Security Analysis of Anonymous Authentication in Apple's Private Cloud Compute

Callista Gratz, Anja Lehmann and Andrey Sidorenko

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

Abstract. Apple's Private Cloud Compute (PCC) offloads on-device computation, such as large language-model prompts, to the cloud while seeking to preserve the privacy users expect from local processing. At its core lies a custom authentication protocol that must reconcile three competing goals: restricting access to registered users within a rate limit, doing so without revealing user identities or linking their requests, and nonetheless retaining the ability to hold misbehaving users accountable. Apple's protocol combines blind RSA signatures in a hierarchical issuance with a salted-hash binding trick, but ships without any formal statement – let alone proof – of the guarantees it is meant to provide. In this work we close this gap by abstracting the desired authentication protocol as PORTA – *pseudonymously openable, rate-limited tokens for anonymous authentication*. We formally define the desired privacy, security, and abuse-mitigation properties such a protocol should satisfy, including a new *binding* notion specific to the protocol's two-step issuance. We then cast Apple's protocol in this model and show that it achieves the rate-limiting goal and, with a small modification, all privacy goals. Its abuse-mitigation mechanism, which is not yet active in the deployed system, would however not provide the desired accountability guarantees even once enabled. Finally, we present an alternative construction based on anonymous credentials that provably achieves all properties and verifies the token binding already at the gateway, so that a pseudonym is only revealed in case of abuse.

Keywords: authentication · anonymous credentials · blind signatures

1 Introduction

Apple's Private Cloud Compute (PCC) extends on-device artificial intelligence to the cloud when a request exceeds what the user's device can handle locally, for example a large language-model prompt that must be processed on more capable hardware [Inc24b]. To preserve the privacy guarantees users expect from on-device processing, PCC is designed so that compute servers handle requests without learning who sent them. At the same time, PCC should be limited to legitimate users that are properly enrolled. Apple uses a custom authentication protocol to provide the required privacy and security balance.

Concretely, a user who wishes to access a PCC service sends their request to a dedicated compute server, the *Node* ($\mathcal{N}$), which is trusted to process the request itself. To justify that trust, Apple constrains $\mathcal{N}$ heavily: it runs only a small, auditable software stack, provides user-verifiable hardware attestations of its configuration, and – crucially – holds no persistent state, clearing all user data after each request. These constraints make $\mathcal{N}$ a poor place to authenticate: doing so would tie request handling to user identities, and a stateless $\mathcal{N}$ cannot track how often a user has authenticated and thus cannot enforce rate

E-mail: callista.gratz@student.hpi.de (Callista Gratz), anja.lehmann@hpi.de (Anja Lehmann), andrey.sidorenko@hpi.de (Andrey Sidorenko)

limits. Authentication is therefore outsourced to a separate *Gateway* ($\mathcal{G}$), which admits only entitled requests and forwards them to $\mathcal{N}$ without learning who sent them. Two further services govern who may use the system at all: an enrollment service $\mathcal{S}_{\mathsf{E}}$ that registers eligible users under a pseudonym *nym*, and a token service $\mathcal{S}_{\mathsf{OTT}}$ that hands out the one-time credentials a user spends to authenticate.

Anonymous rate-limited authentication with accountability. Apple's PCC authentication pursues three goals. First, *rate-limited authentication*: access should be restricted to registered, entitled users, and each user should be able to authenticate only up to a permitted rate. The gateway $\mathcal{G}$ enforces this, admitting a request only if it comes from an entitled user who has not exceeded their limit. Second, this must be achieved *without harming user privacy*: $\mathcal{G}$ should be unable to link a request to a user identity, or to link two requests of the same user, so honest users authenticate anonymously to $\mathcal{G}$. Third, the system must provide *accountability* for abuse: if a request turns out to be malicious, the node $\mathcal{N}$ must be able to recover the responsible user's long-lived pseudonym *nym*, enough to ban them from obtaining further access tokens at $\mathcal{S}_{\mathsf{OTT}}$ but not to identify them. Honest users thus stay anonymous, and even misbehaviour is handled purely pseudonymously.

Apple's authentication protocol. To realize these goals, PCC builds on the Privacy Pass framework [DIW24] instantiated with RSA blind signatures, arranged in a two-step *hierarchical issuance*. During enrollment, the user obtains a blind signature σ_{E} on a self-chosen long-term pseudonym *nym* from $\mathcal{S}_{\mathsf{E}}$; the pair $(nym, \sigma_{\mathsf{E}})$ serves as the user's long-term credential. Using this credential, the user then obtains from $\mathcal{S}_{\mathsf{OTT}}$ a batch of one-time tokens: blind signatures on random one-time messages m_{OTT} that can each be redeemed once at $\mathcal{G}$. Beyond this hierarchy, the protocol uses an unusual construction to tie the two levels together: each one-time message is derived as $m_{\mathsf{OTT}} = \mathsf{H}(nym, \sigma_{\mathsf{E}}, salt)$, a salted hash of the long-term credential. This binding is meant to enable abuse mitigation: $\mathcal{N}$ can recover the pseudonym from a redeemed token, and check that m_{OTT} was correctly derived from the user's credential. The mechanism is not yet active in the deployed system – Apple notes that "the inclusion of the $[(nym, \sigma_{\mathsf{E}})]$ in the body of the encrypted request makes future abuse-mitigation options available, but these options are not currently implemented" [Inc24b] – but its cryptographic foundation is already in place. Once enabled, however, the check requires $\mathcal{N}$ to recover the user's pseudonym on *every* request, whereas such re-identification should be reserved for actual abuse cases, and not routine validation.

Unclear security and privacy guarantees. Despite the care evident in its design, PCC's authentication system comes with no formal definition of the security and privacy guarantees it is meant to provide, and consequently no way to verify whether the protocol actually achieves them. This is particularly relevant for the abuse-mitigation mechanism: before it is activated, it is worth asking whether its cryptographic foundation can actually support the guarantees it is meant to enable – and whether binding can be enforced without routinely re-identifying users.

Beyond PCC. The need to reconcile anonymous access with abuse mitigation is not specific to PCC. Privacy Pass [DIW24] already separates an attester, which establishes a client's eligibility, from an issuer, which hands out anonymous tokens – mirroring the enrollment and token services in Apple's design. The recently proposed PACT and MoLE protocols [Jac26, SJM26] target privacy-preserving access control and rate limiting on the web, and Google's Private AI Compute [GGGG26] and Meta's Private Processing [Met26] combine confidential computing with anonymous tokens or credentials for private cloud inference, similar to PCC. Rate limiting bounds usage quotas, but attributing an individual abusive request to the responsible user – without giving up unlinkability for routine requests – requires additional accountability guarantees. A formal model that separates anonymous validation from pseudonymous opening is thus of interest well beyond Apple's protocol.

Table 1: Corruption settings and achieved properties of the PORTA instantiations. A party may be ● honest, ◐ semi-honest, ◕ corrupt after enrollment, or ○ corrupt. The parenthesized unlinkability requires deterministic RSA-PSS with well-formed signing keys.

Property	$\mathcal{S}_{\mathsf{E}}$	$\mathcal{S}_{\mathsf{OTT}}$	$\mathcal{G}$	$\mathcal{N}$	User	Apple	Ours	Property	$\mathcal{S}_{\mathsf{E}}$	$\mathcal{S}_{\mathsf{OTT}}$	$\mathcal{G}$	$\mathcal{N}$	User	Apple	Ours
E-BLIND	○	○	○	○	●	✓	✓	BIND	●	●	●	○	◕	✗	✓
UNLINK	○	○	○	●	●	(✓)	✓	TRACE	○	●	●	◐	○	✗	✓
RL-UNF	●	●	●	○	○	✓	✓	NFRAME	●	○	○	◐	◕	✗	✓

1.1 Our Contributions

In this work, we formally model and analyze the authentication protocol of PCC. We abstract it into a general primitive, PORTA (*Pseudonymously Openable, Rate-limited Tokens for Anonymous authentication*), and formally define the privacy, rate-limiting, and accountability properties it should provide. We then analyze Apple's protocol in this model: it achieves the rate-limiting goal and, with a small modification, the privacy goals, but none of the accountability goals, even once its abuse-mitigation mechanism is enabled. Finally, we propose an alternative construction that follows Apple's architecture and provably achieves all properties. More concretely, our work makes the following contributions.

A formal model (PORTA). PORTA abstracts the authentication system used in Apple's PCC into four protocol phases (enrollment, token issuance, redemption, and opening) and formalizes six privacy and security notions, grouped into three goals:

Privacy: *enrollment blindness* and *presentation unlinkability* guarantee that honest users stay unlinkable to the issuing services and the gateway.

Rate-limiting: *rate-limited unforgeability* guarantees that no user can authenticate more often than permitted.

Accountability: *traceability* and *non-frameability* guarantee that every abusive request opens to an enrolled pseudonym and that no honest user can be falsely implicated, while a new *binding* property ties each one-time token to the long-term credential it was issued under. This binding is the guarantee that Apple's salted-hash construction reaches for.

Analysis of Apple's protocol (Apple-PORTA). We give a formal description of Apple's protocol as a PORTA scheme, Apple-PORTA, and analyze it in our model (summary in Tab. 1). Apple-PORTA achieves enrollment blindness and rate-limited unforgeability, and presentation unlinkability after a small change to the signature scheme used for the one-time tokens: Apple uses salted, probabilistic RSA-PSS blind signatures, which have to be replaced by the deterministic variant. However, it satisfies *none* of the three accountability properties, even with all checks at $\mathcal{N}$ enabled. We identify three causes. First, $\mathcal{G}$ only verifies the one-time token, but not that it belongs to the user's credential. This check is left to $\mathcal{N}$, where it happens after the request has been accepted (breaks binding), and also requires recovering the pseudonym for every request. Second, one-time tokens are issued blindly, so $\mathcal{S}_{\mathsf{OTT}}$ cannot ensure that they are bound to the credential shown at issuance. A user with two credentials can thus make its requests open to a pseudonym that $\mathcal{S}_{\mathsf{OTT}}$ never saw (breaks traceability). Third, the long-term credential is revealed in plain to $\mathcal{S}_{\mathsf{OTT}}$ and $\mathcal{N}$, enabling them to frame honest users (breaks non-frameability).

Provably secure construction (AC-PORTA). Finally, we present AC-PORTA, a construction based on anonymous credentials that achieves all six PORTA properties. We do not claim particular novelty for this construction; rather, it shows that all privacy and security properties are achievable from well-established building blocks. Architecturally, AC-PORTA mirrors Apple's design, but replaces the salted-hash binding and the plaintext credential showings by a zero-knowledge proof, verified by $\mathcal{G}$, that the token and credential are valid and bound to each other. This moves the binding check from $\mathcal{N}$ to $\mathcal{G}$ and keeps

the credential hidden from all services. Compared to Apple's protocol, which only requires RSA blind signatures, hashing, and encryption, this adds computational cost: presenting a token and verifying it at $\mathcal{G}$ takes about 11 ms in total instead of less than 0.1 ms. With every per-request operation below 7 ms, we nevertheless consider AC-PORTA practical, in particular given the additional guarantees it provides.

1.2 Related Work

Combining anonymous authentication with rate limiting and accountability is not new as such, and several lines of work address parts of it. None of them, however, captures the combination required by PCC and similar systems, namely a two-step issuance by separate enrollment and token services, rate limiting through one-time tokens that are bound to a long-term credential, and pseudonymous opening by a party other than the verifier. We discuss the closest lines of work below, as well as recent empirical analyses of PCC itself.

Privacy Pass and rate-limited anonymous tokens. Privacy Pass [DIW24], the basis of Apple's PCC implementation, is a token-based authentication system whose security has been studied in several works [DGS+18, CDH23a, HLÖ25]. Its Rate-Limited variant [HIP+24, CDH23a] obtains rate-limiting through batch issuance: a user is granted a bounded batch of unlinkable one-time tokens. The Privacy Pass architecture also foresees an additional *attester* role that verifies a client's identity before an issuer hands out tokens, mirroring the enrollment service $\mathcal{S}_{\mathsf{E}}$ in Apple's design. To the best of our knowledge, however, no existing protocol or analysis covers the full attester/issuer split with the user interacting independently with both services. An alternative to batch issuance is to enforce rate limits per presentation using a multi-show anonymous token with a built-in rate-limit predicate [YWFH26, SM26]; we discuss this option in Sec. 5.4 but adopt batch issuance instead, both for simplicity and because rate-limited anonymous credentials are difficult to realize with post-quantum privacy, which batch issuance attains trivially.

Non-transferable tokens. The binding property of PORTA – tying a one-time token to the credential it was issued under – is related to non-transferable anonymous tokens (NTAT) [DMTV24], where redeeming a token requires the secret key of the user it was issued to, so that sharing a token forces sharing the key. NTAT were later extended with betrayability [KL25]. These notions require interactive redemption; it is not clear how to reconcile this with abuse-mitigation at the stateless $\mathcal{N}$. Our work has an explicit model for the hierarchical two-token setting, and abuse-mitigation goals of PCC.

Anonymous credentials and group signatures. Anonymous credentials [CL01, CL04] are a natural fit for privacy-preserving authentication and form the basis of our own construction. Closest to the opening in PORTA are credentials with anonymity revocation [CL01], also known as identity escrow [KP98], and group signatures [BMW03, BSZ05]: a designated party can recover the identity behind an otherwise anonymous credential showing or signature. These notions, however, do not consider the hierarchical issuance and rate limiting of PORTA. We take inspiration from the traceability and non-frameability notions of group signatures, but adapt them to the PORTA setting where the pseudonym (which can be framed) is revealed to some parties in the clear, in contrast to group signatures, where non-frameability rests on a user-held secret. We discuss this difference in Sec. 3.3 and in full detail in Sec. D.2.

A different approach to abuse mitigation is taken by *anonymous blacklisting* schemes, which allow a service to block misbehaving users from future authentications without revealing who they are. In BLAC [TAKS07] and PEREA [TAKS08], users prove in zero knowledge that none of their past authentications has been blacklisted, without relying on a trusted third party. Nymble-like systems [JKTS07, LH12] instead rely on trusted managers that can link a user's subsequent connections once one of them has been reported.

In PORTA, abuse is instead attributed to the pseudonym behind an individual request, which can then be denied further one-time tokens, while authentication at $\mathcal{G}$ remains a simple non-interactive check.

Empirical analyses of PCC. Two concurrent works study the deployed PCC system through reverse engineering, i.e., by analyzing the client software and the messages it exchanges, without a formal model. Zhou et al. [ZZWL26] show that malware on a user's device can extract the long-term credential and use it elsewhere, e.g., to consume the victim's request quota. Dittmar et al. [DSV+26] compare the deployed client with Apple's documentation and report several deviations: more long-term credentials per device than documented, one-time tokens that are accepted several times, and requests with mismatched token pairs that are accepted because the binding check is currently disabled. They also ask whether the salted-hash binding harms unlinkability, which they cannot settle without a formal model. Both works confirm the message flow on which our model of Apple's protocol is based; we relate their findings to our results in Sec. 4.3.

2 The PORTA System Model

In this section, we introduce PORTA as an abstraction of the authentication protocol underlying Apple's PCC system [Inc24b, Inc24a]. We first describe the parties, the two types of cryptographic tokens, and the four phases of the protocol (Sec. 2.1). We then revisit Apple's design choices, from which we derive our design principles and security goals (Sec. 2.2), and present the formal syntax of a PORTA scheme (Sec. 2.3). The formal security model is in Sec. 3.

2.1 Overview of PORTA

A PORTA scheme allows a user to repeatedly and anonymously authenticate to a compute server (the Node), while two distinct authentication services (Enrollment Service and One-Time Token Service) control who may access the system and how often. The Gateway enforces these limits, as it approves requests directed to the Node. On a high level, PORTA provides *rate-limited anonymous authentication*: by default, honest users remain fully anonymous and their requests are unlinkable. Should a request later turn out to be abusive, the Node can recover a long-term pseudonym *nym* for the offending user, which enables out-of-band abuse mitigation, e.g. denying future authentication tokens to *nym*.

Parties and tokens. PORTA involves a user, who wants to access a service offered by the Node, and holds long-term and one-time credentials issued by the services described below. We use Apple's naming wherever it is descriptive and rename the two issuing services for clarity (see [ZZWL26] for an overview in Apple's terminology).

Enrollment Service ($\mathcal{S}_{\mathsf{E}}$) Enrolls eligible users into PORTA. $\mathcal{S}_{\mathsf{E}}$ blindly issues each user a long-term, authenticated pseudonym credential $(nym, \sigma_{\mathsf{E}})$, where *nym* is the user's pseudonym and σ_{E} a corresponding token.

One-Time Token Service ($\mathcal{S}_{\mathsf{OTT}}$) Blindly issues batches of *one-time tokens* $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ to *pseudonymous* users that have been previously enrolled by $\mathcal{S}_{\mathsf{E}}$. $\mathcal{S}_{\mathsf{OTT}}$ enforces *who* – based on the pseudonym *nym* – receives *how many* one-time tokens, thus governing the per-user rate of authentications.

Gateway ($\mathcal{G}$) Verifies authentication requests from the user on behalf of the Node. The authentication towards the Gateway is anonymous in that $\mathcal{G}$ can decide whether a request is valid *and* unspent, without learning anything about the user behind it.

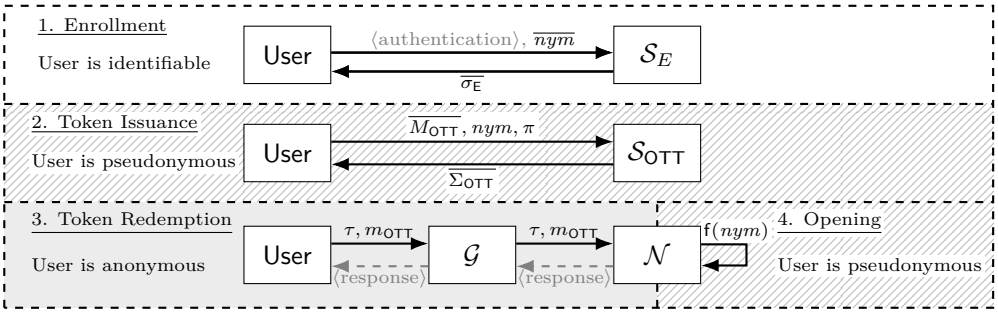

Figure 1: The syntax of our model, relevant parties, and schematic message flow.

Node ($\mathcal{N}$) The actual service the user wants to access. The Node $\mathcal{N}$ is stateless, deleting all user data after processing [Inc24c], so authentication is outsourced to the Gateway. For honest requests, the Node only sees an anonymous, validated request forwarded by the Gateway. If a request turns out to be abusive, the Node can *open* it and recover $\mathsf{f}(nym)$ for out-of-band abuse mitigation, where f is a fixed public function of the user's pseudonym nym (the identity in Apple's protocol, see Sec. 2.3).

We follow Apple in modeling the four backend parties as separate entities, even though in Apple's deployment they are all operated by Apple itself.[1] The separation allows us to reason about different corruption scenarios and to identify the precise trust assumptions each security property requires. PORTA revolves around two cryptographic objects:

Enrollment credential $(nym, \sigma_{\mathsf{E}})$ attesting the pseudonym nym's enrollment by $\mathcal{S}_{\mathsf{E}}$

One-time tokens $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$, each of which may be spent at the Gateway at most once. $\mathcal{S}_{\mathsf{OTT}}$ issues batches of ℓ such tokens per request.

The four phases of PORTA. We split the protocol into four phases, illustrated in Fig. 1.

Phase 1: Enrollment. The user enrolls into PORTA by running an interactive protocol with $\mathcal{S}_{\mathsf{E}}$. The user picks a fresh pseudonym nym and obtains an attestation σ_{E} on it. Crucially, issuance is *blind*: $\mathcal{S}_{\mathsf{E}}$ does not learn nym, and cannot later link an enrollment session to any subsequent appearance of nym or to a redemption involving the resulting $(nym, \sigma_{\mathsf{E}})$.[2] Enrollment is run only once per user [Inc24b].

Phase 2: Token Issuance. Using $(nym, \sigma_{\mathsf{E}})$, the user pseudonymously authenticates to $\mathcal{S}_{\mathsf{OTT}}$ under nym and requests a batch of ℓ one-time tokens $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$. If the user is entitled to additional tokens, $\mathcal{S}_{\mathsf{OTT}}$ returns the batch. As with enrollment, the one-time-token contents are issued *blindly*, so that $\mathcal{S}_{\mathsf{OTT}}$ cannot later link a m_{OTT} presented at $\mathcal{G}$ back to its issuance session. Rate-limiting is *decided* in this phase. Since $\mathcal{S}_{\mathsf{OTT}}$ sees nym, it can refuse further tokens by outputting $\perp$, e.g., when a quota is exhausted or nym is banned (the decision logic is application-specific and not modeled). Tokens issued before a ban remain valid, so a banned user retains at most ℓ requests per outstanding batch.

Phase 3: Redemption. To send a request to $\mathcal{N}$, the user picks an unspent one-time token $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ and produces a presentation (τ, m_{OTT}) that "proves" possession of a valid $(nym, \sigma_{\mathsf{E}})$ together with this token. The user sends the presentation to $\mathcal{G}$, which verifies it and checks that m_{OTT} has not been used in a previously accepted request. These checks *enforce* the rate limit set by $\mathcal{S}_{\mathsf{OTT}}$, as each accepted request consumes a fresh one-time token.[3] If both checks succeed, $\mathcal{G}$ forwards the request to $\mathcal{N}$.

[1] Apple's deployment also features a Third Party Gateway between the user and the Gateway. As it is transparent to the cryptographic protocol, we omit it.

[2] Before this step, $\mathcal{S}_{\mathsf{E}}$ verifies the user's real-world eligibility (e.g. via the user's Apple ID); we treat this out-of-band authentication as out of scope.

[3] As in prior work on anonymous one-time tokens [KLOR20, SS22, CDV23, CDH23b, BLOR22], we treat double-spend prevention as a separate, stateful concern and only model the public verification of τ.

Phase 4: Opening. In case a specific request is found to contain abusive content, $\mathcal{N}$ can link it to the offending user. $\mathcal{N}$ uses its secret key to *open* the presentation, which validates the request and recovers $\mathsf{f}(nym)$. The decision whether to act on $\mathsf{f}(nym)$ – e.g. reporting it to $\mathcal{S}_{\mathsf{OTT}}$ to block token issuance in the future – is an application-level abuse-mitigation policy and outside the cryptographic model.

2.2 Design Choices and Security Goals

We now briefly look at Apple's concrete protocol and implementation choices to derive the core design principles of PORTA. We adopt the first three choices in our syntax and security model, generalizing them where this admits more privacy-friendly instantiations. The placement of the binding check is the only point where we deliberately deviate from Apple's design.

Blind issuance of both credentials. Apple issues both the enrollment credential and the one-time tokens as RSA blind signatures, so that neither service learns what it signs. We model both issuances as three-move blind-issuance protocols (request, issue, finalize).

