

Do You Need a Receipt? Anonymous Credential Revocation at Continental Scale via Private Record Certification

Kasra EdalatNejad, Sebastian Faust, Jonas Hofmann, Philipp-Florens Lehwalder, Thomas Schneider
Technical University of Darmstadt, Germany
{edalat, schneider}@encrypto.cs.tu-darmstadt.de,
{sebastian.faust, jonas.hofmann1, philipp-florens.lehwalder}@tu-darmstadt.de

Abstract

A key challenge in digital credential systems is revocation, that is, the ability to invalidate credentials post-issuance and verify their status upon presentation. While anonymous credentials enhance privacy over classical credentials (e.g., by providing unlinkability), they complicate revocation. Existing revocation schemes for anonymous credentials often suffer from high client or verifier computation, long delays before revocation takes effect (e.g., epoch-based settings), or require updates to all users with each revocation.

We present an efficient, real-time revocation system for anonymous credentials with decentralized revocation authorities based on a novel primitive called *Private Record Certification* (PRC). PRC enables users to obtain a certificate for a record stored in a server-managed database without the servers learning which record was requested. This primitive is of independent interest, and we construct it by combining techniques from private information retrieval and secure multi-party computation. Our revocation scheme outsources its computation to the revocation authorities and has minimal overhead for clients and verifiers, while ensuring the communication costs are sublinear in the number of credentials. We build a prototype and demonstrate that our system achieves sub-second real-time latency for PRC requests, at a scale of over *1 billion* credentials, with an online operational cost of 2.5$ per server for processing *1 million* PRC requests.

1 Introduction

As governments increasingly move public services online, electronic identification (eID) is becoming a critical national infrastructure, serving as the entry point for essential services like taxes, healthcare, mobility, and education. While eID systems ease administrative burdens for citizens, they also raise concerns about mass surveillance and the erosion of fundamental rights, such as privacy. European regulations [49] call for privacy-preserving digital identities that provide strong authenticity without enabling cross-context linkability. The European Digital Identity (EUDI) Wallet initiative seeks to realize these legal requirements, but has major shortcomings and does not yet provide the demanded privacy guarantees [6]. Anonymous credentials [26, 28, 29] have long been known to achieve this securely and efficiently and future EUDIW iterations may adopt anonymous credentials [48].

A central challenge in credential systems is *revocation*: the ability to invalidate credentials after issuance, and to verify a credential's status during presentation. Privacy-preserving revocation is not a technical nicety; it is a two-sided safeguard. It protects citizens against identity theft, lost credentials, misuse, and surveillance, while empowering authorities to enforce accountability in a rights-respecting manner. For instance, suspended driver's licenses or fraudulent professional certifications must be revoked without enabling pervasive profiling of citizens. Privacy can be important for revocation authorities too, especially when they are law enforcement or government agencies (e.g., for driver's licenses). Solutions that rely on public revocation lists disclose the status of all credentials, exposing unnecessary data and creating legal risks. Demonstrating that strong privacy and real-time accountability can be achieved simultaneously is essential to counter the false trade-off between security and civil liberties, which often stalls adoption of privacy-preserving solutions.

While anonymous credentials offer strong privacy, they complicate revocation. Existing revocation schemes [5, 20, 22, 24, 27, 83, 91, 101] incur expensive client updates per revocation, high verifier costs, or delayed revocation via time-based epochs, which create "opportunity windows" for abuse of revoked credentials.

We study the requirements of a practical revocation scheme. Viable systems should: 1) scale for nation-wide deployment, supporting hundreds of millions of credentials, 2) support real-time revocation that eliminates the abuse window, and 3) achieve low latency with 4) minimal computational/communication overhead for clients and verifiers, enabling deployment on mobile devices and constrained networks. We align revocation privacy requirements with those of anonymous credential presentations. Since revocation in-

troduces authorities absent from classic credentials, we also aim to preserve the privacy of the credential status database, which we call *database privacy*. In particular, clients must prove ownership of a credential to learn its revocation status.

We construct a revocation scheme for anonymous credentials that achieves scalability, efficiency, and real-time revocation while maintaining strong privacy guarantees. Clients obtain a non-revocation token from a Revocation Authority (RA) and present it, along with their credential, to verifiers. We target an on-demand scenario where clients request a new token immediately before each credential presentation. Therefore, we must guarantee that non-revocation queries remain unlinkable. We require similar privacy guarantees for RAs as we due for issuers and verifiers in credential systems. This is needed, as otherwise the RA could track or discriminate against users based on their usage patterns, leaked attributes, or revocation status. To achieve this guarantee, we assume multiple RAs that share a replicated database in an anytrust setting, where unlinkability is preserved as long as at least one RA remains honest. The client does not need to know which one is honest. This is a realistic model, as in practice, the distributed RA can be realized as a consortium of different servers, e.g., operated by different governmental and non-governmental institutions.

As a first step toward a revocation scheme, we introduce a new primitive called Private Record Certification (PRC) that enables clients to privately obtain a certificate for a record in a server-managed database. Concretely, it provides a signature on a record $\mathbf{D}[\mathsf{idx}]$ together with its index idx, while hiding the identity of the client, the index itself, and the record value. Clients can use this signature in standard zero-knowledge proofs to demonstrate their record satisfies a desired property. Beyond revocation, PRC is of independent interest for applications such as privacy-preserving proofs of credit score or account balance. Our PRC construction combines techniques from Private Information Retrieval (PIR) [31,42] with Secure Multi-Party Computation (MPC) [55]. Specifically, we introduce a novel conversion from Boolean secret sharing to One-Hot Encoding (OHE) achieving theoretically minimal client-side complexity, logarithmic in the number of database records (i.e., the number of bits needed to represent the index in cleartext). We then combine our secret sharing operations in a new way with homomorphic commitments, enabling a seamless transition to asymmetric operations needed for building a non-revocation token via a (threshold) signature.

Contributions. We summarize our contributions below.

✓ We introduce Private Record Certification (PRC), a cryptographic primitive that enables clients to obtain a private certificate for a record in a server-managed database, along with formal definitions and security properties.

✓ We design a PRC scheme with minimal client and verifier cost complexity, and formally prove its security.

✓ We build a privacy-preserving revocation system for anonymous credentials, instantiated with our PRC.

✓ We implement and evaluate our PRC construction, demonstrating that our revocation system achieves real-time revocation while scaling to *billions* of credentials with *sub-second latency* for revocation status lookups. Our solution is also cost-efficient: the online operation costs per PRC server for *one million queries* are only 2.5$, based on AWS pricing.

1.1 Technical Overview

We provide a high-level technical overview of our solution next. We build a revocation scheme where a decentralized RA keeps a private status list $\mathbf{D}$, such that $\mathbf{D}[\mathsf{idx}] \in \{0,1\}$ is the validity status of the credential with a private index idx (see §6). When clients want to present one of their credentials, they need to prove that $\mathbf{D}[\mathsf{idx}] = 1$, while 1) binding idx to the index embedded in their credential and 2) hiding idx to ensure unlinkability. To achieve this goal, we define a primitive called PRC that allows the client to interact with RAs to privately obtain a signed commitment linking $(\mathsf{idx}, \mathbf{D}[\mathsf{idx}])$, also called a non-revocation token. When clients present their credential, they present a non-revocation token and use zero-knowledge proofs to link the index in the credential to the token (i.e., proof of discrete log equality) and reveal the record value $\mathbf{D}[\mathsf{idx}] = 1$. The verifier validates a) the credential presentation, b) the RA's signature on the token, and c) the ZK-proof proving the token corresponds to the credential. Since PRC provides *unforgeability*, the verifier can be certain that the revealed revocation status belongs to presented credential.

PRC. We define and construct a PRC scheme in §5, where given a private index, the PRC servers compute and then sign a commitment to $(\mathbf{D}[\mathsf{idx}], \mathsf{idx})$ that binds an index to its record value. We design a PRC protocol in the following steps: 1) the client Boolean secret shares its index among PRC servers. 2) the servers convert their boolean shares of idx to arithmetic shares and compute arithmetic shares of the database record $\mathbf{D}[\mathsf{idx}]$. This step has a similar nature as PIR, and we design a custom optimized variant of it, which we discuss below. 3) Each server commits to its arithmetic share of the index and record value via a homomorphic commitment like Pedersen. 4) The PRC servers aggregate the homomorphic commitment. As we deliberately choose the same group order between our commitment and arithmetic sharing, this aggregation acts as a secret reconstruction without revealing the cleartext values to PRC servers. 5) The PRC servers sign the final commitment.

We extend this scheme in §5.4 to achieve database privacy, by requiring clients to add a ZK-proof (similar to the proof used for presenting a non-revocation token) showing that they have an authorization for the requested index.

Record retrieval. In step 2) of our PRC scheme, the PRC servers - each holding a replicated database $\mathbf{D}$ - obtain an arithmetic sharing of the record $\mathbf{D}[\mathsf{idx}]$, where idx is the client's private index. This step resembles multi-server PIR schemes [31] which (Boolean) secret share the OHE of the client's index, an all-zero vector with a single 1 at position idx. The servers

then compute the inner product of this vector and the database and return the result to the client to reconstruct.

Unlike existing schemes that avoid server interaction, our PRC scheme already requires intra-server communication for aggregation and signing. This allows us to leverage MPC to support AND/multiplication on secret-shared values. Instead of representing the database as a 1-dimensional array of size N, we follow Chor et al. [31] and structure it as a 2-dimensional table of size $(\sqrt{N}, \sqrt{N})$. Clients send two OHE selectors for the target record's row and column, reducing retrieval to a two-layer vector-matrix multiplication. Since servers know the plaintext database, the first layer is computed locally without communication. The second layer, where both vectors are secret shared, has a multiplicative depth of 1 and sublinear server-server communication of $O(\sqrt{N})$.

Existing multi-server PIR schemes either use large non-polylog queries [31, 42, 57] to encode the OHE selector, or rely on a Distributed Point Function (DPF) [54], which prevents computing an arithmetic sharing of the index. Because we employ MPC, clients can instead send Boolean shares of their index, which we expand into an OHE vector on the server side. To support this, we design an efficient conversion from Boolean shares to OHE with multiplicative depth $O(\lg \lg N)$, incurring a server-server communication cost of $1.5\sqrt{N}$ to construct an OHE vector of size $\sqrt{N}$. The low depth of our subprotocols allows us to use information-theoretic MPC protocols, such as GMW [55], which avoid online costs dependent on a symmetric security parameter, while achieving sub-second latency even in WAN settings. Because the majority of our computation is based on Boolean secret sharing, we can efficiently pack eight status bits per byte and apply AVX2 vectorized bitwise operations. This use of SIMD enables a high degree of parallelism while running on a single core, allowing us to achieve a low PRC server computation cost.

2 Related Work

Anonymous credential revocation. A prominent revocation approach for anonymous credentials relies on dynamic accumulators [5, 24, 27, 83], which allow users to prove (non-)membership in an accumulator representing the set of all revoked or valid credentials. An inherent drawback is that each accumulator update (e.g., upon revocation) invalidates existing witnesses, forcing users to either update their witness for every revocation or query for a fresh one.

Blacklist-based revocation methods [20, 101] suffer from expensive non-membership proofs. A possible optimization is requiring the authority to maintain a sorted signature list of revoked identifiers, enabling users to prove that their identifier lies between two adjacent signatures [25, 82]. Yet, constructing, updating, and distributing such lists remains costly. One way to reduce this update overhead is by restricting updates to specific time intervals/epochs; however, this sacrifices the timeliness of the revocation. In [22, 91], users can present a limited number of unlinkable non-revocation tokens per epoch without expensive proofs or communication with the RA. However, their approach requires an expensive generation of a blacklist of all revoked tokens for the RA and high costs of non-inclusion checks for verifiers. Furthermore, unlinkability does not hold if the verifier colludes with the RA.

Distributing the RA is a prominent approach to avoid single points of failure [56,61,65]. Our scheme relies on a *distributed RA* under an anytrust assumption, where the user's privacy is guaranteed if at least one out of n RAs does not collude with the adversary. Clients do not need to know which RA is not compromised. We compare our scheme with the recent multi-party revocation scheme ALLOSAUR [65], which is based on accumulators under an anytrust assumption. ALLOSAUR is not designed for deployments at continental scale, and we show in §7.2 that we improve revocation cost by a factor of 20,000× for 10M credentials and a 1% revocation rate.

Private information retrieval. PIR allows a client to retrieve an element from a database without revealing the queried index to the server. PIR schemes fall into two categories: multi-server PIR, which relies on multiple non-colluding servers [31,33,42,45,58,81,85], and single-server PIR, which uses asymmetric cryptography [40, 63, 74, 78, 85]. In multi-server PIR approaches, the introduction of schemes based on distributed point functions and function secret sharing has enabled sublinear communication [15, 16, 54, 72]. Recent years have seen significant progress in single-server PIR, largely due to advances in homomorphic encryption. Early works such as XPIR [77] and SealPIR [2] demonstrated query compression over RLWE (e.g., [19]), achieving sublinear communication. Later schemes, including SimplePIR [63] and FrodoPIR [40], eliminated the ring structure and achieved throughput comparable to multi-server PIR by operating directly on LWE [90], but required clients to download and store large initialization *client-hints*. Later works [62, 74, 80] removed this requirement at the cost of higher per-query overhead. Another line of research [21, 52, 79], combines RLWE with GSW [53] to represent queries in high dimensions, achieving polylogarithmic communication at the expense of heavy computation and large per-client evaluation keys (or *server-hints*).

Despite these advances, many of the PIR techniques are incompatible with our setting. Real-time credential revocation involves frequent database updates, making approaches with expensive, data-dependent client hints impractical, as their costs cannot be amortized across many queries. Similarly, server-side hints such as per-client evaluation keys are infeasible, since reusing the same key would break the anonymity required for anonymous credentials.

While PIR and PRC share similar settings and construction mechanisms, they have orthogonal goals. In §5.2, we highlight these differences and further provide a naive revocation approach based on PIR. We evaluate this approach

using state-of-the-art multi-server PIR and single-server PIR, showing that beyond protecting database privacy, our construction reduces server cost by a factor of $2700\times$ compared to multi-server PIR and $900\times$ compared to single-server PIR.

Committed OT/authenticated MPC. Committed oblivious transfer [34,35,51] is an OT protocol, in which the participants commit to their inputs. Unfortunately, committed OT incurs a cost linear in the number of records. Similarly, authenticated MPC [44] provides a way for participants to authenticate their inputs to an MPC protocol, but these schemes still require communication linear in the input size.

Blind signatures. Similar to PRC, blind signatures [30, 88] let client obtain signatures on a message, without revealing it to the signer. In contrast, in our revocation and PRC setting, the signer must attest to the validity of a data record and not on a freely chosen message. This enforces a specific structure: clients only choose a record index, not its associated content.

Self-sovereign identity. Self-sovereign identity (SSI) is an architectural model for user-centric identity systems allowing users to control and present their credentials without relying on central identity providers [71]. SSI is commonly instantiated using decentralized identifiers and verifiable credentials [76, 95, 97, 98]. Anonymous credentials can be viewed as verifiable credentials that additionally provide anonymity and unlinkability, and SSI systems can be instantiated with them. Revocation mechanisms for non-anonymous credentials, such as the W3C Bitstring Status List [75], rely on credential identifiers as part of the revocation status check, which is incompatible with anonymous credentials where no such identifier can be disclosed to preserve anonymity. We focus on revocation for anonymous credentials; instantiating SSI systems with them is outside the scope of this work.

3 Preliminaries

In this section, we present our notation, preliminaries, and building blocks used in our construction.

Notation. We use $[n]$ to denote the set $\{0,\dots,n-1\}$ and use $(a_i)_{i\in[n]}$ to represent a list of n elements $(a_0,\dots,a_{n-1})$. When size n is known from the context, we omit it. We denote vectors by bold-face letters like $\mathbf{v}$ and refer to their i-th element as $\mathbf{v}[i]$. We denote inner products by $\langle \mathbf{v},\mathbf{w}\rangle$ and when both vectors are k-bit arrays compute it as $p = \bigoplus_{i\in[k]} (\mathbf{v}[i] \wedge \mathbf{w}[i])$. We abuse the notation and still use $\langle \mathbf{v},\mathbf{w}\rangle$ when one of the vectors contains l-bit values. We use $x[a:b)$ to denote the bit slice of $x \in \mathbb{Z}_{2^k}$ from bit position a (inclusive) to b (exclusive), where $0 \le a \le b \le k$. The least significant bit is $x[0:1)$, and we omit the index when the slice starts at 0 or ends at k. We use $y \xleftarrow{\$} S$ to denote that y is sampled uniformly at random from a set S. We write $y \leftarrow \mathsf{A}(x;r)$ to denote that y is assigned the output of algorithm A running on input x and randomness r. We assume that the security parameter λ and public parameters pp, typically containing group parameters, are implicitly given as input to all algorithms. A polynomial-time verifiable binary relation is denoted as $(x,w) \in \mathsf{R}$, where w is a witness for the statement x satisfying the relation. We summarize our notation and common variables in Table 1.

