23]. Privacy (or more precisely *unlinkability*) essentially means that different registrations of the same authenticator can not be linked, either in one server or across several servers. On the other hand, a server may be able to link several authentication sessions to the same authenticator in terms of being related to the same registration event, however different servers may not be able to link different authentication sessions to the same device. Data that is exchanged outside of the protocol is out

of the scope of the definition (e.g., metadata that could be used to link interactions of the authenticator(s)). [HLW23] defines three types of unlinkability: strong, medium, and weak. We adapt their definition to the case of residential credentials, mostly to rule out trivial attacks and to focus on the unlinkability properties of different attestation modes, and resulting in a definition closest to the *medium unlinkability* in [HLW23].

We additionally use the notion of *group unlinkability*, meaning that a protocol provides unlinkability as long as the adversary is restricted to try to link authenticators of the same group G (see Section 4.3). Similarly to the authentication experiment where we use the CORRUPT oracle that receives a group G as input, the definition of *group* may change for each one of the attestation modes.

In [CJR22], the authors propose a new attestation protocol named TokenWeaver that promises higher privacy guarantees and as such is interesting as an additional attestation mode in FIDO2. Therefore, our extension of the unlinkability definition from [HLW23] to cover different attestation modes, is designed to also apply to TokenWeaver.

We define in Figure 6 the unlinkability experiment consisting of three phases:

Phase 1. The Adversary $\mathcal{A}_1$ gets as input the group G and a set of issuers initialised by the experiment, and is allowed to interact with the oracles defined in Figure 4. Its output is stored in st_1.

Phase 2. The Adversary $\mathcal{A}_2$ gets as input st_1 and group G and chooses two target authenticators $T_0, T_1 \in \mathsf{G}$ and two target servers S_L, S_R. The challenger runs InitRL, initialising the oracles RLEFT, ALEFT, RRIGHT, ARIGHT, where S_L, S_R have been assigned to R/ALEFT and R/ARIGHT respectively and T_0 and T_1 have been randomly assigned to either R/ALEFT or R/ARIGHT using a bit b. Additionally, $\mathcal{A}_2$ provides an output st_2.

Phase 3. The Adversary $\mathcal{A}_3$ receives the output st_2 of the previous adversary and is allowed to interact with all the oracles defined so far, except creating new authenticators with NEWT or certify / re-certify the chosen authenticators T_0 and T_1 through oracles CGEN, CRESP, and CCOMPL (see Figure 6). This is to ensure that 1) when we define unlinkability with respect to a group G, both authenticators remain in this group through the next phases of the experiment, and 2) that the whole set of authenticators participating in the experiment is stable. Oracles R/ALEFT and R/ARIGHT are oracles through which the adversary can query either authenticator T_0 or T_1 to run the algorithms meant to be run by the authenticator in the registration (RLEFT, RRIGHT) and authentication (ALEFT, ARIGHT) phases of the ePIAA protocol. Finally, $\mathcal{A}_3$ outputs a bit $\hat{b}$.

Following [HLW23], we define a number of lists using our notation to rule out trivial attacks in the experiments:

$\mathcal{L}^{\mathsf{r}}_{\mathsf{ch}}$**:** all authenticators for which the RRESP oracle was called to register with servers S_L or S_R.

$\mathcal{L}^{\mathsf{a}}_{\mathsf{ch}}$**:** all authenticators for which the ARESP oracle was called to authenticate with servers S_L or S_R.

$\mathcal{L}^{\mathsf{r}}_{\mathsf{lr}}$**:** authenticators $\in \{T_0, T_1\}$ for which the RLeft or RRight oracles were called to register with servers S_L or S_R.
$\mathcal{L}^{\mathsf{a}}_{\mathsf{lr}}$**:** authenticators $\in \{T_0, T_1\}$ for which the ALeft or ARight oracles were called to authenticate with servers S_L or S_R.

The experiment outputs 1 if

1. $\hat{b} = b$ and $S_{\mathsf{unl}^*} = (\mathcal{L}^{\mathsf{r}}_{\mathsf{ch}} \cap \mathcal{L}^{\mathsf{a}}_{\mathsf{lr}}) \cup (\mathcal{L}^{\mathsf{r}}_{\mathsf{lr}} \cap \mathcal{L}^{\mathsf{a}}_{\mathsf{ch}})$ is empty,
2. $\mathcal{A}$ has not corrupted the authenticators T_0 and T_1 (therefore, $\mathcal{A}$ has not gotten access to any credentials stored in such authenticators as part of the registration with servers S_L or S_R), and
3. T_0 and T_1 belong to the same group G of authenticators sharing the same attestation information.

The main differences between this experiment, depicted in Figure 6, and the one in [HLW23] are highlighted in teal. Essentially, the changes arise since we consider attestation modes and residential (instead of non-residential) keys. More concretely, 1) we allow the adversary $\mathcal{A}_1$ to create and corrupt authenticators (with restrictions reflected through $\mathcal{L}_{\mathsf{frsh}}$); 2) we add initialization and certification phases involving an issuer; 3) we consider authenticator groups G defined through shared attestation information; 4) as mentioned above, we consider the medium unlinkability notion from [HLW23], as our experiment conditions imply that the set $\mathcal{L}^{\mathsf{a}}_{\mathsf{lr}} \cup \mathcal{L}^{\mathsf{a}}_{\mathsf{ch}}$ is empty. Still the experiment allows us to analyse unlinkability in terms of servers not being able to link different registrations to the same authenticator.

Definition 2 (Group unlinkability for ePIAA). *Let* $\mathsf{Compl} \in \{\mathsf{PPT}, \mathsf{QPT}\}$. *Let* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *be an Extended Passwordless Authentication Protocol with Attestation. We say that for any group* G *of authenticators sharing the same attestation material* att_m*,* ePIAA *provides* group unlinkability *(*Unl*), if for all* Compl *adversaries* $\mathcal{A}$ *the advantage*

$$\mathsf{Adv}^{\mathsf{Unl}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) := \Pr\left[\mathsf{Expt}^{\mathsf{Unl}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) = 1\right]$$

in winning the game $\mathsf{Expt}^{\mathsf{Unl}}_{\mathsf{ePIAA}}$ *defined in Figure 6 by trying to distinguish authenticators in the same group* G *is negligible in the security parameter* λ*. During* $\mathsf{Expt}^{\mathsf{Unl}}_{\mathsf{ePIAA}}$*,* $\mathcal{A}$ *has access to the following oracles:* cGen*,* cResp*,* cCompl*,* rChall*,* rResp*,* rCompl*,* aChall*,* aResp*,* aCompl*,* NewT*,* NewS*,* Corrupt*, and* CorruptI *(see Figure 4).*

5 WebAuthn and Different Attestations Modes as Instantiations of ePIAA

In this section we explain how we abstract WebAuthn with different attestation modes as ePIAA, and provide the different attestation modes and their analysis regarding their unlinkability and passwordless authentication.

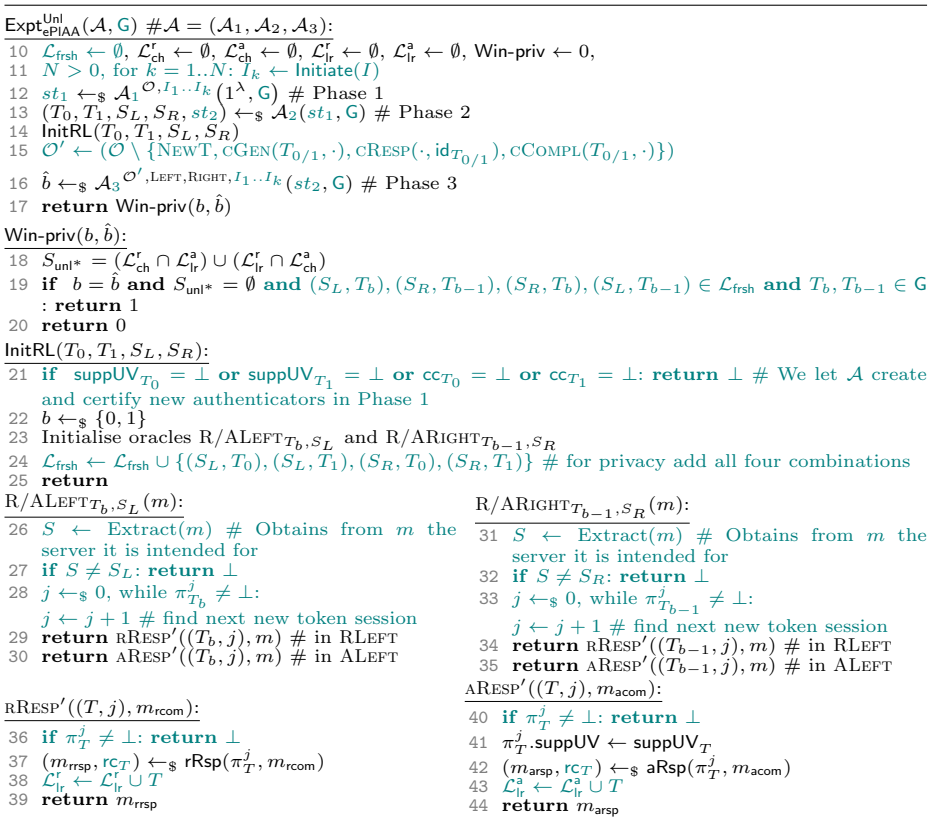

Fig. 6: Group unlinkability experiment for ePlAA Protocols $\mathsf{ePlAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$, with oracles $\mathcal{O}$ defined in Figure 4 and $\mathsf{Compl} \in \{\mathsf{PPT}, \mathsf{QPT}\}$. Differences to [HLW23] are highlighted in teal.