Pseudonym separate from the credential. In Apple's protocol, the user picks a random pseudonym *nym* during enrollment, but effectively the whole credential $(nym, \sigma_{\mathsf{E}})$ serves as the long-term pseudonym. It is sent to $\mathcal{S}_{\mathsf{OTT}}$ with every batch request and encrypted to $\mathcal{N}$ in every presentation. We keep the user-chosen *nym*, but treat only *nym* as the pseudonym. This matches its conceptual role and admits more privacy-friendly instantiations, e.g., where $\mathcal{S}_{\mathsf{OTT}}$ only receives *nym* and a proof of enrollment.

Binding one-time tokens to the credential. Apple derives each one-time message as a salted hash $m_{\mathsf{OTT}} = \mathsf{H}((nym, \sigma_{\mathsf{E}}), salt)$ of the enrollment credential, and $\mathcal{N}$ checks this relation after opening (function `validateOttIsDerivedFromTGT` [App24a]). The intended guarantee is that a one-time token can only be spent together with the credential it was issued for, which we capture as a separate *binding* property. Both reverse-engineering studies [ZZWL26, DSV+26] report that the deployed system already forms one-time messages as salted hashes, but does not perform the check yet and accepts mismatched pairs. Our presentation Apple-PORTA of Apple's protocol assumes that the check is enabled, so our results do not rely on its current absence.

Binding check at the Gateway instead of the Node. Here we deviate from Apple's design. We define binding as a core goal, but require it to be checked by $\mathcal{G}$ instead of $\mathcal{N}$, where Apple's validator places it. A check at $\mathcal{N}$ has two drawbacks. First, $\mathcal{N}$ must open *every* request, including benign ones, and thus learns the user's long-term pseudonym each time, which links all requests of a user. We believe opening should instead be reserved for cases of abuse. Second, $\mathcal{G}$ should prevent invalid authentication requests from reaching $\mathcal{N}$. This is not possible if the (binding) check of valid requests is placed at $\mathcal{N}$ only. When we analyze Apple's protocol in Sec. 4, we keep the check at $\mathcal{N}$ as in Apple's design.

Resulting security goals. We capture Apple's high-level goals of privacy, rate limiting, and accountability in six properties, which we formalize in Sec. 3:

Privacy. *Enrollment blindness* (E-BLIND) ensures that $\mathcal{S}_{\mathsf{E}}$ issues credentials blindly, i.e., it neither learns the pseudonym it certifies nor can link the enrollment session to any later use of it, even when colluding with all other parties. *Presentation unlinkability* (UNLINK) ensures that the authentications of a user can be linked neither to each other nor to the user's pseudonym or token issuance, even if $\mathcal{G}$ colludes with $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$.

Rate-limiting. *Rate-limited unforgeability* (RL-UNF) ensures that users get at most as many presentations accepted as $\mathcal{S}_{\mathsf{OTT}}$ has issued one-time tokens, and tokens for at most as many pseudonyms as $\mathcal{S}_{\mathsf{E}}$ has enrolled.

Accountability. *Traceability* (TRACE) ensures that opening cannot be evaded, as every accepted presentation opens to a pseudonym known to $\mathcal{S}_{\mathsf{OTT}}$. *Non-frameability* (NFRAME) ensures that opening cannot be used against honest users, as no presentation opens to an honest user's pseudonym unless that user created it. *Binding* (BIND) ensures that one-time tokens can only be spent under the credential they were issued for.

2.3 Formal Syntax

We now formalize the four phases of PORTA together with key generation. Following Apple's design, both issuance phases are three-move blind-issuance protocols (request, issue, finalize) between the user and the respective service, as known from blind signatures. Token issuance is only partially blind, since $\mathcal{S}_{\mathsf{OTT}}$ must see the pseudonym to enforce rate limits, and thus additionally takes nym and a proof π of enrollment as input. Redemption and opening are non-interactive. The user produces a presentation that $\mathcal{G}$ verifies publicly, and only $\mathcal{N}$ can open it using its secret key. $\mathbf{E}, \mathbf{OTT}, \mathbf{G}, \mathbf{N}, \mathbf{U}$ indicate which party executes each algorithm.

Definition 1. A PORTA scheme Π is a tuple of algorithms $((\mathbf{E}.\mathsf{KGen}, \mathbf{OTT}.\mathsf{KGen}, \mathbf{N}.\mathsf{KGen}), (\mathbf{U}.\mathsf{Req}, \mathbf{E}.\mathsf{Issue}, \mathbf{U}.\mathsf{Fin}), (\mathbf{U}.\mathsf{ReqBatch}, \mathbf{OTT}.\mathsf{IssueBatch}, \mathbf{U}.\mathsf{FinBatch}), (\mathbf{U}.\mathsf{Present}, \mathbf{G}.\mathsf{Vf}, \mathbf{N}.\mathsf{Open}))$ as described below, together with a public, injective, and efficiently computable function $\mathsf{f}\colon \mathsf{NYM} \to \mathsf{NYM}'$ on the pseudonym space NYM, where $\mathsf{f}(\bot) = \bot$.

Key generation: $\mathbf{E}.\mathsf{KGen}, \mathbf{OTT}.\mathsf{KGen}, \mathbf{N}.\mathsf{KGen}$**.** On input 1^λ, $\mathcal{S}_{\mathsf{E}}$, $\mathcal{S}_{\mathsf{OTT}}$, and $\mathcal{N}$ each generate a key pair: $(sk_{\mathsf{E}}, pk_{\mathsf{E}})$ for enrollment-credential issuance, $(sk_{\mathsf{OTT}}, pk_{\mathsf{OTT}})$ for one-time-token issuance, and $(sk_{\mathsf{N}}, pk_{\mathsf{N}})$ for presentation opening. We assume that public keys are published in an authenticated manner.

Enrollment: The user and $\mathcal{S}_{\mathsf{E}}$ run the following blind-issuance protocol; the user is expected to run enrollment only once:

$\mathbf{U}.\mathsf{Req}(pk_{\mathsf{E}}, nym) \to (\overline{nym}, aux_{\mathsf{E}})$ The user chooses a random nym and produces a blinded request $\overline{nym}$ and auxiliary state aux_{E}.

$\mathbf{E}.\mathsf{Issue}(sk_{\mathsf{E}}, \overline{nym}) \to \overline{\sigma_{\mathsf{E}}}$ $\mathcal{S}_{\mathsf{E}}$ produces a blinded response $\overline{\sigma_{\mathsf{E}}}$.

$\mathbf{U}.\mathsf{Fin}(\overline{\sigma_{\mathsf{E}}}, aux_{\mathsf{E}}) \to \{\sigma_{\mathsf{E}}, \bot\}$ The user unblinds the response to obtain σ_{E}, or outputs $\bot$ if the response is malformed.

Token issuance: The user and $\mathcal{S}_{\mathsf{OTT}}$ run the following partially blind issuance protocol whenever a user needs more tokens:

$\mathbf{U}.\mathsf{ReqBatch}(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}})) \to (\overline{M_{\mathsf{OTT}}}, \pi, aux_{\mathsf{OTT}})$ The user produces a batch of ℓ blinded requests $\overline{M_{\mathsf{OTT}}}$, a proof π that they hold σ_{E} for nym, and state aux_{OTT}.

$\mathbf{OTT}.\mathsf{IssueBatch}(sk_{\mathsf{OTT}}, pk_{\mathsf{E}}, \overline{M_{\mathsf{OTT}}}, nym, \pi) \to \{\overline{\Sigma_{\mathsf{OTT}}}, \bot\}$ $\mathcal{S}_{\mathsf{OTT}}$ checks π against the claimed nym and either produces a batch $\overline{\Sigma_{\mathsf{OTT}}}$ or aborts with $\bot$.

$\mathbf{U}.\mathsf{FinBatch}(\overline{\Sigma_{\mathsf{OTT}}}, aux_{\mathsf{OTT}}) \to \{(M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}}), \bot\}$ The user unblinds the response to obtain $(M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}})$, or outputs $\bot$.

Redemption and opening: Presentation and verification are run for every request, whereas opening is only needed in exceptional cases of abuse.

$\mathbf{U}.\mathsf{Present}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}), (m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})) \to \tau$ The user picks an unspent $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ and produces a presentation τ on the one-time message m_{OTT} and for enrollment credential $(nym, \sigma_{\mathsf{E}})$.

$\mathbf{G}.\mathsf{Vf}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}}) \to \{0, 1\}$ $\mathcal{G}$ publicly verifies (τ, m_{OTT}) using $pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, pk_{\mathsf{N}}$, and outputs 1 on success.

$\underline{\mathbf{N}.\mathsf{Open}(sk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}}) \to \{\mathsf{f}(nym), \bot\}}$ $\mathcal{N}$ uses sk_{N} to recover $\mathsf{f}(nym)$ from (τ, m_{OTT}), or outputs $\bot$.

Pseudonym function f. Note that opening returns $\mathsf{f}(nym)$ instead of nym. Abuse mitigation does not need nym itself. It suffices that $\mathcal{S}_{\mathsf{OTT}}$ computes $\mathsf{f}(nym)$ for a requesting pseudonym and compares it with a deny list of opened values, which requires f to be efficiently computable. Injectivity ensures that an opened value identifies a unique pseudonym, which our traceability and non-frameability notions rely on. In Apple's protocol, f is the identity function, whereas our construction AC-PORTA uses roughly $\mathsf{f}(nym) = g^{nym}$ for a generator g, as its pseudonyms are exponents but $\mathcal{N}$ decrypts group elements.

Shorthand for security model. Some oracles in our security notions run an entirely honest issuance session, for which we write $\mathbf{E}.\mathsf{HonIss}(sk_{\mathsf{E}}, nym) \to \sigma_{\mathsf{E}}$ and $\mathbf{OTT}.\mathsf{HonIssBatch}(sk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}})) \to (M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}})$.

Correctness. Honestly issued tokens must always verify, and every accepted presentation opens to the correct user's pseudonym (see App. C for details).

3 PORTA Security Model

We now formalize the six security properties of a PORTA scheme that we informally described in Sec. 2.2, covering privacy (E-BLIND, UNLINK), rate limiting (RL-UNF), and accountability (TRACE, NFRAME, BIND). Each property is defined through a game-based security notion, where the adversary interacts with the challenger. We start with an overview of the oracles and variables, and then present the formal models (Sec. 3.1 – Sec. 3.3). The full games are presented in Fig. 3.

The challenger presents honest parties in the form of oracles to the adversary. Fig. 2 contains the full descriptions of all oracles across the games (Tab. 4 in App. D summarizes the state). Intuitively the oracles provide the following:

Honest user, honest services ($\mathcal{O}_{\mathsf{Enroll}}, \mathcal{O}_{\mathsf{GetOTT}}, \mathcal{O}_{\mathsf{Present}}$): $\mathcal{O}_{\mathsf{Enroll}}$ registers a user under a fresh nym [4] and optionally returns their $(nym, \sigma_{\mathsf{E}})$ to the adversary (corrupting the user). $\mathcal{O}_{\mathsf{GetOTT}}$ runs an honest OTT issuance. $\mathcal{O}_{\mathsf{Present}}$ produces an honest presentation, referencing an OTT indirectly by batch and position index (j, k), so that honest OTTs need never be disclosed to the adversary.

Honest user, corrupt $\mathcal{S}_{\mathsf{OTT}}$ ($\mathcal{O}_{\mathsf{ReqOTT}}, \mathcal{O}_{\mathsf{FinOTT}}, \mathcal{O}_{\mathsf{LeakOTT}}$): OTT issuance is split into request and finalize so that the adversary can play the $\mathcal{S}_{\mathsf{OTT}}$ side. $\mathcal{O}_{\mathsf{FinOTT}}$ stores the resulting OTTs in the user's state without returning them; the adversary can recover them later, if the experiment allows, via $\mathcal{O}_{\mathsf{LeakOTT}}$.

Honest services, corrupt users ($\mathcal{O}_{\mathsf{IssueE}}, \mathcal{O}_{\mathsf{IssueOTT}}, \mathcal{O}_{\mathsf{Open}}$): $\mathcal{O}_{\mathsf{IssueE}}$ and $\mathcal{O}_{\mathsf{IssueOTT}}$ run the honest token issuance with adversarial input. $q_{\mathsf{E}}, q_{\mathsf{OTT}}$, and Q_{nym} are used in one-more games. $\mathcal{O}_{\mathsf{Open}}$ runs the honest $\mathcal{N}$'s opening.

3.1 Privacy: E-BLIND and UNLINK

Apple sets "user anonymity and non-targetability" [Inc24b] as a core feature and uses blind signatures for both token types to achieve it. We capture this through two properties. Enrollment blindness ensures that $\mathcal{S}_{\mathsf{E}}$ issues enrollment credentials blindly, i.e., it neither learns the pseudonym it certifies nor can link the enrollment session to any later use of it. Presentation unlinkability ensures that the many authentication requests of a user cannot be linked to each other, nor to the user's pseudonym.

[4] We assume that the space of pseudonyms NYM is exponential in λ.

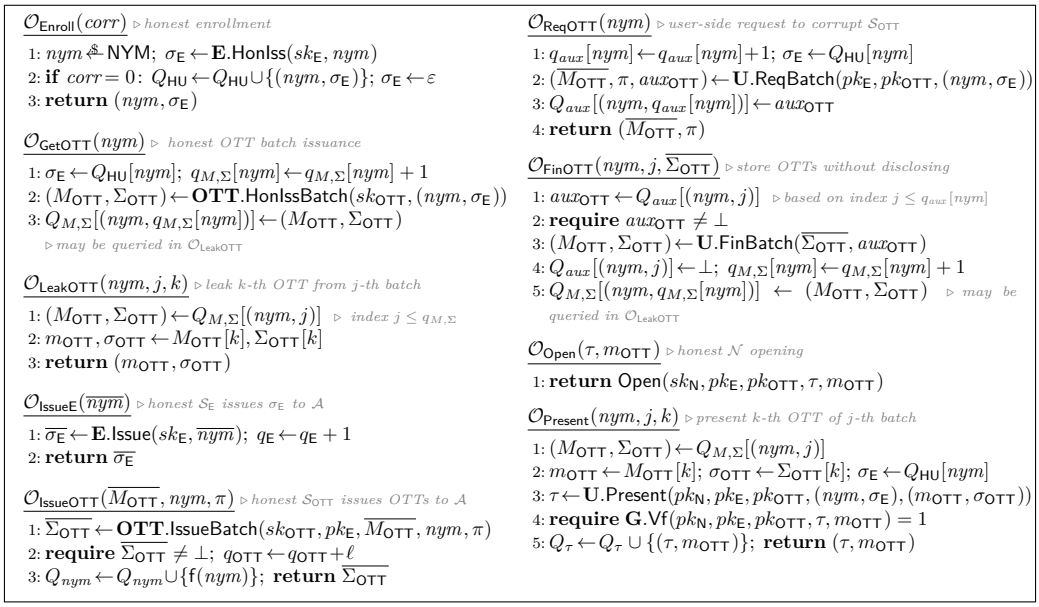

Figure 2: Oracles used across the security experiments of PORTA.

3.1.1 Enrollment Blindness (E-BLIND)

E-BLIND ensures that $\mathcal{S}_{\mathsf{E}}$ does not learn the pseudonym it certifies. Even when colluding with all other parties, $\mathcal{S}_{\mathsf{E}}$ cannot link an enrollment session to any later use of the resulting pseudonym. The game extends the standard blindness game of blind signatures to the later phases of the protocol. The adversary controls $\mathcal{S}_{\mathsf{E}}$, $\mathcal{S}_{\mathsf{OTT}}$, and $\mathcal{N}$, chooses pk_{E} and two pseudonyms nym_0, nym_1, and the challenger enrolls both honestly in an order determined by a secret bit b. The adversary has to guess b.

The adversary never obtains the unblinded σ_{E} directly, but can use the credentials through oracles for token issuance ($\mathcal{O}_{\mathsf{ReqOTT}}$, $\mathcal{O}_{\mathsf{FinOTT}}$, and $\mathcal{O}_{\mathsf{LeakOTT}}$ to obtain the resulting OTTs) and presentation ($\mathcal{O}_{\mathsf{Present}}$). The latter is needed even for leaked OTTs, as $(nym, \sigma_{\mathsf{E}})$ remains a secret input to Present. Direct access to σ_{E} would be unnecessarily strong. It would rule out constructions based on anonymous credentials, where the signature itself is linkable, but users only reveal a zero-knowledge proof of its possession. Finally, $\mathcal{N}$ is *fully* corrupt, as E-BLIND does not hide the pseudonym, only its link to the enrollment session.

3.1.2 Presentation Unlinkability (UNLINK)

UNLINK ensures that the authentications of a user remain unlinkable. Presentations at $\mathcal{G}$ can be linked neither to each other nor to the user's pseudonym or the session in which the used token was issued. This must hold even if $\mathcal{G}$ colludes with $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$, where the latter sees the pseudonym during token issuance. Only $\mathcal{N}$ must be trusted, as it can open any presentation by design.

The game is again left-or-right. The adversary controls $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$ and chooses two pseudonyms nym_0, nym_1, for which the challenger enrolls two honest users. As enrollment privacy is covered by E-BLIND, the adversary obtains both credentials. It then plays $\mathcal{S}_{\mathsf{OTT}}$ in token issuance via $\mathcal{O}_{\mathsf{ReqOTT}}$ and $\mathcal{O}_{\mathsf{FinOTT}}$, and picks two index triples $(nym_0, j_0, k_0), (nym_1, j_1, k_1)$, each selecting the k-th OTT in batch j of a user. The challenger returns the presentation $(\tau_b, m_{\mathsf{OTT}_b})$ for the triple chosen by b.

The adversary can query $\mathcal{O}_{\mathsf{Present}}$ for both challenge users, except for the challenge triples themselves, as m_{OTT} is revealed in clear and would allow a trivial win. For the

same reason, it may not query $\mathcal{O}_{\mathsf{Open}}$ on the challenge presentation, similar to the challenge ciphertext in IND-CCA. Honest users' OTTs are never disclosed, as unlinkability can only hold for unseen one-time messages. Thus, there is no $\mathcal{O}_{\mathsf{LeakOTT}}$, $\mathcal{O}_{\mathsf{FinOTT}}$ stores OTTs silently, and $\mathcal{O}_{\mathsf{Present}}$ takes batch and position indices instead of OTTs.

Definition 2. A PORTA scheme Π is x-secure for $\mathsf{x} \in \{\text{E-BLIND}, \text{UNLINK}\}$ if $\left|\Pr[\text{Exp}^{\text{x-0}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \Pr[\text{Exp}^{\text{x-1}}_{\Pi,\mathcal{A}}(\lambda) = 1]\right| \leq \text{negl}(\lambda)$ for every PPT $\mathcal{A}$.

3.2 Rate-Limited Unforgeability (RL-UNF)

Apple states two related goals, "Limiting which devices or users can access the service" and "Imposing reasonable rate limits on those users" [Inc24b]. The rate-limited unforgeability game RL-UNF bounds the number of successful presentations against both steps of the hierarchical issuance, matching these two goals. $\mathcal{S}_{\mathsf{OTT}}$ should not issue OTTs to more pseudonyms than $\mathcal{S}_{\mathsf{E}}$ has enrolled (limiting access), and $\mathcal{G}$ should not accept more presentations than $\mathcal{S}_{\mathsf{OTT}}$ has issued OTTs (rate limits). Due to blind issuance, both are one-more unforgeability goals. We package them as alternative win conditions in one game.

In the game, $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$ are honest and pk_{N} is adversarial, as $\mathcal{N}$ plays no role in enforcing rate limits. The adversary queries $\mathcal{O}_{\mathsf{IssueE}}$ (counted in q_{E}) and $\mathcal{O}_{\mathsf{IssueOTT}}$ (counted in q_{OTT}), and then outputs presentations $(\tau_i, m_{\mathsf{OTT}_i})$. It wins if it produces either (1) an OTT one-more forgery, i.e., more valid presentations than q_{OTT}, or (2) an enrollment one-more forgery, i.e., $\mathcal{A}$ has tricked $\mathcal{S}_{\mathsf{OTT}}$ into accepting more pseudonyms (counted in $|Q_{nym}|$) than there were honest enrollment sessions q_{E} at $\mathcal{S}_{\mathsf{E}}$.

Winning condition (2) is stronger than strictly necessary, as the adversary already wins if it presents more valid pseudonyms to $\mathcal{S}_{\mathsf{OTT}}$ than $\mathcal{S}_{\mathsf{E}}$ has enrolled, irrespective of whether these yield valid forgeries at $\mathcal{G}$. We adopt this stronger formulation for two reasons. First, it keeps the model simpler, as the alternative would require tracking an upper bound based on the number of token requests for distinct pseudonyms. Second, such over-issuance should be ruled out even if it does not lead to a forgery at $\mathcal{G}$. $\mathcal{S}_{\mathsf{OTT}}$ uses sk_{OTT} whenever it is presented with a valid pseudonym, treating each as a legitimate user, so our version also ensures that sk_{OTT} is never used more than necessary.

3.3 Accountability: TRACE, NFRAME and BIND

Accountability ensures that abuse can be attributed to a pseudonym and sanctioned, e.g., by denying it further tokens at $\mathcal{S}_{\mathsf{OTT}}$. This first requires reliable opening by $\mathcal{N}$. Traceability ensures that misbehaving users cannot evade opening, i.e., every accepted presentation opens to a pseudonym known to $\mathcal{S}_{\mathsf{OTT}}$, which can thus be sanctioned. Non-frameability ensures that opening cannot implicate honest users, i.e., no presentation opens to an honest user's pseudonym unless that user created it. Apple does not state requirements for opening, but group signatures have a similar opener role, from which we take inspiration for both properties [BMW03] (see Sec. D.2 for a comparison). In addition, sanctions and per-user limits are only effective if users cannot spend one-time tokens issued to others, which is ensured by binding.

Binding as a separate property. Binding is not implied by the other properties. Consider a corrupt user B who obtains an honest user A's unspent $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ (e.g., leaked or stolen) and spends it under B's own credential. The number of accepted presentations still equals the number of issued OTTs, the presentation opens to B's enrolled pseudonym, and A is not framed. Still, B exceeds its own quota and, if banned, can continue with tokens collected from others. Binding prevents this by requiring that every accepted presentation ties the spent $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ to the credential under which it is presented. The property is inspired by Apple's token design and has no direct analog for blind or group signatures, where tokens are not issued in two steps.