$(a_i)_{i\in[l}$	List of n elements $(a_0,\dots,a_{n-1})$	qry_i	Query for PRC server $\mathcal{S}_i$
$\langle \mathbf{v},\mathbf{w}\rangle$	Inner product of two vectors	σ_{rec}	Record certificate
$x[a:b$	Bit slice of x from index a (inclusive) to b (exclusive)	σ_{aut}	authorization credential
$[\![x]\!]^{\mathrm{A}}$	Additive share of x	$\mathbf{D}$	Database
$[\![x]\!]^{\mathrm{B}}$	Boolean share of x	N	Size of the database
idx	Index	n	# of RAs/PRC servers
rec	Record value	qst	Query state
rnd	Randomness	co	Commitment
aux	Auxiliary information	pt	Presentation token

Table 1: A summary of the notation used in this work.

One-hot encoding. A One-Hot Encoding (OHE) of an integer x for dimension n, denoted by $\mathbf{v} \leftarrow \mathsf{OHE}(x,n)$, is a bit vector of length n such that $\mathbf{v}[x] = 1$ and $\forall i \neq x, \mathbf{v}[i] = 0$.

Homomorphic commitments. A homomorphic commitment scheme Com consists of the following algorithms:

- $\mathsf{co} \leftarrow \mathsf{Commit}(x;r)$: Generates a commitment co to a value x using randomness r.
- $\mathsf{co} \leftarrow \mathsf{Agg}((\mathsf{co}_i)_{i\in[n]})$: Aggregates the commitments co_i for $i \in [n]$ into a single commitment co.
- $0/1 \leftarrow \mathsf{Open}(\mathsf{co},x,r)$: Verifies whether the commitment co opens to the value x using randomness r.

Homomorphic commitment schemes must satisfy *binding* and *hiding*. Informally, binding ensures that a commitment co can only be opened to the committed value x with the randomness r and hiding ensures that co does not reveal any information about the committed value x. The homomorphism ensures that $\mathsf{Agg}(\mathsf{Commit}(x_1;r_1),\mathsf{Commit}(x_2;r_2)) = \mathsf{Commit}(x_1+x_2;r_1+r_2)$. We will instantiate this homomorphic commitment scheme with Pedersen commitments [86].

(Threshold) signature schemes. We consider both standard and threshold signature schemes, where a standard signature scheme Sig consists of the following algorithms:

- $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$: Generates a signing key sk and a verification key vk.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},m)$: Generates a signature σ for a message m using the signing key sk.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{vk},\sigma,m)$: Verifies the signature σ for a message m using the verification key vk.

A threshold signature scheme TSig in an t-out-of-n setting modifies key generation and signing while keeping compatibility with the standard verification as follows:

- $((\mathsf{sk}_i)_{i\in[n]},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda,n)$: Generates a set of n signing keys sk_i and a common verification key vk. This algorithm might also be defined in a distributed manner.
- $\sigma_i \leftarrow \mathsf{PSign}(\mathsf{sk}_i,m)$: Generates a partial signature σ_i for a message m using the signing key sk_i.

- $\sigma \leftarrow \mathsf{Agg}((\sigma_i)_{i\in[n]})$: Aggregates t partial signatures (σ_i) into a single signature σ.

A (threshold) signature scheme needs to satisfy *correctness* and *unforgeability*. Correctness states that a valid signature σ on some message m can be verified using the verification key vk. Unforgeability ensures that no adversary (knowing at most $t-1$ signing keys sk_i in the threshold setting) can generate a valid, new signature σ on some message m. We instantiate the scheme with threshold Schnorr signatures [94].

Zero-knowledge proof systems. A zero-knowledge proof system ZKP allows a prover to generate a proof π for a statement x that convinces a verifier that the prover knows a witness w satisfying a public relation R such that $(x,w) \in \mathsf{R}$ without revealing any information about the prover's witness w.

A zero-knowledge proof system should satisfy *completeness*, *knowledge-soundness*, and *zero-knowledge*. Completeness states that honestly generated proofs will always pass verification. Knowledge-soundness states that it is impossible to generate a valid proof π for a statement x without knowing a valid witness w. Zero-knowledge ensures that verifier learns no information beyond the prover knowing a valid witness w. We instantiate our ZKP system with efficient Σ-protocols [99].

Anonymous credentials. We specify an anonymous credential system AC as comprising the following algorithms:

- $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$: Generates the secret key sk and verification key vk.
- $\sigma_{\mathsf{cred}} \leftarrow \mathsf{Issue}(\mathsf{sk},(a_i)_{i\in[m]})$: Issues a credential σ_{cred} with attributes $(a_i)_{i\in[m]}$ using the secret key sk.
- $\mathsf{pt} \leftarrow \mathsf{Present}(\sigma_{\mathsf{cred}},(a_i)_{i\in[m]},\mathsf{Rvd})$: Generates a presentation token pt for a credential σ_{cred} while selectively revealing the attributes a_i for $i \in \mathsf{Rvd} \subseteq [m]$.
- $0/1 \leftarrow \mathsf{VerifyPresent}(\mathsf{vk},\mathsf{pt},(a_i)_{i\in\mathsf{Rvd}})$: Verifies the presentation token pt for a set of revealed attributes a_i for $i \in \mathsf{Rvd}$ under the verification key vk.

We require an anonymous credential system to satisfy *correctness*, *unforgeability*, and *unlinkability*. Correctness means that an honestly generated presentation token pt of a valid credential σ_{cred} can be verified using the verification key vk. Unforgeability ensures that no adversary without knowledge of the secret key sk can generate a valid presentation token pt that reveals attributes for which the adversary has not been issued a corresponding credential. Lastly, unlinkability ensures that no adversary controlling the issuer and verifier can tell two presentations pt_1 and pt_2 for the same revealed attribute set Rvd of different credentials apart. This ensures that clients cannot be linked across multiple presentations. We provide formal definitions for these properties in the full version [46].

Secret-sharing. An n-out-of-n secret-sharing scheme SS enables sharing of a value x among n parties such that the parties can compute on the shared value $[\![x]\!]$. Secret-sharing consists of two primary operations:

- $([\![x]\!]_i)_{i\in[n]} \leftarrow \mathsf{SS.Share}(x,n)$: Shares a value x into n secret shares $([\![x]\!]_i)_{i\in[n]}$.
- $x \leftarrow \mathsf{SS.Rec}(([\![x]\!]_i)_{i\in[n]})$: Given n secret shares $([\![x]\!]_i)$, reconstructs the value x.

We require the secret-sharing scheme to satisfy two properties: *correctness* and *hiding*. The correctness property ensures that computation performed on shared values yields the same result as if it were performed on the original value x. The hiding property ensures that any group of less than n parties cannot learn any information about the secret value x. Secret-sharing can be instantiated via the GMW protocol [55] and the online phase can be improved via Beaver triples [7].

We use two types of secret-sharing: 1) *Arithmetic* sharing which we denote as $[\![x]\!]^{\mathrm{A}} \leftarrow \mathsf{SS.Share}^A(x)$ where values are taken from a prime field $x \in \mathbb{Z}_q$ and shared such that $x = \sum_i^n [\![x]\!]_i \bmod q$. 2) *Boolean* sharing denoted as $[\![x]\!]^{\mathrm{B}} \leftarrow \mathsf{SS.Share}^B(x)$ where values are k-bit binary values $x \in \mathbb{Z}_{2^k}$ and shared such that $x = \bigoplus_i^n [\![x]\!]_i^{\mathrm{B}}$ where $\oplus$ is the XOR operation. In addition, we define a conversion operation as follows:

- $[\![x]\!]^{\mathrm{A}} \leftarrow \mathsf{SS.B2A}([\![x]\!]^{\mathrm{B}})$: Takes Boolean secret shares $[\![x]\!]^{\mathrm{B}}$ and interactively converts them to arithmetic shares $[\![x]\!]^{\mathrm{A}}$.

We instantiate this operation via the conversation of Damgård et al. [37], which uses daBits [47, 92] as correlated randomness, and provides a depth-optimized conversion that only requires a single round of interaction.

4 Anonymous Credential Revocation System

This section introduces the system model, requirements, and definitions for anonymous credential revocation.

4.1 System Model

In an anonymous credential revocation system, a client holds a credential and proves its non-revocation status to a verifier. The client first obtains a non-revocation token for its credential from a distributed RA and then presents the token to the verifier together with the credential. For privacy, the RA should not learn for which credential the token was requested.

Conceptually, this approach is similar to the online certificate status protocol (OCSP) with stapling, where web servers obtain a signed OCSP response from an OCSP responder to prove their certificates' validity to a client. In our setting, prior to presenting a credential, a client obtains a non-revocation token from the RA to prove its validity.

The system model (Figure 1) consists of four types of parties: 1) the *issuer*, which issues anonymous credentials to clients; 2) the *client*, which holds a credential and wants to present it while proving that it has not been revoked; 3) the *verifier*, which checks the credential's validity, including non-revocation, before providing a service or granting access; and 4) the *RAs*, which jointly maintain a revocation-status database indicating whether each credential is valid or revoked. We consider distributed RAs: n authorities provide the service jointly, and clients or verifiers may interact with

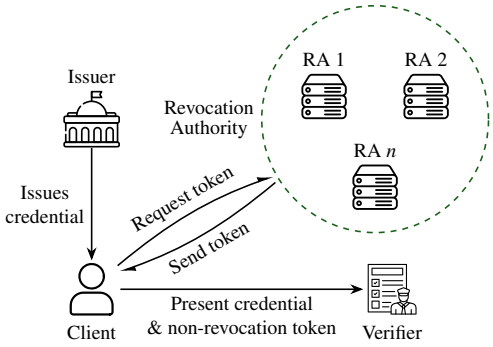

Figure 1: The system model of our anonymous credential revocation system.

them to determine or prove a credential's validity. How the current state of the database is replicated across all RAs is out of scope of our work.

Threat model. We consider adversaries that compromise parties either semi-honestly or maliciously. A *semi-honest* corruption reveals the party's internal state, private values, and view, including all incoming and outgoing messages, but does not allow deviations from the protocol. A *malicious* corruption additionally allows the adversary to deviate from the protocol and to drop or modify the party's messages. For the distributed RAs, we use an any-trust assumption: the adversary may compromise at most $n-1$ RAs, but clients or verifiers *do not* need to know which RA is honest. We assume communication channels provide confidentiality and integrity (e.g., via TLS), and we do not consider side-channel attacks.

4.2 Requirements & Trust Assumptions

We state the functionality, privacy, and deployment requirements for anonymous credential revocation systems. Next, we explain our trust assumptions for these properties. For formal definitions, see D.

Functionality.

RQ.F1: Revocation. Clients can obtain a valid non-revocation token for a credential iff the credential is not revoked, and cannot obtain one for a revoked credential.

RQ.F2: Dynamic Database. The revocation-status database can change at any time, and updates occur frequently.

RQ.F3: Real-time Revocation. Once a credential is revoked, the window in which the holder can still present a non-revocation token must be minimal to prevent abuse. We target validity windows of a few seconds, which is inherently incompatible with settings where both the client and verifier are offline and cannot reach the RAs.

Privacy.

RQ.P1: Client Unlinkability. Adversaries should not be able to link multiple non-revocation presentations of a client.

RQ.P2: Revocation Unlinkability. The RAs should not learn information about the client, including the client's credential validity, revealed attributes, or the verifier to whom the client presents the non-revocation token.

RQ.P3: Database Privacy. The revocation-status database held by the RAs should remain private.

Efficiency.

RQ.D1: Scalability. The system must support nationwide certificate revocation in Europe at a scale exceeding 100M users.

RQ.D2: Lightweight Clients. Clients may run on smartphones with limited resources. Their computation and communication costs must remain small.

RQ.D3: Single Query Efficiency. In on demand scenarios, users typically present a single non-revocation token per interaction. Hence, revocation systems cannot amortize cost over batched queries for one user, and must be efficient for single non-revocation queries.

Trust assumptions. We trust issuers to issue only valid credentials. We trust RAs to revoke a credential iff it is no longer valid, and not to help clients produce valid non-revocation tokens for revoked credentials. In the same way that verifiers trust issuers to honestly issue credentials, they trust RAs to check whether credentials remain valid. We separate issuance from revocation for clarity; in practice, issuers and RAs may be operated by the same parties.

Different properties rely on different adversarial models:

AM.1: Client Unlinkability. The adversary may maliciously compromise the issuer, the verifier, and up to $n-1$ RAs, and try to link different interactions of the same user.

AM.2: Revocation Unlinkability. The adversary may maliciously compromise the issuer and up to $n-1$ RAs, and try to learn attributes, the credential's validity, or to which verifier the client presents the credential.

AM.3: Forging Non-Revocation Tokens. The adversary may maliciously compromise a client and semi-honestly compromise up to $n-1$ RAs, and try to obtain a valid non-revocation token for a revoked credential. Thus, compromised RAs may leak their internal state and views, but do not actively deviate from the protocol to help the client to forge.

Existing non-private revocation systems typically rely on a single RA that is trusted for revocation. Mechanisms such as certificate transparency [73] can provide accountability, but they substantially increase system complexity, and adapting them to a private setting such as ours is non-trivial and costly. We acknowledge that our distributed RA design increases the active-compromise attack surface: unlike a single-RA design, an adversary that actively compromises one of the n RAs may affect revocation correctness.

AM.4: Database Privacy. The adversary may maliciously compromise multiple clients and try to learn information about the revocation-status of non-compromised clients.

4.3 Definition

We provide a formal definition of an anonymous credential system with revocation that fulfills the requirements from §4.2. We denote the credential issuer's keys by $(\mathsf{sk}_I, \mathsf{vk}_I)$ and those of the RA by $((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}})$.

Definition 4.1. *An anonymous credential system with revocation extends an anonymous credential scheme* AC *(§3), adopts the* KGen *and* Issue *algorithms, and extends presentation and verification to include revocation checks through the following algorithms:*

- $((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}, \mathbf{D}) \leftarrow \mathsf{RevSetup}(n)$*: The setup algorithm generates a secret key* $\mathsf{sk}_{\mathcal{R},i}$ *for each of the n RAs, a common verification key* $\mathsf{vk}_{\mathcal{R}}$*, and initializes the revocation status database* $\mathbf{D}$*.*
- $\mathsf{pt}/\perp \leftarrow \mathsf{PresentNonRev}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathit{Rvd}, \mathsf{vk}_{\mathcal{R}})$*: Generates a presentation token* pt *selectively revealing attributes* a_i *for* $i \in \mathit{Rvd}$*, and proving non-revocation of credential* σ_{cred} *under the RA's verification key* $\mathsf{vk}_{\mathcal{R}}$*. If* σ_{cred} *is revoked, it outputs* $\perp$*.*
- $0/1 \leftarrow \mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}, (a_i)_{i\in\mathit{Rvd}})$*: Verifies that a presentation token* pt *with revealed attributes* $(a_i)_{i\in\mathit{Rvd}}$ *belongs to a non-revoked credential under the issuer key* vk_I *and RA key* $\mathsf{vk}_{\mathcal{R}}$*.*

AC.Present is a client-local algorithm, but PresentNonRev requires the client to interact with the RA to produce the presentation token, aborting if the credential is revoked. As in classic anonymous credential schemes that assume issuer-verifier collusion, we account for collusion with the RA.

In contrast with issuance, we *do not require* unlinkability between obtaining a non-revocation token and its corresponding presentation session. While issuance contains information about clients, as issuers often see, choose, or attest client attributes, RAs must not learn any information about clients (see RQ.P2) during the revocation check. Moreover, real-time revocation (see RQ.F3) imposes strict timing constraints on revocation tokens, such as requiring a fresh token before each presentation, inherently creating a timing link between token retrieval and presentation.

We require anonymous credential schemes with revocation to satisfy the following five properties: *correctness, unforgeability, client unlinkability, revocation unlinkability and database privacy*. We summarize these properties below, state their corresponding trust model and defer formal definitions to the appendix (§D). *Correctness* ensures that any honestly generated presentation for a non-revoked credential always verifies, while *unforgeability* ensures that malicious clients cannot produce a valid presentation for a revoked credential even when passively compromising up to $n-1$ RAs (AM.3). Together, revocation correctness and unforgeability capture the functional requirement of revocation (RQ.F1). *Revocation unlinkability*, based on RQ.P2, focuses solely on RAs and requires that a malicious adversary controlling $n-1$ RAs (AM.2) cannot distinguish whether two non-revocation sessions originate from the same or different credentials, *regardless* of their attributes or revocation status. *Credential unlinkability* extends classic credential unlinkability and allows a malicious adversary to compromise $n-1$ RAs in addition to the issuer and verifier (AM.1) to satisfy RQ.P1; it requires distinguishing the origin of a presentation from two credentials—which are chosen by the adversary—to be hard, as long as both presentations have *the same* revealed attributes and revocation status, which are shown to the verifier. *Database privacy* prevents malicious clients (AM.4) from learning the revocation status of credentials they do not own, thus satisfying RQ.P3.

Limitations. Our model relies on an any-trust assumption for the RAs: if all RAs are compromised, they can break client privacy. Moreover, we do not guarantee unforgeability against actively malicious RAs, and actively corrupting a single RA is enough for forging non-revocation tokens, which increases the attack surface. These limitations are not inherent to revocation, but existing solutions that avoid this trust assumption incur orders-of-magnitude higher costs. Our design makes a scalability/trust tradeoff: it supports billions of credentials at the cost of stronger trust assumptions. We note that distributing the RAs does not reduce availability: as long as at least two of the n RAs remain reachable, clients can still obtain non-revocation tokens, and correctness and unforgeability are unaffected by the reduced number of available RAs.

5 Private Record Certification

As the first step toward building a revocation scheme, we define and design Private Record Certification (PRC), a primitive that enables privacy-preserving certification of a data record stored at a specific index in a database.