5.1 WebAuthn with Attestation as ePlAA

It is important to recall that we do not consider the RP's preference enterprise (which would signal to the authenticator to include uniquely identifying information), and we (only) consider mode direct (i.e., the RP is interested in attestation directly from the authenticator and unaltered by the client, see Section 2.1). Therefore, when adding attestation to the abstraction of [BCZ23], only the authenticator's response (rRsp) and the RP's verification (rVrfy) change during Register $= (\mathsf{rChall}, \mathsf{rCom}, \mathsf{rRsp}, \mathsf{rVrfy})$, see Figure 7. In addition, Initiate and Certification change when instantiating WebAuthn with different attestation modes and is therefore described in the following subsections. Authenticate, however, stays the same for all attestation modes and is depicted in Figure 3.

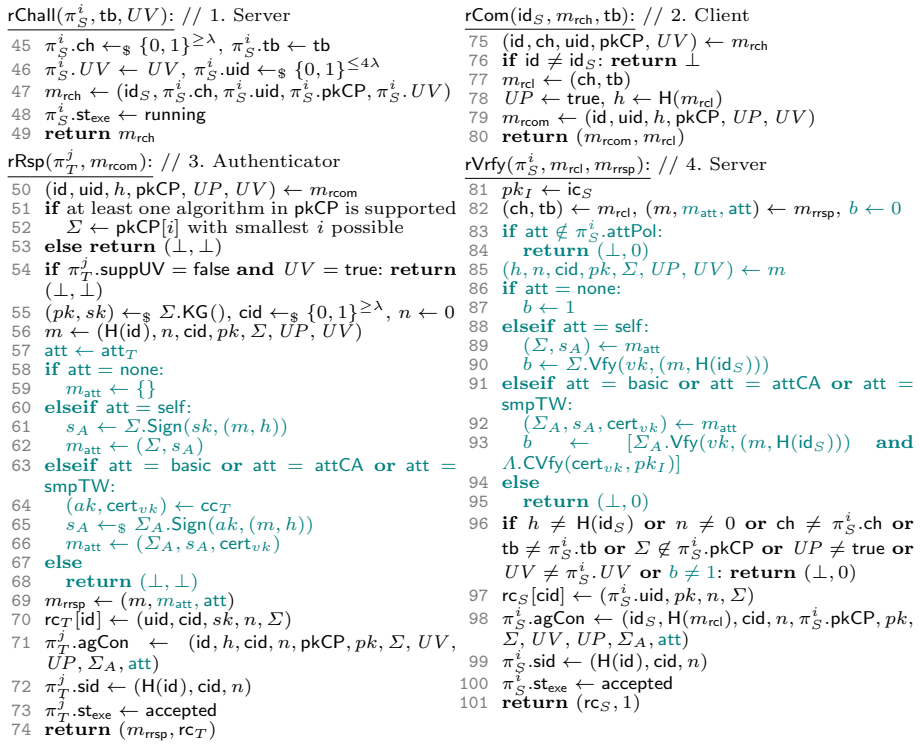

Fig. 7: Registration of authenticators using different attestation modes of WebAuthn as ePIAA = (Initiate, Certification, Register, Authenticate), with Initiate and Certification as described in the figures of the respective attestation modes and Authenticate as in Figure 3; operations needed for attestation are in teal.

We denote the attestation certificate generated (using CG) by the issuer over the public attestation key vk with cert_{vk}. Certification verification is denoted by CVfy using the issuer's public key pk_I.

According to the specifications, during the authenticator's response, it proceeds to create the credential key pair and decides which attestation mode to use. Depending on the attestation, the authenticator proceeds differently as outlined in Figure 7. In all cases it generates an attestation statement m_{att}, which is included in m_{rrsp} together with the chosen attestation mode att.

From the specifications, it is however, not clear how the attestation mode is chosen. To analyse the unlinkability and authentication properties, we assume that during Initiate, the manufacturer defines the attestation mode $\mathsf{att}_T \in \{\mathsf{none}, \mathsf{self}, \mathsf{basic}, \mathsf{attCA}, \mathsf{smpTW}\}$ on each authenticator, depending on its soft- and hardware capabilities. It is important to emphasize that this does not necessarily reflect *all* the attestation modes that an authenticator *can* support. For example, all authenticators can support attestation modes none and self, even if att_T = basic. It is still reasonable to assume that the authenticator always

tries the most advanced attestation, as the authenticator only gets the RP's attestation preference direct, but no further instructions. This is different for RP preference none, as specifications define that in this case the authenticator uses only self or none. Since in our analysis self and none do not have different properties regarding Definition 1 and Definition 2, we decided to refrain from modelling this detail to keep it simpler.

It is interesting to note that while all authenticators should be able to use self attestation, it might be desirable not to. More concretely, while the advantage of self is that the authenticator 'proves' the ownership of the secret credential key, the gain of an adversary registering a public credential key without knowledge of the secret key is rather low. Therefore, the advantage of this extra guarantee is unclear. In addition, the attestation signature (in particular, when considering the potential larger PQ signatures) poses an overhead. Therefore, manufacturers might choose none over self. Indeed, none is the default attestation mode.

Moving on with the description of Figure 7, in rVrfy the RP checks whether att is allowed under its policies (attPol). If yes, it proceeds depending on the attestation mode. Otherwise, it rejects the registration attempt, updates rc_S, outputs $d = 0$, and notifies the user (out-of-band) that the authenticator does not meet the RPs requirements.

5.2 Attestation Mode none and self

For the attestation modes none and self, there is essentially no attestation. As such there is no attestation key pair, and therefore essentially no certification. This is shown in Figure 8.

Instead, an 'empty' attestation signature (in case of none) or a signature generated using the secret credential key (in case of self) is sent. As such, self attestation proves the knowledge of the credential secret key.

Given that the attestation material att_m is empty for these attestation modes, the group G is empty too. For the passwordless authentication experiment, this means that the adversary can corrupt *any* authenticator (however test sessions

Initiate

Initiate(I): // 1. Issuer
102 $\mathsf{ic}_I \leftarrow \{\}$
103 **return**
Initiate(S, π_I^k): // 2. Server
104 $\mathsf{ic}_S \leftarrow \{\}$
105 **return**
Initiate(T, π_I^k): // 3. Authenticator
106 $\mathsf{att}_T \leftarrow$ none or self
107 $\mathsf{att}_m \leftarrow \{\}$
108 $\mathsf{ic}_T \leftarrow \mathsf{ic}_T \cup \{\mathsf{att}_T, \mathsf{att}_m\}$
109 **return**

Certification

cGen(π_T^j): // 1. Token
110 **return** $m_{\mathsf{cgen}} \leftarrow \{\}$
cRsp($\pi_I^k, \mathsf{id}_T, m_{\mathsf{cgen}}$): // 2. Issuer
111 **return** $(m_{\mathsf{crsp}}, \mathsf{cc}_I) \leftarrow (\{\}, \{\})$
cVrfy($\pi_T^j, \mathsf{id}_I, m_{\mathsf{crsp}}$): // 3. Authenticator
112 **return** $(\mathsf{cc}_T, d) \leftarrow (\mathsf{ic}_T, 1)$

Fig. 8: Initialisation and certification of WebAuthn with attestation none and self as an ePIAA = (Initiate, Certification, Register, Authenticate); Register and Authenticate as in Figure 7 and 3, respectively.