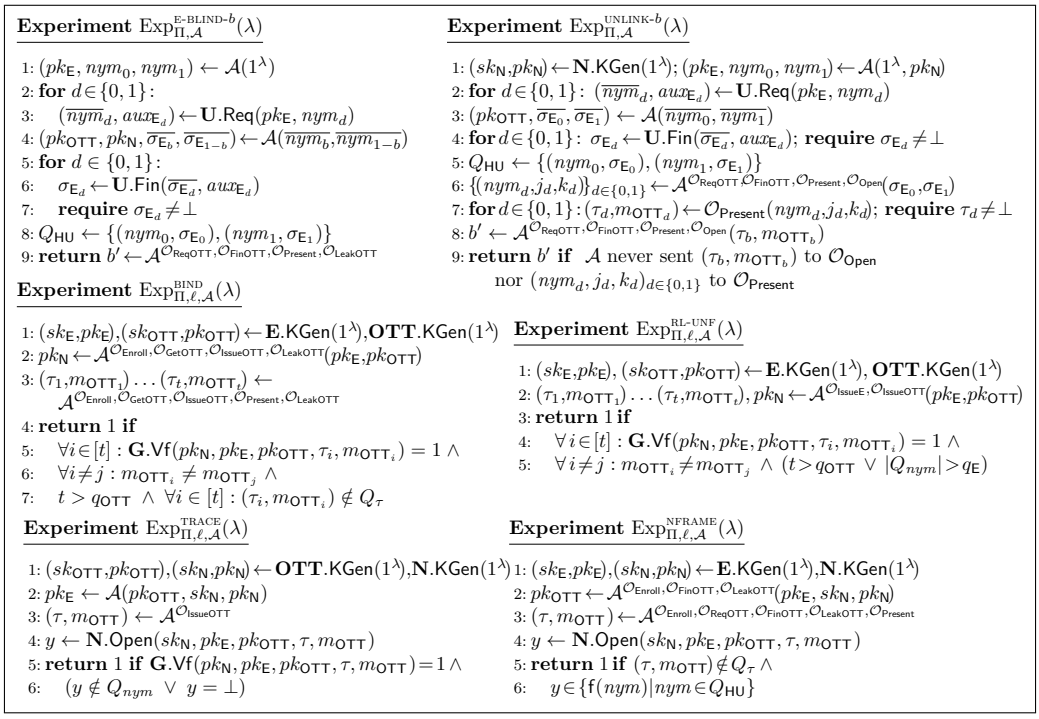

Figure 3: Privacy and security experiments for PORTA.

Semi-honest Node. The winning conditions of traceability and non-frameability run $\mathbf{N}.\mathsf{Open}$, so the key generation of $\mathcal{N}$ is run honestly, ensuring that opening is performed correctly. Still, $\mathcal{A}$ obtains sk_{N} and can otherwise deviate arbitrarily, i.e., $\mathcal{N}$ is semi-honest. A *verifiable opening* with a matching verification algorithm would also capture malicious openings, but at the cost of a more complex syntax and model. We chose the simpler, more established approach.

3.3.1 Traceability (TRACE)

TRACE requires that every presentation accepted by $\mathcal{G}$ opens to a pseudonym that $\mathcal{S}_{\mathsf{OTT}}$ has previously seen during OTT issuance. As abuse mitigation acts via $\mathcal{S}_{\mathsf{OTT}}$, a pseudonym never seen by $\mathcal{S}_{\mathsf{OTT}}$ cannot be sanctioned, and an opening that returns $\bot$ leaves the adversary unaccountable.

The adversary plays a corrupt $\mathcal{S}_{\mathsf{E}}$ (choosing pk_{E}), a semi-honest $\mathcal{N}$ (obtaining sk_{N}), and all corrupt users. $\mathcal{S}_{\mathsf{OTT}}$ is honest and records $\mathsf{f}(nym)$ of all pseudonyms it issues tokens to in Q_{nym}. The adversary outputs $(\tau^*, m^*_{\mathsf{OTT}})$ and wins if it is valid at $\mathcal{G}$ *and* $\mathbf{N}.\mathsf{Open}$ outputs $\bot$ or a value not in Q_{nym}, i.e. not a valid pseudonym representative. $\mathcal{S}_{\mathsf{OTT}}$ must be honest to define when an opened $\mathsf{f}(nym)$ is valid, as it is the only party that sees *nym*s in plain. Due to blind issuance, $\mathcal{S}_{\mathsf{E}}$ cannot take this role and may be fully corrupt.

3.3.2 Non-Frameability (NFRAME)

NFRAME protects honest users, as no adversary should produce a presentation that opens to an honest user's pseudonym. While TRACE requires that opening always returns a *valid* pseudonym, NFRAME requires that it never returns a *wrong* one. This matters even though a corrupt $\mathcal{S}_{\mathsf{OTT}}$ can deny tokens to any pseudonym, as a denial does not attribute abuse to a user, whereas a successful framing does.

$\mathcal{A}$ plays corrupt users against an honest $\mathcal{S}_{\mathsf{E}}$ and a corrupt $\mathcal{S}_{\mathsf{OTT}}$ (choosing pk_{OTT}), with $\mathcal{N}$ again semi-honest. All users are created honestly via $\mathcal{O}_{\mathsf{Enroll}}$. For corrupt users, $\mathcal{A}$ obtains $(nym, \sigma_{\mathsf{E}})$. For honest users, it can run token issuance via $\mathcal{O}_{\mathsf{ReqOTT}}$, $\mathcal{O}_{\mathsf{FinOTT}}$, and $\mathcal{O}_{\mathsf{LeakOTT}}$, and obtain presentations via $\mathcal{O}_{\mathsf{Present}}$ (recorded in Q_τ). It wins if its output $(\tau^*, m^*_{\mathsf{OTT}})$ is not a replay (i.e., not in Q_τ) and $\mathbf{N}.\mathsf{Open}$ returns $\mathsf{f}(nym)$ for an honest user's pseudonym $nym \in Q_{\mathsf{HU}}$.

We highlight two design choices. *Honest enrollment.* Compared to TRACE, we use the reverse corruption setting with $\mathcal{S}_{\mathsf{OTT}}$ corrupt and $\mathcal{S}_{\mathsf{E}}$ honest, as otherwise $\mathcal{A}$ could trivially enroll an honest user's pseudonym again to frame it. This requires all users to enroll honestly, similar to BIND (Sec. 3.3.3). *No verification at $\mathcal{G}$.* Whereas TRACE only concerns presentations accepted by $\mathcal{G}$, we formulate NFRAME in a stronger sense. Regardless of $\mathcal{G}$, honest users are guaranteed that a semi-honest $\mathcal{N}$ never wrongfully accuses them.

3.3.3 Binding (BIND)

BIND rules out *OTT transferability*, i.e., an adversary holding an honest user's unspent $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ should not be able to spend it under a different credential.

The game is again a one-more unforgeability game, but defined over honestly issued OTTs. The counter q_{OTT} only tracks OTTs for which the adversary can trivially create presentations, i.e., those obtained via $\mathcal{O}_{\mathsf{IssueOTT}}$. Honest OTTs (from $\mathcal{O}_{\mathsf{GetOTT}}$ and $\mathcal{O}_{\mathsf{LeakOTT}}$) should not allow the adversary to create presentations and are not counted. The adversary wins by producing more valid presentations than q_{OTT}, none of which matches a recorded honest presentation in Q_τ. $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$ are honest, users are enrolled honestly via $\mathcal{O}_{\mathsf{Enroll}}$ and reveal their credential when corrupted, and $\mathcal{N}$ is corrupt.

As for NFRAME, the game enforces honest enrollment via $\mathcal{O}_{\mathsf{Enroll}}$ even for corrupt users, i.e., there is no $\mathcal{O}_{\mathsf{IssueE}}$ as in RL-UNF. Due to blind enrollment, the adversary could otherwise claim an honest user's pseudonym towards $\mathcal{S}_{\mathsf{E}}$ and spend that user's OTTs. This is also why we treat BIND and RL-UNF as separate properties, as a combined definition would transfer the honest-enrollment requirement to RL-UNF and thus weaken it. A stronger but more complex model would jointly randomize *nym* between $\mathcal{S}_{\mathsf{E}}$ and the user, which we avoid for simplicity.

Definition 3. A PORTA scheme Π is x-secure for $\mathsf{x} \in \{\text{RL-UNF}, \text{TRACE}, \text{NFRAME}, \text{BIND}\}$ if $\Pr[\mathrm{Exp}^{\mathsf{x}}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ for every PPT $\mathcal{A}$.

4 Apple-PORTA: Apple's Instantiation

In this section we present and analyze Apple-PORTA, the blind-signature-based instantiation of PORTA implemented by Apple's Private Cloud Compute system. Apple does not provide a complete specification; our reconstruction draws on the security documentation [App24c], the referenced RFCs, and the public source code (which mostly covers $\mathcal{N}$ [App24b]). The full protocol is given in Fig. 4.

Sec. 4.1 introduces the building blocks, Sec. 4.2 explains their use in PORTA, and Sec. 4.3 analyzes the security. Enrollment blindness and rate-limited unforgeability hold for the deployed primitives; strong presentation unlinkability requires a small modification to the concrete signature scheme, further described below. None of the abuse-mitigation guarantees hold.

4.1 Apple-PORTA Building Blocks

Apple relies on blind RSA signatures, public-key encryption, and a hash function, which we abstract as follows. All formal definitions are given in App. A.

Blind signatures. A blind signature BS extends a standard signature (with $\mathsf{KGen}(1^\lambda)\to(sk,pk)$ and $\mathsf{Vf}(pk,m,\sigma)\to\{0,1\}$) with a blind issuance: $\mathsf{Blind}(pk,m)$ blinds m into $\overline{m}$, with randomness r. $\mathsf{BSign}(sk,\overline{m})$ produces a blind signature. $\mathsf{Unblind}(pk,\overline{\sigma},r)$ unblinds to a signature σ on m.

We require three properties of BS. *Blindness* (BLIND): a message-signature pair (m,σ) cannot be linked to the issuance transcript $(\overline{m},\overline{\sigma})$ that produced it, even against a fully malicious signer. *One-more unforgeability* (OMUF): no adversary can produce valid signatures on more distinct messages m_i^* than the number of BSign queries it made to an honest signer. *Uniqueness*: It is computationally hard to find pk,m,σ,σ' with $\sigma'\neq\sigma$ and $\mathsf{Vf}(pk,m,\sigma)=\mathsf{Vf}(pk,m,\sigma')=1$.

Public-key encryption. A public-key encryption scheme PKE is defined through $\mathsf{KGen}(1^\lambda)\to(sk,pk)$, $\mathsf{Enc}(pk,m)\to c$, and $\mathsf{Dec}(sk,c)\to m$. We require PKE to be IND-CCA-secure: an adversary holding only pk learns nothing about m from c, even given access to a decryption oracle.

Hash function. A hash function $\mathsf{H}:\{0,1\}^*\to\{0,1\}^\lambda$ is a deterministic function. We require it to be *input-hiding* (IH): Intuitively, given only $h=\mathsf{H}(m,s)$ for a random salt $s\xleftarrow{\$}\{0,1\}^\lambda$, h reveals nothing about m. This property is implied if H is a random oracle.

4.2 Apple-PORTA Construction

Apple-PORTA uses two blind signature schemes $\mathsf{BS}_1,\mathsf{BS}_2$ for issuing enrollment credentials and OTTs respectively, a PKE for transporting an enrollment credential to $\mathcal{N}$, and H for binding each OTT to its (nym,σ_E). Splitting BS into $\mathsf{BS}_1,\mathsf{BS}_2$ lets each security theorem state precisely which property of which scheme it relies on. Opening returns the plain pseudonym, i.e., f is the identity.

$\underline{\mathbf{U}.\mathsf{Req}(pk_\mathsf{E},nym)\to(\overline{nym},aux_\mathsf{E})}$
1: **if** $nym=\varepsilon$: $nym\xleftarrow{\$}\{0,1\}^\lambda$
2: $\overline{nym},r\leftarrow\mathsf{BS}_1.\mathsf{Blind}(pk_\mathsf{E},nym)$
3: **return** $(\overline{nym},(nym,r,pk_\mathsf{E}))$

$\underline{\mathbf{E}.\mathsf{Issue}(sk_\mathsf{E},\overline{nym})\to\overline{\sigma_\mathsf{E}}}$
4: **return** $\mathsf{BS}_1.\mathsf{BSign}(sk_\mathsf{E},\overline{nym})$

$\underline{\mathbf{U}.\mathsf{Fin}(\overline{\sigma_\mathsf{E}},aux_\mathsf{E})\to\{\sigma_\mathsf{E},\perp\}}$
5: $(nym,r,pk_\mathsf{E})\leftarrow aux_\mathsf{E};\ \sigma_\mathsf{E}\leftarrow\mathsf{BS}_1.\mathsf{Unblind}(pk_\mathsf{E},\overline{\sigma_\mathsf{E}},r)$
6: **require** $\mathsf{BS}_1.\mathsf{Vf}(pk_\mathsf{E},nym,\sigma_\mathsf{E})=1$; **return** σ_E

$\underline{\mathbf{U}.\mathsf{ReqBatch}(pk_\mathsf{E},pk_\mathsf{OTT},(nym,\sigma_\mathsf{E}))\to(\overline{M_\mathsf{OTT}},\pi,aux_\mathsf{OTT})}$
7: **for** $i=1\dots\ell$:
8: $\quad salt_i\xleftarrow{\$}\{0,1\}^\lambda;\ m_{\mathsf{OTT}_i}\leftarrow\mathsf{H}((nym,\sigma_\mathsf{E}),salt_i)$
9: $\quad\overline{m_{\mathsf{OTT}_i}},r_i\leftarrow\mathsf{BS}_2.\mathsf{Blind}(pk_\mathsf{OTT},m_{\mathsf{OTT}_i})$
10: $\overline{M_\mathsf{OTT}}\leftarrow(\overline{m_{\mathsf{OTT}_1}}\dots\overline{m_{\mathsf{OTT}_\ell}});\ \pi\leftarrow\sigma_\mathsf{E}$
11: $aux_\mathsf{OTT}\leftarrow(pk_\mathsf{OTT},(m_{\mathsf{OTT}_1},r_1,salt_1)\dots(m_{\mathsf{OTT}_\ell},r_\ell,salt_\ell))$
12: **return** $(\overline{M_\mathsf{OTT}},\pi,aux_\mathsf{OTT})$

$\underline{\mathbf{OTT}.\mathsf{IssueBatch}(sk_\mathsf{OTT},pk_\mathsf{E},\overline{M_\mathsf{OTT}},nym,\pi)\to\{\overline{\Sigma_\mathsf{OTT}},\perp\}}$
13: $\sigma_\mathsf{E}\leftarrow\pi$; **require** $\mathsf{BS}_1.\mathsf{Vf}(pk_\mathsf{E},nym,\sigma_\mathsf{E})=1$
14: $(\overline{m_{\mathsf{OTT}_1}}\dots\overline{m_{\mathsf{OTT}_\ell}})\leftarrow\overline{M_\mathsf{OTT}}$
15: **for** $i=1\dots\ell$: $\overline{\sigma_{\mathsf{OTT}_i}}\leftarrow\mathsf{BS}_2.\mathsf{BSign}(sk_\mathsf{OTT},\overline{m_{\mathsf{OTT}_i}})$
16: **return** $\overline{\Sigma_\mathsf{OTT}}\leftarrow(\overline{\sigma_{\mathsf{OTT}_1}}\dots\overline{\sigma_{\mathsf{OTT}_\ell}})$

$\underline{\mathbf{U}.\mathsf{FinBatch}(\overline{\Sigma_\mathsf{OTT}},aux_\mathsf{OTT})\to\{(M_\mathsf{OTT},\Sigma_\mathsf{OTT}),\perp\}}$
17: $(pk_\mathsf{OTT},(m_{\mathsf{OTT}_1},r_1,\underline{salt_1})\dots(m_{\mathsf{OTT}_\ell},r_\ell,salt_\ell))\leftarrow aux_\mathsf{OTT}$
18: $(\overline{\sigma_{\mathsf{OTT}_1}}\dots\overline{\sigma_{\mathsf{OTT}_\ell}})\leftarrow\overline{\Sigma_\mathsf{OTT}}$
19: **for** $i=1\dots\ell$:
20: $\quad\sigma'_{\mathsf{OTT}_i}\leftarrow\mathsf{BS}_2.\mathsf{Unblind}(pk_\mathsf{OTT},\overline{\sigma_{\mathsf{OTT}_i}},r_i)$
21: $\quad$**require** $\mathsf{BS}_2.\mathsf{Vf}(pk_\mathsf{OTT},m_{\mathsf{OTT}_i},\sigma'_{\mathsf{OTT}_i})=1$
22: $M_\mathsf{OTT}\leftarrow(m_{\mathsf{OTT}_1}\dots m_{\mathsf{OTT}_\ell});\ \Sigma_\mathsf{OTT}\leftarrow((\sigma'_{\mathsf{OTT}_1},salt_1)\dots(\sigma'_{\mathsf{OTT}_\ell},salt_\ell))$
23: **return** $(M_\mathsf{OTT},\Sigma_\mathsf{OTT})$

$\underline{\mathbf{U}.\mathsf{Present}(pk_\mathsf{N},pk_\mathsf{E},pk_\mathsf{OTT},(nym,\sigma_\mathsf{E}),(m_\mathsf{OTT},\sigma_\mathsf{OTT}))\to\tau}$
24: $(\sigma'_\mathsf{OTT},salt)\leftarrow\sigma_\mathsf{OTT}$
25: $c\leftarrow\mathsf{PKE}.\mathsf{Enc}(pk_\mathsf{N},((nym,\sigma_\mathsf{E}),salt));\ \tau\leftarrow(c,\sigma'_\mathsf{OTT})$
26: **return** τ

$\underline{\mathbf{G}.\mathsf{Vf}(pk_\mathsf{N},pk_\mathsf{E},pk_\mathsf{OTT},\tau,m_\mathsf{OTT})\to\{0,1\}}$
27: $(c,\sigma'_\mathsf{OTT})\leftarrow\tau$
28: **return** $\mathsf{BS}_2.\mathsf{Vf}(pk_\mathsf{OTT},m_\mathsf{OTT},\sigma'_\mathsf{OTT})$

$\underline{\mathbf{N}.\mathsf{Open}(sk_\mathsf{N},pk_\mathsf{E},pk_\mathsf{OTT},\tau,m_\mathsf{OTT})\to\{\mathsf{f}(nym),\perp\}}$
29: **require** $\mathbf{G}.\mathsf{Vf}(pk_\mathsf{N},pk_\mathsf{E},pk_\mathsf{OTT},\tau,m_\mathsf{OTT})=1$ ▷ *Verify OTT*
30: $(c,\sigma'_\mathsf{OTT})\leftarrow\tau;\ p\leftarrow\mathsf{PKE}.\mathsf{Dec}(sk_\mathsf{N},c)$
31: $((nym,\sigma_\mathsf{E}),salt)\leftarrow p$; **require** $p\neq\perp$
32: **require** $\mathsf{BS}_1.\mathsf{Vf}(pk_\mathsf{E},nym,\sigma_\mathsf{E})=1$
33: **require** $\mathsf{H}((nym,\sigma_\mathsf{E}),salt)=m_\mathsf{OTT}$; **return** nym

Figure 4: Apple-PORTA using the blind signatures $\mathsf{BS}_1,\mathsf{BS}_2$, public-key encryption PKE, and hash H. Setup: $\mathbf{E}.\mathsf{KGen}=\mathsf{BS}_1.\mathsf{KGen}$, $\mathbf{OTT}.\mathsf{KGen}=\mathsf{BS}_2.\mathsf{KGen}$, $\mathbf{N}.\mathsf{KGen}=\mathsf{PKE}.\mathsf{KGen}$.

Phase 1: Enrollment. The user samples nym and runs the BS_1 blind-signing protocol with $\mathcal{S}_\mathsf{E}$ to obtain σ_E on nym.
Phase 2: Token Issuance. The user prepares ℓ one-time messages $m_{\mathsf{OTT}_i}\leftarrow\mathsf{H}((nym,\sigma_\mathsf{E}),salt_i)$ for fresh salts $salt_i$, blinds each into $\overline{m_{\mathsf{OTT}_i}}$, and sends them to $\mathcal{S}_\mathsf{OTT}$ along with $\pi\leftarrow\sigma_\mathsf{E}$. $\mathcal{S}_\mathsf{OTT}$ verifies σ_E under pk_E and blindly signs each $\overline{m_{\mathsf{OTT}_i}}$ with BS_2. Embedding $\mathsf{H}((nym,\sigma_\mathsf{E}),salt_i)$ inside each m_OTT is the trick that blindly ties each OTT

to a specific $(nym, \sigma_{\mathsf{E}})$: $\mathcal{S}_{\mathsf{OTT}}$ never sees the hash, only its blinded form, while the salts ensure the ℓ messages are mutually unlinkable.
Phase 3: Redemption. The user encrypts $((nym, \sigma_{\mathsf{E}}),salt)$ under pk_{N} to obtain c, and sets $\tau \leftarrow (c,\sigma_{\mathsf{OTT}})$. It sends (τ,m_{OTT}) to $\mathcal{G}$, which verifies the OTT signature, but does not check c. This design choice will impact binding and traceability.
Phase 4: Opening. $\mathcal{N}$ decrypts c to recover $((nym, \sigma_{\mathsf{E}}),salt)$, verifies both tokens, and checks $\mathsf{H}((nym, \sigma_{\mathsf{E}}),salt) = m_{\mathsf{OTT}}$. On success, $\mathcal{N}$ outputs nym; otherwise it outputs $\perp$.

4.3 Security Analysis of Apple-PORTA

We first prove E-BLIND and RL-UNF, and prove UNLINK under the theorem's uniqueness assumption. We then explain why the deployed probabilistic signature falls short of that assumption and why Apple-PORTA fails BIND, TRACE, and full NFRAME. All theorems assume correctness of the building blocks.