5.1 Definition

We define a PRC scheme as a protocol between a client and n PRC servers $\mathcal{S}_0, \ldots, \mathcal{S}_{n-1}$, with the following sub-protocols:

- $((\mathsf{sk}_i)_{i\in[n]}, \mathsf{vk}) \leftarrow \mathsf{KGen}(n)$: Generates a secret key sk_i for each PRC server $i \in [n]$, and a verification key vk.
- $((\mathsf{qry}_i)_{i\in[n]}, \mathsf{qst}) \leftarrow \mathsf{Request}(\mathsf{idx}, \sigma_{\mathsf{auth}})$: For an index idx and an optional authorization credential σ_{auth}, the Request algorithm outputs a query qry_i for each PRC server $\mathcal{S}_i$ and a client-local query state qst (e.g. randomness).
- $(\sigma_{\mathsf{rec}}, \mathsf{aux}) \leftarrow \mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i, \mathsf{qry}_i, \mathbf{D})_{i\in[n]}, \mathsf{vk}_I)$: The interactive response protocol is executed between all PRC servers $\mathcal{S} = (\mathcal{S}_i)_{i\in[n]}$, each taking their secret key sk_i, query share qry_i, optionally an authorization verification key vk_I, and the database $\mathbf{D}$ as input. As output, the protocol produces a record certificate σ_{rec} and auxiliary data aux.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{qst}, \mathsf{idx}, \mathsf{rec}, \mathsf{aux})$: For a record certificate σ_{rec} with a query state qst, the verification algorithm

checks whether σ_{rec} is valid for the verification key vk, i.e., certifies that $\mathbf{D}[\mathsf{idx}] = \mathsf{rec}$ with auxiliary data aux.

Properties. We require PRC schemes to satisfy *correctness, unforgeability, unlinkability*, and optionally *database privacy*. We give an informal description of these properties here and defer formal definitions to §C. Correctness requires that any honest execution yields a valid record certificate. Unforgeability ensures that no adversary can produce a valid certificate for a falsified record. In particular, it prevents modifying either the auxiliary data, which may include metadata such as a timestamp, or the record relation, i.e., verifying a certificate for a pair $(\mathsf{idx}, \mathsf{rec})$ where the relation $\mathbf{D}[\mathsf{idx}] = \mathsf{rec}$ does not hold. Unlinkability ensures that the PRC servers learn no information about the clients' identity or record. We model this by requiring that no subset of PRC servers can determine the order of two requests from client's chosen by them. Finally, *database privacy* prevents clients from learning record values for indices for which they do not hold an authorization credential.

Threat Model & Trust Assumptions. PRC is a cryptographic primitive, in which we allow for *malicious* and *semi-honest* corruptions, as defined in §4.1. We use the same threat model as in §4.1. As described above, we consider different adversarial models for different properties. Consider the following models: 1) *Unforgeability:* An adversary maliciously compromises the client and up to $n-1$ PRC servers are semi-honest (resembling AM.1). 2) *Unlinkability:* An adversary maliciously compromises up to $n-1$ PRC servers (corresponding to AM.2, AM.3). 3) *DB privacy:* An adversary maliciously compromises the client (similar to AM.4).

5.2 Our PRC Construction at a High Level

We provide a high-level overview of our PRC protocol, but first, we study a naive solution to discuss how our problem differs from PIR and how this impacts the challenges we face.

At first glance, PRC resembles PIR, where a client retrieves a record $\mathsf{rec} = \mathbf{D}[\mathsf{idx}]$ without revealing the index idx to the servers. However, there is a fundamental difference: In PIR, the client learns an unknown value rec, whereas in PRC, the client already knows rec and seeks a certificate attesting a statement about a record held at a private index $\mathbf{D}[\mathsf{idx}]$ to third parties. For example, downloading the entire database satisfies the PIR functionality, but offers no benefit for PRC, where the client aims to prove a statement about the database value at a private index. For our revocation use case, we additionally want certificates to admit efficient proofs-of-knowledge, avoiding the need to reveal idx and rec for verification.

Naive solution. A straightforward approach to construct a PRC protocol is to have the server(s) maintain a secondary database holding signed commitments to each record and its index. For a suitable signature scheme, certificates instantiated this way admit efficient proofs. This reduces the PRC problem to clients retrieving the corresponding signature on a commitment via PIR. Concretely, the commitment database contains $\mathbf{CD}[i] = (\sigma_i, \mathsf{rnd}_i)$, where σ_i is a signature on $(\mathsf{co}_i \| \mathsf{aux})$ such that $\mathsf{co}_i = \mathsf{Com.Commit}((i, \mathbf{D}[i]); \mathsf{rnd}_i)$ and aux is auxiliary information, such as a timestamp. However, this naive construction has three major issues: 1) For frequently updated or time-sensitive databases (e.g., using a timestamp or expiration date as auxiliary data), PRC servers must re-sign all records in each time epoch. 2) The records in $\mathbf{CD}$ are often substantially larger than the original records, especially in cases of binary records such as validity status. Each entry includes a signature σ_i (e.g., 64 bytes for Schnorr [94] or 112 bytes for BBS+ [4]) and the commitment randomness rnd_i (32 bytes) resulting in an expansion of over $768\times$ in the database size for binary records and a proportional increase in PIR cost, which scales linearly with the database size. 3) Extending the naive PIR solutions to provide database privacy (often called Symmetric PIR or Oblivious Transfer) i.e., preventing clients from querying records they are unauthorized to access, is expensive.

We implement and benchmark this naive approach using state-of-the-art PIR schemes in both single- and multi-server settings in §7. Even ignoring issues (1) and (3), high update costs and lack of database privacy, our proposed solution outperforms this naive baseline by a factor of $1000\times$.

Our high-level approach. To overcome these issues, we construct our PRC protocol such that the PRC servers only hold a database $\mathbf{D}$ of the actual record values and compute record certificates on the fly using a tailored, efficient, secure MPC protocol. At a high level, our PRC scheme works as follows.

The client sends a Boolean secret sharing of their requested index idx and an arithmetic secret sharing of a randomness rnd to the PRC servers $\mathcal{S}_i$. Given these shares, the PRC servers collectively compute the secret sharing of the requested record value $[\![\mathsf{rec} = \mathbf{D}[\mathsf{idx}]]\!]^{\mathrm{A}}$ combining techniques from PIR and MPC. Next, each $\mathcal{S}_i$ commits to its (arithmetic) share of the index and record value using the randomness share provided by the client as $\mathsf{co}_i \leftarrow \mathsf{Commit}(([\![\mathsf{idx}]\!]^{\mathrm{A}}_i, [\![\mathsf{rec}]\!]^{\mathrm{A}}_i); [\![\mathsf{rnd}]\!]^{\mathrm{A}}_i)$. By ensuring that our arithmetic sharing operates in a prime modulus $\mathbb{Z}_q$ consistent with the group order of our homomorphic commitment scheme Com, aggregating the individual commitments implicitly reconstructs to the index and record value in the commitment $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Agg}((\mathsf{co}_i)_{i\in[n]}) = \mathsf{Commit}((\mathsf{idx}, \mathsf{rec}); \mathsf{rnd})$. Lastly, the PRC servers compute a threshold signature σ_{rec} on the aggregated commitment $\mathsf{co}_{\mathsf{rec}}$ and auxiliary information aux. Due to the hiding property of the commitment, the PRC servers do not learn any information about the committed index and record shares of other servers. The client can now prove statements about the relation between their index idx and its corresponding record rec, contained in the same commitment $\mathsf{co}_{\mathsf{rec}}$, on which they have obtained a signature σ_{rec} valid under the verification key vk.

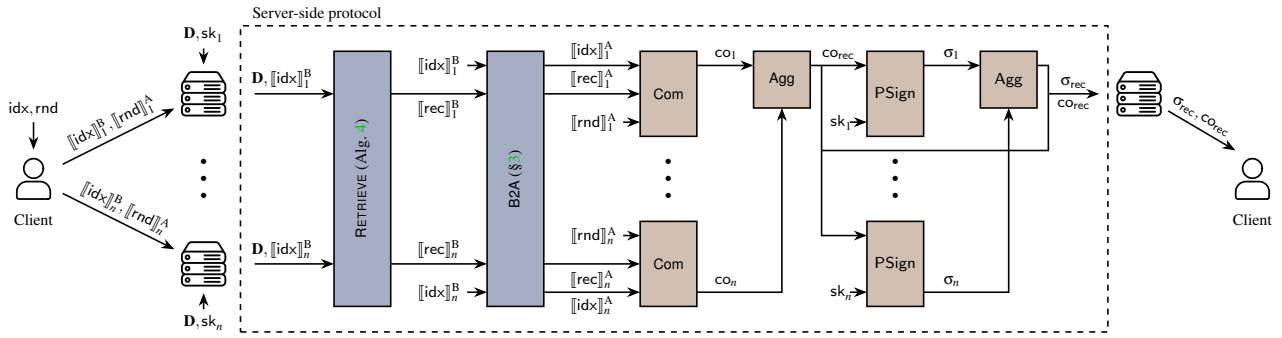

Figure 2: Overview of our PRC construction. Algorithms based on secret sharing are shown in blue, and those using asymmetric operations are shown in brown. For simplicity, auxiliary information aux is omitted.

Algorithm 1 Key generation, request, and verify procedures of PRC.

function $\mathsf{KGen}(n)$
1: $((sk_i)_{i\in[n]}, \mathsf{vk}) \leftarrow \mathsf{KGen}(n)$
2: **return** $((sk_i)_{i\in[n]}, \mathsf{vk})$

function $\mathsf{Request}(\mathsf{idx}, \sigma_{\mathsf{auth}} = \bot)$
3: $\mathsf{rnd} \xleftarrow{\$} \mathbb{Z}_q$
4: $[\![\mathsf{rnd}]\!]^{\mathrm{A}} \leftarrow \mathsf{SS.Share}^{A}(\mathsf{rnd})$
5: $[\![\mathsf{idx}]\!]^{\mathrm{B}} \leftarrow \mathsf{SS.Share}^{B}(\mathsf{idx})$
6: **return** $((\mathsf{qry}_i)^n := ([\![\mathsf{rnd}]\!]^{\mathrm{A}}_i, [\![\mathsf{idx}]\!]^{\mathrm{B}}_i)^n, \mathsf{qst} := \mathsf{rnd})$

function $\mathsf{Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{qst}, \mathsf{idx}, \mathsf{rec}, \mathsf{aux})$
7: $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Commit}((\mathsf{idx}, \mathsf{rec}); \mathsf{qst})$
8: **return** $\mathsf{TSig.Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}} \| \mathsf{aux})$

5.3 Constructing Private Record Certification

We now construct a basic version of PRC, as defined in §5.1, omitting the optional authorization credential σ_{auth}. In §5.4, we then extend this protocol to achieve database privacy. We present PRC algorithms executed by a single party in Alg. 1, and the more complex multi-party Response protocol separately in Alg. 2. The full construction is illustrated in Figure 2. We construct our PRC scheme as follows:

- $((\mathsf{sk}_i)_{i\in[n]}, \mathsf{vk}) \leftarrow \mathsf{KGen}(n)$: The key generation uses a threshold signature scheme TSig to generate secret keys sk_i for each of the n servers and a public verification key vk.
- $((\mathsf{qry}_i)_{i\in[n]}, \mathsf{qst}) \leftarrow \mathsf{Request}(\mathsf{idx}, \bot)$: The request algorithm samples commitment randomness $\mathsf{rnd} \xleftarrow{\$} \mathbb{Z}_q$ and then secret shares rnd via an arithmetic sharing mod q, where q is the commitment group order, and the index idx via Boolean sharing. The client sends the shares $\mathsf{qry}_i = ([\![\mathsf{rnd}]\!]^{\mathrm{A}}_i, [\![\mathsf{idx}]\!]^{\mathrm{B}}_i)$ to the corresponding server and stores the state $\mathsf{qst} := \mathsf{rnd}$.
- $(\sigma_{\mathsf{rec}}, \mathsf{aux}) \leftarrow \mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i, \mathsf{qry}_i, \mathbf{D})_{i\in[n]}, \bot)$: The following response protocol (Alg. 2) is run among the servers: 1) Given the Boolean-shared index $[\![\mathsf{idx}]\!]^{\mathrm{B}}_i$ for each PRC server $\mathcal{S}_i$, the PRC servers interactively compute their sharing of the record value as $[\![\mathsf{rec}]\!]^{\mathrm{B}} \leftarrow \textsc{Retrieve}(\mathbf{D}, [\![\mathsf{idx}]\!]^{\mathrm{B}})$ using Alg. 4. We explain the retrieval algorithm in more detail below. 2) The PRC servers convert the Boolean-shared index $[\![\mathsf{idx}]\!]^{\mathrm{B}}_i$ and record value $[\![\mathsf{rec}]\!]^{\mathrm{B}}_i$ to arithmetic shares $[\![\mathsf{idx}]\!]^{\mathrm{A}}_i$ and $[\![\mathsf{rec}]\!]^{\mathrm{A}}_i$ using B2A, see §3. 3) They commit to their arithmetic index and record share using the randomness shares provided by the client as $\mathsf{co}_i \leftarrow \mathsf{Com.Commit}(([\![\mathsf{idx}]\!]^{\mathrm{A}}_i, [\![\mathsf{rec}]\!]^{\mathrm{A}}_i); [\![\mathsf{rnd}]\!]^{\mathrm{A}}_i)$ and 4) aggregate all commitments as $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Agg}((\mathsf{co}_i)_{i\in[n]})$. 5) They individually compute partial signatures $\sigma_i \leftarrow \mathsf{TSig.PSign}(\mathsf{sk}_i, \mathsf{co}_{\mathsf{rec}} \| \mathsf{aux})$ on the aggregated commitment $\mathsf{co}_{\mathsf{rec}}$ and auxiliary information aux. 6) Lastly, they aggregate the partial signatures as $\sigma_{\mathsf{rec}} \leftarrow \mathsf{TSig.Agg}((\sigma_i)_{i\in[n]})$, and output $(\sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}}, \mathsf{aux})$ to the client.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{qst}, \mathsf{idx}, \mathsf{rec}, \mathsf{aux})$: The verification algorithm computes $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Commit}((\mathsf{idx}, \mathsf{rec}); \mathsf{qst})$ and verifies that $\mathsf{TSig.Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}} \| \mathsf{aux})$.

Algorithm 2 Multiparty response protocol for privately signing a commitment to a database index idx and its record $\mathsf{rec} = \mathbf{D}[\mathsf{idx}]$.

Public server input: Database $\mathbf{D} \in \mathbb{Z}_{2^l}^N$, auxiliary data aux, key vk.
Private input of $\mathcal{S}_i$: Secret key sk_i, query share $q_i = ([\![\mathsf{rnd}]\!]^{\mathrm{A}}_i, [\![\mathsf{idx}]\!]^{\mathrm{B}}_i)$.
Output: Tuple $(\sigma_{\mathsf{rec}}, \mathsf{aux})$, where $\mathsf{TSig.Verify}(\mathsf{vk}, \sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}} \| \mathsf{aux})$, for $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Commit}((\mathsf{idx}, \mathbf{D}[\mathsf{idx}]); \mathsf{rnd})$.
Response protocol:

1. All servers interactively compute $[\![\mathsf{rec}]\!]^{\mathrm{B}} \leftarrow \textsc{Retrieve}(\mathbf{D}, [\![idx]\!]^{\mathrm{B}})$.
2. All servers interactively convert their binary shares to arithmetic shares: $[\![\mathsf{idx}]\!]^{\mathrm{A}} \leftarrow \mathsf{SS.B2A}([\![\mathsf{idx}]\!]^{\mathrm{B}})$, $[\![\mathsf{rec}]\!]^{\mathrm{A}} \leftarrow \mathsf{SS.B2A}([\![\mathsf{rec}]\!]^{\mathrm{B}})$.
3. Each server $\mathcal{S}_i$ commits to its index and record share as $\mathsf{co}_i \leftarrow \mathsf{Com.Commit}(([[\![\mathsf{idx}]\!]^{\mathrm{A}}_i, [\![\mathbf{D}[\mathsf{idx}]]\!]^{\mathrm{A}}_i]); [\![\mathsf{rnd}]\!]^{\mathrm{A}}_i)$ and sends it to $\mathcal{S}_0$.
4. Server $\mathcal{S}_0$ aggregates all commitments: $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Agg}((\mathsf{co}_i)_{i\in[n]})$. Since the commitment is additively homomorphic and has the same group order as our arithmetic sharing, it holds that: $\mathsf{co}_{\mathsf{rec}} = \mathsf{Com.Commit}([\mathsf{idx} = \sum_i [\![\mathsf{idx}]\!]^{\mathrm{A}}_i, \mathsf{rec} = \sum_i [\![\mathsf{rec}]\!]^{\mathrm{A}}_i]; r = \sum_i [\![\mathsf{rnd}]\!]^{\mathrm{A}}_i)$.
5. Server $\mathcal{S}_0$ broadcasts commitment $\mathsf{co}_{\mathsf{rec}}$ and auxiliary data aux. Each $\mathcal{S}_i$ computes $\sigma_i \leftarrow \mathsf{TSig.PSign}(\mathsf{sk}_i, \mathsf{co}_{\mathsf{rec}} \| \mathsf{aux})$ and sends it to $\mathcal{S}_0$.
6. Server $\mathcal{S}_0$ aggregates all signatures as $\sigma_{\mathsf{rec}} \leftarrow \mathsf{TSig.Agg}((\sigma_i)_{i\in[n]})$ and sends signature σ_{rec} and auxiliary data aux to the client.