$\text{REG}((S,i),(T,j),\mathsf{tb},UV)$:

113 **if** $\mathsf{pkCP}_S = \bot$ **or** $\mathsf{attInfo}_S = \bot$ **or** $\mathsf{suppUV}_T = \bot$ **or** $\pi_S^i \neq \bot$ **or** $\pi_T^j \neq \bot$ **or** $\mathsf{rc}_T[S] \neq \bot$ **or** $\mathsf{ic}_T = \bot$ **or** $\mathsf{cc}_T = \bot$: **return** $\bot$
114 $\pi_S^i.\mathsf{pkCP} \leftarrow \mathsf{pkCP}_S$, $\pi_S^i.\mathsf{attInfo} \leftarrow \mathsf{attInfo}_S$, $\pi_T^j.\mathsf{suppUV} \leftarrow \mathsf{suppUV}_T$
115 $m_{\mathsf{rch}} \leftarrow_\$ \mathsf{rChall}(\pi_S^i, \mathsf{tb}, UV)$
116 $(m_{\mathsf{rcom}}, m_{\mathsf{rcl}}) \leftarrow \mathsf{rCom}(\mathsf{id}_S, m_{\mathsf{rch}}, \mathsf{tb})$
117 $(m_{\mathsf{rrsp}}, \mathsf{rc}_T, \mathsf{sid}, \mathsf{agCon}) \leftarrow_\$ \mathsf{rRsp}(\pi_T^j, m_{\mathsf{rcom}})$
118 $(\mathsf{rc}_S, d, \mathsf{sid}, \mathsf{agCon}) \leftarrow_\$ \mathsf{rVrfy}(\pi_S^i, m_{\mathsf{rcl}}, m_{\mathsf{rrsp}})$
119 $\mathcal{L}_{\mathsf{frsh}} \leftarrow \mathcal{L}_{\mathsf{frsh}} \cup \{(S,T)\}$
120 **if** $S \in \{S_L, S_R\}$: $\mathcal{L}^r_{\mathsf{ch}} \leftarrow \mathcal{L}^r_{\mathsf{ch}} \cup (T)$ // only if in $\mathsf{Expt}^{\mathsf{Unl}}_{\mathsf{ePIAA}}()$
121 **return** $(m_{\mathsf{rch}}, m_{\mathsf{rcl}}, m_{\mathsf{rcom}}, m_{\mathsf{rrsp}}, d)$

Fig. 9: Oracle description of REG; differences to [BCZ23] are highlighted in teal.

that originate from a call to this oracle are excluded from the winning conditions). During the unlinkability experiment the adversary can choose any two authenticators.

Since attestation modes none and self do not give any security assurance during Register, trust on first use has been assumed in [BCZ23]. In Definition 1 of PAuth, we, however, allow an active adversary. This definition can not be satisfied by WebAuthn with attestation mode none or self. Therefore, we define a weaker variant of PAuth (called PAuth-w), assuming a passive adversary as in [BCZ23].

The difference between PAuth and PAuth-w is that instead of accessing the oracles RCHALL, RRESP, and RCOMPL, the adversary is only allowed access to an oracle REG given in Figure 9. By invoking the REG oracle, $\mathcal{A}$ is able to eavesdrop on honest registrations between servers and authenticators of its choice. From the definition of PAuth and PAuth-w it is clear that protocols providing PAuth, also provide PAuth-w.

We can prove WebAuthn with attestation types none or self to be PAuth-w secure and provide the theorem and proof in Appendix A. This theorem is essentially the same as [BCZ23, Theorem 1] adapted to our ePIAA class definition.

Likewise, we prove in Appendix A that WebAuthn with none or self is unlinkable. The proof idea is simply that since there is no attestation, there is also no information provided that can be used to link the tokens.

5.3 Attestation Mode basic

In the basic attestation mode, the authenticator's attestation key pair is shared with at least 99,999 other authenticators [W3C21, Sec 14.4.1], forming a *batch* (e.g., all devices from a specific model). The secret attestation key is embedded in the authenticator, together with a certificate for the public attestation key. Therefore, essentially there is no Certification and most operations are done during Initiate as shown in Figure 10, e.g., the attestation key is generated and put on all hardware authenticators of a batch during Initiate.

We define the attestation material att_m as consisting of the certificate of the respective batch attestation public key cert_B, and the issuer's public key vk_I. This means that in the passwordless authentication experiment the adversary

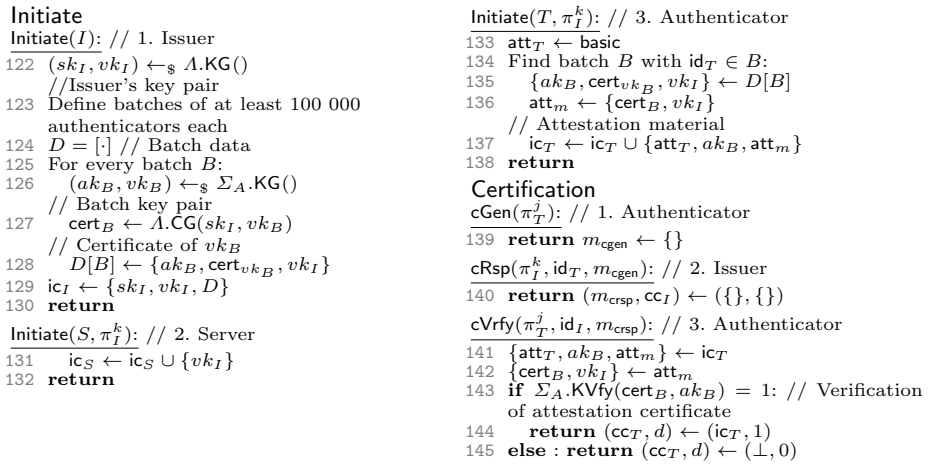

Fig. 10: Initialisation and certification of WebAuthn with attestation mode basic as ePIAA = (Initiate, Certification, Register, Authenticate); Register and Authenticate as in Figure 7 and 3, respectively.

won't be able to corrupt authenticators from the same batch as the one for which a successful authentication needs to happen according to the winning conditions. In the unlinkability experiment, the adversary will try to distinguish between two authenticators from the same batch.

During rRsp (see Figure 7), data is signed using the secret attestation key ak and sent together with m_{rrsp} and the certificate cert_{vk}. During rVrfy, the server will check that the attestation type matches those supported by the server, and verify the attestation signature and the attestation certificate. We assume (see Figure 10), the issuer public key is shared out of band with the server in advance.

Passwordless Authentication. An adversary making a CORRUPT query for an authenticator of the same batch as the winning authenticator, would be able to generate attestation signatures. To exclude this attack, we allow the adversary to corrupt only authenticators of other batches than the test session. More concretely, we define the authenticator group G as the batch of the winning authenticator. This enables us to allow malicious adversaries also during registration, formalised in the next theorem. It is important to emphasize that while restricting authenticator corruptions to the group limits the adversary's power, the statement is still a significant improvement over previous analysis that needed to assume an honest registration.

Theorem 1 (PAuth **of WebAuthn with** basic)**.** *Let WebAuthn with attestation mode* basic *be an instantiation of an ePlAA* ePIAA = (Initiate, Certification, Register, Authenticate) *as in Figure 10, 7, and 3. Let* 2^{λ_1} *and* 2^{λ_2} *be the sizes of the value spaces for credential id* cid *and the challenge nonce sampled during*

authentication respectively. Moreover, Let G *be a group of authenticators sharing the same attestation materials* att_m*, i.e., the same batch. Assume that the underlying function* H *is* $\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}$*-collision resistant, and the signature schemes* Σ, Σ_A *are* $\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *and* $\epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *secure against* $\mathsf{PPT/QPT}$ *adversaries. For any* $\mathsf{PPT/QPT}$ *adversary* $\mathcal{A}$ *against* PAuth *of* ePIAA *for a test session* π*, it holds that*

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathtt{WebAuthn\text{-}basic}}(\mathcal{A}, \mathsf{G}) \leq \binom{q_{\mathrm{RRESP}}}{2} 2^{-\lambda_1} + \binom{q_{\mathrm{ACHALL}}}{2} 2^{-\lambda_2} + \epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}} + 2q_{\mathrm{RRESP}} \cdot (\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}).$$

Proof sketch. The main part of the proof is very similar to that for the mode none (cf. Appendix A), which is based on the analysis in [BCZ23]. Namely, we first assume that credential ids cid and random challenges ch are unique, and that hash functions with different inputs produce different outputs. Assuming unique identifiers and collision-resistant hash functions, we can rule out an adversary winning via conditions 1 and 2. $\mathcal{A}$ needs to forge a valid authentication message m_{arsp} without the collaboration of an authenticator in order to win with condition 3. For that, either $\mathcal{A}$ forges the assertion signature (i.e., it needs to be able to break the $\mathsf{euf\text{-}cma}$ security of the assertion signature scheme), or it registers a different credential during the registration phase (i.e., it needs to forge a valid registration message m_{rrsp}). Given that $\mathcal{A}$ is not allowed to corrupt authenticators of the same group, which share the same attestation key that signs such message, $\mathcal{A}$ would need to break the $\mathsf{euf\text{-}cma}$ security of the attestation signature scheme. Similarly, $\mathcal{A}$ may win by forging the attestation signature via condition 4.

Unlinkability. Analysing the unlinkability of WebAuthn with attestation mode basic is similar to the mode none (cf. Appendix A) as well, except that for basic the adversary is restricted to choose authenticators T_0 and T_1 from the group G (i.e., the same batch B containing at least two authenticators). We provide the formal statement next.