Theorem 1. *Apple-PORTA is* E-BLIND*-secure if* BS_1 *is* BLIND*.*

Proof. Enrollment in Apple-PORTA is structurally the blind issuance of BS_1 on nym, so BLIND directly hides the link between an enrollment session and the resulting $(nym, \sigma_{\mathsf{E}})$. The reduction forwards the adversary's $(pk_{\mathsf{E}}, nym_0, nym_1)$ and its two signing responses to the BS_1 blindness challenger. Once both sessions succeed, it obtains both enrollment credentials and answers $\mathcal{O}_{\mathsf{ReqOTT}}$, $\mathcal{O}_{\mathsf{FinOTT}}$, $\mathcal{O}_{\mathsf{LeakOTT}}$, and $\mathcal{O}_{\mathsf{Present}}$ honestly, as none of them needs the enrollment blinding randomness. These later oracles thus add no further linkage to the enrollment transcript. The reduction forwards the adversary's guess, and as the views are identical (including enrollment failures), the advantages coincide. □

Theorem 2. *Apple-PORTA is* UNLINK*-secure if* BS_2 *is* BLIND *and has unique signatures,* PKE *is* IND-CCA*, and* H *is* IH*.*

Proof sketch. To win, the adversary needs to determine which triple (nym, j, k) corresponds to the challenge (τ, m_{OTT}). A presentation (τ, m_{OTT}) with $\tau = (c, \sigma_{\mathsf{OTT}})$ has three components, each individually hiding the user's credential and issuance session. The pair $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ is unlinkable to its issuance transcript by BLIND of BS_2, lifted across batches and positions via the fresh blinding randomness of each token. The message $m_{\mathsf{OTT}} = \mathsf{H}((nym, \sigma_{\mathsf{E}}), salt)$ hides $(nym, \sigma_{\mathsf{E}})$ by IH of H, as the salt is random and never exposed to $\mathcal{A}$. The ciphertext c hides $((nym, \sigma_{\mathsf{E}}), salt)$ by IND-CCA of PKE. The $\mathcal{O}_{\mathsf{Open}}$ oracle, which the adversary may query on anything except the challenge, does not help. A modified c is non-malleable by IND-CCA, a different σ_{OTT} that still verifies on m_{OTT} contradicts uniqueness of BS_2, and a different m_{OTT} fails the hash check at $\mathcal{N}$. The full proof in Sec. E.1 formalizes this through a sequence of hybrids. It first replaces the ciphertexts of honest presentations by encryptions of a fixed dummy plaintext. As these can no longer be decrypted meaningfully, the reduction records message, signature, and pseudonym of each honest presentation, and answers $\mathcal{O}_{\mathsf{Open}}$ on a recorded ciphertext only if the entire presentation matches. This is exact, as decryption fixes $((nym, \sigma_{\mathsf{E}}), salt)$ and thus the message, and uniqueness of BS_2 fixes the only valid signature on it. All other ciphertexts are opened via the IND-CCA decryption oracle. Only then are the hashes of honest OTTs replaced by hashes of a fixed dummy credential, as input hiding requires the salts, which are contained in the ciphertexts, to remain secret. Finally, the reduction embeds the two challenge issuance sessions into the blindness game of BS_2, losing a factor quadratic in the number of OTTs for guessing these sessions.

Unlinkability of the deployed scheme. Thm. 2 requires BS_2 to have unique signatures, which Apple's deployed scheme does not provide, as it uses salted, i.e., probabilistic, RSA-PSS. A corrupt $\mathcal{S}_{\mathsf{OTT}}$ knowing sk_{OTT} can then re-sign the message of the challenge

presentation, query the modified presentation to $\mathcal{O}_{\mathsf{Open}}$, and learn the challenge bit. Without access to $\mathcal{O}_{\mathsf{Open}}$ this attack does not apply, and Apple-PORTA achieves a weaker, CPA-style variant of unlinkability. To achieve unlinkability as in Def. 2, we propose to use deterministic, saltless RSA-PSS instead. As its signatures are only unique for well-formed keys, $\mathcal{S}_{\mathsf{OTT}}$ additionally has to prove in zero knowledge that pk_{OTT} is well-formed, which then meets the uniqueness premise of Thm. 2.

Theorem 3. *Apple-PORTA is* RL-UNF*-secure if both* $\mathsf{BS}_1, \mathsf{BS}_2$ *are* OMUF*.*

Proof. To win, the adversary must either register more pseudonyms at $\mathcal{O}_{\mathsf{IssueOTT}}$ than at $\mathcal{O}_{\mathsf{IssueE}}$, or provide more than q_{OTT} valid presentations. The first case yields a one-more forgery of BS_1. The reduction embeds the BS_1 key, forwards every $\mathcal{O}_{\mathsf{IssueE}}$ query, and signs OTTs itself. Each distinct pseudonym accepted by $\mathcal{O}_{\mathsf{IssueOTT}}$ comes with a valid enrollment signature, so $|Q_{nym}| > q_{\mathsf{E}}$ gives signatures on more distinct messages than signing queries. The second case yields a one-more forgery of BS_2. The reduction generates the enrollment key itself and forwards the ℓ signing requests of each accepted batch, i.e., it makes q_{OTT} signing queries. Each of the $t > q_{\mathsf{OTT}}$ accepted presentations contains a valid OTT signature on a distinct message. Both simulations are exact and the Node key plays no role, so a union bound over the two cases proves the claim. □

Missing properties. The remaining properties BIND, TRACE, and NFRAME all fail in Apple-PORTA (Thm. 4). We sketch the attacks here and give them in full in Sec. E.2.

Binding. $\mathcal{G}$ verifies only the OTT signature and never inspects the ciphertext c carrying the user's credential to $\mathcal{N}$, so it cannot check that the token belongs to this credential. A corrupt user can thus present a leaked, unspent OTT of an honest user together with an arbitrary ciphertext, and $\mathcal{G}$ accepts. The salted-hash check at $\mathcal{N}$ catches the mismatch, but only after $\mathcal{G}$ has accepted and forwarded the request.

Traceability. The missing check at $\mathcal{G}$ also allows pairing a validly issued OTT with a malformed ciphertext, which $\mathcal{G}$ accepts and $\mathcal{N}$ opens to $\perp$. A second attack passes the check at $\mathcal{N}$ and exploits that OTTs are issued blindly, so $\mathcal{S}_{\mathsf{OTT}}$ cannot ensure that they are bound to the credential presented at issuance. An adversary holding two credentials $(nym_0, \sigma_{\mathsf{E}_0})$ and $(nym_1, \sigma_{\mathsf{E}_1})$ authenticates to $\mathcal{S}_{\mathsf{OTT}}$ with $(nym_1, \sigma_{\mathsf{E}_1})$, but requests OTTs on hashes of $(nym_0, \sigma_{\mathsf{E}_0})$. These tokens open to nym_0, which $\mathcal{S}_{\mathsf{OTT}}$ has never seen.

Non-frameability. The user's credential is sent in plain to $\mathcal{S}_{\mathsf{OTT}}$ with every batch request and encrypted to $\mathcal{N}$ in every presentation. A corrupt $\mathcal{S}_{\mathsf{OTT}}$ or $\mathcal{N}$ can thus create fresh presentations that open to an honest user, i.e., it obtains a long-term framing capability against every honest user it has interacted with.

We stress that the binding and traceability attacks require a user to deviate from the protocol at the cryptographic level. Against users that follow the protocol, the checks would go through. Since abuse mitigation is, by definition, meant to protect against malicious users, we do not regard such weakened guarantees as practically meaningful. The non-frameability attack, in contrast, does not need a malicious user but a single corrupt service, and Apple-PORTA only achieves non-frameability if all services are honest.

Theorem 4. *Apple-PORTA does not achieve* BIND*,* TRACE*, or* NFRAME*.*

A weaker variant of non-frameability (wNFRAME), in which all services are honest and only users are corrupt, *is* achieved by Apple-PORTA if BS_1 is UNF-CMA, BS_2 has unique signatures, PKE is IND-CCA, and H is modeled as a random oracle. The formal details are given in Sec. E.3.

Relation to reverse-engineering findings. Our results also relate to the observations of the two reverse-engineering studies of PCC [ZZWL26, DSV+26]. Zhou et al. [ZZWL26] extract the enrollment credential from a user's device with malware and use it on another device. In our model, this corresponds to corrupting that user. No scheme can prevent a party holding the full credential from acting as that user, and protecting the credential on the device is outside our model. Dittmar et al. [DSV+26] observe that one-time messages are salted hashes of the credential and ask whether this harms unlinkability. Our analysis answers this question. The salted hash does not affect privacy, but UNLINK in our strong sense fails for the deployed probabilistic RSA-PSS variant and holds for the saltless variant with a proof that the signing key is well-formed. They also report that the deployed system accepts mismatched token pairs, as the binding check is currently disabled. Our binding attack shows that enabling this check at $\mathcal{N}$ would not suffice for BIND, as $\mathcal{G}$ accepts and forwards such requests before $\mathcal{N}$ rejects them. Finally, they report more enrollment credentials per device than documented and one-time tokens that can be redeemed several times. Both concern how eligibility and double-spending are enforced, which our model assumes rather than analyzes.

5 AC-PORTA: An AC-Based Instantiation

We now present AC-PORTA, which follows the approach of anonymous credentials. Instead of revealing their signatures, users prove knowledge of them in zero knowledge, which also allows proving that the enrollment credential, the one-time token, and the ciphertext for $\mathcal{N}$ are consistent. We do not claim particular novelty for AC-PORTA, as it follows well-established techniques from anonymous credentials. This is precisely what we want to show. All six properties of PORTA can be achieved from relatively standard building blocks, and the construction stays close enough to Apple's design to serve as a concrete upgrade path. Sec. 5.1 first gives the necessary preliminaries; Sec. 5.2 describes the construction and Sec. 5.3 states the security results. The full instantiation is given in Fig. 5.

5.1 AC-PORTA Building Blocks

AC-PORTA uses two variants of blind signatures, a weakly blind signature for the long-term credential and a partially blind multi-message signature for the OTTs, as well as non-interactive zero-knowledge proofs and public-key encryption with an encoding function. All formal definitions are given in App. A.

Weakly and partially blind signatures. AC-PORTA adapts the blind signature scheme BS from Sec. 4.1 in two ways. First, we relax blindness for both signatures we use: instead of full blindness, we only require unlinkability to hold if the signature σ is not revealed to the adversary, a notion we call WBLIND (Def. 5). This relaxation is exactly what is needed for anonymous credentials with blind issuance – the holder never shows σ in the clear, but only proves possession of it in zero knowledge (see below). The long-term credential is issued with such a weakly blind signature scheme SIG, which is otherwise syntactically identical to BS. For SIG, we additionally require standard unforgeability (UNF-CMA), i.e., no adversary can produce a valid signature on a message it has not queried to the signing oracle, which runs blind issuance honestly on the plaintext message (Def. 6).

Second, the one-time tokens are issued with a *partially blind multi-message* signature scheme MSIG that signs a pair of messages (m_0, m_1), of which only m_0 is blinded. Compared to BS, the signing and verification algorithms additionally take the plaintext m_1: $\mathsf{BSign}(sk, (\overline{m}_0, m_1))$ signs a blinded m_0 and a plaintext m_1. $\mathsf{Vf}(pk, (m_0, m_1), \sigma)$ now verifies a message tuple. The remaining algorithms are as in SIG. The two security notions adapt accordingly: WBLIND is required only with respect to the blinded m_0, while m_1 may be

revealed. One-more unforgeability counts signing requests separately for each m_1. OMUF holds if after q signing requests for m_1, the adversary can neither find a one-more forgery on (m_0^*, m_1), nor a forgery on $(\cdot, m_1^*)$ for a fresh m_1^*, i.e. $q = 0$. Formal details are in App. A.

We retain the blind-signature syntax despite AC-PORTA proving possession of σ in zero-knowledge, instead of showing it. This is done for simplicity and comparability, by reusing the same blind issuance syntax across both protocols.

Zero-knowledge proofs. We write $\pi \leftarrow \mathsf{NIZK}\{(\sigma, m) : \mathsf{SIG.Vf}(pk, m, \sigma) = 1\}[ctx]$ to denote a proof of knowledge of a signature σ on a message m that verifies under pk, revealing nothing about σ, m beyond the truth of the statement. The proof is bound to its intended context ctx so it cannot be reused outside it. We require completeness, zero-knowledge, and online simulation-extractability [Gro06, FKMV12] (a valid witness can be extracted without rewinding from any accepting, not-simulated proof). For the Fiat-Shamir transform, context binding is achieved by hashing an unambiguous encoding of ctx, along with the public statement, and the prover's commitment into the challenge.

Public-key encryption with encoding. Opening only needs a unique and checkable value of the pseudonym, not the pseudonym itself. We capture this generically by public-key encryption PKE with an encoding. Encryption $\mathsf{Enc}(pk, m; v)$ takes a message $m \in \mathcal{M}_{\mathsf{PKE}}$ and randomness v, which we make explicit for use in NIZK statements. Decryption $\mathsf{Dec}(sk, c)$ returns the encoding $\mathsf{f}(m) \in \mathcal{M}'_{\mathsf{PKE}}$ under a public, injective, and efficiently computable function $\mathsf{f}\colon \mathcal{M}_{\mathsf{PKE}} \to \mathcal{M}'_{\mathsf{PKE}}$, i.e., correctness requires $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m; v)) = \mathsf{f}(m)$. For the identity f, this is standard public-key encryption. In AC-PORTA, we encrypt pseudonyms, i.e., $\mathcal{M}_{\mathsf{PKE}} = \mathsf{NYM}$, and f is the function of Def. 1. A non-trivial encoding allows, e.g., to encrypt pseudonyms that are exponents with ElGamal, where decryption recovers $\mathsf{f}(m) = g^m$, which is exactly what our concrete instantiation uses (App. B). Keeping pseudonyms as exponents limits our zero-knowledge proofs to statements over exponents, whereas statements over group elements would require more expensive proof techniques. We only require IND-CPA security, which is defined as for standard public-key encryption, with messages from $\mathcal{M}_{\mathsf{PKE}}$ (Def. 7).

5.2 AC-PORTA Construction

AC-PORTA keeps the four-phase structure of Apple-PORTA but replaces the blind-signature *showings* with zero-knowledge proofs. As a result, $(nym, \sigma_{\mathsf{E}})$ is no longer disclosed in plaintext to $\mathcal{S}_{\mathsf{OTT}}$ or $\mathcal{N}$, and the binding between credential and OTT can be enforced at $\mathcal{G}$ instead of only at $\mathcal{N}$. All building blocks operate on the same pseudonym space NYM. Enrollment credentials are signatures of a weakly blind signature scheme SIG on pseudonyms, and OTTs are signatures of a partially blind multi-message signature scheme MSIG on pairs $(m_{\mathsf{OTT}}, nym) \in \mathcal{M} \times \mathsf{NYM}$. Public-key encryption PKE with message space NYM transports the pseudonym to $\mathcal{N}$, and its encoding defines the function f of AC-PORTA, i.e., opening returns $\mathsf{f}(nym)$. Two proof systems NIZK_1 and NIZK_2 prove possession of the credentials during token issuance and redemption.

Phase 1: Enrollment. The user samples a fresh nym and runs the SIG blind signing protocol with $\mathcal{S}_{\mathsf{E}}$ to obtain σ_{E}.

Phase 2: Token Issuance. The user prepares ℓ blinded one-time messages $\overline{m_{\mathsf{OTT}_i}}$ on random m_{OTT_i}, and accompanies them with a NIZK_1 proof π of holding σ_{E} on the claimed nym (bound to $\overline{M_{\mathsf{OTT}}}$ as context). The user sends $(\overline{M_{\mathsf{OTT}}}, nym, \pi)$ to $\mathcal{S}_{\mathsf{OTT}}$. After verifying π, $\mathcal{S}_{\mathsf{OTT}}$ signs each joint message $(m_{\mathsf{OTT}_i}, nym)$ – nym as plaintext, m_{OTT_i} blinded – using MSIG. The user unblinds and verifies the batch. In contrast to Apple-PORTA, only nym (not the full $(nym, \sigma_{\mathsf{E}})$) is revealed to $\mathcal{S}_{\mathsf{OTT}}$, and $\mathcal{S}_{\mathsf{OTT}}$ – not the user – cryptographically binds m_{OTT} to nym.

Phase 3: Redemption. To spend an OTT, the user encrypts nym under pk_{N} as $c \leftarrow \overline{\mathsf{PKE.Enc}(pk_{\mathsf{N}}, nym; v)}$ and produces a NIZK_2 proof τ' of the conjunctive statement: "σ_{E}

is a valid SIG-signature on nym, σ_{OTT} is a valid MSIG-signature on (m_{OTT}, nym), and c encrypts nym, all for the same nym". The presentation is $\tau \leftarrow (c, \tau')$, sent to $\mathcal{G}$. Verification at $\mathcal{G}$ now covers the OTT, the $(nym, \sigma_{\mathsf{E}})$, and the binding to nym in a single proof.

Phase 4: Opening. $\mathcal{N}$ decrypts c to recover $\mathsf{f}(nym)$. A valid NIZK_2 ensures nym is consistent with both the $(nym, \sigma_{\mathsf{E}})$ and the OTT. $\mathcal{N}$ only opens presentations of abusive requests and learns nothing about the pseudonym behind all other requests.

$\underline{\mathbf{U}.\mathsf{Req}(pk_{\mathsf{E}}, nym) \to (\overline{nym}, aux_{\mathsf{E}})}$
1: **if** $nym = \varepsilon$: $nym \xleftarrow{\$} \mathsf{NYM}$
2: $(\overline{nym}, r) \leftarrow \mathsf{SIG.Blind}(pk_{\mathsf{E}}, nym)$
3: **return** $(\overline{nym}, (nym, r, pk_{\mathsf{E}}))$

$\underline{\mathbf{E}.\mathsf{Issue}(sk_{\mathsf{E}}, \overline{nym}) \to \overline{\sigma_{\mathsf{E}}}}$
4: **return** $\mathsf{SIG.BSign}(sk_{\mathsf{E}}, \overline{nym})$

$\underline{\mathbf{U}.\mathsf{Fin}(\overline{\sigma_{\mathsf{E}}}, aux_{\mathsf{E}}) \to \{\sigma_{\mathsf{E}}, \bot\}}$
5: $(nym, r, pk_{\mathsf{E}}) \leftarrow aux_{\mathsf{E}}$; $\sigma_{\mathsf{E}} \leftarrow \mathsf{SIG.Unblind}(pk_{\mathsf{E}}, \overline{\sigma_{\mathsf{E}}}, r)$
6: **require** $\mathsf{SIG.Vf}(pk_{\mathsf{E}}, nym, \sigma_{\mathsf{E}}) = 1$; **return** σ_{E}

$\underline{\mathbf{U}.\mathsf{ReqBatch}(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}})) \to (\overline{M_{\mathsf{OTT}}}, \pi, aux_{\mathsf{OTT}})}$
7: **for** $i = 1 \dots \ell$: $m_{\mathsf{OTT}_i} \xleftarrow{\$} \mathcal{M}$
8: $\overline{m_{\mathsf{OTT}_i}}, r_i \leftarrow \mathsf{MSIG.Blind}(pk_{\mathsf{OTT}}, m_{\mathsf{OTT}_i})$
9: $\overline{M_{\mathsf{OTT}}} \leftarrow (\overline{m_{\mathsf{OTT}_1}} \dots \overline{m_{\mathsf{OTT}_\ell}})$
10: $\pi \leftarrow \mathsf{NIZK}_1\{(\sigma_{\mathsf{E}}) : \mathsf{SIG.Vf}(pk_{\mathsf{E}}, nym, \sigma_{\mathsf{E}}) = 1\}[\overline{M_{\mathsf{OTT}}}]$
11: $aux_{\mathsf{OTT}} \leftarrow (nym, pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}_1}, r_1) \dots (m_{\mathsf{OTT}_\ell}, r_\ell))$
12: **return** $(\overline{M_{\mathsf{OTT}}}, \pi, aux_{\mathsf{OTT}})$

$\underline{\mathbf{OTT}.\mathsf{IssueBatch}(sk_{\mathsf{OTT}}, pk_{\mathsf{E}}, \overline{M_{\mathsf{OTT}}}, nym, \pi) \to \{\overline{\Sigma_{\mathsf{OTT}}}, \bot\}}$
13: **require** π verifies w.r.t. $(nym, pk_{\mathsf{E}}, \overline{M_{\mathsf{OTT}}})$
14: $(\overline{m_{\mathsf{OTT}_1}} \dots \overline{m_{\mathsf{OTT}_\ell}}) \leftarrow \overline{M_{\mathsf{OTT}}}$
15: **for** $i = 1 \dots \ell$: $\overline{\sigma_{\mathsf{OTT}_i}} \leftarrow \mathsf{MSIG.BSign}(sk_{\mathsf{OTT}}, (\overline{m_{\mathsf{OTT}_i}}, nym))$
16: **return** $\overline{\Sigma_{\mathsf{OTT}}} \leftarrow (\overline{\sigma_{\mathsf{OTT}_1}} \dots \overline{\sigma_{\mathsf{OTT}_\ell}})$

$\underline{\mathbf{U}.\mathsf{FinBatch}(\overline{\Sigma_{\mathsf{OTT}}}, aux_{\mathsf{OTT}}) \to \{(M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}}), \bot\}}$
17: $(\overline{\sigma_{\mathsf{OTT}_1}} \dots \overline{\sigma_{\mathsf{OTT}_\ell}}) \leftarrow \overline{\Sigma_{\mathsf{OTT}}}$
18: $(nym, pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}_1}, r_1) \dots (m_{\mathsf{OTT}_\ell}, r_\ell)) \leftarrow aux_{\mathsf{OTT}}$
19: **for** $i = 1..\ell$:
20: $\quad \sigma_{OTT_i} \leftarrow \mathsf{MSIG.Unblind}(pk_{\mathsf{OTT}}, \overline{\sigma_{\mathsf{OTT}_i}}, r_i)$
21: $\quad$ **require** $\mathsf{MSIG.Vf}(pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}_i}, nym), \sigma_{\mathsf{OTT}_i}) = 1$
22: $M_{\mathsf{OTT}} \leftarrow (m_{\mathsf{OTT}_1} \dots m_{\mathsf{OTT}_\ell})$; $\Sigma_{\mathsf{OTT}} \leftarrow (\sigma_{\mathsf{OTT}_1} \dots \sigma_{\mathsf{OTT}_\ell})$
23: **return** $M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}}$

$\underline{\mathbf{U}.\mathsf{Present}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}), (m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})) \to \tau}$
24: $c \leftarrow \mathsf{PKE.Enc}(pk_{\mathsf{N}}, nym; v)$
25: $\tau' \leftarrow \mathsf{NIZK}_2\left\{ \begin{pmatrix} nym, \\ v, \sigma_{\mathsf{E}}, \\ \sigma_{\mathsf{OTT}} \end{pmatrix} : \begin{matrix} \mathsf{SIG.Vf}(pk_{\mathsf{E}}, nym, \sigma_{\mathsf{E}}) = 1 \\ \mathsf{MSIG.Vf}(pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}}, nym), \sigma_{\mathsf{OTT}}) = 1 \\ c = \mathsf{PKE.Enc}(pk_{\mathsf{N}}, nym; v) \end{matrix} \right\}$
26: $\tau \leftarrow (c, \tau')$; **return** τ

$\underline{\mathbf{G}.\mathsf{Vf}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}}) \to \{0, 1\}}$
27: $(c, \tau') \leftarrow \tau$
28: **return** 1 **if** τ' verifies w.r.t. $(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, m_{\mathsf{OTT}}, c)$

$\underline{\mathbf{N}.\mathsf{Open}(sk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}}) \to \{\mathsf{f}(nym), \bot\}}$
29: **require** $\mathbf{G}.\mathsf{Vf}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}}) = 1$
30: $(c, \tau') \leftarrow \tau$; $\mathsf{f}(nym) \leftarrow \mathsf{PKE.Dec}(sk_{\mathsf{N}}, c)$
31: **require** $\mathsf{f}(nym) \neq \bot$; **return** $\mathsf{f}(nym)$

Figure 5: Our AC Instantiation (AC-PORTA) using SIG, MSIG, PKE, and NIZK systems $\mathsf{NIZK}_1, \mathsf{NIZK}_2$. Setup: $\mathbf{E}.\mathsf{KGen} = \mathsf{SIG.KGen}$, $\mathbf{OTT}.\mathsf{KGen} = \mathsf{MSIG.KGen}$, $\mathbf{N}.\mathsf{KGen} = \mathsf{PKE.KGen}$

On hiding the OTT signature. It is somewhat surprising that the one-time signature σ_{OTT} is not revealed, given its one-time nature. It is instead kept hidden and proven in zero-knowledge via NIZK_2. This is needed for UNLINK. Concretely, revealing σ_{OTT} would allow $\mathcal{S}_{\mathsf{OTT}}/\mathcal{G}$ collusion to test candidate nyms by checking $\mathsf{MSIG.Vf}(pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}}, nym), \sigma_{\mathsf{OTT}}) = 1$, thereby leaking linkage information across executions. This is mostly an artifact of adopting the blind signature style from the Apple construction, an alternative would be to introduce a dedicated building block combining partially blind issuance with partially blind and unlinkable presentations. This would allow revealing σ_{OTT} while shifting consistency checks to a more lightweight NIZK. Such a redesign would increase the burden for the generic construction, but could yield a more efficient instantiation.