Private retrieval. Now we explain how given a database $\mathbf{D}$ of N l-bit records and a Boolean-sharing of index $[\![\mathsf{idx}]\!]^{\mathrm{B}}$, PRC servers compute a secret sharing of record $[\![\mathsf{rec}]\!]^{\mathrm{B}} =$

Algorithm 3 Converting a boolean secret sharing $[\![x]\!]^{\mathrm{B}}$ of an k-bit value $x \in \mathbb{Z}_{2^k}$ to a boolean secret sharing of its one-hot encoding.

function $\mathsf{Bin2OHE}(k, [\![x]\!]^{\mathrm{B}})$
1: **if** $k = 1$ **then**
2: **return** $[\neg[\![x]\!]^{\mathrm{B}}, [\![x]\!]^{\mathrm{B}}]$
3: $m \leftarrow \lceil k/2 \rceil$
4: $[\![\mathbf{l}]\!]^{\mathrm{B}} \leftarrow \mathsf{Bin2OHE}(m, [\![x[:m)]\!]^{\mathrm{B}})$
5: $[\![\mathbf{r}]\!]^{\mathrm{B}} \leftarrow \mathsf{Bin2OHE}(k-m, [\![x[m:)]\!]^{\mathrm{B}})$
6: **for** $i \leftarrow [2^m]$ **do**
7: **for** $j \in [2^{k-m}]$ **do**
8: $[\![\mathbf{o}[i+j2^m]]\!]^{\mathrm{B}} = [\![\mathbf{l}[i]]\!]^{\mathrm{B}} \wedge [\![\mathbf{r}[j]]\!]^{\mathrm{B}}$
9: **return** $[\![\mathbf{o}]\!]^{\mathrm{B}}$

Algorithm 4 Given a plaintext database $\mathbf{D} \in \mathbb{Z}_{2^l}^N$ of N l-bit records and a boolean secret shared index $[\![\mathsf{idx}]\!]^{\mathrm{B}}$, computes the secret sharing of the requested record $[\![\mathsf{rec}]\!]^{\mathrm{B}} = [\![\mathbf{D}[\mathsf{idx}]]\!]^{\mathrm{B}}$.

function $\textsc{Retrieve}(\mathbf{D}, [\![\mathsf{idx}]\!]^{\mathrm{B}})$
1: $\eta \leftarrow \lceil \lg(N) \rceil$
2: $m \leftarrow \lceil (\eta + \lg(l))/2 \rceil$
3: $\mathbf{D}'[i][j] \leftarrow \mathbf{D}[i2^m + j]$ ▷ Rearrange $\mathbf{D}$ as a 2-dim table
4: $[\![\mathbf{r}]\!]^{\mathrm{B}} \leftarrow \mathsf{Bin2OHE}(m, [\![\mathsf{idx}[:m)]\!]^{\mathrm{B}})$ ▷ Get row idx share
5: $[\![\mathbf{c}]\!]^{\mathrm{B}} \leftarrow \mathsf{Bin2OHE}(\eta - m, [\![\mathsf{idx}[m:)]\!]^{\mathrm{B}})$ ▷ Get column idx share
6: **for** $i \leftarrow [2^{\eta-m}]$ **do**
7: $[\![\mathbf{a}[i]]\!]^{\mathrm{B}} = \langle \mathbf{D}'[\mathbf{i}], [\![\mathbf{r}]\!]^{\mathrm{B}} \rangle$ ▷ Local operation as $\mathbf{D}'[i]$ is plain
8: $[\![\mathsf{rec}]\!]^{\mathrm{B}} = \langle [\![\mathbf{a}]\!]^{\mathrm{B}}, [\![\mathbf{c}]\!]^{\mathrm{B}} \rangle$
9: **return** $[\![\mathsf{rec}]\!]^{\mathrm{B}}$

$[\![\mathbf{D}[\mathsf{idx}]]\!]^{\mathrm{B}}$ via our $\textsc{Retrieve}$ algorithm given in Alg. 4. As we have multiple PRC servers, we base our construction on multi-server PIR protocols [31, 42, 57]. In classical multi-server PIR schemes, the client creates a secret-shared OHE of its index, which is of database size N. It then sends the share to the PRC servers, who non-interactively compute their response. We avoid this load on clients by allowing them to send Boolean shares of their index to the PRC servers, who then collaboratively compute the OHE. This expansion requires interaction, which is acceptable in our case as PRC servers already interact for the aggregation and threshold signature. This enables us to achieve a minimal client-side communication and computation cost of $\lg N$.

We start by building a conversion $\textsc{Bin2OHE}(k, [\![x]\!]^{\mathrm{B}})$ that takes a k-bit Boolean secret shared index x and recursively computes a sharing of $[\![\mathsf{OHE}(k,x)]\!]^{\mathrm{B}}$, as shown in Alg. 3. Having a 1-bit shared index $[\![x]\!]^{\mathrm{B}}$, building the OHE vector, which has two elements, is trivial as $[\![\mathsf{OHE}(1, [\![x]\!]^{\mathrm{B}})]\!]^{\mathrm{B}} = [\neg[\![x]\!]^{\mathrm{B}}, [\![x]\!]^{\mathrm{B}}]$. To build $[\![\mathsf{OHE}(k, [\![x]\!]^{\mathrm{B}})]\!]^{\mathrm{B}}$ for a k-bit index x, we treat x as vector $[\![\mathbf{x}_{\mathbf{i}\in[k]}]\!]^{\mathrm{B}}$ of k secret shared bits and split the shared bits of index into lower $[\![\mathbf{x}[:k/2)]\!]^{\mathrm{B}}$ and upper $[\![\mathbf{x}[k/2:)]\!]^{\mathrm{B}}$ halves, then recursively compute their OHEs. Finally, we build an OHE vector of size 2^k, where each bit is computed by a single AND operation, for each combination of elements between the lower and upper OHE vectors.

Now that we can convert secret shared indices to OHEs, we can apply the conversion to the client-shared index and perform a classic XOR PIR. However, expanding the index to a vector of size N requires communication linear in the number of records. Instead, we follow Chor's guideline [31] and rearrange the database as a two-dimensional table. This enables us to split the OHE index generation into two parts of size approximately $\sqrt{N}$, supporting sublinear communication cost for the OHE expansion: given an index $\mathsf{idx} = c2^m + r$ represented as $([\![c]\!]^{\mathrm{B}}, [\![r]\!]^{\mathrm{B}})$, we compute two OHE vectors $[\![\mathbf{c}]\!]^{\mathrm{B}} = \textsc{Bin2OHE}(\eta - m, c)$ and $[\![\mathbf{r}]\!]^{\mathrm{B}} = \textsc{Bin2OHE}(m, [\![r]\!]^{\mathrm{B}})$ where $\eta = \lg(N)$. The first vector $[\![\mathbf{c}]\!]^{\mathrm{B}}$ corresponds to the column index of the record in the database, while the second vector $[\![\mathbf{r}]\!]^{\mathrm{B}}$ corresponds to the row index. By performing an XOR PIR via the row index, through $[\![\mathbf{a}[i]]\!]^{\mathrm{B}} = \langle \mathbf{D}'[\mathbf{i}], [\![\mathbf{r}]\!]^{\mathrm{B}} \rangle$, we can select the desired row in the database and reduce the dimension to one. As the PRC servers hold the plaintext database $\mathbf{D}$, this operation only requires performing AND with known values, which can be done locally by each $\mathcal{S}_i$. Next, the PRC servers interact to compute the inner product of the resulting vector $[\![\mathbf{a}[i]]\!]^{\mathrm{B}}$ with the column index to retrieve the desired record value $[\![\mathsf{rec}]\!]^{\mathrm{B}} = \langle [\![\mathbf{a}]\!]^{\mathrm{B}}, [\![\mathbf{c}]\!]^{\mathrm{B}} \rangle$.

PRC assumes that clients know the record value. If this assumption does not hold, clients can either brute-force the value by verifying the token against the record domain (e.g., valid or revoked for a revocation status), or PRC servers can attach encrypted shares of the record $[\![\mathsf{rec}]\!]^{\mathrm{B}}$ to the response, effectively yielding a PIR query.

Table 2: The number of AND gates, $C(k)$, used for a k-bit $\textsc{Bin2OHE}(k,x)$ conversion. We denote overhead over the output size 2^k in parenthesis.

i	$C(i)$	i	$C(i)$	i	$C(i)$
1	0	7	164 (+28%)	13	8444 (+3.1%)
2	4 (0)	8	304 (+19%)	14	16712 (+2%)
3	12 (+50%)	9	584 (+14%)	15	33236 (+1.4%)
4	24 (+50%)	10	1120 (+9.8%)	16	66144 (+0.92%)
5	48 (+50%)	11	2184 (+6.6%)	17	131960 (+0.68%)
6	88 (+38%)	12	4272 (+4.3%)	18	263312 (+0.45%)

Performance. We analyze the computation and communication costs of our $\textsc{Retrieve}$ algorithm for PRC servers next.

First, we analyze the cost of $\textsc{Bin2OHE}$ in Alg. 3 using induction. Let $C(k)$ denote the number of AND gates required to compute $\textsc{Bin2OHE}(k,x)$, a k-bit conversion, with base case $C(1) = 0$. In lines 4–5, we recursively call the conversion, halving the index's input size, and incurring costs of $C(\lceil k/2 \rceil)$ and $C(\lfloor k/2 \rfloor)$, while the loop on lines 6–8 incurs a fixed cost of 2^k AND gates. Combined, this results in a cost of $C(k) = C(\lceil k/2 \rceil) + C(\lfloor k/2 \rfloor) + 2^k$ AND gates. We report the concrete cost of converting a k-bit index to its OHE representation and the resulting overhead relative to the 2^k-sized output vector in Table 2. The cost is bounded by $C(k) \leq 1.5 \cdot 2^k$ ANDs gates, but this overhead diminishes rapidly with increasing k.

The $\textsc{Retrieve}$ algorithm in Alg. 4 calls $\textsc{Bin2OHE}$ twice on

lines 4–5, incurring communication and computation costs of $1.5(2^{\eta-m}+2^m)$. The loop on line 6 performs an inner product between a secret-shared row index vector and the plaintext database $\mathbf{D}$. As the database is known, this computation can be performed locally leading to a computation cost $O(2^{\eta-m} \cdot l2^m = lN)$ and *no communication cost*. The final, inner product on line 8 is between two secret shared vectors of size $2^{\eta-m}$ but gets repeated for each bit of our l-bit record, leading to a computation and communication cost of $l \cdot 2^{\eta-m}$. This leads to a total computation cost of $O(lN)$, as required by the lower bound of private retrieval [59], and a communication cost of $O(2^m + (l+1) \cdot 2^{\eta-m})$. We set m in an optimized manner such that $2^m \approx l \cdot 2^{\eta-m}$, which leads to a total communication cost of $O(\sqrt{lN})$ that is sublinear in the size of the database.

We analyze the multiplicative depth of our protocol (i.e., the minimal number of sequential AND layers) as it determines the round complexity and affects system latency. Our BIN2OHE conversion has a depth of $\lg k$, as it runs two instances of size $k/2$ in parallel, then performs one AND layer. Our RETRIEVE algorithm calls two BIN2OHE conversions in parallel, then runs one local (line 7) and one interactive (line 8) AND layer to compute inner products. This results in an overall depth of $1+\lg\lg N$ as $k = \lg N$ in BIN2OHE. This low depth lets us use high-latency communication-efficient MPC protocols like GMW [55] while still achieving real-time performance (§7).

Using a threshold signature. We use threshold signatures even though our PRC construction does not protect against malicious PRC servers forging certificates, as they can modify $[\![\mathsf{rec}]\!]^{\mathrm{B}}$ in our semi-honest MPC protocol. The threshold design mainly improves availability, enables load-balancing aggregation, and simplifies verifier key management when the set of active PRC servers changes. Nevertheless, threshold signatures can be trivially replaced with other signature schemes.

5.4 Extension for Database Privacy

We extend our basic construction to support database privacy by restricting client queries to records for which they hold a valid authorization credential σ_{auth}. The authorization credential can be instantiated with a signature or anonymous credential, as long as it is unforgeable and allows for efficient zero-knowledge proofs of knowledge. To verify σ_{auth}, we assume its verification key vk_I to be known to all parties. We outline the changes in our PRC construction next.

- $((\mathsf{qry}_i)_{i\in[n]}, \mathsf{qst}) \leftarrow \mathsf{Request}(\mathsf{idx}, \sigma_{\mathsf{auth}})$: The client holds a certificate σ_{auth} on its index idx. In addition to performing the basic Request in Alg. 1, it creates a zero-knowledge proof π_{auth} for relation $\mathsf{R}_{\mathsf{auth}}$ (see Rel. 1 in §A). Specifically, it samples extra randomness $\mathsf{rnd}_{\mathsf{idx}} \xleftarrow{\$} \mathbb{Z}_q$ and secret shares for all the PRC servers, enabling to produce a second commitment on the index $\mathsf{co}_{\mathsf{idx}} \leftarrow \mathsf{Com.Commit}(\mathsf{idx}; \mathsf{rnd}_{\mathsf{idx}})$ similar to $\mathsf{co}_{\mathsf{rec}}$. The client precomputes this commitment $\mathsf{co}_{\mathsf{idx}}$ and computes the zero-knowledge proof π_{auth} proving knowledge of a certificate σ_{auth} on the same idx as in $\mathsf{co}_{\mathsf{idx}}$. It extends each query as $\mathsf{qry}_i = ([\![\mathsf{rnd}]\!]_i^{\mathrm{A}}, [\![\mathsf{rnd}_{\mathsf{idx}}]\!]^{\mathrm{A}}, [\![\mathsf{idx}]\!]_i^{\mathrm{B}}, \pi_{\mathsf{auth}})$.
- $(\sigma_{\mathsf{rec}}, \mathsf{aux}) \leftarrow \mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i, \mathsf{qry}_i, \mathbf{D})_{i\in[n]}, \mathsf{vk}_I)$: In a similar process to $\mathsf{co}_{\mathsf{rec}}$ in Alg. 2, the $\mathcal{S}_i$ compute $\mathsf{co}_{\mathsf{idx}} \leftarrow \mathsf{Com.Commit}(\sum[\![\mathsf{idx}]\!]_i^{\mathrm{A}}; \sum[\![\mathsf{rnd}_{\mathsf{idx}}]\!]_i^{\mathrm{A}})$. Then, they verify the zero-knowledge proof π_{auth} using $\mathsf{co}_{\mathsf{idx}}$ and vk_I, abort if verification fails, and send the final response to the client.

This modification does not impact the security of the protocol, which we study next. In particular, the zero-knowledge property of the proof system ensures that no information about the index idx is revealed, thus preserving unlinkability.

5.5 Security of our PRC Construction

We present proof sketches for our PRC construction here, and defer formal proofs to the full version of the paper [46].

Informal Theorem 5.1 (Unforgeability)**.** *Our PRC construction achieves unforgeability if the threshold signature scheme* TSig *is unforgeable, and the commitment scheme* Com *is binding.*

For unforgeability, we consider a malicious client and up to $n-1$ semi-honest PRC servers. Since all PRC servers follow the protocol, the client can only obtain signatures on commitments to an index and its corresponding record value and auxiliary data. A certificate forgery thus either requires opening a valid certificate to different content (commitment binding) or forging a signature.

Informal Theorem 5.2 (Unlinkability)**.** *Our PRC construction achieves unlinkability if the commitment scheme* Com *is hiding and the correlated randomness is private.*

We require the correlated randomness to be private, i.e., randomness shares of honest parties are uniform and unknown to the adversary. Our security argument is similar to the approach of Araki et. al. [3] for malicious privacy. Given private correlated randomness, we obtain protection against malicious PRC servers even though we use a semi-honest MPC protocol [55]. An adversary may tamper with shares, breaking integrity and introducing additive errors; however, since no output value is reconstructed, these errors remain hidden and leak no information. In more detail, our construction consists of two components: 1) a semi-honest MPC protocol computing $([[\![\mathsf{idx}]\!]_i^{\mathrm{A}}, [\![\mathbf{D}[\mathsf{idx}]]\!]_i^{\mathrm{A}}])$ and 2) a custom-made protocol using asymmetric operations (commitment and signature). At the end of the MPC computation, each party commits to its share using the client-chosen randomness rnd_i and publishes the commitment co_i. Assume that $\mathcal{S}_h$ is the non-colluding PRC server. By the hiding property of the commitment, co_h reveals no information about its content $([\![\mathsf{idx}]\!]_h^{\mathrm{A}}, [\![\mathsf{rec}]\!]_h^{\mathrm{A}}; [\![\mathsf{rnd}]\!]_h^{\mathrm{A}})$. We assume that the commitment aggregation is adversarial. If the aggregation includes co_h,

then the randomness rnd_h protects the output $\mathsf{co}_{\mathsf{rec}}$ even if the adversary tampers with co_h. Otherwise, the resulting aggregation contains no share from $\mathcal{S}_h$, and is therefore independent of the client's input by the security of the secret sharing scheme.