Theorem 2 (Unl **of WebAuthn with** basic**).** *Let WebAuthn with attestation mode* basic *be an instantiation of an ePIAA* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *as in Figure 10, 7, and 3. Let* G *be the group of authenticators in the same batch* B*. Then for a* $\mathsf{PPT/QPT}$ *adversary* $\mathcal{A}$ *it holds*

$$\mathsf{Adv}^{\mathsf{Unl}}_{\mathtt{WebAuthn\text{-}basic}}(\mathcal{A}, \mathsf{G}) = 0.$$

Proof sketch. Given that $\mathcal{A}$ is restricted to select two authenticators of the same batch, the attestation key used to sign the registration message and the certificate created by the issuer will be the same for both authenticators. Therefore, no information is provided to the adversary that can be used to distinguish T_0 and T_1. This is true even if the adversary corrupts issuers or if it provides invalid information to the authenticators or servers during initialization as the registration will fail.

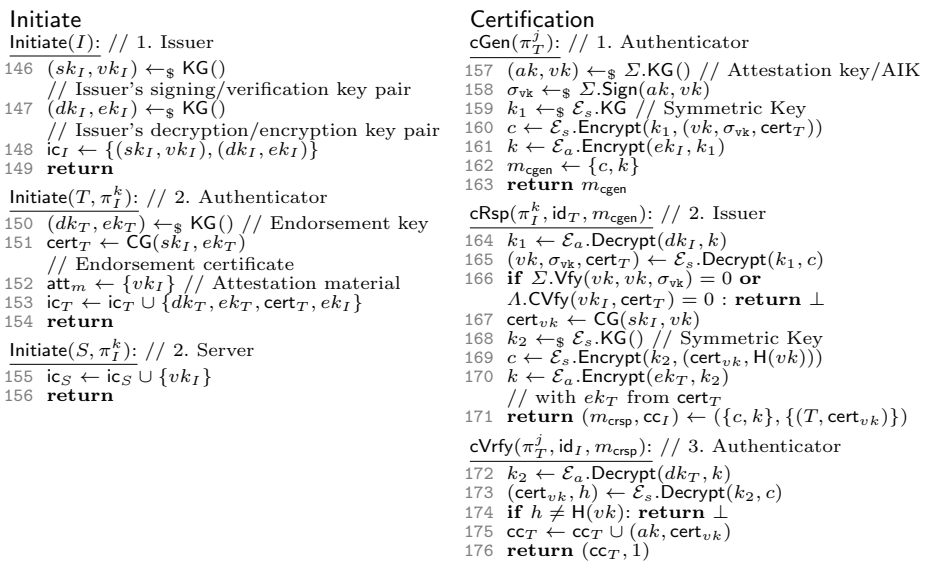

Fig. 11: Initialisation and certification of WebAuthn with attestation attCA as an ePIAA = (Initiate, Certification, Register, Authenticate); Register and Authenticate as in Figure 7 and 3, respectively.

5.4 Attestation Mode attCA

For attestation mode attCA, it is assumed that the authenticator is based on a trusted platform module (TPM) that holds a specific endorsement key (EK) [W3C21; Gro11]. This key is used to authenticate subsequent communications with the attestation CA. A TPM may create multiple attestation identity key (AIK) pairs and asks the issuer for a certificate for each AIK. The Initiate and Certification phases of this attestation mode are depicted in Figure 11.

The attestation material att_m is defined as the issuer's public key vk_I. In the passwordless authentication experiment, the adversary is not allowed to corrupt authenticators with AIKs certified by the same issuer; in the unlinkability experiment, the adversary will have to try to distinguish between two authenticators with AIKs certified by the same issuer.

Passwordless Authentication. The authentication security analysis of Webauthn with attCA given formally in the next theorem is very similar to using attestation mode basic. The main difference is that the adversary can also try to win the experiment by forging the certificate of the endorsement key.

Theorem 3 (PAuth **of WebAuthn with** attCA)**.** *Let WebAuthn with attestation mode* attCA *be an instantiation of an ePIAA* ePIAA = (Initiate, Certification, Register, Authenticate) *as in Figure 11, 7, and 3. Let* 2^{λ_1} *and* 2^{λ_2} *be the sizes*

of the value spaces for credential id cid *and the challenge nonce sampled during authentication, respectively. Moreover, Let* G *be a group of authenticators sharing the same attestation materials* att_m *(i.e., authenticators with attestation and endorsement keys certified by the same issuer signing keypair). Assume that the underlying function* H *is* $\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}$*-collision resistant, the signature schemes* Σ, Σ_A *are* $\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *and* $\epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *secure, and the certificate generation scheme* Λ *is also* $\epsilon_{\Sigma_\Lambda}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *secure against* $\mathsf{PPT/QPT}$ *adversaries. For any* $\mathsf{PPT/QPT}$ *adversary* $\mathcal{A}$ *against* PAuth *of* ePIAA *for a test session* π*, it holds that*

$$\mathsf{Adv}^{\mathsf{PAuth\text{-}w}}_{\mathtt{WebAuthn\text{-}attCA}}(\mathcal{A}, \mathsf{G}) \leq \binom{q_{\mathrm{RResp}}}{2} 2^{-\lambda_1} + \binom{q_{\mathrm{AChall}}}{2} 2^{-\lambda_2}$$
$$+\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}} + 2q_{\mathrm{RResp}} \cdot (\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}) + 2q_{\mathrm{CGen}} \cdot \epsilon_{\Sigma_\Lambda}^{\mathsf{euf\text{-}cma}}.$$

Proof sketch. The proof is very similar to the proof of Theorem 1 with the addition that the adversary can also win via conditions 3 and 4 as follows. The adversary could forge a valid certificate cert_T (i.e., it breaks the $\mathsf{euf\text{-}cma}$ security of the certificate generation scheme) for an endorsement key pair by its own choice. Moreover, it generates an AIK and asks the issuer to certify the AIK. The issuer will generate the certificate for the public AIK, and encrypts it under the adversary-chosen endorsement key.

Unlinkability. Different from the other attestation modes, during WebAuthn with attCA the issuer is trusted. In particular, issuers keep track of authenticators and their attestation certificates. As such, an adversary corrupting an issuer via CorruptI can trivially win the Unl experiment. Therefore, we do not give access to CorruptI, defining a weaker unlinkability notion (called $\mathsf{Unl\text{-}w}$). From the definition of Unl and $\mathsf{Unl\text{-}w}$ it is clear that protocols providing Unl, also provide $\mathsf{Unl\text{-}w}$.

Under the assumptions defined for $\mathsf{Unl\text{-}w}$, analysing the unlinkability of WebAuthn with attestation mode attCA is the same as in the previous modes. The authenticator group G essentially corresponds to all authenticators with attestation certificates generated by the same issuer. As such $\mathcal{A}$ has to choose two authenticators with attestation keys certified with the same issuer key pair (i.e., $\forall T \in \mathsf{G}$, $(ak_T, \mathsf{cert}_{vk^T})$ are such that $\Lambda.\mathsf{CVfy}(vk_I, \mathsf{cert}_{vk^T}) = 1$). When calling to the registration oracles, these authenticators provide attestation signatures generated with different attestation keys. They can, however, be verified with the same issuer public key vk_I. This way, there is no additional information provided to $\mathcal{A}$ that provides an advantage for winning the game.

Theorem 4 ($\mathsf{Unl\text{-}w}$ **of WebAuthn with** attCA**).** *Let WebAuthn with attestation mode* attCA *be an instantiation of an ePlAA* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *as in Figure 11, 7, and 3. Let* G *be the group of authenticators where* $\forall T \in \mathsf{G}$*,* $(ak_T, \mathsf{cert}_{vk^T})$ *are such that* $\Lambda.\mathsf{CVfy}(vk_I, \mathsf{cert}_{vk^T}) = 1$*. Then for a* $\mathsf{PPT/QPT}$ *adversary* $\mathcal{A}$ *it holds*

$$\mathsf{Adv}^{\mathsf{Unl\text{-}w}}_{\mathtt{WebAuthn\text{-}attCA}}(\mathcal{A}, \mathsf{G}) = 0.$$