5.3 Security Analysis of AC-PORTA

AC-PORTA achieves all six PORTA properties. All theorems assume correctness of the underlying building blocks.

Theorem 5. *AC-PORTA is* E-BLIND*-secure if* SIG *is* WBLIND *and* $\mathsf{NIZK}_1, \mathsf{NIZK}_2$ *are zero-knowledge.*

Proof. Enrollment is run as the blind issuance of SIG, so weak blindness directly hides the link between an enrollment session and the resulting pseudonym. Every later oracle uses σ_{E} only as a witness inside NIZK_1 and NIZK_2. The reduction therefore first replaces all honest proofs by simulated ones, at the cost of the zero-knowledge advantages of both proof systems. It then embeds the two enrollment sessions into the weak-blindness experiment

of SIG, forwarding the adversarial key, requests, and signing responses, including joint finalization failure. Although this experiment withholds the signatures, no remaining oracle needs them. Token requests and finalizations use the known pseudonyms and fresh local token state, $\mathcal{O}_{\mathsf{LeakOTT}}$ returns these tokens, and $\mathcal{O}_{\mathsf{Present}}$ encrypts the known pseudonym and simulates its proof. A distinguisher in the simulated experiment thus breaks weak blindness, also for adversarial keys of $\mathcal{S}_{\mathsf{OTT}}$ and $\mathcal{N}$. □

Theorem 6. *AC-PORTA is* UNLINK*-secure if* MSIG *is* WBLIND*,* NIZK_2 *is online simulation-extractable and zero-knowledge, and* PKE *is* IND-CPA*-secure.*

Proof sketch. A presentation has the form $((c, \tau'), m_{\mathsf{OTT}})$ with $c = \mathsf{Enc}(pk_{\mathsf{N}}, nym_b; v)$. The three components separately reveal nothing about nym_b. The message m_{OTT} is sampled uniformly, and its link to the issuance session is hidden by weak blindness of MSIG, as σ_{OTT} is never revealed. The proof τ' leaks nothing by zero-knowledge of NIZK_2, and c hides nym_b by IND-CPA of PKE. The subtle point is that IND-CPA suffices despite the presence of $\mathcal{O}_{\mathsf{Open}}$, as the reduction never decrypts. It records the simulated statement–proof pairs of honest presentations and answers $\mathcal{O}_{\mathsf{Open}}$ on an exact replay of such a pair from this record. For every other accepting presentation, it extracts the pseudonym from τ' by online simulation-extractability, which also covers new proofs for previously simulated statements. As τ' proves that c encrypts the extracted pseudonym, this equals the real opening by correctness. The challenge is excluded from $\mathcal{O}_{\mathsf{Open}}$. The full proof is in Sec. F.1.

Theorem 7. *AC-PORTA is* RL-UNF*-secure if* SIG *and* MSIG *are* OMUF*, and* $\mathsf{NIZK}_1, \mathsf{NIZK}_2$ *are online simulation-extractable.*

Proof. To win, the adversary must either present more distinct pseudonyms to $\mathcal{S}_{\mathsf{OTT}}$ than $\mathcal{S}_{\mathsf{E}}$ has enrolled, or provide more than q_{OTT} valid presentations. In both cases, the reduction forwards the blind signing queries of one scheme to its OMUF challenger and generates the other key itself, while the Node key plays no role. In the first case, online extraction from NIZK_1 yields an enrollment signature on the claimed pseudonym of each accepted OTT request, so $|Q_{nym}| > q_{\mathsf{E}}$ gives a one-more forgery of SIG. In the second case, extraction from each accepting NIZK_2 proof yields a valid MSIG-signature on $(m_{\mathsf{OTT}_i}, nym_i)$. As one-more unforgeability of MSIG is counted per pseudonym, let $q(x)$ count the signing queries with pseudonym x and $t(x)$ the extracted signatures with $nym_i = x$. Distinct one-time messages imply distinct first components for each x. Since $\sum_x t(x) = t > q_{\mathsf{OTT}} = \sum_x q(x)$, some x has $t(x) > q(x)$, which is a one-more forgery of MSIG (possibly with $q(x) = 0$). A union bound over both cases and the extraction errors proves the claim. □

Theorem 8. *AC-PORTA is* BIND*-secure if* SIG *is* UNF-CMA *and* MSIG *is* OMUF*, and* NIZK_2 *is both zero-knowledge and online simulation-extractable.*

Proof sketch. Each OTT is signed by $\mathcal{S}_{\mathsf{OTT}}$ on (m_{OTT}, nym), so a presentation commits to the pseudonym under which the OTT was issued. By simulation-extractability of NIZK_2, each accepted presentation yields an MSIG-signature on $(m_{\mathsf{OTT}}, nym^*)$ and an SIG-signature on nym^*. If nym^* is the pseudonym of a corrupt user, exceeding q_{OTT} accepted presentations is a one-more forgery of MSIG, counted per pseudonym as for RL-UNF. If nym^* belongs to an honest user or was never enrolled, the corresponding σ_{E} has never been exposed, as honest users only use it inside simulated proofs. The extracted SIG-signature is then an UNF-CMA forgery. The full proof is given in Sec. F.2.

Theorem 9. *AC-PORTA is* TRACE*-secure if* MSIG *is* OMUF *and* NIZK_2 *is online simulation-extractable.*

Proof. A winning presentation is accepted by $\mathcal{G}$ and opens either to $\perp$ or to a value not in Q_{nym}. The reduction embeds the honest MSIG key, forwards every accepted batch to its signing oracle with the requested pseudonym as plaintext component, and records $\mathsf{f}(nym)$ in

Q_{nym}. By online simulation-extractability, it extracts from the final NIZK_2 proof a witness $(nym^*, v^*, \sigma^*_{\mathsf{E}}, \sigma^*_{\mathsf{OTT}})$, where σ^*_{OTT} verifies on $(m^*_{\mathsf{OTT}}, nym^*)$ and $c^* = \mathsf{Enc}(pk_{\mathsf{N}}, nym^*; v^*)$. The $\perp$ case cannot arise, as correctness of PKE forces opening to return $\mathsf{f}(nym^*)$. Hence $\mathsf{f}(nym^*) \notin Q_{nym}$, i.e., no signing query used nym^*, and the extracted signature is a one-more forgery of MSIG with zero queries for nym^*. This holds even for an adversarial enrollment key and adversarial users, and bounds the winning probability by the extraction error plus the OMUF advantage. □

Theorem 10. *AC-PORTA is* NFRAME*-secure if* SIG *is* UNF-CMA*,* NIZK_2 *is online simulation-extractable, and both* $\mathsf{NIZK}_1, \mathsf{NIZK}_2$ *are zero-knowledge.*

Proof sketch. A winning presentation opens to an honest user's pseudonym and, by simulation-extractability of NIZK_2, contains a valid SIG-signature on the pseudonym nym^* that its ciphertext encrypts. By correctness, opening returns $\mathsf{f}(nym^*)$, and by injectivity of f, nym^* is the honest user's pseudonym. Honest users' σ_{E} is never exposed to the adversary, as it is only used inside NIZK_1 and NIZK_2, which are simulated via zero-knowledge. Unless an honest and a corrupt user enrolled with the same pseudonym, which only happens with negligible probability, the extracted signature on this never-signed pseudonym is an UNF-CMA forgery on SIG. Replays of honest presentations are excluded via Q_τ, and simulation-extractability also covers new proofs of previously simulated statements. The full proof is given in Sec. F.3.

5.4 Instantiation and Comparison

For instantiating AC-PORTA, we propose BBS signatures [BBS04, TZ23, LKWL26] over BLS12-381 for both SIG and MSIG, as BBS natively signs message vectors, supports efficient Schnorr proofs of possession, and admits (partially) blind issuance. BBS is secure under the q-SDH assumption [TZ23]. For PKE, we use ElGamal in $\mathbb{G}_1$ of BLS12-381, which is IND-CPA-secure under SXDH [BGdMM05]. NIZK_1 and NIZK_2 are Fiat–Shamir-transformed Schnorr proofs, which are online simulation-extractable for the required relations in the algebraic group model and random oracle model (see [Orr25] and App. G). The concrete instantiation is given in App. B. For Apple-PORTA, Apple fixes $\mathsf{BS}_1, \mathsf{BS}_2$ to RSA-2048 blind signatures [DJW23], PKE to DH-based HPKE [BBLW22], and H to SHA-256. Neither protocol provides post-quantum privacy, as both rely on DL-based encryption.

Performance. We benchmarked our implementations of Apple-PORTA and AC-PORTA[5] on an AMD Ryzen AI 7 350 (8 cores, 2 GHz, 32 GB RAM). The presentation round trip costs roughly $0.05 + 0.02 \approx 0.07$ ms for Apple-PORTA versus $4.23 + 6.50 \approx 11$ ms for AC-PORTA, with the gap dominated by $\mathcal{G}$ verifying the NIZK_2 proof. Presentation sizes are comparable ($\approx$ 669 B for Apple-PORTA, $\approx$ 736 B for AC-PORTA). All frequent AC-PORTA operations run below 7 ms per party, which we consider practical. Detailed numbers, including the more expensive but amortized token issuance, are reported in App. B. The additional computation mainly falls on $\mathcal{G}$, where verification can be distributed across horizontally scaled replicas. This increases the Gateway capacity required for a given request rate, but ensures that requests with mismatched OTT–credential pairs are rejected before they reach an attested inference Node. In Apple-PORTA, such requests reach $\mathcal{N}$ and must be decrypted and checked there. In AC-PORTA, $\mathcal{G}$ already verifies the binding, so routine requests do not require opening at $\mathcal{N}$. The measured opening cost of 6.54 ms includes re-running the Gateway verification, but as opening is only needed for abusive requests, this cost does not affect routine requests.

[5] See https://anonymous.4open.science/r/porta-impl-59B1/ for detailed numbers of computation, communication, network, and memory cost.

Alternative design choices. Two alternative design choices for AC-PORTA are worth mentioning. First, token issuance could be based on the anonymous rate-limited credentials (ARC) being standardized in IETF Privacy Pass [DIW24, YWFH26], which fold rate limiting into the token itself. We adopted batch issuance for simplicity and to mirror Apple's design. Second, AC-PORTA could use keyed-verification anonymous credentials (KVAC) [CMZ14, CDVW24, Orr25] to avoid pairings, which may appear natural for a single-operator deployment like Apple's. However, KVAC replaces public verification by verification with the issuer's secret key. $\mathcal{G}$ would then hold the keys of $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$, and a compromised Gateway could issue credentials and tokens itself, removing the separation between issuing and verifying parties that our model relies on.

6 Conclusion

We abstracted Apple's PCC authentication as PORTA, a primitive for rate-limited anonymous authentication with pseudonymous opening, and gave precise definitions of its security and privacy goals. Beyond PCC, PORTA can be seen as an extension of the Privacy Pass architecture that makes the attester role explicit and adds accountable opening, which may also be useful for abuse mitigation in other anonymous authentication systems, such as the recently proposed PACT and MoLE [Jac26, SJM26]. Our main result is that Apple's protocol achieves none of the three accountability properties, even with the Node-side check enabled. As this mechanism is not yet active, the deployed system is not affected today, but the flaws should be addressed before abuse mitigation is enabled. In addition, Apple's salted, i.e., probabilistic, RSA-PSS variant does not provide full presentation unlinkability. A deterministic variant closes this gap without additional online cost, provided that $\mathcal{S}_{\mathsf{OTT}}$ also proves that its signing key is well-formed. We have disclosed our findings and possible mitigations to Apple. AC-PORTA shows that all six properties are achievable within the same architecture. Interesting follow-ups are lighter-weight building blocks, in particular for the one-time tokens, which currently require a full anonymous credential, and a post-quantum instantiation, as both Apple-PORTA and AC-PORTA rely on DL-based assumptions for privacy.

Acknowledgements

Andrey Sidorenko was supported by the HPI Research School on Systems Design.

We thank the students Paulin Deupmann and Margarete Dippel for contributions to early stages of this work during a Master's project at the Hasso Plattner Institute. We also thank Jiska Classen and Yannik Dittmar, who provided insights on applied security aspects.

References

[App24a] Apple Inc. Apple security - private cloud compute repository, 2024. Accessed: 2026-01-29. URL: https://github.com/apple/security-pcc/blob/main/CloudBoard/Sources/CloudBoardJobHelperCore/Auth/TokenGrantingTokenValidator.swift#L89.

[App24b] Apple Inc. Apple security - private cloud compute repository, 2024. Accessed: 2025-05-01. URL: https://github.com/apple/security-pcc.

[App24c] Apple Inc. Private cloud compute: Private cloud compute, 2024. URL: https://security.apple.com/documentation/private-cloud-compute/.

[BBLW22] R. Barnes, K. Bhargavan, B. Lipp, and C. Wood. Hybrid Public Key Encryption. RFC 9180 (Informational), February 2022. URL: https://www.rfc-editor.org/rfc/rfc9180.txt, doi:10.17487/RFC9180.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_3.

[BCD+17] Michael Till Beck, Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 17: 22nd Australasian Conference on Information Security and Privacy, Part I*, volume 10342 of *Lecture Notes in Computer Science*, pages 437–452. Springer, Cham, July 2017. doi:10.1007/978-3-319-60055-0_23.

[BGdMM05] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. URL: https://eprint.iacr.org/2005/417.

[BLOR22] Fabrice Benhamouda, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Publicly verifiable anonymous tokens with private metadata bit. Cryptology ePrint Archive, Report 2022/004, 2022. URL: https://eprint.iacr.org/2022/004.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, Berlin, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_38.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, Berlin, Heidelberg, February 2005. doi:10.1007/978-3-540-30574-3_11.

[CDH23a] Hien Chu, Khue Do, and Lucjan Hanzlik. On the security of rate-limited privacy pass. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 2871–2885. ACM Press, November 2023. doi:10.1145/3576915.3616619.

[CDH23b] Hien Chu, Khue Do, and Lucjan Hanzlik. On the security of rate-limited privacy pass. Cryptology ePrint Archive, Report 2023/1805, 2023. URL: https://eprint.iacr.org/2023/1805.

[CDV23] Melissa Chase, F. Betül Durak, and Serge Vaudenay. Anonymous tokens with stronger metadata bit hiding from algebraic MACs. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 418–449. Springer, Cham, August 2023. doi:10.1007/978-3-031-38545-2_14.

[CDVW24] S. Celi, A. Davidson, S. Valdez, and C. A. Wood. Privacy Pass Issuance Protocols. RFC 9578 (Proposed Standard), June 2024. URL: https://www.rfc-editor.org/rfc/rfc9578.txt, doi:10.17487/RFC9578.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, Berlin, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_7.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_4.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 1205–1216. ACM Press, November 2014. doi:10.1145/2660267.2660328.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018(3):164–180, July 2018. doi:10.1515/popets-2018-0026.

[DIW24] A. Davidson, J. Iyengar, and C. A. Wood. The Privacy Pass Architecture. RFC 9576 (Informational), June 2024. URL: https://www.rfc-editor.org/rfc/rfc9576.txt, doi:10.17487/RFC9576.

[DJW23] F. Denis, F. Jacobs, and C. A. Wood. RSA Blind Signatures. RFC 9474 (Informational), October 2023. URL: https://www.rfc-editor.org/rfc/rfc9474.txt, doi:10.17487/RFC9474.

[DMTV24] F. Betül Durak, Laurane Marco, Abdullah Talayhan, and Serge Vaudenay. Non-transferable anonymous tokens by secret binding. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 2460–2474. ACM Press, October 2024. doi:10.1145/3658644.3670338.

[DSV+26] Yannik Dittmar, Marvin Jerome Stephan, Thomas Völkl, Matthias Hollick, and Jiska Classen. Unlocking apple's private cloud compute: An analysis of privacy-preserving artificial intelligence. In *Proceedings of the 19th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '26, page 63–73, New York, NY, USA, 2026. Association for Computing Machinery. doi:10.1145/3765613.3811691.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, Berlin, Heidelberg, December 2012. doi:10.1007/978-3-642-34931-7_5.

[GGGG26] Google Platforms and Devices, Google DeepMind, Google Core, and Google Cloud. Google private AI compute: Enabled with secure server-side memory. Technical brief, Google, September 2026. URL: https://services.google.com/fh/files/misc/private_ai_compute_technical_brief.pdf.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, Berlin, Heidelberg, December 2006. doi:10.1007/11935230_29.

[HIP+24] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Rate-Limited Token Issuance Protocol. Internet-Draft draft-ietf-privacypass-rate-limit-tokens-06, Internet Engineering Task Force, April 2024. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-privacypass-rate-limit-tokens/06/.

[HLÖ25] Konrad Hanff, Anja Lehmann, and Cavit Özbay. Security analysis of privately verifiable privacy pass. In *ACM CCS 2025: 32nd Conference on Computer and Communications Security*, pages 2922–2936. ACM Press, November 2025. doi:10.1145/3719027.3765172.

[Inc24a] Apple Inc. Pcc documentation: Attested request handling, 2024. URL: https://security.apple.com/documentation/private-cloud-compute/requesthandling.

[Inc24b] Apple Inc. Pcc documentation: Request flow, 2024. URL: https://security.apple.com/documentation/private-cloud-compute/requestflow.

[Inc24c] Apple Inc. Pcc documentation: Stateless computation and enforceable guarantees, 2024. URL: https://security.apple.com/documentation/private-cloud-compute/statelessandenforcable.

[Jac26] Dennis Jackson. PACT: Anonymous credentials for the web, 2026. URL: https://hacks.mozilla.org/2026/06/pact-anonymous-credentials-for-the-web/.

[JKTS07] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous IP-address blocking. In Nikita Borisov and Philippe Golle, editors, *PET 2007: 7th International Symposium on Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 113–133. Springer, Berlin, Heidelberg, June 2007. doi:10.1007/978-3-540-75551-7_8.

[KL25] David Kretzler and Yong Li. No honor among crooks: Non-transferable anonymous tokens from betrayability. Cryptology ePrint Archive, Report 2025/1795, 2025. URL: https://eprint.iacr.org/2025/1795.

[KLOR20] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 308–336. Springer, Cham, August 2020. doi:10.1007/978-3-030-56784-2_11.

[KP98] Joe Kilian and Erez Petrank. Identity escrow. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. Springer, Berlin, Heidelberg, August 1998. doi:10.1007/BFb0055727.

[LH12] Peter Lofgren and Nicholas Hopper. BNymble: More anonymous blacklisting at almost no cost (a short paper). In George Danezis, editor, *FC 2011: 15th International Conference on Financial Cryptography and Data Security*, volume 7035 of *Lecture Notes in Computer Science*, pages 268–275. Springer, Berlin, Heidelberg, February / March 2012. doi:10.1007/978-3-642-27576-0_22.

[LKWL26] Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Technical report, 2026. URL: https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/10/.

[Met26] Meta. Private processing for WhatsApp overview. Technical white paper and security guide, Meta, October 2026. Version 3, updated October 20, 2026. URL: https://ai.meta.com/static-resource/private-processing-technical-whitepaper.

[Orr25] Michele Orrù. Revisiting keyed-verification anonymous credentials. In *ACM CCS 2025: 32nd Conference on Computer and Communications Security*, pages 1188–1199. ACM Press, November 2025. doi:10.1145/3719027.3765163.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Berlin, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_9.

[SJM26] Samuel Schlesinger, Dennis Jackson, and Thibault Meunier. MoLE protocols. Internet-Draft draft-jms-mole-protocols-latest, Internet Engineering Task Force, 2026. Work in Progress. URL: https://moderation-of-unlinkable-endorsements.github.io/internet-drafts/draft-jms-mole-protocols.html.

[SM26] S. Schlesinger and T. Meunier. Privacy pass issuance protocol for anonymous credit tokens, 2026. Accessed: 2026-04-27. URL: https://www.ietf.org/archive/id/draft-schlesinger-privacypass-act-01.html.

[SS22] Tjerand Silde and Martin Strand. Anonymous tokens with public metadata and applications to private contact tracing. In Ittay Eyal and Juan A. Garay, editors, *FC 2022: 26th International Conference on Financial Cryptography and Data Security*, volume 13411 of *Lecture Notes in Computer Science*, pages 179–199. Springer, Cham, May 2022. doi:10.1007/978-3-031-18283-9_9.

[TAKS07] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007: 14th Conference on Computer and Communications Security*, pages 72–81. ACM Press, October 2007. doi:10.1145/1315245.1315256.

[TAKS08] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. PEREA: towards practical TTP-free revocation in anonymous authentication. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 333–344. ACM Press, October 2008. doi:10.1145/1455770.1455813.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 691–721. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_24.

[YWFH26] C. Yun, C. A. Wood, and A. Faz-Hernandez. Anonymous rate-limited credentials cryptography, 2026. Accessed: 2026-04-27. URL: https://www.ietf.org/archive/id/draft-ietf-privacypass-arc-crypto-01.html.

[ZZWL26] Haoling Zhou, Shixuan Zhao, Chao Wang, and Zhiqiang Lin. Too private to tell: Practical token theft attacks on apple intelligence, 2026. URL: https://arxiv.org/abs/2604.15637, arXiv:2604.15637.

Ethical Considerations

This work analyzes a deployed system and identifies shortcomings in its security guarantees, so we treated it as a potential vulnerability. We disclosed our findings to Apple through Apple Security Research in June 2026. Apple responded, stating that they do not classify our results as a vulnerability, as the abuse mechanism is not in use yet. However, our results may be taken into consideration for the future design of abuse mitigation.

Our analysis shows that Apple's deployed PCC protocol does not satisfy the abuse-mitigation goals we formalize, but we judge the real-world risk to be low, as the affected mechanism is not yet active in the deployed system. As this work involves no human subjects or personal data, it did not require review by an institutional ethics board.

AI Use

The authors accept responsibility for the veracity and correctness of all material in this paper, including any AI-assisted material.

The model, definitions, analysis, and construction are the authors' work, as is an initial version of the manuscript including proof sketches. We made substantial use of AI tools (ChatGPT Edu, Claude, and Copilot) in writing the present version, namely to rewrite and restructure text from our drafts, to work out the full proofs from our proof sketches, and for proofreading and consistency checks. The proof-of-concept implementation was generated entirely using AI and checked by the authors.