Informal Theorem 5.3 (Database Privacy)**.** *Our extended protocol (§5.4) achieves database privacy if the authorization credential is unforgeable and the zero-knowledge proof system is knowledge-sound.*

As all PRC servers are honest, the protocol response only contains information about the requested index. By the knowledge-soundness of the proof system, clients can only generate valid proofs if they know an authorization credential. Further, if the authorization credentials are unforgeable, clients learn nothing about unauthorized indices.

6 Anonymous Credential Revocation via PRC

We construct an anonymous credential revocation system, as defined in §4.3, based on our PRC scheme from §5.2 and an anonymous credential scheme that supports efficient zero-knowledge proofs. Our system includes an issuer I that issues credentials and a decentralized RA $\mathcal{R}$ of n PRC servers that manage a binary revocation status database $\mathbf{D}$. W.l.o.g., we assume that the issuer sets the first attribute a_0 of each credential to the corresponding revocation index idx and a credential is valid if $\mathbf{D}[\mathsf{idx}] = 1$. As our system is focused on revocation, we omit the issuer in the rest of the section. Our construction works as follows:

- $((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}, \mathbf{D}) \leftarrow \mathsf{RevSetup}(n)$: Initialize a status database $\mathbf{D}$ and a PRC instance $((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}) \leftarrow \mathsf{PRC.KGen}(n)$, which produces the verification key $\mathsf{vk}_{\mathcal{R}}$.
- $\mathsf{pt}/\perp \leftarrow \mathsf{PresentNonRev}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd}, \mathsf{vk}_{\mathcal{R}})$: The client generates a presentation including a non-revocation proof for a credential σ_{cred} with index $\mathsf{idx} = a_0$ as follows: 1) the client performs a PRC request for idx with the RAs, who set $\mathsf{aux} \leftarrow \mathsf{time}$ as a timestamp. It receives $(\sigma_{\mathsf{rec}}, \mathsf{time})$ and stores the query state qst. 2) the client checks if $\mathsf{PRC.Verify}(\mathsf{vk}_{\mathcal{R}}, (\sigma_{\mathsf{rec}}, \mathsf{qst}), \mathsf{idx} = a_0, \mathsf{rec} = 1, \mathsf{time})$, returning $\perp$ if the check fails. 3) the client computes a zero-knowledge proof π_{rev} for relation $\mathsf{R}_{\mathsf{rev}}$ (cf. Rel. 2), proving knowledge of a valid credential σ_{cred} with revealed attributes $(a_i)_{i\in\mathsf{Rvd}}$, where the PRC commitment $\mathsf{co}_{\mathsf{rec}}$ opens to a valid status ($\mathsf{rec} = 1$) and the same index idx as the hidden attribute a_0. 4) the client outputs $\mathsf{pt} := (\sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}}, \mathsf{time}, \pi_{\mathsf{rev}})$.
- $0/1 \leftarrow \mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}, (a_i)_{i\in\mathsf{Rvd}})$: The verifier parses $\mathsf{pt} := (\sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}}, \mathsf{time}, \pi_{\mathsf{rev}})$ and checks 1) if the timestamp is valid and within an application-specific limit, 2) if $\mathsf{TSig.Verify}(\mathsf{vk}_{\mathcal{R}}, \sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}} || \mathsf{time})$, i.e., the PRC certificate is valid, and 3) if proof π_{rev}, which links the credential presentation to the non-revocation token, is valid.

Our revocation scheme can be extended to achieve database privacy and satisfy RQ.P3 by using σ_{cred} as an authorization credential in the extended PRC construction (§5.4). We also support multi-show revocation tokens, by employing a ZK-friendly or re-randomizable signature in the PRC construction. We discuss these extensions in more detail in §B.

Security analysis. Next, we provide an intuition why our construction fulfills the security properties defined in §4.3. Other credential properties of interest, such as device-binding, are inherited by the underlying credential scheme and out of scope of our discussion. Our construction links a PRC non-revocation token to a credential presentation with a zero-knowledge proof, ensuring that both components use the same index idx without revealing it. Given the completeness and knowledge-soundness of our proof system, revocation correctness and unforgeability follow directly from the properties of PRC and the underlying credential scheme. In addition, database privacy trivially follows from PRC. Revocation unlinkability follows from the unlinkability of the PRC scheme, since the adversary only observes the non-revocation token retrieval and not the credential presentation, which an honest verifier obtains. Client unlinkability requires the adversary to choose challenge credentials that have the same revealed attributes and revocation status, leading to the following two cases: 1) If both credentials are revoked, the presentations abort after retrieving an invalid non-revocation token, and unlinkability follows from PRC. 2) If both credentials are valid, the client retrieves a non-revocation token with a fixed status $\mathsf{rec} = 1$ and presents it alongside their credential. While the adversary knows the one-to-one mapping between retrieval and presentation sessions, unlinkability follows from combining the unlinkability of the PRC and the credential scheme.

Instantiation. We can efficiently instantiate our PRC scheme with a simple signature scheme like Schnorr [94], or threshold variants like [9, 32, 70], and the Pedersen commitment scheme [86]. The anonymous credential scheme may be instantiated with BBS/BBS+ [23, 99], or PS [87] credentials. These schemes support efficient zero-knowledge proofs over relation $\mathsf{R}_{\mathsf{rev}}$ (see Rel. 2), proving the credential and commitment both contain the same index, via a standard Σ-proof of discrete logarithm equality.

7 Implementation & Evaluation

We implement a prototype of our construction from §5.3, evaluate its performance, and compare the cost of our credential revocation against the state-of-the-art revocation schemes and the naive PIR-based baseline in §5.2.

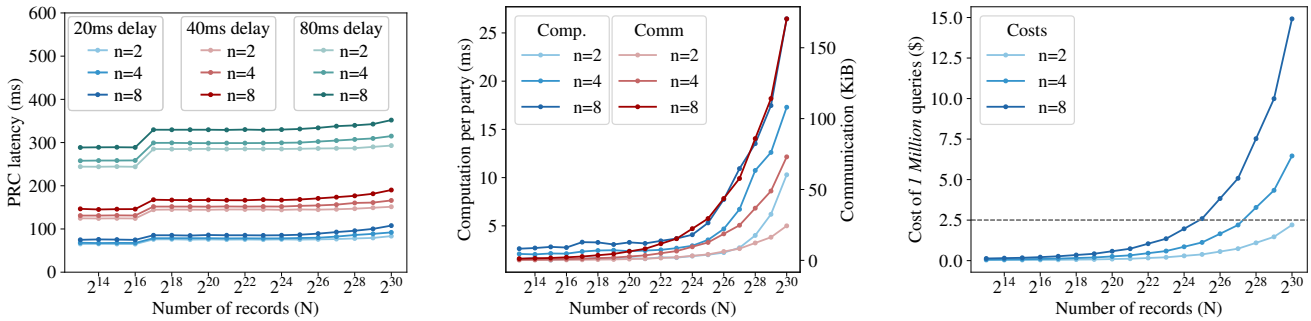

Figure 3: Evaluating the PRC latency (left), *single-core* computation and one-way communication per PRC server (middle), and the operational cost per PRC server (right) over various number of records in database (N) and number of PRC servers (n).

7.1 Implementation and Setup

We provide an open-source implementation of our PRC construction for single-bit records in Rust[1]. We rely on AVX2 to massively parallelize our operations and pack eight status bits in each single byte. AVX2 enables vectorized bitwise operations over 256-bit registers, supporting up to 256 logical operations per cycle while running on a single CPU core.

Experimental setup. We run the experiments on a machine with an AMD Ryzen AI 7 PRO 350 and 64 GB of RAM. We emulate running n PRC servers on a single machine. We rely on TCP for communication in our prototype, while using the 'tc' command to shape the network and set the round-trip time (RTT). All experiments are performed in a single core setting, and we use User+System time to report the computation time.

Communication channel security. Our prototype omits TLS management and certificate handling, and uses TCP for simplicity; a real deployment, however, must use TLS or an equivalent secure channel. A single TLS handshake already exceeds the client's total communication cost for an entire PRC execution, highlighting the communication efficiency of our client-side protocol. In contrast, PRC servers can maintain persistent communication channels among themselves because they handle many requests, and are therefore not impacted by the cost of initiating secure connections. Client-side TLS overhead can be reduced by opening a single TLS connection, onion-encrypting the queries for the other PRC servers, and letting one server relay them over persistent connections, similarly to our offline-client extension (see §B).

Instantiation. We instantiate our MPC using the GMW protocol [55], our homomorphic commitment with the Pedersen scheme over the prime-order Ristretto group [41, 60], and our signatures with Ed25519 [8, 66]. While GMW achieves highly efficient communication without the linear security parameter expansion common in garbled circuits [102], it requires one interaction round per layer of AND gates, often resulting in high latency since typical circuits have thousands of layers. By designing tailor-made protocols with depth $O(\lg\lg N)$ (see §5.3), we achieve both low communication and low latency.

Preprocessing. Efficient SMC protocols heavily rely on preprocessing to reduce their latency and online phase cost. We use Beaver triples [7] for performing AND operations and daBits [47, 92] for B2A conversions. The majority of MPC protocols have expensive preprocessing phases requiring GiBs of communication per execution, however, our protocol only requires sub-linear preprocessing material; specifically, $\sim 3\sqrt{N}$ Beaver triples and $\lg(N)+l$ dabits requiring a transfer of 13.5 KiB for all PRC servers combined when generated via a dealer in our largest setting with $N = 1$ billion credentials. Since our preprocessing is generic and function-independent, we can generate material in large batches and use it across many requests, for lower amortized costs. Correlation generation heavily depends on the deployment scenario, as the cost and threat model often differ between preprocessing and online phases. Benchmarking generic pre-processing generation is out of scope of this work. We use a dealer with pre-shared keys and discuss costs and alternatives in the appendix §E.

7.2 Performance

We evaluate the performance of our PRC and revocation system, comparing it with the baseline and existing solutions.

Latency. Our system is designed to achieve low latency, and our MPC computation only requires six rounds of interactions. Most of this round complexity arises from the Boolean-sharing to one-hot-encoding conversion (Alg. 3), which has depth $\lg\lg(N)$ and requires four rounds for databases with $N = 1$ billion records. The retrieval step (Alg. 4) requires a single round to compute the second inner product. We further adopt a depth-optimized, single-round, Boolean to arithmetic conversion based on [37] using DaBit [92]. We evaluate the latency of our PRC from receiving a query to generating the token σ_{rec}. The left plot in Figure 3 shows system latency as the number of PRC servers varies, with RTTs ranging from 20 ms to 80 ms. Our solution achieves *real-time* responses

[1] https://zenodo.org/records/20433787

with latency below 400 ms. The plot also indicates that latency is dominated by the network delay rather than the number of credentials, servers, or computation cost. We observe a step increase on all curves after 2^{16} records, consistent with our theoretical analysis, as BIN2OHE requires an additional round once the index size exceeds 2^8 bits.

Cost. We measure the computation and communication cost of our scheme for different numbers of PRC servers and database records and report the results in the middle plot of Figure 3. We observe a constant cost of $\sim$1 ms for small databases, beyond which the cost grows linearly with the number of records. Using vectorized AVX2 operations, our system achieves a minimal single-core computation cost lower than 25 ms even for $N = 1$ billion records. The communication cost of PRC is sublinear in the database size, $O(\sqrt{Nl})$, as shown in §5.3, and concretely requires about $3\sqrt{N}$ AND gates and two conversion gates for 1-bit and $\lg(N)$-bit values in our revocation system. We use pairwise broadcast to reconstruct Beaver triples and daBits among n PRC servers, which incurs a quadratic cost in n, but has low impact in practice since n is usually small. As uploads and downloads are symmetric, we report only the one-way communication cost per PRC server.

The client cost is minimal and independent of the number of revocations—revocation simply changes values in the database with no impact on size—requiring $36n + 72$ Bytes of communication and negligible computation. Clients sends a query consisting of secret shares of randomness rnd (32 B) and index idx (4 B) to every one of n PRC servers. A single server responds with a token containing the signature (64 B) and auxiliary data (an 8 B timestamp); we omit the commitment since the client knows rnd and the record value, and can locally compute the commitment and verify the signature.

Monetary cost. We use Amazon EC2 pricing [1] to estimate the cost of operating a PRC server. Each server uses a single CPU core and less than 2 GiB of RAM, so we consider low-end EC2 instances priced at \$0.04 per computation hour, with free inbound data transfer and \$0.09 per 1 GiB of outbound traffic. We report the estimated cost under this model in the right plot of Figure 3. The cost scales linearly with the number of records. Our protocol is highly cost-effective, processing *one million requests* for under \$2.5 per PRC server.

Presenting credentials. Adding revocation support introduces the overhead of obtaining a PRC token and linking it to the credential via a proof over the equality of the committed index (see Rel. 2 on page). As shown in §5.3, the PRC cost is minimal for clients, and the same holds for the proof, which can be realized using a single discrete-log equality proof. The verifier overhead is limited to verifying a signature and the equality proof, both small and comparable to a non-anonymous revocation check. Since these costs are constant and dominated by the underlying credential and signature schemes, we omit benchmarks for clients and verifiers.

Comparison. We compare our work with ALLOSAUR [65] and the PIR-based baseline from §5. ALLOSAUR is a credential revocation scheme with a threshold threat model, but with cost linear in the number of revocations. We provide a detailed comparison in §F for varying number of credentials and revocations. Since ALLOSAUR does not scale for 10 million or more revocations, we extrapolate its server runtime for $N = 1$ billion credentials and a 1% revocation rate to 2663 seconds, while our solution requires only 10.294 ms. We instantiate the PIR baseline with the state-of-the-art single-server (via Hintless [74]) and multi-server (via DPF [11]) PIR schemes with 96 byte tokens $\mathbf{CD}[i] = (\sigma_i, \mathsf{rnd}_i)$. For multi-party settings, we use $n = 2$ servers and report per server revocation cost in Figure 4. As expected, all costs grow linearly with the number of credentials. Existing PIR implementations cannot handle the large databases required in our setting (e.g., 96 GiB for one billion credentials), so we extrapolate costs when systems crash and denote these regions with dashed lines. For $N = 1$ billion credentials, our solution improves revocation computation cost by a factor of 2700× over [11] and 900× over [74]. Excluding the prohibitive cost of maintaining such large and frequently changing databases.

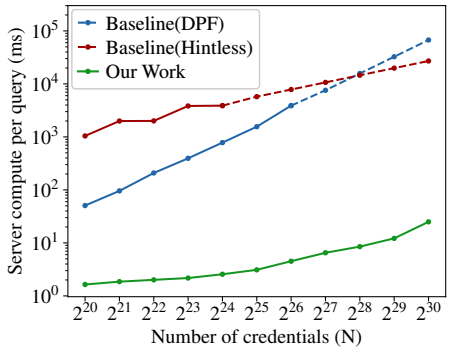

Figure 4: Per-server compute for a query of the PIR baselines, instantiated with Hintless [74] and a DPF [11], varying the number of credentials (N).

8 Conclusion

We have presented an anonymous credential revocation system that satisfies all requirements defined in §4.2. We have shown that our scheme achieves the revocation functionality (RQ.F1) and the privacy properties in §5.5. We now demonstrate that the remaining requirements are also met.

Functionality. In our system, credentials can be dynamically revoked with a single bit-flip, with no overhead for clients or verifiers (RQ.F2). We achieve latency below 400 ms with precise timestamps and full privacy, eliminating long token validity periods and enabling real-time revocation (RQ.F3).

Efficiency. Our system scales to $N = 1$ billion credentials with under 25 ms of computation per RA, far exceeding the

requirement to support 100 million users (RQ.D1). As we do not amortize costs, our system also satisfies single-query efficiency (RQ.D3). Requesting a non-revocation token requires negligible computation and 144 B of transfer from clients in a two-server setting, overshadowed by TLS overhead. Token presentation is comparable to non-anonymous revocation, adding a minimal overhead of verifying a signature and a discrete-log equality proof, thus supporting lightweight clients and verifiers (RQ.D2).

Acknowledgment. The authors have used LLM-based tools to refine the text and improve flow, conciseness, and phrasing. This project received funding from the ERC under the EU's research and innovation programs Horizon Europe (PRIVTOOLS/101124778 and CRYPTOLAYER/101044770). It was co-funded by the DFG within SFB 1119 CROSSING/236615297, and supported by BMFTR and HMWK within ATHENE.

Ethical Considerations

We provide a stakeholder-based ethical analysis below.

Stakeholder Identification and Impacts. Entities that may be impacted by our research include the research team, the research society as a whole, users of anonymous credential systems, and finally, operators, including governments, of (anonymous) credential systems. For each of the stakeholders, we consider the ethical implications of our work, using the principles listed in the Menlo Report.

Ethical Principles. We consider the principles listed in the Menlo Report to argue about the ethical impact of our work.
Beneficience We propose a new system with beneficial properties, which we assess using benchmarks. All of our benchmarks were conducted in a controlled environment and did not impair any real humans or system users in any way.
Respect for Persons We did not interfere with the personal rights of any of the stakeholders. All experiments were conducted in a controlled environment using synthetic data and do not process, collect, or analyze any personal or otherwise sensitive information.
Justice The research outcomes do not exclude or disadvantage any specific group. Our prototype code is open-source, which enables others to build on top of our results.
Respect for Law and Public Interest Our work does not violate any laws or regulations. Our tests were run using open-source software or using our own implementation. Our research helps to mitigate accountability issues of anonymous credentials, which protect users from surveillance and support users' right to privacy. Our research not only follows European regulation, but it also helps to achieve the protection measures desired in the regulation, while improving the scalability and efficiency.