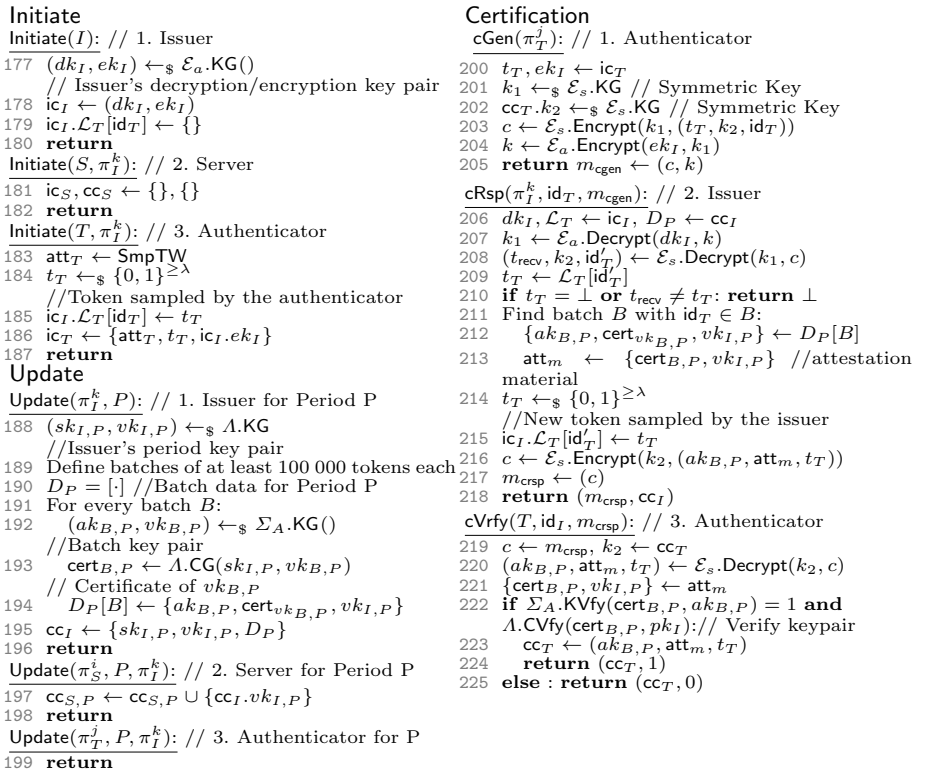

Fig. 12: Initialisation and certification of WebAuthn with the new attestation type smpTW.

Proof sketch. The proof follows from the proof of Theorem 8 with the change that no access is given to CorruptI.

6 Simple TokenWeaver as a New Attestation Mode for WebAuthn

WebAuthn with attestation mode basic is currently the only mode that provides both PAuth and Unl. One limitation is that Unl is limited for authenticators in the same (large) batch. The requirement for large batch size leads to a more significant limitation—PAuth is preserved as long asno other authenticator in the batch has been corrupted. Compromising a single authenticator means that we lose PAuth for any server that supports the batch's attestation key. The only way to recover security is to revoke the entire batch of at least 100,000 authenticators! In this section, we propose a new attestation mode called smpTW that is able to overcome this limitation.

$\underline{\text{OUPDATEI}(I, P)}$:	$\underline{\text{OUPDATES}(S, I, P)}$:
226 **if** $\mathsf{ic}_I = \bot$: **return** $\bot$	228 **if** $\mathsf{cc}_I = \bot$ **or** $\mathsf{ic}_I = \bot$: **return** $\bot$
227 $\mathsf{Update}(I, P)$	229 $\mathsf{Update}(S, P)$

Fig. 13: Description of new oracles to run Update for issuers and servers.

6.1 The smpTW att Mode

WebAuthn with smpTW is very similar to WebAuthn with attestation mode basic, but with an added mechanism that allows for periodic provisioning of new attestation keys to achieve PCS. smpTW is described in Figure 12.

As in basic, the same attestation keys and certificates are provided to a batch of authenticators, who will use them to sign the messages during registration in a remote web server. However, unlike basic, in smpTW the attestation keys are only valid for a limited time period. During Initiate, the authenticator is provisioned only with a secret one-time token, and the issuer's public encryption key.

For each period, the Update algorithms are called to generate the new period's attestation keys for all batches, and provision the servers with the new public keys. The new attestation keys are provisioned to the authenticators during Certification. Communication during Certification is protected using symmetric keys generated by the authenticator and sent to the issuer encrypted under its public key. During Certification, the authenticator "spends" its one-time token in exchange for a new valid token and the next period's attestation keys.

This periodic certification protocol allows for the security of the entire batch to "heal" after an authenticator from the batch and its attestation keys were compromised. If a compromised authenticator runs the Certification algorithms before the adversary it will "lock" the adversary out (as the token can only be used once) and security will be recovered. If the adversary runs Certification first, the user will learn that it was compromised as they are "locked out". The user can then ask the issuer to revoke the compromised authenticator and security will be recovered after the next period's Update. We combine this new mode, with a requirement for periodic re-registration of authenticators by the remote servers to fully "heal" and achieve PCS for the authentication process. For a full discussing of the PCS property we refer to [CJR22].

smpTW is based on the TokenWeaver general attestation scheme proposed in [CJR22]. While the original scheme allows for instant revocation of attestation keys of specific users and might allow for faster detection or recovery from compromises, smpTW is simpler to implement and—contrary to TokenWeaver—it is based on cryptographic primitives with post-quantum variants that are currently being standardized.

For the security and privacy analysis of this attestation mode, we introduce in Figure 13 two new oracles to support the new Update algorithms. In Appendix D, we provide the full analysis of the PAuth and Unl properties of smpTW and we prove the following theorems:

Theorem 5 (PAuth **Security of WebAuthn with** smpTW)**.** *Let* Compl $\in \{\mathsf{PPT}, \mathsf{QPT}\}$*. Let WebAuthn with attestation mode* smpTW *be an instantiation of an ePlAA* ePlAA $= (\mathsf{Initiate}, \mathsf{Update}*, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *as in Figure 12, 7, and 3. Let* 2^{λ_1}*,* 2^{λ_2} *and* 2^{λ_3} *be the sizes of the value spaces for credential id* cid*, the challenge nonce sampled during authentication, and tokens* t_T *used during certification, respectively. Moreover, let* G *be a group of authenticators sharing the same attestation materials* att_m *(i.e., the same batch). Assume that the underlying function* H *is* $\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}$*-collision resistant, and the signature schemes* Σ *and* Σ_A *and the encryption schemes* $\mathcal{E}_a$ *and* $\mathcal{E}_s$ *are* $\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}$*-* *and* $\epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}$-euf-cma *secure, and* $\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}}$*-* *and* $\epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}$-ind-cpa *secure, respectively, against* Compl *adversaries. For any* Compl *adversary* $\mathcal{A}$ *against* PAuth *of* ePlAA *for a test session* π*, it holds that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PAuth}}_{\texttt{WebAuthn-simpleTW}}(\mathcal{A}, \mathsf{G}) \leq & \binom{q_{\mathrm{RResp}}}{2} 2^{-\lambda_1} + \binom{q_{\mathrm{AChall}}}{2} 2^{-\lambda_2} + \epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}} \\ & + 2q_{\mathrm{RResp}} \cdot (\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}) + 2q_{\mathrm{CResp}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + 2^{-\lambda_3}) \\ & + 2q_{\mathrm{CGen}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + \epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}).\end{aligned}$$

Proof sketch. The proof is based on the one for the mode basic (cf. Appendix C), which is based on the analysis in [BCZ23]. The main idea is that for winning via conditions 3 or 4, $\mathcal{A}$ needs to be able to register a new public key, for which it will need to break either the euf-cma security of the assertion signature scheme or of the attestation signature scheme. Alternatively, $\mathcal{A}$ may try to get a valid attestation key pair to register the public key. In the proof, we restrict $\mathcal{A}$ to not being able to corrupt authenticators from the batch B which form the group G, and for it being able to either impersonate an authenticator from that group in front of an issuer, or learn the attestation key pair the issuer sends to an honest authenticator from that group during the Update phase, $\mathcal{A}$ needs to break the security of the encryption schemes used for securing the communications between issuer and authenticators. $\mathcal{A}$ winning via conditions 1 and 2 is ruled out by initial assumptions regarding unique identifiers and collission-resistant hash functions.

Theorem 6 (Unl **of WebAuthn with** smpTW)**.** *Let* Compl $\in \{\mathsf{PPT}, \mathsf{QPT}\}$*. Let WebAuthn with attestation mode* smpTW *be an instantiation of an ePlAA* ePlAA $= (\mathsf{Initiate}, \mathsf{Update}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *as in Figure 12, 7, and 3. Let* G *be the group of authenticators in the same batch* B*. Then,*

$$\mathsf{Adv}^{\mathsf{Unl}}_{\texttt{WebAuthn-smpTW}}(\mathcal{A}, \mathsf{G}) = 0.$$

Proof sketch. Note that in the unlinkability experiment the adversary is restricted to choose two authenticators from the same group G, and after that the adversary is not allowed to query any certification oracles (at least for such authenticators). Then, unless we reach the next epoch and the attestation / issuer public key certificates expire before the target tokens are registered in the target servers, the behaviour of the experiment is exactly the same as in basic

mode. Otherwise, in case these certificates expired, the registration in the target servers would fail equally for the target authenticators, and therefore there isn't any additional advantage for $\mathcal{A}$. In case $\mathcal{A}$ runs the OUpdateI and OUpdateS oracles from Figure 13 in phase 3, registration in the target servers would fail as well for both target authenticators and $\mathcal{A}$ wouldn't get an additional advantage.