A Additional Definitions

A.1 Full Syntax for SIG, MSIG, and PKE

Weakly blind signatures. A scheme SIG with message space $\mathcal{M}$ consists of the following algorithms:

$\mathsf{KGen}(1^\lambda) \to (sk, pk)$**:** generates a signing key and a verification key.

$\mathsf{Blind}(pk, m) \to (\overline{m}, r)$**:** blinds a message $m \in \mathcal{M}$ and outputs the blinded message and private blinding state.

$\mathsf{BSign}(sk, \overline{m}) \to \overline{\sigma}$**:** signs the blinded message.

$\mathsf{Unblind}(pk, \overline{\sigma}, r) \to \sigma$**:** uses the blinding state to obtain a signature on m.

$\mathsf{Vf}(pk, m, \sigma) \to \{0, 1\}$**:** outputs 1 if σ is a valid signature on m under pk, and 0 otherwise.

The syntax is identical to that of BS, but we require only weak blindness (Def. 5). The plaintext signing oracle used for UNF-CMA executes blinding, signing, and unblinding honestly on its input message (Def. 6).

Partially blind multi-message signatures. A scheme MSIG signs an ordered pair $(m_0, m_1) \in \mathcal{M}_0 \times \mathcal{M}_1$, with m_0 blinded and m_1 supplied to the signer in the clear. It consists of the following algorithms:

$\mathsf{KGen}(1^\lambda) \to (sk, pk)$**:** generates a signing key and a verification key.

$\mathsf{Blind}(pk, m_0) \to (\overline{m}_0, r)$**:** blinds the first message component and outputs the blinded component and private blinding state.

$\mathsf{BSign}(sk, (\overline{m}_0, m_1)) \to \overline{\sigma}$**:** signs the blinded first component together with the plaintext second component.

$\mathsf{Unblind}(pk, \overline{\sigma}, r) \to \sigma$**:** uses the blinding state to obtain a signature on (m_0, m_1).

$\mathsf{Vf}(pk, (m_0, m_1), \sigma) \to \{0, 1\}$**:** outputs 1 if σ is a valid signature on the ordered pair under pk, and 0 otherwise.

Weak blindness protects only m_0, while m_1 is visible to the signer (Def. 5). One-more unforgeability is counted separately for each plaintext value m_1: if $q(m_1)$ signing-oracle queries use this value, producing valid signatures on $q(m_1)+1$ pairs (m_0, m_1) with distinct first m_0 constitutes a forgery (Def. 8). A signature for a fresh m_1 is the case $q(m_1) = 0$.

Correctness. Each signature scheme $\Sigma \in \{\mathsf{BS}, \mathsf{SIG}, \mathsf{MSIG}\}$ is perfectly correct if, for every honestly generated key pair and every admissible message $\mathbf{m}$ (an ordered pair (m_0, m_1) for MSIG), honest blinding, blind signing, and unblinding produce a signature on $\mathbf{m}$ that verifies with probability 1 over all issuance randomness.

Public-key encryption with an encoding. A scheme PKE has a message space $\mathcal{M}_{\mathsf{PKE}}$, an encoded message space $\mathcal{M}'_{\mathsf{PKE}}$, and a ciphertext space $\mathbf{C}$. It is parameterized by a public, efficiently computable, injective function $\mathsf{f} : \mathcal{M}_{\mathsf{PKE}} \to \mathcal{M}'_{\mathsf{PKE}}$, whose inverse need not be efficient (see also Sec. 2.3). The algorithms are:

$\mathsf{KGen}(1^\lambda) \to (sk, pk)$**:** generates a decryption key and an encryption key.

$\mathsf{Enc}(pk, m; v) \to c$**:** encrypts $m \in \mathcal{M}_{\mathsf{PKE}}$ using randomness v, producing $c \in \mathbf{C}$. The algorithm is deterministic for fixed v; omitting v denotes sampling it uniformly.

$\mathsf{Dec}(sk, c) \to \mathcal{M}'_{\mathsf{PKE}} \cup \{\perp\}$**:** deterministically returns an encoded message or failure $\perp$.

Perfect correctness requires that, for every security parameter, every $m \in \mathcal{M}_{\mathsf{PKE}}$, and every v, $\Pr\left[\ \mathsf{Dec}(sk, \mathsf{Enc}(pk, m; v)) = \mathsf{f}(m) \quad (sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)\ \right] = 1$. Taking f to be the identity produces standard public-key encryption. In AC-PORTA, we use $\mathcal{M}_{\mathsf{PKE}} = \mathsf{NYM}$ and $\mathcal{M}'_{\mathsf{PKE}} = \mathsf{NYM}'$: recovering the unique value $\mathsf{f}(m)$ suffices for deny-list checks, and permits ElGamal encryption of scalar pseudonyms through their group encodings (Sec. 5.1).

The IND-CPA and IND-CCA experiments in Fig. 6 apply to this syntax: the adversary chooses admissible, equal-length messages $m_0, m_1 \in \mathcal{M}_{\mathsf{PKE}}$, and the challenge is $\mathsf{Enc}(pk, m_b)$. In the IND-CCA experiment, the decryption oracle returns values in $\mathcal{M}'_{\mathsf{PKE}} \cup \{\perp\}$ and cannot be queried on the challenge ciphertext; the IND-CPA experiment provides no decryption oracle. Neither experiment requires computing f^{-1}.

A.2 Standard Security Definitions

This section provides complementary definitions of standard notions. The experiments are defined in Fig. 6.

Definition 4. We call a BS scheme Π blind if there exists a negligible function $\mathrm{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$ with:

$$\left|\Pr\left[\mathrm{Exp}^{\text{BLIND-0}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{BLIND-1}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

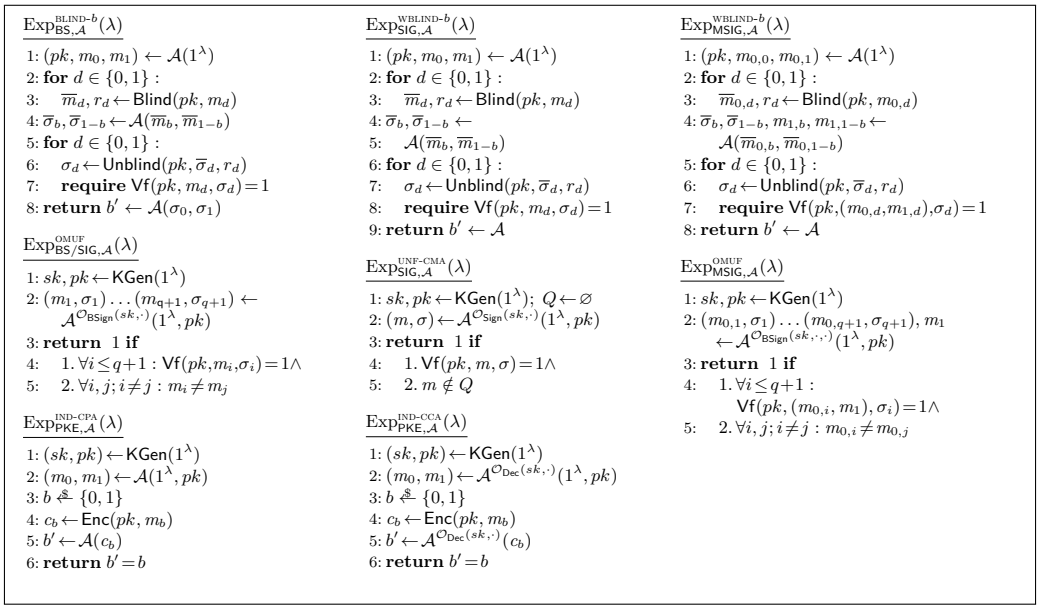

Figure 6: Security experiments for SIG/MSIG weak blindness, BS blindness, BS/SIG/MSIG (one-more-)unforgeability, and PKE IND-CPA/IND-CCA.

Weak blindness is defined using experiments similar to the above blindness definition – the core difference is that the adversary does not learn the unblinded signatures σ_b. Accordingly, blindness implies weak blindness. In the case of a multi-message scheme MSIG, (weak) partial blindness is modeled by the adversary providing the plaintext inputs for which the signature is to be valid.

Definition 5. We call a SIG or MSIG scheme Π weakly blind if there exists a negligible function $\mathrm{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$ with:

$$|\Pr\left[\mathrm{Exp}^{\text{WBLIND-0}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{WBLIND-1}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]| \le \mathrm{negl}(\lambda).$$

Definition 6. We call a BS or SIG scheme Π UNF-CMA-secure if there exists a negligible function $\mathrm{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$ with: $\Pr\left[\mathrm{Exp}^{\text{UNF-CMA}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \le \mathrm{negl}(\lambda)$. The oracle $\mathcal{O}_{\mathsf{Sign}}$ adds each input message m to Q. For schemes with blind issuance, $\mathcal{O}_{\mathsf{Sign}}$ performs: $\overline{m}, r \leftarrow \mathsf{Blind}(pk, m)$, then $\overline{\sigma} \leftarrow \mathsf{BSign}(sk, \overline{m})$, and finally returns σ from $\sigma \leftarrow \mathsf{Unblind}(pk, \overline{\sigma}, r)$. For BS, the experiment and the oracle use the corresponding algorithms of BS.

Definition 7. We call a PKE scheme Π x-secure for $\mathsf{x} \in \{\text{IND-CPA}, \text{IND-CCA}\}$ if there exists a negligible function $\mathrm{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$ with:

$$\Pr\left[\mathrm{Exp}^{\mathsf{x}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \le \frac{1}{2} + \mathrm{negl}(\lambda).$$

The oracle $\mathcal{O}_{\mathsf{Dec}}$ may not be queried with c_b. For public-key encryption with an encoding f (Sec. A.1), the experiments are unchanged, with $m_0, m_1 \in \mathcal{M}_{\mathsf{PKE}}$ and $\mathcal{O}_{\mathsf{Dec}}$ returning values in $\mathcal{M}'_{\mathsf{PKE}} \cup \{\perp\}$.

Definition 8. We call a BS, SIG, or MSIG scheme Π one-more-unforgeable if there exists a negligible function $\mathrm{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$, which makes at most q calls to $\mathcal{O}_{\mathsf{BSign}}$ (for MSIG, q denotes the number of $\mathcal{O}_{\mathsf{BSign}}$ queries with input $(\cdot, m_1)$), with:

$$\Pr\left[\mathrm{Exp}^{\text{OMUF}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \le \mathrm{negl}(\lambda).$$

Definition 9. We call a BS scheme Π unique [BCD+17] if for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|c} \sigma_0 \neq \sigma_1 \wedge & (pk, m, \sigma_0, \sigma_1) \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{Vf}(pk, m, \sigma_0) = \mathsf{Vf}(pk, m, \sigma_1) = 1 & \end{array}\right] \leq \mathrm{negl}(\lambda).$$

Definition 10. A hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is computationally input hiding, if for all messages $m_0, m_1 \in \mathcal{M}$ and a uniformly random secret salt $s \xleftarrow{\$} \{0,1\}^\lambda$, the distributions $\{h, m_0, m_1 \mid h \leftarrow \mathsf{H}(m_0, s)\} \approx \{h, m_0, m_1 \mid h \leftarrow \mathsf{H}(m_1, s)\}$ are computationally indistinguishable, where $\mathsf{H}(x, y)$ is the concatenation $\mathsf{H}(x||y)$.

Definition 11 (NIZK security)**.** We define a Non-Interactive Zero-Knowledge (NIZK) proof system as a tuple of efficient algorithms $\mathsf{NIZK} = (\mathsf{P}, \mathsf{Vf})$. The prover holds a public statement x, which is known to both prover and verifier, and a secret witness w for x. The prover runs $\mathsf{P}(x, w)$ to generate a proof π. The verifier runs $\mathsf{Vf}(x, \pi)$ and outputs a bit, with 1 signifying acceptance. A NIZK proof system for an NP relation $\mathcal{R}$ satisfies the following properties for all $\lambda \in \mathbb{N}$:

Completeness: For all $(x, w) \in \mathcal{R}$: $\Pr\left[\ 1 \leftarrow \mathsf{NIZK.Vf}(x, \pi) \quad \pi \leftarrow \mathsf{NIZK.P}(x, w)\ \right] = 1$

Non-interactive zero-knowledge: Our definition is similar to [FKMV12]. In the random-oracle model, the stateful pair $(\mathcal{S}_1, \mathcal{S}_2')$ simulates random-oracle queries ($\mathcal{S}_1$) and proofs ($\mathcal{S}_2'$). The oracles share a state. We write H for the random oracle. For every PPT distinguisher $\mathcal{D}$ that adaptively requests polynomially many proofs for pairs $(x, w) \in \mathcal{R}$, its joint view with the real random oracle and honest proofs $\mathsf{P}(x, w)$ is computationally indistinguishable from its view with $\mathcal{S}_1$ and $\mathcal{S}_2'$. Thus, there exists a negligible function $\mathrm{negl}(\lambda)$ such that $\left|\Pr\left[1 \leftarrow \mathcal{D}^{\mathsf{H},\mathsf{P}^{\mathsf{H}}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathcal{D}^{\mathcal{S}_1,\mathcal{S}_2'}(1^\lambda)\right]\right| \leq \mathrm{negl}(\lambda)$.

Online simulation-extractability: We use full simulation extractability [FKMV12] for the same joint simulator. In the extraction experiment, $\mathcal{S}_2'$ may be queried adaptively on arbitrary statements, including false ones. Let T record every statement–proof pair that it returns. There exists a PPT extractor $\mathcal{E}$ such that, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|c} (x^*, \pi^*) \notin T & (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2'}(1^\lambda) \\ \wedge\ \mathsf{Vf}^{\mathcal{S}_1}(x^*, \pi^*) = 1 & w^* \leftarrow \mathcal{E}(st, x^*, \pi^*) \\ \wedge\ (x^*, w^*) \notin \mathcal{R} & \end{array}\right] \leq \mathrm{negl}(\lambda).$$

The extractor uses the joint simulator's state and the submitted statement–proof pair, without rewinding $\mathcal{A}$. The probability is over the randomness of the simulator, adversary, and extractor. Only exact pairs in T are excluded: a new proof for a previously simulated statement remains subject to extraction.

B Instantiation Details for AC-PORTA

$\underline{\mathsf{NIZK}_1.\mathsf{P}((w_1, w_2, e, r, \{m_{\mathsf{OTT}_i}, r_i\}), (nym, \overline{A}, \overline{B}, \overline{C}, \overline{M_{\mathsf{OTT}}}))}$

1: $\pi' \leftarrow \mathsf{NIZK}_1 \left\{ \begin{pmatrix} w_1, w_2, e, r, \\ \{m_{\mathsf{OTT}_i}, r_i\} \end{pmatrix} : \begin{array}{l} \overline{B} = \overline{C}^{w_2} \cdot \overline{A}^{-e} \wedge D = \overline{C}^{w_1^{-1}} \cdot h^{-r} \\ \bigwedge_{i=1}^{\ell} p^{r_i} \cdot p_1^{m_{\mathsf{OTT}_i}} = \overline{m_{\mathsf{OTT}_i}} \end{array} \right\} \left[\begin{array}{l} \overline{M_{\mathsf{OTT}}}, nym, \\ \overline{A}, \overline{B}, \overline{C} \end{array} \right]$

▷ $\pi' = (k, u_{w_1}, u_{w_2}, u_e, u_r, u_{r_1}, u_{m_{\mathsf{OTT}_1}} \ldots u_{r_\ell}, u_{m_{\mathsf{OTT}_\ell}} \in \mathbb{Z}_q^{5+2\ell})$

2: **return** π'

$\underline{\mathsf{NIZK}_2.\mathsf{P}((w_1, w_2, w_1', w_2', e, r, e', r', nym, v), (\overline{A}, \overline{B}, \overline{C}, \overline{A'}, \overline{B'}, \overline{C'}, m_{\mathsf{OTT}}, c_0, c_1))}$

3: $\tau'' \leftarrow \mathsf{NIZK}_2 \left\{ \begin{pmatrix} w_1, w_2, e, r, \\ w_1', w_2', e', r', \\ nym, v \end{pmatrix} : \begin{array}{l} \overline{B} = \overline{C}^{w_2} \cdot \overline{A}^{-e} \wedge g = \overline{C}^{w_1^{-1}} \cdot h^{-r} \cdot h_0^{-nym} \\ \overline{B'} = \overline{C'}^{w_2'} \cdot \overline{A'}^{-e'} \wedge D' = \overline{C'}^{w_1'^{-1}} \cdot p^{-r'} \cdot p_0^{-nym} \\ c_0 = t^v \wedge c_1 = s^{nym} \cdot Z^v \end{array} \right\} \left[\begin{array}{l} c, m_{\mathsf{OTT}} \\ \overline{A}, \overline{B}, \overline{C}, \\ \overline{A'}, \overline{B'}, \overline{C'} \end{array} \right]$

▷ $\tau'' = (k, u_{w_1}, u_{w_2}, u_{w_1'}, u_{w_2'}, u_e, u_r, u_{e'}, u_{r'}, u_{nym}, u_v) \in \mathbb{Z}_q^{11}$

4: **return** τ''

Figure 7: NIZK statements for the BBS/ElGamal AC-PORTA instantiation.

Pairings. We denote a pairing as a function $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$ for a prime q. We use multiplicative notation for all three groups. To differentiate between $\mathbb{G}_1$ and $\mathbb{G}_2$, we write elements as $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$. We assume a type-3 pairing, where no efficient isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known.

Schemes and comparison. We propose using the BBS scheme [BBS04, CL04, TZ23] as the underlying construction for both SIG and MSIG. For PKE, ElGamal can be used in $\mathbb{G}_1$, encrypting a pseudonym $nym \in \mathbb{Z}_q$ as the group element $\mathsf{f}(nym) = s^{nym}$ for a public generator $s \in \mathbb{G}_1$. The full construction is given in Fig. 9, with the NIZKs detailed in Fig. 7. See Tab. 2 and Tab. 3 for a comparison of the computation and communication costs of Apple-PORTA and AC-PORTA.

Assumptions. We summarize the assumptions for the combined PORTA security of AC-PORTA. Both SIG and MSIG are weakly blind WBLIND, one-more unforgeable OMUF, and UNF-CMA. Both are based on blind BBS signatures, which are unconditionally blind due to the Pedersen-style blinding [Ped92], which implies WBLIND. Assuming q-SDH, they are unforgeable (UNF-CMA) [TZ23], and in the algebraic group model (AGM) they are also one-more unforgeable [TZ23]. The one-more-unforgeability result extends to MSIG: the decomposition of the blinded message provided by Req allows the reduction to apply standard unforgeability to the component associated with p_0 and one-more unforgeability to the component associated with p_1. ElGamal encryption in $\mathbb{G}_1$ is IND-CPA if the decisional Diffie-Hellman (DDH) assumption holds in $\mathbb{G}_1$, which is implied by SXDH. NIZK_1 and NIZK_2 have to satisfy completeness, zero-knowledge, and online simulation-extractability. Since our proofs are Σ-protocols made non-interactive with the Fiat–Shamir transform using H, they are complete and zero-knowledge in the random oracle model (ROM) [TZ23, FKMV12]. Fiat–Shamir NIZKs are online simulation-extractable in the AGM+ROM if only linear relations are proven, which is the case for our statements (App. G). Recall that the AGM is already required for BBS one-more unforgeability.

Table 2: Full version of the computation times (ms) for Apple-PORTA and AC-PORTA. Bracketed numbers in the AC column use batched verification during OTT finalization.

Batch size	Operation	Apple	AC	[batched AC]	AC / Apple
	U.Req	0.05	0.25		537.55%
	E.Issue	2.07	0.13		6.41%
	U.Fin	0.04	2.05		4804.17%
	U.Present	0.05	4.23		9245.34%
	G.Vf	0.02	6.50		30479.41%
	N.Open	0.08	6.54		8031.72%
	U.ReqBatch	0.05	2.05		4150.55%
$\ell = 1$	**OTT**.IssueBatch	2.08	3.90		187.46%
	U.FinBatch	0.04	2.22	[2.47]	6267.58%
	U.ReqBatch	0.48	6.94		1435.39%
$\ell = 10$	**OTT**.IssueBatch	20.83	9.96		47.82%
	U.FinBatch	0.36	22.13	[8.22]	6154.57%
	U.ReqBatch	4.92	55.09		1120.06%
$\ell = 100$	**OTT**.IssueBatch	203.02	70.15		34.55%
	U.FinBatch	3.78	221.88	[65.67]	5865.79%

Table 3: Communication cost comparison ($\ell = 1$), assuming RSA2048 and HPKE based on AES/P256 for Apple-PORTA and BLS12-381 for AC-PORTA.