Harms and Mitigations. This work is purely constructive and does not involve any attacks, exploitation of vulnerabilities, or interaction with deployed systems or real users. For the research team, the research society, and the users of anonymous credential systems, there is no harm in publishing this work. For operators of anonymous credential systems, there may be financial considerations necessary when deciding whether to implement our work in a production environment. Additional infrastructure may be required, which may be supplied by the credential operator. We believe that a more efficient system, potentially leading to a larger adoption, is only in the best interest of the credential operator.

Decision to Proceed with Publication. In summary, this work does not raise any ethical concerns. This work addresses anonymous revocation, which is a critical challenge in anonymous credential systems. We improve on both accountability by enabling authorities to revoke invalid or fraudulent credentials in a right-respecting manner and users to revoke lost or stolen credentials while keeping the strong privacy of anonymous credentials intact. As this work benefits users and society without creating any major risk, we have decided to conduct and publish our findings.

Open Science

Our prototype is available at https://zenodo.org/records/20433787. It implements our PRC construction and includes a Docker container for benchmarking. We fully reproduce the PRC cost measurements shown in Figure 3. Presenting a credential with non-revocation requires generating an anonymous credential presentation together with a zero-knowledge proof of equality of discrete logarithm in addition to obtaining a non-revocation token via PRC. This additional constant client cost depends only on the credential scheme and can be added externally. For comparisons with the naive PIR approach (cf. Figure 4) and ALLOSAUR (cf. Figure 14), we document the external runs and cost extrapolations without including third-party repositories in our artifact.

References

[1] Amazon Web Services. Amazon EC2 Pricing, 2025. https://aws.amazon.com/ec2/pricing/, Accessed: 2025-06-04.

[2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy*, pages 962–979. IEEE Computer Society, 2018.

[3] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *CCS*, pages 805–817. ACM, 2016.

[4] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic *k*-taa. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.

[5] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *EuroS&P*, pages 301–315. IEEE, 2017.

[6] Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, René Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, et al. Cryptographers' feedback on the eu digital identity's arf. *Tech. Rep.*, 2024.

[7] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[8] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 124–142. Springer, 2011.

[9] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[10] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *CRYPTO*, volume 11694 of *Lecture Notes in Computer Science*, pages 67–97. Springer, 2019.

[11] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *SP*, pages 762–776. IEEE, 2021.

[12] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In *CRYPTO*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633. Springer, 2022.

[13] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518. Springer, 2019.

[14] Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Ariel Nof. Preprocessing for life: Dishonest-majority MPC with a trusted or untrusted dealer. In *SP*, pages 2433–2452. IEEE, 2025.

[15] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Information-theoretic distributed point functions. In *ITC*, volume 230 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[16] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Programmable distributed point functions. In *CRYPTO*, volume 13510 of *Lecture Notes in Computer Science*, pages 121–151. Springer, 2022.

[17] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Sublinear gmw-style compiler for MPC with preprocessing. In *CRYPTO*, volume 12826 of *Lecture Notes in Computer Science*, pages 457–485. Springer, 2021.

[18] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.

[19] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[20] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415. Springer, 2007.

[21] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-rate PIR for databases with small records. In *CCS*, pages 1463–1477. ACM, 2024.

[22] Jan Camenisch, Manu Drijvers, and Jan Hajny. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs. In *WPES@CCS*, pages 123–133. ACM, 2016.

[23] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *TRUST*, volume 9824 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2016.

[24] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2009.

[25] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In *SAC*, volume

9566 of *Lecture Notes in Computer Science*, pages 3–24. Springer, 2015.

[26] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.

[27] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002.

[28] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN*, Lecture Notes in Computer Science. Springer, 2002.

[29] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[30] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.

[31] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, pages 41–50. IEEE Computer Society, 1995.

[32] Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical schnorr threshold signatures without the algebraic group model. In *CRYPTO*, volume 14081 of *Lecture Notes in Computer Science*, pages 743–773. Springer, 2023.

[33] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, volume 12105 of *Lecture Notes in Computer Science*, pages 44–75. Springer, 2020.

[34] Claude Crépeau. Verifiable disclosure of secrets and applications (abstract). In *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 150–154. Springer, 1989.

[35] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123. Springer, 1995.

[36] Elizabeth C. Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In *ASIACRYPT*, volume 14439 of *Lecture Notes in Computer Science*, pages 348–382. Springer, 2023.

[37] Ivan Damgård, Daniel Escudero, Tore Kasper Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. In *IEEE Symposium on Security and Privacy*, pages 1102–1120. IEEE, 2019.

[38] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[39] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[40] Alex Davidson, Gonçalo Pestana, and Sofía Celi. Frodopir: Simple, scalable, single-server private information retrieval. *Proc. Priv. Enhancing Technol.*, 2023(1):365–383, 2023.

[41] Henry de Valence, Jack Grigg, Mike Hamburg, Isis Lovecruft, George Tankersley, and Filippo Valsorda. The ristretto255 and decaf448 groups. RFC 9496, Internet Engineering Task Force, 2023.

[42] Daniel Demmler, Amir Herzberg, and Thomas Schneider. RAID-PIR: practical multi-server PIR. In *CCSW*, pages 45–56. ACM, 2014.

[43] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold BBS+ signatures for distributed anonymous credential issuance. In *SP*, pages 773–789. IEEE, 2023.

[44] Moumita Dutta, Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Compute, but verify: Efficient multiparty computation over authenticated inputs. In *ASIACRYPT*, volume 15489 of *Lecture Notes in Computer Science*, pages 133–166. Springer, 2024.

[45] Zeev Dvir and Sivakanth Gopi. 2-server PIR with subpolynomial communication. *J. ACM*, 63(4):39:1–39:15, 2016.

[46] Kasra Edalatnejad, Sebastian Faust, Jonas Hofmann, Philipp-Florens Lehwalder, and Thomas Schneider. Do you need a receipt? anonymous credential revocation at continental scale via private record certification. *IACR Cryptol. ePrint Arch.*, 2026.

[47] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In

CRYPTO, volume 12171 of *Lecture Notes in Computer Science*, pages 823–852. Springer, 2020.

[48] European Data Protection Supervisor. TechDispatch #3/2025 - Digital Identity Wallets, 2025. https://www.edps.europa.eu/data-protection/our-work/publications/techdispatch/2025-12-15-techdispatch-32025-digital-identity-wallets_en, Accessed: 2026-03-21.

[49] European Parliament and Council of the European Union. Regulation (EU) 2024/1183 of 11 april 2024 amending regulation (EU) no 910/2014 as regards establishing the european digital identity framework, April 2024. Official Journal of the European Union, L 202401183. ELI: http://data.europa.eu/eli/reg/2024/1183/oj.

[50] Sebastian Faust, Carmit Hazay, David Kretzler, Leandro Rometsch, and Benjamin Schlosser. Non-interactive threshold BBS+ from pseudorandom correlations. In *CT-RSA*, volume 15598 of *Lecture Notes in Computer Science*, pages 198–222. Springer, 2025.

[51] Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316. Springer, 2004.

[52] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464. Springer, 2019.

[53] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[54] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658. Springer, 2014.

[55] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[56] Michael T. Goodrich, Roberto Tamassia, and Jasminka Hasic. An efficient dynamic and distributed cryptographic accumulator. In *ISC*, volume 2433 of *Lecture Notes in Computer Science*, pages 372–388. Springer, 2002.

[57] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. Gpu-accelerated PIR with client-independent preprocessing for large-scale applications. In *USENIX Security Symposium*, pages 1759–1776. USENIX Association, 2022.

[58] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. Gpu-accelerated PIR with client-independent preprocessing for large-scale applications. In *USENIX Security Symposium*, pages 1759–1776. USENIX Association, 2022.

[59] Iftach Haitner, Jonathan J. Hoch, and Gil Segev. A linear lower bound on the communication complexity of single-server private information retrieval. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2008.

[60] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In *CRYPTO*, volume 9215 of *Lecture Notes in Computer Science*, pages 705–723. Springer, 2015.

[61] Lukas Helminger, Daniel Kales, Sebastian Ramacher, and Roman Walch. Multi-party revocation in sovrin: Performance through distributed trust. In *CT-RSA*, volume 12704 of *Lecture Notes in Computer Science*, pages 527–551. Springer, 2021.

[62] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In *SOSP*, pages 396–416. ACM, 2023.

[63] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security Symposium*, pages 3889–3905. USENIX Association, 2023.

[64] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[65] Samuel Jaques, Hart Montgomery, and Michael Lodder. ALLOSAUR: accumulator with low-latency oblivious sublinear anonymous credential updates with revocations. In *AsiaCCS*. ACM, 2024.

[66] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). RFC 8032, Internet Engineering Task Force, January 2017.

[67] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[68] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *CCS*, pages 830–842. ACM, 2016.

[69] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pages 158–189. Springer, 2018.

[70] Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In *SAC*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2020.

[71] Evan Krul, Hye-young Paik, Sushmita Ruj, and Salil S. Kanhere. Sok: Trusting self-sovereign identity. *Proc. Priv. Enhancing Technol.*, 2024(3):297–313, 2024.

[72] Maximilian Lam, Jeff Johnson, Wenjie Xiong, Kiwan Maeng, Udit Gupta, Yang Li, Liangzhen Lai, Ilias Leontiadis, Minsoo Rhu, Hsien-Hsin S. Lee, Vijay Janapa Reddi, Gu-Yeon Wei, David Brooks, and G. Edward Suh. Gpu-based private information retrieval for on-device machine learning inference. In *ASPLOS*, pages 197–214. ACM, 2024.

[73] Ben Laurie. Certificate transparency. *Commun. ACM*, 57(10):40–46, 2014.

[74] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz. Hintless single-server private information retrieval. In *CRYPTO*, volume 14928 of *Lecture Notes in Computer Science*, pages 183–217. Springer, 2024.

[75] Dave Longley, Manu Sporny, and Orie Steele. Bitstring status list v1.0. W3c recommendation, World Wide Web Consortium (W3C), 2025.

[76] Carlo Mazzocca, Abbas Acar, Selcuk Uluagac, Rebecca Montanari, Paolo Bellavista, and Mauro Conti. A survey on decentralized identifiers and verifiable credentials. *IEEE Communications Surveys & Tutorials*, 2025.

[77] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private information retrieval for everyone. *Proc. Priv. Enhancing Technol.*, 2016(2):155–174, 2016.

[78] Samir Jordan Menon and David J. Wu. SPIRAL: fast, high-rate single-server PIR via FHE composition. In *SP*, pages 930–947. IEEE, 2022.

[79] Samir Jordan Menon and David J. Wu. SPIRAL: fast, high-rate single-server PIR via FHE composition. In *SP*, pages 930–947. IEEE, 2022.

[80] Samir Jordan Menon and David J. Wu. YPIR: high-throughput single-server PIR with silent preprocessing. In *USENIX Security Symposium*. USENIX Association, 2024.

[81] Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server PIR. In *CCS*, pages 2292–2306. ACM, 2021.

[82] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. 5443:463–480, 2009.

[83] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.

[84] Jean Claude Paillès. New protocols for electronic money. In *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 1992.

[85] Andrew Park, Trevor Leong, Francisco Maturana, Wenting Zheng, and K. V. Rashmi. Communication-efficient, fault tolerant PIR over erasure coded storage. In *SP*, pages 4331–4347. IEEE, 2024.

[86] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[87] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *CT-RSA*, volume 9610 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2016.

[88] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, 2000.

[89] Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In *CRYPTO*, volume 14084 of *Lecture Notes in Computer Science*, pages 602–632. Springer, 2023.

[90] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

[91] Leandro Rometsch, Philipp-Florens Lehwalder, Anh-Tu Hoang, Dominik Kaaser, and Stefan Schulte. Uppr: Universal privacy-preserving revocation. In *2025 IEEE International Conference on Blockchain (Blockchain)*, pages 161–170. IEEE, 2025.

[92] Dragos Rotaru and Tim Wood. Marbled circuits: Mixing arithmetic and boolean circuits with active security. In *INDOCRYPT*, volume 11898 of *Lecture Notes in Computer Science*, pages 227–249. Springer, 2019.

[93] Lawrence Roy. Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In *CRYPTO*, volume 13507 of *Lecture Notes in Computer Science*, pages 657–687. Springer, 2022.

[94] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[95] Reza Soltani, Uyen Trang Nguyen, and Aijun An. A survey of self-sovereign identity ecosystem. *Security and Communication Networks*, 2021, 2021.

[96] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. 2019.

[97] Manu Sporny, Dave Longley, David Chadwick, and Ivan Herman. Verifiable credentials data model v2.0. W3C recommendation, World Wide Web Consortium, 2025.

[98] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized identifiers (DIDs) v1.0. W3C recommendation, World Wide Web Consortium, 2022.

[99] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In *EUROCRYPT*, volume 14008 of *Lecture Notes in Computer Science*, pages 691–721. Springer, 2023.

[100] Justin Thaler. Proofs, arguments, and zero-knowledge. *Found. Trends Priv. Secur.*, 4(2-4):117–660, 2022.

[101] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS*, pages 72–81. ACM, 2007.

[102] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE Computer Society, 1986.

A Zero-Knowledge Proof Relations

Relation 1 (Database Authorization)**.** *Relation* $\mathsf{R}_{\mathsf{auth}}$ *is used to prove knowledge of an authorization credential* σ_{auth}*, valid under a verification key* vk_I*, such that the same index* idx *is included in both the certificate and the commitment* $\mathsf{co}_{\mathsf{idx}}$*. More formally, the relation can be defined as follows:*

$$\mathsf{R}_{\mathsf{auth}} := \left\{ \begin{array}{c} ((\mathsf{vk}_I, \mathsf{co}_{\mathsf{idx}}), (\sigma_{\mathsf{auth}}, \mathsf{idx}, \mathsf{rnd})) : \\ \mathit{Cert.Verify}(\mathsf{vk}_I, \sigma_{\mathsf{auth}}, \mathsf{idx}) = 1 \\ \wedge\ \mathit{Com.Open}(\mathsf{co}_{\mathsf{idx}}, \mathsf{idx}, \mathsf{rnd}) = 1 \end{array} \right\}$$

We require the authorization credential σ_{auth} to be verifiable via a verification algorithm $\mathsf{Cert.Verify}$ for a verification key vk, and we can efficiently prove over this algorithm.

Relation 2 (Credential Non-Revocation)**.** *Relation* $\mathsf{R}_{\mathsf{rev}}$ *is used to prove knowledge of an anonymous credential* σ_{cred} *valid under the verification key* vk_I*, having (revealed) attributes* $(a'_i)_{i \in \mathit{Rvd}}$*, such that* a_0 *(the index attribute) and* $b = 1$ *(status is valid) are contained in the PRC commitment* $\mathsf{co}_{\mathsf{rec}}$*. More formally, the relation is defined as follows:*

$$\mathsf{R}_{\mathsf{rev}} := \left\{ \begin{array}{c} ((\mathsf{vk}_I, (a'_i)_{i \in \mathit{Rvd}}, \mathsf{co}_{\mathsf{rec}}), \\ (\sigma_{\mathsf{cred}}, (a_i)_{i \in [m]}, \mathsf{idx}, \mathsf{rnd})) : \\ \mathit{AC.Verify}(\mathsf{vk}_I, \sigma_{\mathsf{cred}}, (a_i)_{i \in [m]}) = 1 \\ \wedge\ a'_i = a_i \text{ for } i \in \mathit{Rvd} \\ \wedge\ \mathit{Com.Open}(\mathsf{co}_{\mathsf{rec}}, (a_0, b), \mathsf{rnd}) = 1 \end{array} \right\}$$

We require that the credential σ_{cred} can be directly verified using a verification algorithm $\mathsf{AC.Verify}$ over which we can efficiently prove. This step would implicitly be performed in the presentation algorithm $\mathsf{Present}$.

B Extensions of our AC Revocation System

We discuss three extensions of our anonymous credential revocation system in §6: enabling offline clients to delegate contacting RAs to the verifier, achieving multi-show presentations of non-revocation tokens, and database privacy.

Extension with offline clients. Real-time revocation (see RQ.F3) requires connection to the RAs. However, as long as the verifier can reach the RAs, our construction can support offline clients that perform a one-round presentation with a non-revocation check and no privacy degradation. This improves support for constrained clients without internet access, such as scenarios with a phone without mobile data.

Observe that in a usual presentation with a non-revocation check, a client 1) performs a PRC request to obtain a non-revocation token and 2) generates a zero-knowledge proof linking the token to its credential. In this proof, the client only needs to know the commitment value $\mathsf{co}_{\mathsf{rec}}$ inside the token, but does not need to prove over the token (i.e., signature) itself, which is revealed to the verifier in the clear. Since the commitment randomness is chosen by the client, and it can assume that its credential is valid (i.e., $\mathbf{D}[\mathsf{idx}] = 1$), the client can pre-compute the commitment without actually receiving the non-revocation token. Further, processing the

PRC response does not require any private client input and can be fully revealed to the verifier.