7 Conclusion

As summarised in Table 3, the four different attestation modes provide different (group) security and privacy guarantees. It turns out that none and self are the most privacy preserving attestation modes as they satisfy Unl under no restrictions to the authenticator group (as att_m is empty). However, Webauthn with attestation modes none or self is not PAuth secure. It can provide the weaker notion PAuth-w where there is no active adversary during Register.

attCA on the other hand, provides passwordless authentication security PAuth (if no authenticator certified by the same issuer as the winning authenticator has been corrupted) but relies strongly on the trust of the issuer and as such only satisfies the weaker notion Unl-w where issuers are not allowed to be corrupted.

Lastly, basic is the only current WebAuthn attestation mode that provides both PAuth and Unl. However, PAuth is preserved only if no other authenticator in the batch has been corrupted. Given the large batch size of 100,000 or more, this poses both a security risk and an inconvenience in case one authenticator is compromised and the entire batch has to be replaced. Our proposal for a new attestation mode smpTW, which also satisfies both PAuth and Unl, overcomes this limitation by providing a healing mechanism.

All the modes analysed rely on cryptographic primitives that already have PQ standardization candidates selected by NIST [NIS] and are thus "PQ ready." However, future work is required to design efficient instantiations with PQ secure primitives that take into account the increase in size and computational cost required by these new primitives.

Table 3: Summary of authentication security and unlinkability proven.

Attestation mode	PAuth-w	PAuth	Unl-w	Unl	att_m
none	✓	✗	✓	✓	{}
self	✓	✗	✓	✓	{}
basic	✓	✓	✓	✓	cert_B
attCA	✓	✓	✓	✗	vk_I
smpTW	✓	✓	✓	✓	$\mathsf{cert}_{B,P}, vk_{I,P}$

Acknowledgments

Icons in Figure 1 from flaticon with premium account. The fourth author is partly supported by ISF grant no. 1807/23 and the Len Blavatnik and the Blavatnik Family Foundation.

References

[Alla] F. Alliance. *Passkeys FAQ*. Last checked February 2023. URL: https://fidoalliance.org/passkeys/#faq (cit. on p. 7).

[Allb] F. Alliance. *What is FIDO?* Last checked April 2023. URL: https://fidoalliance.org/what-is-fido/ (cit. on p. 3).

[Bar+21] M. Barbosa, A. Boldyreva, S. Chen, and B. Warinschi. "Provable Security Analysis of FIDO2". In: *Advances in Cryptology, 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*. Ed. by T. Malkin and C. Peikert. Vol. 12827. Lecture Notes in Computer Science. Springer, 2021, pp. 125–156. DOI: 10.1007/978-3-030-84252-9_5. URL: https://doi.org/10.1007/978-3-030-84252-9_5 (cit. on pp. 4, 5, 8, 10, 12).

[BCZ23] 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. DOI: 10.1109/SP46215.2023.10179454. URL: https://doi.org/10.1109/SP46215.2023.10179454 (cit. on pp. 4, 5, 8–13, 15–18, 21, 24, 26, 31, 35, 36).

[CDL16] J. Camenisch, M. Drijvers, and A. Lehmann. "Universally Composable Direct Anonymous Attestation". In: *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*. Ed. by C. Cheng, K. Chung, G. Persiano, and B. Yang. Vol. 9615. Lecture Notes in Computer Science. Springer, 2016, pp. 234–264. DOI: 10.1007/978-3-662-49387-8_10. URL: https://doi.org/10.1007/978-3-662-49387-8_10 (cit. on p. 8).

[CGCG16] K. Cohn-Gordon, C. Cremers, and L. Garratt. "On post-compromise security". In: *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE. 2016, pp. 164–178 (cit. on p. 9).

[CJR22] C. Cremers, C. Jacomme, and E. Ronen. "TokenWeaver: Privacy Preserving and Post-Compromise Secure Attestation". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1691. URL: https://eprint.iacr.org/2022/1691 (cit. on pp. 5, 9, 19, 30).

[Gro11] T. I. W. Group. *A CMC Profile for AIK Certificate Enrollment, Version 1.0, Revision 7*. 2011. URL: https://trustedcomputinggroup.org/wp-content/uploads/IWG_CMC_Profile_Cert_Enrollment_v1_r7.pdf (cit. on p. 27).

[HLW23] L. Hanzlik, J. Loss, 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. DOI: 10.1109/SP46215.2023.10179373. URL: https://doi.org/10.1109/SP46215.2023.10179373 (cit. on pp. 4–6, 8, 10, 13, 15, 18–21, 39).

[NIS] NIST. *Selected Algorithms 2022*. Last checked April 2023. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022 (cit. on p. 32).

[Pop+18] A. Popov, M. Nystroem, D. Balfanz, and J. Hodges. *The Token Binding Protocol Version 1.0*. 2018. URL: https://www.rfc-editor.org/rfc/rfc8471#section-1 (cit. on p. 11).

[SRW22] 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 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 251–268. ISBN: 978-1-939133-31-1. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/shakevsky (cit. on p. 4).

[W3C21] W3C. *Web Authentication: An API for accessing Public Key Credentials Level 2*. 2021. URL: https://www.w3.org/TR/webauthn-2/ (cit. on pp. 6, 11, 24, 27).

[W3C23] W3C. *Web Authentication: An API for accessing Public Key Credentials Level 3*. 2023. URL: https://w3c.github.io/webauthn/ (cit. on p. 6).

A Full Analysis of WebAuthn With none

For completeness we give the definition of PAuth-w next and depict the differences in oracle access in Table 4.

Definition 3 (PAuth-w for ePIAA). *Let* $\mathsf{Compl} \in \{\mathsf{PPT}, \mathsf{QPT}\}$. *Let* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *be an Extended Passwordless Authentication Protocol with Attestation. We say that for any group* G *of authenticators sharing the same attestation material* att_m, ePIAA *provides* weak secure passwordless authentication, *or* PAuth-w *for short, if for all* Compl *adversaries* $\mathcal{A}$ *the advantage*

$$\mathsf{Adv}^{\mathsf{PAuth\text{-}w}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) := \Pr\left[\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}(\mathcal{A}, \mathsf{G}) = 1\right]$$

in winning the game $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}$ *defined in Figure 5 is negligible in the security parameter* λ. *During the experiment* $\mathsf{Expt}^{\mathsf{PAuth}}_{\mathsf{ePIAA}}$, $\mathcal{A}$ *has access to the following oracles:* cGen, cResp, cCompl, Reg, aChall, aResp, aCompl, NewT, NewS, *and* Corrupt.

We provide the security statement for WebAuthn with attestation type none that also applies to all other attestation types next. This theorem is essentially the same as [BCZ23, Theorem 1] adapted to our ePIAA class definition.

Theorem 7 (PAuth-w of WebAuthn with none). *Let WebAuthn with attestation mode* none *be an instantiation of an ePIAA* $\mathsf{ePIAA} = (\mathsf{Initiate}, \mathsf{Certification}, \mathsf{Register}, \mathsf{Authenticate})$ *as in Figure 8, 7, and 3. Let* 2^{λ_1} *and* 2^{λ_2} *be the sizes of the*

Phase	Oracle access			
	PAuth	PAuth-w	Unl	Unl-w
Initiate	ic_S		Initiate (T) Initiate (S)	
Certification	cGen cResp cCompl			
Register	rChall rResp rCompl	Reg	rChall rResp rCompl	
Authenticate	aChall aResp aCompl			
All phases	NewT NewS Corrupt		CorruptI	

Table 4: Oracle accesses given to the adversary in the Passwordless Authentication and Unlinkability experiments.

value spaces for credential id cid *and challenge, respectively. Let* G *be an empty group. Assume that the underlying function* H *is* $\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}$*-collision resistant, and the signature scheme* Σ *is* $\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}$-$\mathsf{euf\text{-}cma}$ *secure against* $\mathsf{PPT}/\mathsf{QPT}$ *adversaries. For any* $\mathsf{PPT}/\mathsf{QPT}$ *adversary* $\mathcal{A}$ *against* PAuth *of* ePIAA *for a test session* π*, it holds that*

$$\mathsf{Adv}^{\mathsf{PAuth\text{-}w}}_{\mathtt{WebAuthn\text{-}none}}(\mathcal{A},\mathsf{G}) \leq \binom{q_{\mathrm{Reg}}}{2} 2^{-\lambda_1} + \binom{q_{\mathrm{aChall}}}{2} 2^{-\lambda_2} + \epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}} + 2q_{\mathrm{Reg}}\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}.$$

Proof. We prove the theorem through a sequence of games, following [BCZ23] closely. In particular Games 0 to 3 are the same. Let Adv_i denote the advantage of the $\mathsf{PPT}/\mathsf{QPT}$ Adversary $\mathcal{A}$ in winning Game i.