	Value	Apple (bytes)	AC (bytes)	AC vs. Apple
Storage cost	nym	32	32	100.00%
	σ_{E}	256	112	43.75%
	m_{OTT}	32	32	100.00%
	σ_{OTT}	256	112	43.75%
Network cost	$\overline{m_{\mathsf{E}}}$	256	48	18.75%
	$\overline{\sigma_{\mathsf{E}}}$	256	80	31.25%
	$\overline{m_{\mathsf{OTT}}}$	256	48	18.75%
	$\overline{\sigma_{\mathsf{OTT}}}$	256	80	31.25%
	π	256	$368\,(304 + 64{\cdot}\ell)$	143.75%
	τ	669	736	110.01%
	c *(part of* τ*)*	413	96	23.24%

Key Generation:
$(sk_{\mathsf{E}}, pk_{\mathsf{E}}) \leftarrow \mathbf{E}.\mathsf{KGen}(1^\lambda)$
$(sk_{\mathsf{OTT}}, pk_{\mathsf{OTT}}) \leftarrow \mathbf{OTT}.\mathsf{KGen}(1^\lambda)$
$(sk_{\mathsf{N}}, pk_{\mathsf{N}}) \leftarrow \mathbf{N}.\mathsf{KGen}(1^\lambda)$

Enrollment:
$(\overline{nym}, aux_{\mathsf{E}}) \leftarrow \mathbf{U}.\mathsf{Req}(pk_{\mathsf{E}}, nym)$
$\overline{\sigma_{\mathsf{E}}} \leftarrow \mathbf{E}.\mathsf{Issue}(sk_{\mathsf{E}}, \overline{nym})$
$\{\sigma_{\mathsf{E}}, \perp\} \leftarrow \mathbf{U}.\mathsf{Fin}(\overline{\sigma_{\mathsf{E}}}, aux_{\mathsf{E}})$

Token Issuance:
$(\overline{M_{\mathsf{OTT}}}, \pi, aux_{\mathsf{OTT}}) \leftarrow \mathbf{U}.\mathsf{ReqBatch}(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}))$
$\{\overline{\Sigma_{\mathsf{OTT}}}, \perp\} \leftarrow \mathbf{OTT}.\mathsf{IssueBatch}(sk_{\mathsf{OTT}}, pk_{\mathsf{E}}, \overline{M_{\mathsf{OTT}}}, nym, \pi)$
$\{(M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}}), \perp\} \leftarrow \mathbf{U}.\mathsf{FinBatch}(\overline{\Sigma_{\mathsf{OTT}}}, aux_{\mathsf{OTT}})$

Redemption and Opening:
$\tau \leftarrow \mathbf{U}.\mathsf{Present}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}), (m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}}))$
$\{0, 1\} \leftarrow \mathbf{G}.\mathsf{Vf}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}})$
$\{\mathsf{f}(nym), \perp\} \leftarrow \mathbf{N}.\mathsf{Open}(sk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}})$

Experiment $\mathrm{Exp}^{Corr}_{\Pi,\ell}(\lambda)$

$(sk_{\mathsf{E}}, pk_{\mathsf{E}}) \leftarrow \mathbf{E}.\mathsf{KGen}(1^\lambda)$
$(sk_{\mathsf{OTT}}, pk_{\mathsf{OTT}}) \leftarrow \mathbf{OTT}.\mathsf{KGen}(1^\lambda)$
$(sk_{\mathsf{N}}, pk_{\mathsf{N}}) \leftarrow \mathbf{N}.\mathsf{KGen}(1^\lambda)$
$nym \xleftarrow{\$} \mathsf{NYM}$
$(\overline{nym}, aux_{\mathsf{E}}) \leftarrow \mathbf{U}.\mathsf{Req}(pk_{\mathsf{E}}, nym)$
$\overline{\sigma_{\mathsf{E}}} \leftarrow \mathbf{E}.\mathsf{Issue}(sk_{\mathsf{E}}, \overline{nym})$
$\sigma_{\mathsf{E}} \leftarrow \mathbf{U}.\mathsf{Fin}(\overline{\sigma_{\mathsf{E}}}, aux_{\mathsf{E}})$
require $\sigma_{\mathsf{E}} \neq \perp$
$(\overline{M_{\mathsf{OTT}}}, \pi, aux_{\mathsf{OTT}}) \leftarrow \mathbf{U}.\mathsf{ReqBatch}(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}))$
$\overline{\Sigma_{\mathsf{OTT}}} \leftarrow \mathbf{OTT}.\mathsf{IssueBatch}(sk_{\mathsf{OTT}}, pk_{\mathsf{E}}, \overline{M_{\mathsf{OTT}}}, nym, \pi)$
$(M_{\mathsf{OTT}}, \Sigma_{\mathsf{OTT}}) \leftarrow \mathbf{U}.\mathsf{FinBatch}(\overline{\Sigma_{\mathsf{OTT}}}, aux_{\mathsf{OTT}})$
require $\Sigma_{\mathsf{OTT}} \neq \perp$
$j \xleftarrow{\$} \{1, \ldots, \ell\}$
$m_{\mathsf{OTT}_j} \leftarrow M_{\mathsf{OTT}}[j]$; $\sigma_{\mathsf{OTT}_j} \leftarrow \Sigma_{\mathsf{OTT}}[j]$
$\tau \leftarrow \mathbf{U}.\mathsf{Present}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, (nym, \sigma_{\mathsf{E}}), (m_{\mathsf{OTT}_j}, \sigma_{\mathsf{OTT}_j}))$
require $\mathbf{G}.\mathsf{Vf}(pk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}_j}) = 1$
$y \leftarrow \mathbf{N}.\mathsf{Open}(sk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}_j})$
require $y = \mathsf{f}(nym)$
return 1

Figure 8: Left: Algorithms and Syntax of PORTA. Right: Correctness for PORTA.

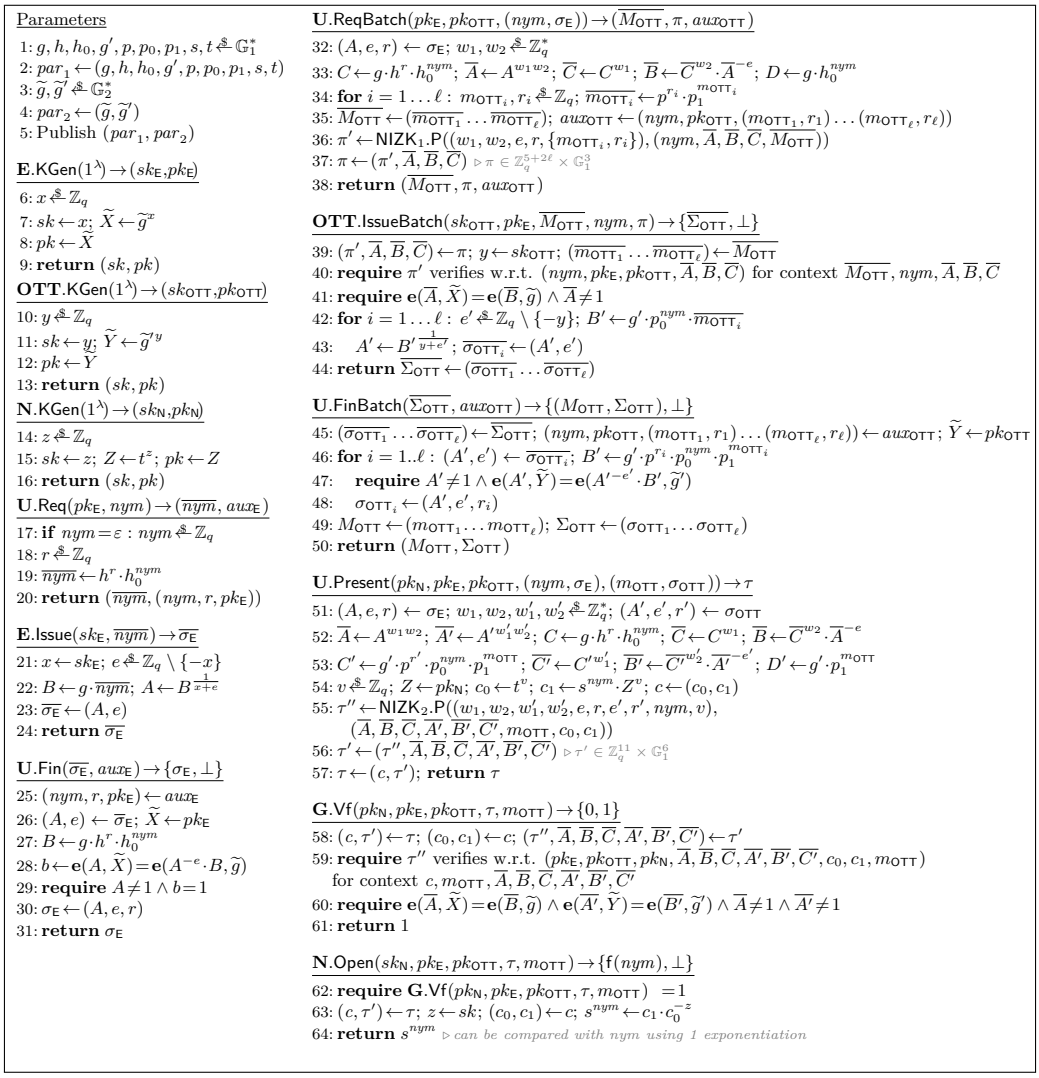

Figure 9: Our AC Instantiation (AC-PORTA) using BBS and ElGamal.

C Syntax and Correctness of PORTA

We provide an overview of all algorithms of our syntax in Fig. 8.

Definition 12 (Correctness)**.** A PORTA scheme is correct if, for all λ and all internal randomness, the game $\mathrm{Exp}^{Corr}_{\Pi,\ell}(\lambda)$ in Fig. 8 returns 1.

D Additional Material for our Security Model

D.1 Challenger State

Tab. 4 lists the variables that the challenger uses to keep state in the security experiments.

D.2 Comparison with Group Signatures

In group signatures, non-frameability and traceability exist as well [BMW03], with non-frameability typically the stronger notion as it additionally allows the group manager/issuer

Table 4: Challenger state during PORTA experiments. All sets and counters are initialized to $\varnothing/0$, respectively.

Var	Description	Var	Description
Q_{HU}	Set of honest users, maps nym to $(nym, \sigma_{\mathsf{E}})$	q_{E}	Counter of enrollments issued by honest $\mathcal{S}_{\mathsf{E}}$
Q_{τ}	Set of honest presentations (τ, m_{OTT})	q_{OTT}	Counter of OTTs issued by honest $\mathcal{S}_{\mathsf{OTT}}$
Q_{nym}	Set of values $\mathsf{f}(nym)$ of all pseudonyms nym seen by honest $\mathcal{S}_{\mathsf{OTT}}$	q_{aux}, $q_{M,\Sigma}$	Per-pseudonym maps of pending and finalized issuance-session counters
Q_{aux}	Map (nym, j) to state for the j-th pending token-issuance session of nym		
$Q_{M,\Sigma}$	Map (nym, j) to the j-th finalized OTT batch of nym		

to be corrupt. In PORTA such an implication does not exist, due to the hierarchical issuance and the role pseudonyms play. First, NFRAME in PORTA is defined w.r.t. nym, which is revealed with every pseudonymous token request to $\mathcal{S}_{\mathsf{OTT}}$; in group signatures that achieve non-frameability the corresponding value is a high-entropy user secret that is never exposed. Consequently, our NFRAME must assume that $\mathcal{S}_{\mathsf{E}}$ – and, more generally, the entire enrollment process – is honest, otherwise an adversary could sign nym of an honest user and frame them at will. Second, TRACE requires expressing what a "valid" pseudonym is, and in PORTA the only party that knows the valid pseudonyms is $\mathcal{S}_{\mathsf{OTT}}$; thus TRACE crucially requires $\mathcal{S}_{\mathsf{OTT}}$ to be honest, while $\mathcal{S}_{\mathsf{E}}$ can be corrupt since the goal is only to prevent untraceable presentations. Overall, TRACE and NFRAME are almost complementary in PORTA and expressed in their strongest individual corruption settings.

E Full Proofs for Apple-PORTA

E.1 Proof of Thm. 2: Presentation Unlinkability

Proof. In the UNLINK game, $\mathcal{A}$ selects two honestly issued OTTs and must distinguish which one underlies the challenge presentation, while having access to $\mathcal{O}_{\mathsf{Open}}$ on every presentation except the challenge itself. In the proof, we hide the challenge bit by successively removing any link to the ciphertext, input to the hash, and the issuance session.

Let Q bound the number of honestly requested OTTs. For every honest presentation, store $T[c] = (m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}}, nym)$, where σ_{OTT} is the raw BS_2 signature. After public verification, answer $\mathcal{O}_{\mathsf{Open}}$ on a recorded c by its stored nym exactly when the entire triple matches, and by $\perp$ otherwise; open every fresh c normally. This is exact: decryption fixes $((nym, \sigma_{\mathsf{E}}), salt)$, hence the deterministic digest, and uniqueness fixes the valid signature on that digest.

Hybrid by hybrid, replace every recorded ciphertext by an encryption of a fixed admissible plaintext D_c. The IND-CCA reduction answers its challenge ciphertext from T and sends every other fresh ciphertext to its decryption oracle. The following standard consequence justifies all collision aborts used below. For fixed D_c, the expected maximum point probability of $\mathsf{Enc}(pk_{\mathsf{N}}, D_c)$ is negligible: otherwise, comparing an independent encryption of D_c with an IND-CPA challenge encryption of D_c or a distinct plaintext would distinguish them, since correctness makes the two ciphertext supports disjoint. Formally, if $p(pk)$ is that maximum probability and $\mathsf{Col}(pk)$ is the collision probability, then $p(pk)^2 \leq \mathsf{Col}(pk)$; averaging over generated keys gives the claim. Thus polynomially many dummy ciphertexts collide only negligibly.

Next, for every adaptively requested honest OTT, replace $\mathsf{H}((nym, \sigma_{\mathsf{E}})_i, salt_i)$ by $\mathsf{H}(D_h, salt_i)$ for a fixed admissible D_h. A request-order hybrid reduces this to input hiding; the challenge digest suffices for blinding, finalization, and presentation, while a fresh $\mathcal{O}_{\mathsf{Open}}$ query reveals its purported preimage for direct checking.

At challenge generation, perform the stored-token lookup and all validity checks, including both internal $\mathcal{O}_{\mathsf{Present}}$ calls, but postpone sampling the unrevealed candidate's

independent dummy ciphertext. Abort if this hidden ciphertext occurs anywhere in the complete $\mathcal{O}_{\mathsf{Open}}$ log. Conditioning on each query prefix, its probability is bounded by the maximum point probability above, so the abort is negligible. We may henceforth omit that ciphertext. Only the association between two issuance slots and their now user-independent message–signature pairs remains. Equal selected slots give identical challenge distributions. Otherwise, guess their ordered individual request ordinals among the at most Q requests. Submit two fresh values $m_d = \mathsf{H}(D_h, s_d)$ to the two-session BS_2 blindness game, place the returned blinded messages at the guessed positions, and generate all other positions locally. Store the two target responses until both are available, then forward them to the challenger. A target batch may be stored provisionally after checking its non-target positions: $\mathcal{O}_{\mathsf{FinOTT}}$ has no output, and both target responses verify whenever the guess is the eventual eligible pair.

Let E be the event that the selected triples, in order, resolve to the guessed finalized requests; both embedded responses and both internal $\mathcal{O}_{\mathsf{Present}}$ calls succeed; neither selected triple is adversarially presented; and the revealed challenge is never opened. Failed guesses return 0. One blindness distinguisher returns 1_E; another returns the UNLINK experiment's final bit on E and 0 otherwise. This pair avoids conditioning on a possibly bit-dependent continuation. On a correct guess, the challenger supplies the valid signatures on m_0, m_1; reveal (m_0, σ_0) with a fresh dummy ciphertext. The swap during issuance is exactly the UNLINK bit. The table rejects any nonexact reuse of its ciphertext, and an exact $\mathcal{O}_{\mathsf{Open}}$ query is forbidden. Guessing loses at most $Q(Q-1)$, while blindness bounds both distinguishers' gaps. The final view is bit-independent. Summing the blindness, input-hiding, IND-CCA, and negligible collision and hidden-ciphertext terms proves Thm. 2. □

E.2 Proof of Thm. 4: Missing Properties

Thm. 4 asserts that Apple-PORTA does not achieve BIND, TRACE, or NFRAME. We prove each failure separately below. For each, we first state the corresponding lemma, followed by the full attack proof.

E.2.1 Failure of BIND

Lemma 1. *Apple-PORTA does not satisfy* BIND *(Def. 3).*

Proof. We construct an adversary $\mathcal{A}$ that wins the BIND experiment with probability 1. The attack exploits the fact that $\mathbf{G}.\mathsf{Vf}$ in Apple-PORTA verifies only the one-time signature and never inspects the ciphertext c carrying the long-term credential to $\mathcal{N}$. Consequently, $\mathcal{A}$ can pair an honestly issued $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ with an arbitrary fresh ciphertext, and the result still passes $\mathcal{G}$. Recall the game's setup: $\mathcal{S}_{\mathsf{E}}$ and $\mathcal{S}_{\mathsf{OTT}}$ are honest, $\mathcal{A}$ corrupts $\mathcal{N}$ and picks pk_{N} (with the matching sk_{N}), and honest users are registered via $\mathcal{O}_{\mathsf{Enroll}}$.

The attack. $\mathcal{A}$ generates $(sk_{\mathsf{N}}, pk_{\mathsf{N}}) \leftarrow \mathsf{PKE.KGen}(1^\lambda)$ and enrolls an honest user via $\mathcal{O}_{\mathsf{Enroll}}$, receiving only its pseudonym nym. Then:

1. $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{GetOTT}}(nym)$, triggering an honest batch issuance.
2. $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{LeakOTT}}(nym, 1, 1)$ to learn an OTT $(m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}})$ from that batch.
3. $\mathcal{A}$ chooses any valid ciphertext c^*, sets $\tau^* = (c^*, \sigma_{\mathsf{OTT}})$, and outputs $(\tau^*, m_{\mathsf{OTT}})$.

Verification of the win condition. $\mathbf{G}.\mathsf{Vf}$ checks $\mathsf{BS}_2.\mathsf{Vf}(pk_{\mathsf{OTT}}, m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}}) = 1$, which holds because the OTT was honestly issued; c^* is never inspected. Moreover, $q_{\mathsf{OTT}} = 0$ because $\mathcal{A}$ never queried $\mathcal{O}_{\mathsf{IssueOTT}}$, whereas $t = 1$; pairwise message distinctness is vacuous; and $(\tau^*, m_{\mathsf{OTT}}) \notin Q_\tau$ because $\mathcal{A}$ never queried $\mathcal{O}_{\mathsf{Present}}$. Hence, $\mathcal{A}$ wins with probability 1. □

E.2.2 Failure of TRACE

Lemma 2. *Apple-PORTA does not satisfy* TRACE *(Def. 3).*

Proof. We construct an adversary $\mathcal{A}$ that wins the TRACE experiment with probability 1. The attack exploits a structural weakness of Apple-PORTA's binding: the one-time message $m_{\mathsf{OTT}_i} = \mathsf{H}((nym, \sigma_{\mathsf{E}}), salt_i)$ is computed by the user *inside* the blinded batch, so $\mathcal{S}_{\mathsf{OTT}}$ never sees which $(nym, \sigma_{\mathsf{E}})$ a token is bound to. $\mathcal{A}$ can therefore have $\mathcal{S}_{\mathsf{OTT}}$ register one pseudonym while binding the issued tokens to a different one; the resulting presentation opens to a pseudonym $\mathcal{S}_{\mathsf{OTT}}$ has never witnessed. Recall the game's setup: $\mathcal{A}$ corrupts $\mathcal{S}_{\mathsf{E}}$ and picks pk_{E} (knowing the matching sk_{E}, e.g. by running $\mathbf{E}.\mathsf{KGen}$ honestly), $\mathcal{S}_{\mathsf{OTT}}$ is honest with tracked set Q_{nym}, $\mathcal{N}$ is semi-honest and $\mathcal{A}$ holds sk_{N}.

The attack. $\mathcal{A}$ chooses two distinct pseudonyms nym_0, nym_1 and signs each honestly with sk_{E} to obtain $\sigma_{\mathsf{E}_0}, \sigma_{\mathsf{E}_1}$ valid under pk_{E}. $\mathcal{A}$ then:

1. Runs $\mathbf{U}.\mathsf{ReqBatch}$ on the credential $(nym_0, \sigma_{\mathsf{E}_0})$, producing a blinded batch $\overline{M_{\mathsf{OTT}}}$ whose underlying messages are $m_{\mathsf{OTT}_i} = \mathsf{H}((nym_0, \sigma_{\mathsf{E}_0}), salt_i)$ for fresh salts $salt_i$, together with auxiliary state aux_{OTT}.
2. Queries $\mathcal{O}_{\mathsf{IssueOTT}}(\overline{M_{\mathsf{OTT}}}, nym_1, \sigma_{\mathsf{E}_1})$. The oracle verifies $\mathsf{BS}_1.\mathsf{Vf}(pk_{\mathsf{E}}, nym_1, \sigma_{\mathsf{E}_1}) = 1$ (which holds), records nym_1 into Q_{nym}, and signs the blinded batch with sk_{OTT}.
3. Unblinds via $\mathbf{U}.\mathsf{FinBatch}$ to obtain ℓ pairs $(m_{\mathsf{OTT}_i}, \sigma_{\mathsf{OTT}_i})$ valid under pk_{OTT}.
4. Computes $c \leftarrow \mathsf{PKE.Enc}(pk_{\mathsf{N}}, ((nym_0, \sigma_{\mathsf{E}_0}), salt_1))$, sets $\tau \leftarrow (c, \sigma_{\mathsf{OTT}_1})$, and outputs $(\tau, m_{\mathsf{OTT}_1})$.

Verification of the win condition. $\mathbf{G}.\mathsf{Vf}$ checks $\mathsf{BS}_2.\mathsf{Vf}(pk_{\mathsf{OTT}}, m_{\mathsf{OTT}_1}, \sigma_{\mathsf{OTT}_1}) = 1$, which holds since the token was honestly signed. $\mathbf{N}.\mathsf{Open}$ decrypts c to recover $((nym_0, \sigma_{\mathsf{E}_0}), salt_1)$; verifies $\mathsf{BS}_1.\mathsf{Vf}(pk_{\mathsf{E}}, nym_0, \sigma_{\mathsf{E}_0}) = 1$ (holds, since $\mathcal{A}$ signed it under sk_{E}); verifies $\mathsf{H}((nym_0, \sigma_{\mathsf{E}_0}), salt_1) = m_{\mathsf{OTT}_1}$ (holds by construction); and returns nym_0. But $nym_0 \notin Q_{nym}$ – $\mathcal{S}_{\mathsf{OTT}}$ only ever recorded nym_1. Hence $\mathbf{G}.\mathsf{Vf}$ accepts and $\mathbf{N}.\mathsf{Open}$ returns $nym_0 \notin Q_{nym}$, so the TRACE winning condition is satisfied. □

E.2.3 Failure of NFRAME

Lemma 3. *Apple-PORTA does not satisfy* NFRAME *(Def. 3).*

Proof. We construct an adversary $\mathcal{A}$ that wins the NFRAME experiment with probability 1. The attack exploits the fact that an honest user's $(nym, \sigma_{\mathsf{E}})$ is leaked in plaintext to a corrupt $\mathcal{S}_{\mathsf{OTT}}$ via the proof $\pi = \sigma_{\mathsf{E}}$ sent during batch issuance. Once $\mathcal{A}$ holds σ_{E} for an honest nym, it can mint a fresh presentation that opens to nym entirely on its own. Recall the game's setup: $\mathcal{S}_{\mathsf{E}}$ is honest, $\mathcal{A}$ corrupts $\mathcal{S}_{\mathsf{OTT}}$ and picks pk_{OTT} (with the matching sk_{OTT}), and $\mathcal{N}$ is semi-honest so $\mathcal{A}$ holds sk_{N}.

The attack. $\mathcal{A}$ generates $(sk_{\mathsf{OTT}}, pk_{\mathsf{OTT}}) \leftarrow \mathsf{BS}_2.\mathsf{KGen}(1^\lambda)$ and enrolls an honest user via $\mathcal{O}_{\mathsf{Enroll}}$, receiving only the pseudonym nym (which is recorded in Q_{HU}). Then:

1. $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{ReqOTT}}(nym)$. The honest user runs $\mathbf{U}.\mathsf{ReqBatch}$ internally and returns $(\overline{M_{\mathsf{OTT}}}, \pi)$ to $\mathcal{A}$. Since Apple-PORTA instantiates $\pi \leftarrow \sigma_{\mathsf{E}}$, $\mathcal{A}$ now holds the honest user's full $(nym, \pi) = (nym, \sigma_{\mathsf{E}})$.
2. $\mathcal{A}$ constructs a fresh presentation from scratch: it picks a salt $salt^* \xleftarrow{\$} \{0,1\}^\lambda$, computes $m^*_{\mathsf{OTT}} \leftarrow \mathsf{H}((nym, \sigma_{\mathsf{E}}), salt^*)$, signs $\sigma^*_{\mathsf{OTT}} \leftarrow \mathsf{BS}_2.\mathsf{Sign}(sk_{\mathsf{OTT}}, m^*_{\mathsf{OTT}})$ (i.e. performing $\mathsf{Blind}, \mathsf{BSign}, \mathsf{Unblind}$) using its own sk_{OTT}, encrypts $c^* \leftarrow \mathsf{PKE.Enc}(pk_{\mathsf{N}}, ((nym, \sigma_{\mathsf{E}}), salt^*))$, and sets $\tau^* \leftarrow (c^*, \sigma^*_{\mathsf{OTT}})$.
3. $\mathcal{A}$ outputs $(\tau^*, m^*_{\mathsf{OTT}})$.