In an offline setting, the client can thus prepare the PRC request, encrypt each query under the public key of the corresponding RA, and delegate the interaction with the RAs to the verifier. By also providing the zero-knowledge proof over the pre-computed commitment, the verifier can check the presentation without a further round with the client. To prevent replay attacks and in extension prevent a verifier from repeatedly checking the revocation status of a client, we encrypt add a nonce to the queries. We further assume that the encryption scheme provides confidentiality and integrity, binding the nonce to its query. If one of the RAs receives a nonce twice, they can abort the protocol. Due to the integrity protection, no verifier can change the nonce associated with a query. In the following, we assume that each RA is equipped with a public key encryption key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. The offline presentation and verification proceed as follows:

- $\mathsf{opt} \leftarrow \mathsf{OfflinePresentNonRev}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd}, \mathsf{vk}_{\mathcal{R}})$: The client produces an offline presentation opt for credential σ_{cred} with revocation index idx as follows: 1) compute a PRC request $((\mathsf{qry}_i)_{i\in[n]}, \mathsf{qst}{=}\mathsf{rnd}) \leftarrow \mathsf{Request}(\mathsf{idx}, \sigma_{\mathsf{auth}})$ and its corresponding commitment $\mathsf{co}_{\mathsf{rec}} \leftarrow \mathsf{Com.Commit}([\mathsf{idx}, \mathsf{rec}{=}1]; \mathsf{rnd})$; 2) choose a nonce η and encrypt each query under the corresponding PRC server's public key, $\mathsf{eqry}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, (\mathsf{qry}_i, \eta))$; 3) compute the zero-knowledge proof π_{rev} for relation $\mathsf{R}_{\mathsf{rev}}$ (cf. Rel. 2)using the pre-computed $\mathsf{co}_{\mathsf{rec}}$; and 4) output $\mathsf{opt} := ((\mathsf{eqry}_i)_{i\in[n]}, \mathsf{co}_{\mathsf{rec}}, \pi_{\mathsf{rev}})$.
- $0/1 \leftarrow \mathsf{OfflineVerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{opt}, (a_i)_{i\in\mathsf{Rvd}})$: The verifier combines the PRC interaction with presentation verification as follows: 1) the verifier parses $\mathsf{opt} := ((\mathsf{eqry}_i)_{i\in[n]}, \mathsf{co}_{\mathsf{rec}}, \pi_{\mathsf{rev}})$ and sends each encrypted query eqry_i to the corresponding RA; 2) each RA acting as a PRC server decrypts its query $(\mathsf{qry}_i, \eta) \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \mathsf{eqry}_i)$, computes the response $(\sigma_{\mathsf{rec}}, \mathsf{time}) \leftarrow \mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i, \mathsf{qry}_i, \mathbf{D})_{i\in[n]}, \perp)$, and send it to the verifier; 3) the verifier forms $\mathsf{pt} := (\sigma_{\mathsf{rec}}, \mathsf{co}_{\mathsf{rec}}, \mathsf{time}, \pi_{\mathsf{rev}})$ and accepts iff $\mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}, (a_i)_{i\in\mathsf{Rvd}})$ accepts.

Correctness is straightforward: if the client's credential is not revoked, the pre-computed commitment matches the PRC response, and verification accepts. The security argument follows the base construction. Compared with the base version, there are two differences. First, the client cannot verify the PRC response and therefore cannot detect that its credential is revoked and abort contacting the verifier. This is an inherent limit of one-round offline presentations; a two-round variant could trivially solve this. Second, the verifier learns the encrypted queries eqry_i in addition to pt. The encryption ensures that the verifier query content remains confidential, and nonces prevent reusing queries.

Extension with multi-show presentations. In our construction, we assume that the record certificate σ_{rec}, which attests the revocation status of a credential, is only valid for a short period and is only used once for a presentation to the verifier. If the client reuses the same record certificate σ_{rec} for multiple presentations, it would make them linkable. However, one could instantiate the (threshold) signature scheme used in the PRC scheme with a scheme that supports efficient zero-knowledge proofs such as threshold PS [96] or threshold BBS [43, 50] or that is self-blindable [36] and let the client only prove knowledge of such a record certificate instead of revealing it to the verifier. Naturally, this change also requires reducing the precision of the timestamp time of the record certificate, such that sufficiently coarse intervals ensure unlinkability within the corresponding anonymity sets.

Extension with database privacy. As the credentials in our system σ_{cred} already contain the revocation index, one can similarly use it as the authorization credential σ_{auth} to achieve database privacy as described in the extended PRC construction §5.4. Thus, during PresentNonRev, the client uses its credential σ_{cred} as the authorization credential when generating the PRC request. Note that the authorization proof π_{auth} for the relation $\mathsf{R}_{\mathsf{auth}}$ (see §A) is essentially constructed the same way as the proof of non-revocation for the presentation π_{cred}; both prove knowledge of a signature containing the same index as in a given commitment. The only difference is that the presentation proof π_{cred} may prove additional statements about other attributes of the credential.

In the following, we define and prove database privacy of this modified construction.

Definition B.1 (Database Privacy)**.** *An anonymous credential revocation scheme* Π *fulfills database privacy, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that for all* $\lambda, n \in \mathbb{N}$*,*

$$Adv^{\text{REV-DBP}}_{\mathcal{A},\Pi}(1^\lambda, n) \leq \mathsf{negl}(1^\lambda)$$

where $\text{REV-DBP}_{\mathcal{A},\Pi}$ *is defined in Figure 5.*

Theorem B.1 (Database Privacy)**.** *Our anonymous credential revocation scheme* Π_{rev} *in §6 achieves database privacy if the base credential scheme* AC *is unforgeable, and the proof system ZKP fulfills knowledge-soundness.*

Proof. We show that for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$, such that for all $\lambda, n \in \mathbb{N}$,

$$\text{Adv}^{\text{REV-DBP}}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \text{Adv}^{\text{AC-UF}}_{\mathcal{B},\mathsf{AC}}(1^\lambda) + \mathsf{negl}(1^\lambda).$$

As our anonymous credential revocation scheme builds upon our PRC construction Π_{PRC} in §5.2, instantiating the generic authentication certificate with AC, we can reduce to the database privacy of Π_{PRC}.

As shown in §C.3, if the proof system ZKP used in Π_{PRC} fulfills knowledge-soundness, and Π_{auth} is unforgeable, for

GAME REV-DBP$_{\mathcal{A},\Pi}(1^\lambda, n)$:

1 : $(\mathsf{sk}_I, \mathsf{vk}_I) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}, _) \leftarrow \mathsf{RevSetup}(1^\lambda, n)$
3 : $\mathbf{D}_0, \mathbf{D}_1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}})$
4 : $b \xleftarrow{\$} \{0,1\}$
5 : $b' \leftarrow \mathcal{A}^{O^{\mathsf{Issue}}, O^{\mathsf{Present}}}()$
6 : **return** $b = b'$

$O^{\mathsf{Issue}}((a_i)_{i\in[m]})$:

1 : **if** $(\mathbf{D}_0[a_0] \neq \mathbf{D}_1[a_0])$ **then return** $\perp$
2 : **return** $\mathsf{Issue}(\mathsf{sk}_I, (a_i)_{i\in[m]})$

$O^{\mathsf{Present}}_{\mathcal{A}}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd})$:

1 : $(\mathsf{pt}, \perp) \leftarrow \mathsf{PresentNonRev}_{\mathcal{A},\mathcal{R}}\left(\begin{matrix}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd}, \mathsf{vk}_{\mathcal{R}}),\\ (\mathsf{sk}_{\mathcal{R},i}, \mathbf{D}_b, \mathsf{vk}_{\mathcal{R}})_{i\in[n]}\end{matrix}\right)$
2 : **return** pt

Figure 5: Game-based definition of database privacy for an anonymous credential revocation scheme Π. By λ, we denote the security parameter, and n denotes the number of RAs.

all PPT adversaries $\mathcal{C}$, there exists a PPT adversary $\mathcal{D}$, such that for all λ, n

$$\mathrm{Adv}^{\text{PRC-DBP}}_{\mathcal{C},\Pi_{\mathsf{PRC}db},\Pi_{\mathsf{auth}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{AUTH-UF}}_{\mathcal{B},\Pi_{\mathsf{auth}}}(1^\lambda) + \mathsf{negl}(1^\lambda).$$

In the following, we argue that Π_{rev} adheres to the definition of an unforgeable authorization credential, allowing us to reduce directly to PRC. Similar to Rel. 2, we require an explicit verification algorithm AC.Verify for the credential scheme AC that is implicitly used in the presentation algorithm Present and allows to verify a credential σ_{cred} on a set of attributes $(a_i)_{i\in[m]}$ under a verification key vk_I. Instantiating AUTH-UF with AC is trivially possible, as the KGen, Sign and Verify algorithms follow the definition of authorization credentials(§C.1). Further, an adversary that can break AUTH-UF$_{\mathsf{AC}}$ may also break AC-UF$_{\mathsf{AC}}$, simply by computing a presentation proof of the forgery it generates for AUTH-UF and sending the proof to its AC-UF$_{\mathsf{AC}}$ challenger. By this argument, we can consider AC to be an unforgeable authorization credential.

We prove the theorem via a series of game hops, and finally a reduction to the unforgeability of Π_{PRC}. Let GAME$_0$ = REV-DBP$_{\mathcal{A},\Pi_{\mathsf{rev}}}$.

Game$_1$. We define GAME$_1$ to be identical to GAME$_0$, but with the following distinction: instead of obtaining a credential on an arbitrary set of attributes, an adversary $\mathcal{A}_1$ in GAME$_1$ only obtains credentials for its database index (attribute a_0). Clearly, attribute a_0 is the only attribute used to prove authorization in REV-DBP. An adversary that can forge credentials on attributes other than a_0 will not gain any advantage in breaking REV-DBP. It holds that

$$\mathrm{Adv}^{\text{GAME}_0}_{\mathcal{A},\Pi_{\mathsf{rev}db}}(1^\lambda, n) = \mathrm{Adv}^{\text{GAME}_1}_{\mathcal{A}_1,\Pi_{\mathsf{rev}db},\Pi_{\mathsf{auth}}}(1^\lambda, n).$$

Given adversary $\mathcal{A}_1$ in GAME$_1$, we can build an adversary $\mathcal{C}$ for PRC-DBP$_{\mathcal{C},\Pi_{\mathsf{PRC}db},\mathsf{AC}}$ as follows. $\mathcal{C}$ obtains the PRC and AC setup from its challenger, which it can use to simulate the interaction in GAME$_1$. Initially, $\mathcal{C}$ obtains a set of AC credentials/authorization credentials on each index. These can be used to answer issuance oracle queries made by $\mathcal{A}_1$, which in GAME$_1$ only request credentials on a database index. Further, $\mathcal{C}$ can simulate the Response oracle by querying its PRC oracle and forwarding the output to $\mathcal{A}_1$. As the $\Pi_{\mathsf{rev}db}$ scheme combines only the authorization check and non-revocation token issuance of Π_{PRC} with a presentation proof of AC, $\mathcal{B}$ can perfectly simulate GAME$_1$ to $\mathcal{A}$. As both games use exactly the same authentication mechanism (knowledge of an AC credential), we find that

$$\mathrm{Adv}^{\text{GAME}_1}_{\mathcal{A},\Pi_{\mathsf{rev}db}}(1^\lambda, n) = \mathrm{Adv}^{\text{PRC-DBP}}_{\mathcal{C},\Pi_{\mathsf{PRC}db},\mathsf{AC}}(1^\lambda, n).$$

By §C.3, this implies the existence of a PPT adversary $\mathcal{B}$, such that

$$\mathrm{Adv}^{\text{PRC-DBP}}_{\mathcal{C},\Pi_{\mathsf{PRC}db},\mathsf{AC}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{AUTH-UF}}_{\mathcal{B},\Pi_{\mathsf{auth}}}(1^\lambda) + \mathsf{negl}(1^\lambda),$$

which concludes the proof. □

C Security of our PRC Scheme

In this section, we provide formal, game-based definitions of the security properties for PRC schemes introduced in §5. Further, we provide full security proofs for the PRC scheme presented in §5.2. We will now make explicit use of the security parameter 1^λ. In the following, we let interactive protocols output a random variable View$_i$ for each participating party i, which contains the transcript of all incoming/outgoing messages including the local randomness and inputs of party i during the protocol execution. This allows us to formally capture the notion of semi-honest adversaries in our security games.

We add another algorithm $(\mathsf{idx}, \mathsf{qst}) \leftarrow \mathsf{QryExt}((\mathsf{qry}_i)_{i\in[n]})$ to the definition of PRC schemes which takes the set of all query shares generated by Request and extracts the requested index idx and query state $\mathsf{qst} = \mathsf{rnd}$. In our concrete construction this function uses SS.Rec() to reconstruct query shares. We require QryExt only in the security games to simplify notation.

We assume that adversaries implicitly keep state throughout executions and use $\mathrm{Adv}^{\text{GAME}}_{\mathcal{A},\Pi}$ to denote the advantage of adversary $\mathcal{A}$ in winning a security game GAME, instantiated

with scheme Π. If clear from context, we omit Π as a parameter. Let params be the arguments of the security game. We define:

$$\mathrm{Adv}^{\mathrm{GAME}}_{\mathcal{A},\Pi}(\mathsf{params}) = \Pr[\mathrm{GAME}_{\mathcal{A},\Pi}(\mathsf{params}) = 1]$$

for games with computational win condition, and

$$\mathrm{Adv}^{\mathrm{GAME}}_{\mathcal{A},\Pi}(\mathsf{params}) = |\Pr[\mathrm{GAME}_{\mathcal{A},\Pi}(\mathsf{params}) = 1]| - \frac{1}{2}$$

for games with distinguishing win condition.

C.1 Additional Definitions

We introduce necessary definitions for primitives used in our PRC construction.

Commitment Schemes. A commitment scheme Com as defined in §3 has to fulfill binding and hiding, captured by the security games GAMEBND and GAMEHID, respectively. For the binding game GAMEBND, the adversary has to output a commitment co and two pairs of openings (x_0, r_0) and (x_1, r_1) such that $\mathsf{Open}(\mathsf{co}, x_0, r_0) = 1$ and $\mathsf{Open}(\mathsf{co}, x_1, r_1) = 1$ and $x_0 \neq x_1$. For the hiding game GAMEHID, the adversary chooses two messages x_0 and x_1 and receives a commitment co on one of them, and has to guess which one was committed to. We refer to a formal definition to [67].

Zero-knowledge Proof System. A zero-knowledge proof system ZKP consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$. The setup algorithm $\mathsf{Setup}(1^\lambda, \mathsf{R})$ generates public parameters pp for a relation R. The algorithm $\mathsf{Prove}(x, w)$ inputs a statement x and a witness w s.t. $(x, w) \in \mathsf{R}$ and outputs a proof π. The verification algorithm $\mathsf{Verify}(x, \pi)$ outputs a bit indicating whether the proof π is valid for the statement x.

We require the zero-knowledge proof system to satisfy *completeness*, *knowledge-soundness*, and *zero-knowledge*. Completeness states that honestly generated proofs will always pass verification. Knowledge-soundness ensures the existence of an extractor Ext that can extract a valid witness w from any prover generating a valid proof π for a statement x. The probability that extraction fails is negligible in the security parameter λ. Zero-knowledge ensures that simulated proofs are indistinguishable from honestly generated proofs, i.e., a proof does not reveal any more information than the validity of the statement itself. For a formal definition, we refer to [100].

authorization credentials. We define a scheme for generating authorization credentials denoted as Π_{auth} as a tuple of algorithms $(\mathsf{KGen}, \mathsf{Issue}, \mathsf{Verify})$. The key generation algorithm $\mathsf{KGen}(1^\lambda)$ generates an issuing secret key sk_I and a verification key vk_I, the issuing algorithm $\mathsf{Issue}(\mathsf{sk}_I, \mathsf{idx})$ generates an authorization credential σ_{auth} for an index idx using the secret key sk_I, and the verification algorithm $\mathsf{Verify}(\mathsf{vk}_I, \sigma_{\mathsf{auth}}, \mathsf{idx})$ checks whether σ_{auth} is a valid authorization credential for index idx w.r.t. a verification key vk_I.

GAME AUTH-UF$_{\mathcal{A},\Pi_{\mathsf{auth}}}(1^\lambda)$:

1 : $(\mathsf{sk}_I, \mathsf{vk}_I) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $(\mathsf{idx}, \sigma_{\mathsf{auth}}) \leftarrow \mathcal{A}^{O^{\mathsf{Issue}}}(\mathsf{vk}_I)$
3 : **return** $(\mathsf{idx} \notin \mathcal{U}) \wedge \mathsf{Verify}(\mathsf{vk}_I, \sigma_{\mathsf{auth}}, \mathsf{idx})$

$O^{\mathsf{Issue}}(\mathsf{idx})$:

1 : $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathsf{idx}\}$
2 : **return** $\mathsf{Issue}(\mathsf{sk}_I, \mathsf{idx})$

Figure 6: The authorization credential unforgeability game with λ denoting the security parameter.

We demand that the authorization credential scheme fulfills correctness, i.e., honestly generated certificates will always pass verification, and unforgeability, which we define below.

Definition C.1 (authorization credential Unforgeability)**.** *We say that an authorization credential scheme Π_{auth} is unforgeable, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$Adv^{\mathrm{AUTH\text{-}UF}}_{\mathcal{A},\Pi_{\mathsf{auth}}}(1^\lambda) \leq \mathsf{negl}(1^\lambda),$$

where AUTH-UF$_{\mathcal{A},\Pi_{\mathsf{auth}}}$ *is defined in Figure 6*

C.2 Property Definitions

Recall that PRC schemes are required to satisfy correctness, unforgeability, unlinkability, and optionally database privacy. Correctness is straightforward, and we provide formal game-based definitions for the remaining properties.