Game 0. This game is identical to the original experiment, therefore

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{PAuth\text{-}w}}_{\mathtt{WebAuthn\text{-}none}}(\mathcal{A}). \tag{1}$$

Game 1. The game aborts and the adversary wins if there exist two credential identifiers cid that collide with each other.

Otherwise, given an honest authenticator and an authenticator corrupted by the adversary with the same cid, the adversary will be able to register a credential in a server slot that collides with the honest authenticator credential (winning condition 4) and potentially authenticate as the honest authenticator.

It is important to note that credential identifiers are only created during the calls to the Reg oracle, which an adversary can query q_{Reg} times. There

is a maximum of $\binom{q_{\text{REG}}}{2}$ pairs of cid which are sampled from the set $\{0,1\}^{\leq\lambda_1}$. Therefore,

$$\mathsf{Adv}_0 - \mathsf{Adv}_1 = \binom{q_{\text{REG}}}{2} 2^{-\lambda_1}. \tag{2}$$

Game 2. This game aborts and the adversary wins if there are two random challenge nonces during the authentication that collide.

Otherwise the adversary could reuse a response m_{arsp} from a honest authenticator to impersonate it when authenticating to the server (winning condition 3). Even if the server could potentially detect such reuse due to the increment on the counter n, $\mathcal{A}$ could block the authentication response of the honest authenticator in order to make this control ineffective. Alternatively, in case the server sent the same nonce to both an honest and a corrupt authenticator, the corrupt authenticator could end up registering as the honest one.

The nonce is sampled by the server in the aCHALL oracle, which can be queried by the adversary up to q_{ACHALL} times. Similarly to case 1, there are $\binom{q_{\text{ACHALL}}}{2}$ pairs of challenges sampled from the set $\{0,1\}^{\leq\lambda_2}$. Therefore,

$$\mathsf{Adv}_1 - \mathsf{Adv}_2 = \binom{q_{\text{ACHALL}}}{2} 2^{-\lambda_2}. \tag{3}$$

Game 3. Game 3 is the final game over which we perform the analysis. The game aborts and the adversary wins if there exist two hash values that collide on different inputs.

There are several situations where an adversary could take advantage of finding hash collisions. One is during registration. Namely, an adversary that finds a collision in $\mathsf{H}(\mathsf{id})$ can potentially make an authenticator register and later authenticate to server S' with $\mathsf{id}_{S'} \neq \mathsf{id}_S$ (winning conditions 2 and 3). During authentication, an adversary that finds a collision in $\mathsf{H}(m_{\mathsf{acl}}) = \mathsf{H}(\mathsf{ch}, \mathsf{tb})$ could reuse the authentication response of an honest authenticator in order to impersonate it in future requests (winning condition 3).

Let's denote $\epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}$ the probability that an adversary finds a hash collision. Hence,it holds that

$$\mathsf{Adv}_2 - \mathsf{Adv}_3 = \epsilon_{\mathsf{H}}^{\mathsf{coll\text{-}res}}. \tag{4}$$

Next we analyse the probability that $\mathcal{A}$ wins in the last game. It is important to note that $\mathcal{A}$ can only win if one of the four winning conditions hold.

Case 1. The session identifier for the authenticator session $\pi_{T_1}^{j_1}.\mathsf{sid}$ for any pair (T, j) includes the credential identifier cid which we make sure in Game 1 that will be unique per each registration context. This means that two session identifiers using different registration contexts won't be equal. Furthermore, the session identifier also depends on a counter n, which is equal to 0 during registration, and will be incremented by the authenticator by 1 before a new session identifier is set during authentication. Therefore, two authenticator sessions using the same registration context won't share the same sid. Hence, $\mathsf{Adv}_3^{case1} = 0$.

Case 2. The session identifier for the server session $\pi_S^i.\mathsf{sid}$ for any pair (S, i) includes $\mathsf{H}(\mathsf{id})$. We assume that each server has a unique identity id and in

Game 3 we assumed there is no collision of two hashes for different inputs, meaning two different servers can't have the same session id. Within the same server, the session id for registration is clearly distinguishable from those for authentication due to the presence of $H(m_{acl})$ in the latter, which depends on the challenge. Given that we ensure in Game 2 that all challenges ch are unique and in Game 3 no collision of hashes with different inputs occur, no session identifiers of different sessions of the same server can be identical. Hence, $\mathsf{Adv}_3^{case2} = 0$.

Case 3. In case 3 there is an authenticator T that registers with the server S and hence the authenticator registration context $\mathsf{rc}_T[\mathsf{id}_S]$ is set. When the server runs the ACompl oracle in order to process the authentication response, it verifies the signature over the information sent by the authenticator using the corresponding public key pk associated to a given credential id cid. Then, part of this information is used to compute the session id $\pi_S^i.\mathsf{sid}$. We note that in order for the authenticator not to have a matching session id, the information must not have been produced by any of the authenticator sessions. Given that the authenticator was not corrupted, the adversary didn't have access to the private key sk. Therefore, an adversary that wins this game can be used to break the $\mathsf{euf\text{-}cma}$ security of the signature scheme. The adversary has to target a specific session where the authenticator creates a key pair during registration, hence this attack can be potentially be done q_{REG} times. Therefore,

$$\mathsf{Adv}_3^{case3} = q_{\text{REG}} \epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}. \tag{5}$$

Another potential strategy to be followed by the adversary would be to register another public key pk' on the server under the same credential id cid. Given that credential identifiers cid are unique and that the adversary does not have access to the individual registration oracles, this strategy is impossible.

Case 4. Note that by case 1 and case 2, we make sure that if an authenticator and a server share the same session identifier, they are each other's partners. In the registration phase, the adversary is limited to a Reg query where $\pi_S^i.\mathsf{agCon} = \pi_T^j.\mathsf{agCon}$ by definition. In the authentication phase, by case 3 we have to consider that a server will partner with an authenticator if and only if T is the registration partner of S, except with probability Adv_3^{case3} (and in this case agCon in authenticator and server may not match!). Assuming they are registration partners, similarly to case 3, we see that $\pi_S^i.\mathsf{agCon}$ contains information that was presumably digitally signed by the authenticator and verified by the server. For it not to match, the adversary has been able to forge a valid signature under pk and hence it can be used to break the $\mathsf{euf\text{-}cma}$ security of the signature scheme, which can be done with probability at most $q_{\text{REG}} \epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}$. Therefore,

$$\mathsf{Adv}_3^{case4} \leq \mathsf{Adv}_3^{case3} + \mathsf{Adv}_3^{case3} \leq 2 q_{\text{REG}} \epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}. \tag{6}$$

By combining all cases it follows that

$$\mathsf{Adv}_3 \leq 2 q_{\text{REG}} \epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}}, \tag{7}$$

which proves the theorem statement.

□

As in the case of the PAuth security analysis, we prove unlinkability related to the notion of a group G which contains all the authenticators participating in the experiment.

Theorem 8 (Unl of WebAuthn with none). *Let WebAuthn with attestation mode* none *be an instantiation of an ePlAA* ePlAA = (Initiate, Certification, Register, Authenticate) *as in Figure 8, 7, and 3. Let* G *be the group of all authenticators participating in the experiment. Then for a* PPT/QPT *adversary* $\mathcal{A}$ *it holds that*

$$\mathsf{Adv}^{\mathsf{Unl}}_{\texttt{WebAuthn-none}}(\mathcal{A}, \mathsf{G}) = 0.$$

Proof. We first look at the unlinkability proof in [HLW23] for the key wrapping mode kwrPA. The main difference between kwrPA and the key derivation mode (kdfPA) is that in the latter the authenticator will provide a successful answer even if queried with a credential id cid that wasn't registered in a server, which is not the case in our scenario with resident credentials. The proof is given as a sequence of games, gradually removing the information about authenticators T_b and T_{b-1} from the oracles R/ALeft and R/ARight, such that at the end the information the adversary receives is independent of the decision bit b.

In *Game 1* the challenger tries to guess the authenticators T_0 and T_1, and the adversary will select and replace them by T_0^*, T_1^*.

In *Game 2* the two authenticators guessed by the challenger, which will be used in the R/ALeft and R/ARight oracles, behave in the following way. Credential ids cid are generated at random, $\mathsf{cid} \leftarrow_{\$} \{0,1\}^{\geq \lambda}$; when RLeft or RRight oracles are called, the credential keypair (pk, sk) is created and an entry is stored in a map L_0 or L_1—depending on the authenticator we are using—indexed by cid. Later on, during authentication, the authenticator will look for the corresponding entry given the cid, to retrieve the credentials to answer the challenge.

In *Game 3*, the map used by the authenticator T_b is split in 2: L_b to be used exclusively with the rResp and aResp oracles, and L'_b to be used only with oracles R/ALeft.