Verification of the win condition. $\mathbf{G}.\mathsf{Vf}$ checks $\mathsf{BS}_2.\mathsf{Vf}(pk_{\mathsf{OTT}}, m^*_{\mathsf{OTT}}, \sigma^*_{\mathsf{OTT}}) = 1$, which holds since $\mathcal{A}$ signed σ^*_{OTT} honestly under sk_{OTT}. $\mathbf{N}.\mathsf{Open}$ decrypts c^* to recover $((nym, \sigma_{\mathsf{E}}), salt^*)$; verifies $\mathsf{BS}_1.\mathsf{Vf}(pk_{\mathsf{E}}, nym, \sigma_{\mathsf{E}}) = 1$, which holds since σ_{E} was honestly issued by $\mathcal{S}_{\mathsf{E}}$; verifies $\mathsf{H}((nym, \sigma_{\mathsf{E}}), salt^*) = m^*_{\mathsf{OTT}}$, which holds by construction; and returns nym. Finally, $(\tau^*, m^*_{\mathsf{OTT}}) \notin Q_\tau$ because $\mathcal{A}$ never called $\mathcal{O}_{\mathsf{Present}}$, so $Q_\tau = \emptyset$. Together with $nym \in Q_{\mathsf{HU}}$, the NFRAME winning condition is satisfied.

An equivalent attack via sk_{N}. The same framing is also reachable through corruption of $\mathcal{N}$ alone. Rather than calling $\mathcal{O}_{\mathsf{ReqOTT}}$, $\mathcal{A}$ asks an honest user to make a presentation via $\mathcal{O}_{\mathsf{Present}}$, then decrypts the resulting ciphertext using sk_{N} to recover $((nym, \sigma_{\mathsf{E}}), salt)$, and proceeds as in step 2 above. This shows that corrupting *either* $\mathcal{S}_{\mathsf{OTT}}$ or $\mathcal{N}$ alone is enough to break NFRAME – a long-lived framing capability follows from any single interaction with either service. □

E.3 Weak Non-Frameability

We use the weak variant of non-frameability in which all services are honest and only users may be corrupt.

Definition 13. A PORTA scheme Π is wNFRAME-secure if, for every PPT $\mathcal{A}$, $\Pr[\mathrm{Exp}^{w\mathrm{NFRAME}}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$, where the experiment is identical to NFRAME except that $\mathcal{S}_{\mathsf{OTT}}$ and $\mathcal{N}$ are honest as well (Fig. 10), i.e., $\mathcal{A}$ obtains OTTs via the oracles for an honest $\mathcal{S}_{\mathsf{OTT}}$ and has access to $\mathcal{O}_{\mathsf{Open}}$ instead of sk_{N}.

Theorem 11. *Apple-PORTA is w*NFRAME*-secure if* BS_1 *is* UNF-CMA*,* BS_2 *has unique signatures,* PKE *is* IND-CCA*, and* H *is a random oracle.*

Experiment $\mathrm{Exp}^{w\mathrm{NFRAME}}_{\Pi,\mathcal{A}}(\lambda)$

$(sk_{\mathsf{E}}, pk_{\mathsf{E}}) \leftarrow \mathbf{E}.\mathsf{KGen}(1^\lambda)$; $(sk_{\mathsf{OTT}}, pk_{\mathsf{OTT}}) \leftarrow \mathbf{OTT}.\mathsf{KGen}(1^\lambda)$
$(sk_{\mathsf{N}}, pk_{\mathsf{N}}) \leftarrow \mathbf{N}.\mathsf{KGen}(1^\lambda)$ ▷ *All system parties honest*
$(\tau, m_{\mathsf{OTT}}) \leftarrow$
  $\mathcal{A}^{\mathcal{O}_{\mathsf{Enroll}}, \mathcal{O}_{\mathsf{GetOTT}}, \mathcal{O}_{\mathsf{LeakOTT}}, \mathcal{O}_{\mathsf{IssueOTT}}, \mathcal{O}_{\mathsf{Present}}, \mathcal{O}_{\mathsf{Open}}}(pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, pk_{\mathsf{N}})$
$y \leftarrow \mathbf{N}.\mathsf{Open}(sk_{\mathsf{N}}, pk_{\mathsf{E}}, pk_{\mathsf{OTT}}, \tau, m_{\mathsf{OTT}})$
return 1 **if** $(\tau, m_{\mathsf{OTT}}) \notin Q_\tau \wedge y \in \{\mathsf{f}(nym) | nym \in Q_{\mathsf{HU}}\}$ ▷ *Same win condition*

Figure 10: Security experiment for weak non-frameability.

Here UNF-CMA for BS_1 is defined as in Def. 6. The BS_1 signing oracle runs blind issuance honestly on its input, as in UNF-CMA. We use the non-programmable random oracle: the reduction observes queries and samples random values for an unqueried hidden input, stopping with a forgery before answering an exact query.

Proof. In the wNFRAME game, all services are honest, and $\mathcal{A}$ wins by producing a fresh accepting presentation that opens to an honestly enrolled pseudonym. We replace the honest presentation ciphertexts and token hashes by simulatable values, then show that any winning output contains a new valid BS_1 signature and therefore yields a UNF-CMA forgery.

For each honest $\mathcal{O}_{\mathsf{Present}}$ output, store $T[c] = (m_{\mathsf{OTT}}, \sigma_{\mathsf{OTT}}, nym)$ and apply the same exact opening rule as above, including to the final output: the recorded triple opens to nym, any other use of its ciphertext returns $\perp$, and fresh ciphertexts are opened normally. Uniqueness of BS_2 makes the rule exact. IND-CCA hybrids then replace every recorded ciphertext by an encryption of D_c, at the negligible collision cost established above. An exact final replay lies in Q_τ, whereas a modified use of a recorded ciphertext cannot open.

We reduce the remaining game to UNF-CMA of BS_1. Given pk_{E}, the reduction generates the other keys and invokes its plaintext signing oracle only for corrupt enrollments. For an

honest enrollment it samples and records *nym* without obtaining the signature, associating the record with an opaque handle. It aborts on an enrollment collision, at cost at most $\binom{q_E}{2}/|\mathsf{NYM}|$, and scans all earlier protocol-domain random-oracle queries whenever it samples an honest pseudonym.

For each honest $\mathcal{O}_{\mathsf{GetOTT}}$ token, choose *salt* and associate an independent uniform digest with the pair consisting of its handle and *salt*; repeated pairs reuse that digest. Run the honest BS_2 protocol on it and store the resulting OTT. Hence, $\mathcal{O}_{\mathsf{LeakOTT}}$ and $\mathcal{O}_{\mathsf{Present}}$ are simulated without the missing enrollment signature. Meanwhile, inspect every credential in (i) random-oracle queries, (ii) $\mathcal{O}_{\mathsf{IssueOTT}}$ inputs, (iii) plaintexts recovered from fresh $\mathcal{O}_{\mathsf{Open}}$ queries, and (iv) the final plaintext. A valid BS_1 signature on an honestly enrolled *nym* is immediately a forgery, since after the collision abort that message was never signed.

This simulation is exact until such a forgery. In the real game the digest for an honest token is $\mathsf{H}((nym, \sigma_{\mathsf{E}}), salt)$. If that point is unqueried, its random-oracle value is independent and uniform, so it couples with the stored digest. An exact query supplies the missing valid credential; the reduction outputs it before answering. Thus, it neither programs nor changes an oracle value, and revealing *salt* is harmless.

Finally, a winning ciphertext is fresh by the table argument. Successful opening therefore exposes, through one of the monitored plaintexts, a valid enrollment signature for the honest pseudonym to which it opens. This yields the UNF-CMA forgery. Adding the IND-CCA and UNF-CMA advantages and the negligible probabilities of enrollment and ciphertext collisions proves Thm. 11. □

F Full Proofs for AC-PORTA

Proof-system convention. When a NIZK is assumed to be both zero-knowledge and online simulation-extractable, we use the standard joint interface: the adaptive multi-theorem simulator whose proofs are indistinguishable from honest proofs is the simulator relative to whose state online extraction is guaranteed. Independent proof systems use independent setup and simulator state; equivalently, Fiat–Shamir realizations use disjoint domain tags. Consequently, the NIZK_1 and NIZK_2 simulation hybrids below compose.

F.1 Proof of Thm. 6: Presentation Unlinkability

Proof. In the UNLINK game, $\mathcal{A}$ selects two honestly issued OTTs and must identify which one produced the challenge presentation, despite its $\mathcal{O}_{\mathsf{Open}}$ queries. We simulate the proofs and rely on the indistinguishability of the encryption to remove the link to the $(nym, \sigma_{\mathsf{E}})$, and then reduce the remaining OTT issuance session linkability to weak blindness of MSIG.

First reformulate challenge generation so that only the revealed candidate is formed. Retain the stored-token lookup and every validity check for both selected triples, including success of both internal $\mathcal{O}_{\mathsf{Present}}$ calls; for the unrevealed valid triple, omit only encryption, proof generation, and its unused Q_τ insertion. This is exact in the abstract NIZK model because proving is stateless, the output is never shown, and Q_τ is not subsequently consulted. In the random-oracle model, view the oracle as a random function fixed at setup: the discarded honest prover only evaluates that function and exposes neither its queries nor local state, so omitting those evaluations leaves every later oracle answer jointly unchanged. This reformulation precedes every simulation hybrid.

Now simulate all honest NIZK_2 proofs and record each complete statement–proof pair with its intended $\mathsf{f}(nym)$. On $\mathcal{O}_{\mathsf{Open}}$, reject a failed verification, answer an exact recorded pair from the table, and online-extract every other accepting pair. Extraction yields $(x, v, \sigma_{\mathsf{E}}, \sigma_{\mathsf{OTT}})$ satisfying the full relation, including $\mathsf{SIG.Vf}(pk_{\mathsf{E}}, x, \sigma_{\mathsf{E}}) = 1$, $c = \mathsf{PKE.Enc}(pk_{\mathsf{N}}, x; v)$, so correctness makes $\mathsf{f}(x)$ the real opening. In particular, a new proof

for an old statement is extracted; only the exact simulator output is table-resolved. This replaces decryption up to the zero-knowledge and online-extraction errors.

Replace every honest encryption of nym by an encryption of a fixed $D \in \mathsf{NYM}$, retaining the intended opening values $\mathsf{f}(nym)$ in the table and simulating the now possibly false statement. Each hop is IND-CPA because all nonrecorded accepting queries are opened by extraction. Correctness and injectivity of f give the negligible dummy-ciphertext collision bound as in the proof of Thm. 2; abort such collisions.

It remains to hide which issuance slot contains each independent uniform m_{OTT}. Equal selected slots give identical distributions. Otherwise, guess their ordered individual request ordinals among at most Q. Submit fresh $m_0, m_1 \xleftarrow{\$} \mathcal{M}$ to the two-session weak-blindness game for MSIG, embed the returned blinded messages at the guessed positions, and generate the rest locally. When both target responses arrive, forward them together with the clear pseudonyms of the respective sessions. Until then, provisionally store each target batch after checking all non-target positions; $\mathcal{O}_{\mathsf{FinOTT}}$ returns nothing, and the challenger verifies both targets on the correctly guessed eligible branch.

Let E say that the selected triples, in order, resolve to those finalized targets; their responses and both internal $\mathcal{O}_{\mathsf{Present}}$ checks succeed; neither triple is adversarially presented; and the revealed challenge is never opened. Failed guesses return 0. An indicator distinguisher returns 1_E, while a main distinguisher returns the UNLINK final bit on E and 0 otherwise, avoiding conditioning on a bit-dependent continuation. Although weak blindness withholds the target signatures, they are unneeded: reveal m_0 with a dummy ciphertext and simulated proof, and use extraction for every nonexact $\mathcal{O}_{\mathsf{Open}}$ query. The hidden issuance permutation is precisely the UNLINK bit; an exact challenge $\mathcal{O}_{\mathsf{Open}}$ query is forbidden. Guessing costs at most $Q(Q-1)$, and weak blindness bounds both gaps. The resulting view is bit-independent. Combining these bounds proves Thm. 6. □

F.2 Proof of Thm. 8: Binding

Proof. In the BIND game, $\mathcal{A}$ wins by producing more than q_{OTT} distinct accepting presentations outside Q_τ. After extracting the credential and OTT signatures from every output, we show that either one credential was never issued to a corrupt user, yielding an SIG forgery, or some corrupt pseudonym appears in more outputs than signing queries, yielding an MSIG one-more forgery.

Let C be the set of pseudonyms returned by corrupt-enrollment queries and H the set returned by honest-enrollment queries. Abort if these sets intersect; because all enrollment pseudonyms are sampled independently, the abort probability is at most $\binom{q_E}{2}/|\mathsf{NYM}|$.

First replace every honest presentation's NIZK_2 proof by a simulator output. To answer an honest $\mathcal{O}_{\mathsf{GetOTT}}$ call, sample each m_{OTT}, run $\mathsf{MSIG.Blind}$, and invoke $\mathsf{MSIG.BSign}$ (locally in the SIG reduction and through the signing oracle in the MSIG reduction), then unblind, verify, and store the batch. The internal NIZK_1 proof and its verification are omitted because their transcript is not exposed to $\mathcal{A}$ and completeness makes honest issuance succeed. Thus, this step uses zero knowledge only of NIZK_2, as required by the theorem.

Suppose $\mathcal{A}$ outputs $t > q_{\mathsf{OTT}}$ pairwise distinct public messages whose presentations all verify and are all outside Q_τ. Apply the online extractor to their NIZK_2 proofs. Except with negligible aggregate extraction error, this yields, for every $i \in [t]$, a pseudonym x_i and valid signatures $\mathsf{SIG.Vf}(pk_{\mathsf{E}}, x_i, \sigma_{\mathsf{E}_i}) = 1$, $\mathsf{MSIG.Vf}(pk_{\mathsf{OTT}}, (m_{\mathsf{OTT}_i}, x_i), \sigma_{\mathsf{OTT}_i}) = 1$. Freshness from Q_τ excludes every exact simulated statement–proof pair. In particular, a new proof for a statement seen before remains extractable.

If some $x_i \notin C$, embed the SIG UNF-CMA key. Query its plaintext signing oracle exactly for corrupt enrollments, and sample honest pseudonyms without signing them. The preceding simulation handles honest $\mathcal{O}_{\mathsf{GetOTT}}$ and $\mathcal{O}_{\mathsf{Present}}$ calls without those signatures; the OTT key is generated locally. The extracted signature on x_i is therefore a valid

signature on a message never submitted to the signing oracle, whether x_i is an honest pseudonym or was never enrolled. It is an SIG forgery.

It remains to consider the event that every $x_i \in C$. Embed the MSIG OMUF key and generate the enrollment key locally. Forward every accepted individual invocation of $\mathsf{MSIG.BSign}$ to the OMUF oracle. For $x \in C$, let $q(x)$ count the forwarded requests whose clear component is x, and let $t(x)$ count extracted outputs with $x_i = x$. Honest $\mathcal{O}_{\mathsf{GetOTT}}$ requests have clear components in H, which is disjoint from C after the abort. Hence, $\sum_{x\in C} q(x) \le q_{\mathsf{OTT}} < t = \sum_{x\in C} t(x)$. There is an $x \in C$ with $t(x) > q(x)$. The BIND experiment requires the m_{OTT_i} to be pairwise distinct, so the $t(x)$ extracted signatures have distinct first components and the same clear component x. Selecting any $q(x)+1$ of them violates the per-clear-component OMUF definition of MSIG. Repeated signing queries only increase $q(x)$ and do not affect the argument. The two cases cover every successful extraction. A union bound over the two signature reductions, the zero-knowledge and extraction errors, and the enrollment-collision event proves Thm. 8. □

F.3 Proof of Thm. 10: Non-Frameability

Proof. In the NFRAME game, $\mathcal{A}$ controls $\mathcal{S}_{\mathsf{OTT}}$, receives sk_{N}, and wins by producing a fresh presentation outside Q_τ that opens to an honestly enrolled pseudonym.

Replace all exposed honest NIZK_1 and NIZK_2 proofs by outputs of their compatible adaptive simulators. $\mathcal{O}_{\mathsf{FinOTT}}$ continues to validate and store the corrupt $\mathcal{S}_{\mathsf{OTT}}$'s responses, $\mathcal{O}_{\mathsf{LeakOTT}}$ returns the stored OTTs, and $\mathcal{O}_{\mathsf{Present}}$ encrypts the honest user's actual pseudonym while simulating only its proof. Thus the ciphertexts remain correct even though $\mathcal{A}$ receives the honestly generated key sk_{N}. The change is bounded by the zero-knowledge advantages of the two proof systems.

Embed the SIG UNF-CMA public key, generate the key of $\mathcal{N}$ honestly, and let $\mathcal{A}$ choose pk_{OTT} as in the experiment. On a corrupt enrollment, query the plaintext signing oracle and return the resulting credential. On an honest enrollment, sample and record the pseudonym without querying the signing oracle. Abort if any two enrollment pseudonyms collide; for at most q_E enrollments this costs at most $\binom{q_E}{2}/|\mathsf{NYM}|$. After the preceding simulation, no oracle requires an honest user's enrollment signature: NIZK_1 and NIZK_2 proofs are simulated, while OTT finalization, leakage, and the presentation ciphertext depend only on the pseudonym and OTT state.

Let $((c^*, \pi^*), m^*_{\mathsf{OTT}})$ be a winning final presentation. Open first verifies π^*. Moreover, the presentation is outside Q_τ, so its complete statement–proof pair is not an exact simulator output. Online simulation-extractability therefore yields $(x^*, v^*, \sigma^*_{\mathsf{E}}, \sigma^*_{\mathsf{OTT}})$ satisfying the NIZK_2 relation, including $\mathsf{SIG.Vf}(pk_{\mathsf{E}}, x^*, \sigma^*_{\mathsf{E}}) = 1$, $c^* = \mathsf{PKE.Enc}(pk_{\mathsf{N}}, x^*; v^*)$. Encryption correctness makes the actual opening equal $\mathsf{f}(x^*)$. Since the presentation wins, this value is $\mathsf{f}(y)$ for some honestly enrolled pseudonym y. Injectivity of f gives $x^* = y$. Assuming no collisions, y was never submitted to the signing oracle. The reduction outputs $(y, \sigma^*_{\mathsf{E}})$ as its UNF-CMA forgery. The collision event is bounded by $\binom{q_E}{2}/|\mathsf{NYM}|$; adding it to the zero-knowledge and extraction errors and the UNF-CMA advantage proves Thm. 10. □

G AGM Extractors

We omit the description of the AGM and refer the reader to [Orr25]. As stated in [Orr25], Fiat–Shamir (FS) proofs of the form shown in (1) are online simulation-extractable in the AGM+ROM. For a matrix of group elements F, write $F \odot w$ for the row-wise vector of multi-exponentiations, whose j-th entry is $\prod_i F_{j,i}^{w_i}$. The statement to be proven is then shown in (2).

$$\begin{bmatrix} F_{1,1} & \dots & F_{1,b} \\ \vdots & \vdots & \vdots \\ F_{a,1} & \dots & F_{a,b} \end{bmatrix} \odot \begin{bmatrix} w_1 \\ \vdots \\ w_b \end{bmatrix} = \begin{bmatrix} W_1 \\ \vdots \\ W_a \end{bmatrix} \quad (1) \qquad \begin{bmatrix} \prod_{i=1}^{b} F_{1,i}^{w_i} \\ \vdots \\ \prod_{i=1}^{b} F_{a,i}^{w_i} \end{bmatrix} = \begin{bmatrix} W_1 \\ \vdots \\ W_a \end{bmatrix} \quad (2)$$

The extraction is possible as long as (1) for every row $(F_{j,1} \dots F_{j,b})$ it is hard to find $(w_1 \dots w_b) \neq (0 \dots 0)$ with $\prod_{i=1}^{b} F_{j,i}^{w_i} = 1$ and (2) all distinct elements $F_{j,i}$ that are not the identity are independent.

The NIZK statements can be expressed in accordance with both conditions as shown in (3) for NIZK_1 and (4) for NIZK_2, so that they are extractable by the aforementioned AGM+ROM result:

$$\begin{bmatrix} p & p_1 & & & & & & & & \\ & p & p_1 & & & & & & & \\ & & & \ddots & & & & & & \\ & & & & p & p_1 & & & & \\ & & & & & & \overline{C} & \overline{A} & & \\ & & & & & & & & \overline{C} & h \end{bmatrix} \begin{bmatrix} r_1 \\ m_{\mathsf{OTT}_1} \\ r_2 \\ m_{\mathsf{OTT}_2} \\ \vdots \\ r_\ell \\ m_{\mathsf{OTT}_\ell} \\ w_2 \\ -e \\ w_1^{-1} \\ -r \end{bmatrix} \begin{bmatrix} \overline{m_{\mathsf{OTT}_1}} \\ \overline{m_{\mathsf{OTT}_2}} \\ \vdots \\ \overline{m_{\mathsf{OTT}_\ell}} \\ \overline{B} \\ D \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \overline{C} & \overline{A} & & & & & & & & \\ & & \overline{C} & h & h_0 & & & & & \\ & & & & & \overline{C'} & \overline{A'} & & & \\ & & & & p_0 & & & \overline{C'} & p & \\ & & & & & & & & & t \\ & & & & s^{-1} & & & & & Z \end{bmatrix} \begin{bmatrix} w_2 \\ -e \\ w_1^{-1} \\ -r \\ -nym \\ w_2' \\ -e' \\ w_1'^{-1} \\ -r' \\ v \end{bmatrix} \begin{bmatrix} \overline{B} \\ g \\ \overline{B'} \\ D' \\ c_0 \\ c_1 \end{bmatrix} \quad (4)$$

We note two differences to [Orr25] in our instantiation. First, there are additional group elements present, namely $\widetilde{g}, \widetilde{g}' \in \mathbb{G}_2$, which have no impact on the extraction, since all elements of the proven statements are in $\mathbb{G}_1$. Second, for extracting a BBS signature, [Orr25] relies on a DDH oracle. Since our construction uses pairings, these can be used instead of the DDH oracle. The corresponding checks are in line 42 (**OTT**.IssueBatch) and line 63 (**G**.Vf) of the instantiation in Fig. 9.