Game 4 does the same for the authenticator T_{b-1}. At this point, the information the adversary receives is independent of the bit b.

We next turn to the specifics of the attestation mode none. First, we define that for the attestation mode none, the group G from which the adversary can choose authenticators is the entire authenticator set $\mathcal{T}$ of authenticators participating in the experiment. Then, we use the games explained above with our instantiation, starting with *Game 0* where $\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{Unl}}_{\texttt{WebAuthn-none}}(\mathcal{A}, \mathsf{G})$.

Note that our instantiation matches the scenario depicted in *Game 2*, but instead of indexing the maps L_0, L_1 by cid, they are indexed by id_S. Hence, $\mathsf{Adv}_0 - \mathsf{Adv}_2 = 0$.

Then, given our condition of $S_{\mathsf{unl}^*} = \emptyset$ to rule out trivial attacks, it is evident that the map entries used in the rResp and aResp oracles for T_b do not match

those used in R/ALeft, and the same applies for T_{b-1} and R/ARight, meaning that $\mathsf{Adv}_2 - \mathsf{Adv}_3 = 0$ and $\mathsf{Adv}_3 - \mathsf{Adv}_4 = 0$. Finally,

$$\mathsf{Adv}_4 = \mathsf{Adv}^{\mathsf{Unl}}_{\texttt{WebAuthn-none}}(\mathcal{A}, \mathsf{G}) = 0.$$

With resident credentials and attestation mode none, the amount of information from the authenticator that a potential adversary receives are credential IDs and public keys which are different per server/relying party, and provided they are fully independent, they can't be used to link authenticator registrations. □

B Analysis of WebAuthn with self

The attestation mode self is very similar to none in terms of security and privacy guarantees; the attestation material att_m is empty. As for none, there is no attestation key pair, instead, the credential private key is used to generate the attestation signature. As such, self attestation proves the knowledge of the credential secret key. However, it does not give any additional benefits against a PAuth adversary, and Theorem 7 applies. Likewise, there is no additional advantage for the Unl adversary as the credential keys are independent of each other and Theorem 8 applies.

C Full Analysis of WebAuthn with basic

We give the proof for Theorem 1 next.

Proof. We give the proof through a sequence of games. Games 0-3 are the same as those for mode none (cf. Appendix A). We summarise that

$$\mathsf{Adv}^{\mathsf{auth}}_{\texttt{WebAuthn-basic}}(\mathcal{A}, \mathsf{G}) - \mathsf{Adv}_3 = \binom{q_{\text{RRESP}}}{2} 2^{-\lambda_1} + \binom{q_{\text{ACHALL}}}{2} 2^{-\lambda_2} + \epsilon^{\mathsf{coll\text{-}res}}_{\mathsf{H}}.$$

It is important to note that we use q_{RRESP} instead of q_{REG} here, due to the fact that the adversary is given access to oracles for the individual registration algorithms in this mode.

Then we look at the probability of $\mathcal{A}$ winning the last game, for which $\mathcal{A}$ has exactly the same options in case 1 and case 2 as in mode none. Therefore, $\mathsf{Adv}^{case1}_3 = \mathsf{Adv}^{case2}_3 = 0$. For cases 3 and 4, the adversary can follow strategies in addition to the ones described in mode none outlined next.

Case 3. During registration, $\mathcal{A}$ can register another public key pk' for which the corresponding sk' is also known by the adversary. This requires forging a valid input message m_{rrsp} that will be accepted by the oracle rCompl. Given that we restrict the adversary to corrupt authenticators from *other* batches through the definition of G (which won't share the same attestation key pair), the only way to succeed is to break the euf-cma security of the attestation signature scheme

Σ_A used during registration which can be done with probability at most $\epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma_A}$. Therefore,

$$\mathsf{Adv}_3^{case3} = q_{\mathrm{RRESP}} \cdot (\epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma} + \epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma_A}).$$

Case 4. In the registration phase, for agCon values not to match between authenticator and server, the adversary can follow the new strategy of forging a valid message m_{rrsp} which, given the collision-resistance property of the hash function stated in Game 3, and the restriction of not corrupting authenticators from the same batch, requires the adversary to break the $\mathsf{euf\text{-}cma}$ security of Σ_A. Therefore,

$$\mathsf{Adv}_3^{case4} \leq \mathsf{Adv}_3^{case3} + \mathsf{Adv}_3^{case3} \leq 2q_{\mathrm{RRESP}} \cdot (\epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma} + \epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma_A}).$$

By combining all cases where adversary wins we have that:

$$\mathsf{Adv}_3 \leq 2q_{\mathrm{RRESP}} \cdot (\epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma} + \epsilon^{\mathsf{euf\text{-}cma}}_{\Sigma_A}).$$

which proves the theorem statement. □

D Full Analysis of WebAuthn With smpTW

As in basic, we define the attestation material att_m as the certificate of the respective batch attestation public key $\mathsf{cert}_{B,P}$, and the issuer's public key $vk_{I,P}$. In the authentication security experiment the adversary won't be able to corrupt authenticators from the same batch as the one for which it may get a successful authentication to win. Moreover, for simplicity we assume batches are constant across periods and that there isn't any authenticator in group G that was compromised in a previous period. Although we don't include this in the experiment, this can be considered realistic given the PCS property of the scheme. In the unlinkability experiment, the adversary will try to distinguish between two authenticators from the same batch.

Next we provide the proof of Theorem 5.

Proof. We base the proof for this mode on the proof for basic (cf. Appendix C). The main difference between the two modes is that in basic the attestation keys are set in the authenticator during initialization, while in smpTW these are sent from the issuer to the authenticator over an insecure channel controlled by $\mathcal{A}$ when running the certification algorithms. We also take into account the two new oracles from Figure 13. Following the authentication security proof for basic, at the end of Game 3 we have that:

$$\mathsf{Adv}^{\mathsf{PAuth}}_{\mathtt{WebAuthn\text{-}simpleTW}}(\mathcal{A}, \mathsf{G}) - \mathsf{Adv}_3 = \binom{q_{\mathrm{RRESP}}}{2} 2^{-\lambda_1} + \binom{q_{\mathrm{ACHALL}}}{2} 2^{-\lambda_2} + \epsilon^{\mathsf{coll\text{-}res}}_{\mathsf{H}}.$$

Then we review the strategies for winning in cases 1-4. In smpTW, the adversary has two new additional strategies to win in *Case 3* by updating the

credential public key pk and forging a valid input message m_{rrsp}. In both strategies, $\mathcal{A}$ gets a valid attestation keypair assigned to the batch B of authenticators which form the group G: *a)* $\mathcal{A}$ recovers $(ak_{B,P}, \mathsf{att}_m)$ from m_{crsp} during a honest interaction between an authenticator $T \in \mathsf{G}$ and an issuer I, or *b)* $\mathcal{A}$ impersonates an authenticator $T \in \mathsf{G}$ and runs cGen to get an attestation key pair, which requires $\mathcal{A}$ to guess a valid token t_T corresponding to id_T.

An adversary that wins according to strategy *a)* plays against the ind-cpa property of the encryption scheme $\mathcal{E}_a$ (actually ind-cpa is a stronger notion since it does not allow recovering a single bit, while $\mathcal{A}$ needs to recover the whole message), which can be broken with probability at most $\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}}$, or against ind-cpa of the encryption scheme $\mathcal{E}_s$, which can be broken with probability at most $\epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}$. Strategy *b)* can be successful with probability 2^{-n}, where n is the length of tokens t_T. Therefore:

$$\mathsf{Adv}_3^{case3} = q_{\text{RResp}}(\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}) + q_{\text{cResp}}(\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + 2^{-\lambda_3}) + q_{\text{cGen}}(\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + \epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}).$$

Similarly, we add the same 2 strategies for winning in *case 4*, resulting in

$$\begin{aligned}\mathsf{Adv}_3^{case4} =& \mathsf{Adv}_3^{case3} + \mathsf{Adv}_3^{case3} = 2q_{\text{RResp}} \cdot (\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}) \\ &+ 2q_{\text{cResp}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + 2^{-\lambda_3}) + 2q_{\text{cGen}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + \epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}).\end{aligned}$$

By combining all cases where adversary wins we have that:

$$\begin{aligned}\mathsf{Adv}_3 \leq& 2q_{\text{RResp}} \cdot (\epsilon_{\Sigma}^{\mathsf{euf\text{-}cma}} + \epsilon_{\Sigma_A}^{\mathsf{euf\text{-}cma}}) + 2q_{\text{cResp}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + 2^{-\lambda_3}) \\ &+ 2q_{\text{cGen}} \cdot (\epsilon_{\mathcal{E}_a}^{\mathsf{ind\text{-}cpa}} + \epsilon_{\mathcal{E}_s}^{\mathsf{ind\text{-}cpa}}).\end{aligned}$$

□