The first property that we define is unforgeability. We present the PRC unforgeability game PRC-UF in Figure 7. In this setting, the client is malicious and up to $n-1$ servers are semi-honest. Intuitively, the adversary's goal in this game is to produce a valid record certificate on a tuple of index, record value and auxiliary information that was not previously requested during an oracle call to O^{Response}. This captures adversaries that forge valid certificates for incorrect record values, as well as those that forge the corresponding auxiliary information.

Definition C.2 (PRC Unforgeability)**.** *We say that a PRC scheme Π is unforgeable, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda, n \in \mathbb{N}$,*

$$Adv^{\mathrm{PRC\text{-}UF}}_{\mathcal{A},\Pi}(1^\lambda, n) \leq \mathsf{negl}(1^\lambda),$$

where PRC-UF$_{\mathcal{A},\Pi}$ *is defined in Figure 7.*

Our PRC unlinkability game, presented in Figure 8, intuitively captures the fact that adversaries cannot distinguish

GAME PRC-UF$_{\mathcal{A},\Pi}(\lambda,n)$:

$((\mathsf{sk}_i)_{i\in[n]},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda,n)$

$\mathbf{D},\mathsf{Corr} \leftarrow \mathcal{A}(1^\lambda,n,\mathsf{vk})$

$\mathcal{U} \leftarrow \emptyset$

$(\sigma^*_{\mathsf{rec}},\mathsf{qst}^*,\mathsf{idx}^*,\mathsf{rec}^*,\mathsf{aux}^*) \leftarrow \mathcal{A}^{O^{\mathsf{Response}}}\left((\mathsf{sk}_i)_{i\in\mathsf{Corr}}\right)$

return $\mathsf{Verify}(\mathsf{vk},\sigma^*_{\mathsf{rec}},\mathsf{qst}^*,\mathsf{idx}^*,\mathsf{rec}^*,\mathsf{aux}^*)$
$\wedge(|\mathsf{Corr}|<n)\wedge(\mathsf{idx}^*,\mathsf{rec}^*,\mathsf{aux}^*)\notin\mathcal{U}$

$O^{\mathsf{Response}}((\mathsf{qry}_i)_{i\in[n]})$:

$(\sigma_{\mathsf{rec}},\mathsf{aux}),\ \{\mathsf{View}_i\}_{i\in[n]} \leftarrow \mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i,\mathsf{qry}_i,\mathbf{D})_{i\in[n]})$

$(\mathsf{idx},\mathsf{qst}) \leftarrow \mathsf{QryExt}((\mathsf{qry}_i)_{i\in[n]})$

$\mathcal{U} \leftarrow \mathcal{U}\cup\{(\mathsf{idx},\mathbf{D}[\mathsf{idx}],\mathsf{aux})\}$

return $(\sigma_{\mathsf{rec}},\mathsf{aux},\{\mathsf{View}_i\}_{i\in\mathsf{Corr}})$

Figure 7: The unforgeability game for PRC schemes (§5). We denote the security parameter by λ, and by n the number of servers. In this setting, the client is malicious and up to $n-1$ RAs are semi-honest. By View_i we denote the transcript of all messages sent and received by server i when executing the Response protocol.

GAME PRC-UL$_{\mathcal{A},\Pi}(\lambda,n)$:

$(\mathbf{D},\mathsf{Corr},(\mathsf{sk}_i)_{i\in[n]},\mathsf{vk}) \leftarrow \mathcal{A}(1^\lambda,n)$

$\{\mathsf{idx}_j\}_{j\in\{0,1\}} \leftarrow \mathcal{A}()$

$b \xleftarrow{\$} \{0,1\}$

for $j\in\{0,1\}$ **do** // b changes iteration order

$(\mathsf{qry}_{i,j\oplus b})_{i\in[n]},\mathsf{st} \leftarrow \mathsf{Request}(\mathsf{idx}_{j\oplus b})$

$_ \leftarrow \mathsf{Response}_{\mathcal{A},(\mathcal{S}_i)_{i\in[n]\setminus\mathsf{Corr}}}\left((\mathsf{sk}_i,\mathsf{qry}_{i,j\oplus b},\mathbf{D})_{i\in[n]}\right)$

$b' \leftarrow \mathcal{A}()$

return $(b'=b)\ \wedge\ (|\mathsf{Corr}|<n)$

Figure 8: The unlinkability game for PRC schemes (§5). By λ, we denote the security parameter, while n denotes the number of servers. In this setting, up to $n-1$ servers are malicious.

between two different queried indices, even when a subset of the servers is malicious. In detail, the adversary's goal is to guess the order in which two indices are requested, given only the information available to the corrupted servers.

Definition C.3 (PRC Unlinkability)**.** *We say that a PRC scheme Π is unlinkable, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all $\lambda,n\in\mathbb{N}$,*

$$Adv^{\text{PRC-UL}}_{\mathcal{A},\Pi}(1^\lambda,n) \le \mathsf{negl}(1^\lambda),$$

GAME PRC-DBP$_{\mathcal{A},\Pi,\Pi_{\mathsf{auth}}}(1^\lambda,n)$:

$((\mathsf{sk}_i)_{i\in[n]},\mathsf{vk}) \leftarrow \mathsf{KGen}(n)$

$(\mathsf{sk}_I,\mathsf{vk}_I) \leftarrow \Pi_{\mathsf{auth}}.\mathsf{KGen}(1^\lambda)$

$\mathbf{D}_0,\mathbf{D}_1 \leftarrow \mathcal{A}(1^\lambda,n,\mathsf{vk},\mathsf{vk}_I)$

for $i\in[N]$ **do**

$\sigma_{\mathsf{auth},i} \leftarrow \Pi_{\mathsf{auth}}.\mathsf{Issue}(\mathsf{sk}_I,i)$

$b \xleftarrow{\$} \{0,1\}$

$b' \leftarrow \mathcal{A}^{O^{\mathsf{Response}}}\left((\sigma_{\mathsf{auth},i})_{\{i\in[N]\mid\mathbf{D}_0[i]=\mathbf{D}_1[i]\}}\right)$

return $b=b'$

$O^{\mathsf{Response}}((\mathsf{qry}_i)_{i\in[n]})$:

return $\mathsf{Response}_{\mathcal{S}}((\mathsf{sk}_i,\mathsf{qry}_i,\mathbf{D}_b)_{i\in[n]},\mathsf{vk}_I)$

Figure 9: The database privacy game for PRC schemes (§5). By λ, we denote the security parameter, and by n, the number of servers. The game is additionally parameterized by an authorization credential scheme Π_{auth} (§C.1).

where PRC-UL$_{\mathcal{A},\Pi}$ *is defined in Figure 8.*

Optionally, PRC schemes may fulfill database privacy. This notion captures the fact that malicious clients may only access database entries, for which they hold an authorization credential. We define the database privacy game in Figure 9 for an authorization credential scheme Π_{auth} as defined in §C.1. The adversary's goal is to distinguish between two databases, for which the adversary is only given authorization credentials on indices where the record values are identical in both databases.

Definition C.4 (PRC database privacy)**.** *We say a PRC scheme Π fulfills database privacy for an authorization credential scheme Π_{auth} (§C.1) if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all $\lambda,n\in\mathbb{N}$,*

$$Adv^{\text{PRC-DBP}}_{\mathcal{A},\Pi,\Pi_{\mathsf{auth}}}(1^\lambda,n) \le \mathsf{negl}(1^\lambda),$$

where PRC-DBP$_{\mathcal{A}}$ *is defined in Figure 9.*

C.3 Security Proofs for our PRC Construction

We now give security proofs for the PRC scheme Π_{PRC} presented in §5.2, showing that it satisfies unforgeability and unlinkability as per Definitions C.2 and C.3. We further show that the extended PRC construction $\Pi_{\mathsf{PRC}db}$ in §5.4 fulfills database privacy as per Definition C.4.

Theorem 5.1 (PRC Unforgeability)**.** *When instantiated with a binding commitment scheme* Com *and an unforgeable threshold signature scheme* TSig*, our PRC construction Π_{PRC} (§5.2) is unforgeable as per Definition C.2.*

Proof. We prove the theorem via a union-bound argument, reducing to the unforgeability of the underlying threshold signature scheme TSig and the binding property of commitment Com. We show that for all PPT adversaries $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}$ and $\mathcal{C}$, such that for all $\lambda, n \in \mathbb{N}$,

$$\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GameBnd}}_{\mathcal{B},\mathsf{Com}}(1^\lambda) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{C},\mathsf{TSig}}(1^\lambda),$$

where $\text{PRC-UF}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$ is defined in Figure 7.

Let i be the index of an O^{Response} query made by adversary $\mathcal{A}$. By $(\mathsf{idx}_i, \mathsf{rec}_i, \mathsf{aux}_i)$, we denote the tuple added to $\mathcal{U}$ for the ith oracle query. Further, we use $\mathsf{qst}_i = \mathsf{rnd}_i$ to denote the query state/randomness used by $\mathcal{A}$ in the ith query, reconstructed using QryExt in line 2 of the oracle.

Let $\mathcal{A}$ be a PPT adversary in $\text{PRC-UF}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$. For any forgery $(\sigma^*_{\mathsf{rec}}, \mathsf{qst}^*, \mathsf{idx}^*, \mathsf{rec}^*, \mathsf{aux}^*)$ output by $\mathcal{A}$, one of the following statements is true:

Type 1) There exists an oracle query with index i, such that

$$\begin{aligned}\mathsf{co}_i =& \mathsf{Com.Commit}(\mathsf{idx}_i, \mathsf{rec}_i; \mathsf{rnd}_i)\\ =& \mathsf{Com.Commit}(\mathsf{idx}^*, \mathsf{rec}^*; \mathsf{rnd}^*) = \mathsf{co}^*.\end{aligned}$$

Type 2) For all oracle queries i,

$$\begin{aligned}\mathsf{co}_i =& \mathsf{Com.Commit}(\mathsf{idx}_i, \mathsf{rec}_i; \mathsf{rnd}_i)\\ \neq& \mathsf{Com.Commit}(\mathsf{idx}^*, \mathsf{rec}^*; \mathsf{rnd}^*) = \mathsf{co}^*.\end{aligned}$$

Let $\mathcal{B}$ be an adversary playing the binding game GameBnd for Com. $\mathcal{B}$ obtains $\mathcal{A}$ and simulates a $\text{PRC-UF}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$ by sampling the PRC setup including all signing keys. This allows $\mathcal{B}$ to perfectly simulate a O^{Response} response. If $\mathcal{A}$ finally outputs a type 1) forgery, the tuples $(\mathsf{idx}_i, \mathsf{rec}_i; \mathsf{rnd}_i)$ and $(\mathsf{idx}^*, \mathsf{rec}^*; \mathsf{rnd}^*)$ constitute two different openings of the same commitment $\mathsf{co}_i = \mathsf{co}^*$. $\mathcal{B}$ can send $(\mathsf{idx}_i, \mathsf{rec}_i; \mathsf{rnd}_i)$ and $(\mathsf{idx}^*, \mathsf{rec}^*; \mathsf{rnd}^*)$ to its challenger, breaking the binding property of Com. It holds that

$$\begin{aligned}&\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \cdot \Pr[\text{Type 1) forgery}]\\ &\leq \mathrm{Adv}^{\text{GameBnd}}_{\mathcal{B},\mathsf{Com}}(1^\lambda).\end{aligned}$$

Consider now an adversary $\mathcal{C}$ playing the unforgeability game EUF-CMA for threshold signature scheme TSig. $\mathcal{C}$ obtains $\mathcal{A}$ and can simulate $\text{PRC-UF}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$ as follows. $\mathcal{C}$ obtains the keys from its challenger and can use its corruption oracle to query the secret keys of servers $\mathcal{S}_i$ for $i \in \mathsf{Corr}$. When $\mathcal{A}$ queries O^{Response}, $\mathcal{C}$ can simulate the protocol interaction up to the signature generation step. By querying its partial signing oracle on the commitments output by the PRC participants, $\mathcal{C}$ can perfectly simulate the protocol to $\mathcal{A}$. If $\mathcal{A}$ outputs a forgery of type 2), none of the previous queries to the partial signing oracle (for which co_i was the message to be signed) requested a signature on the forgery commitment co^*. Therefore, $(\mathsf{co}^*, \sigma^*_{\mathsf{auth}})$ is a forgery in EUF-CMA, and $\mathcal{C}$ is successful. We find that

$$\begin{aligned}&\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \cdot \Pr[\text{Type 2) forgery}]\\ &\leq \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{C},\mathsf{TSig}}(1^\lambda)\end{aligned}$$

Combining both results, and knowing that $\Pr[\text{Type 1) forgery}] + \Pr[\text{Type 2) forgery}] = 1$, we can conclude that

$$\begin{aligned}&\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \cdot \Pr[\text{Type 1) forgery}]\\ &+\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \cdot \Pr[\text{Type 2) forgery}])\\ &\leq \mathrm{Adv}^{\text{GameBnd}}_{\mathcal{B},\mathsf{Com}}(1^\lambda) + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{C},\mathsf{TSig}}(1^\lambda).\end{aligned}$$

□

The $\mathcal{F}_{\mathsf{CRG}}$ hybrid model. For our unlinkability proof, we model the generation of correlated randomness by an ideal functionality $\mathcal{F}_{\mathsf{CRG}}$. In practice, this ideal functionality may be implemented by outsourcing the computation to a trusted third party, or by implementing it via a maliciously secure MPC protocol. Formally, $\mathcal{F}_{\mathsf{CRG}}$ implements a single interface: The Gen interface outputs secret shares of a beaver triple and a daBit to each party i. We now show that in this hybrid model, where the real protocol interacts with an idealized functionality, our construction achieves unlinkability.

Theorem 5.2 (PRC Unlinkability)**.** *Let Com be a hiding commitment scheme. Then, our PRC construction Π_{PRC} from §5.2 is unforgeable as per Definition C.2, in the $\mathcal{F}_{\mathsf{CRG}}$-hybrid model.*

Proof. We show that for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$, such that for all $\lambda, n \in \mathbb{N}$

$$\mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GameHid}}_{\mathcal{B},\mathsf{Com}}(1^\lambda),$$

where PRC-UL is defined in Figure 8 and GameHid is the hiding game of a commitment scheme. To prove the statement, we consider a series of game-hops, bounding the adversary's success probability in each step. Let $\text{Game}_0 = \text{PRC-UL}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$.

Game_1. We define Game_1 to be identical to Game_0, except for the following distinction. Instead of committing to the record value shares $[\![\mathsf{rec}]\!]^{\mathrm{A}}_i$ as $\mathsf{co}_i \leftarrow \mathsf{Commit}((\![\mathsf{idx}]\!]^{\mathrm{A}}, [\![\mathsf{rec}]\!]^{\mathrm{A}}); [\![\mathsf{rnd}]\!]^{\mathrm{A}})$, we let servers $\mathcal{S}_i$ output $\mathsf{co}_i \leftarrow \mathsf{Commit}(r; [\![\mathsf{rnd}]\!]^{\mathrm{A}})$ for $r \xleftarrow{\$} \mathbb{Z}_q^2$. Any adversary $\mathcal{A}$, whose success probability in Game_0 and Game_1 differs by more than a negligible amount, may be used to break hiding of the commitment scheme as follows: Let $\mathcal{B}$ be a PPT adversary for GameHid. $\mathcal{B}$ samples the secret keys for PRC and simulates PRC-UL to adversary $\mathcal{A}$. After choosing a random bit b, when running the response protocol for the two queries, $\mathcal{B}$ forwards $([\![\mathsf{idx}_b]\!]^{\mathrm{A}}, [\![\mathsf{rec}_b]\!]^{\mathrm{A}})$ and r as the pair of messages to its challenger and receives a commitment co on one of them. Knowing the final combined commitment co, $\mathcal{B}$ can emulate a set of commitments, one for each honest party, by randomly choosing $n - |\mathsf{Corr}| - 1$ group elements at random. The remaining commitment for the last honest party can be computed as the quotient of co over the product of

the randomly chosen commitments. By this strategy, $\mathcal{B}$ perfectly simulates GAME_0, if the GAMEHID challenge bit is 0. Otherwise, $\mathcal{B}$ perfectly simulates GAME_1. If $\mathcal{A}$ is successful in GAME_0, but not GAME_1, $\mathcal{B}$ is successful in GAMEHID. It follows that:

$$\text{Adv}^{\text{GAME}_0}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) = \text{Adv}^{\text{GAME}_1}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) + \text{Adv}^{\text{GAMEHID}}_{\mathcal{B},\text{Com}}(\lambda).$$

Game$_2$. We let GAME_2 be identical to GAME_1, but replace the honest parties' input shares to the Response protocol by uniformly random values. Importantly, our Response protocol does not reconstruct any of the honest parties' input values. In fact, the output shares are discarded, since the commitments in GAME_1 contain random values. Further, as an initial step in Response, each party uses the correlated randomness issued by ideal functionality $\mathcal{F}_{\text{CRG}}$ to perfectly mask their input for the protocol interaction. Let msg_j be an arbitrary message sent by an honest server $\mathcal{S}_i$ during the interactive phase of the Response protocol. The correlated randomness perfectly hides $\mathcal{S}_i$'s input, and $\mathcal{S}_i$ never uses its reconstruction share in the interactive phase. As the commitments are independent of the MPC result, msg_j is distributed uniformly at random over its domain. The same holds if $\mathcal{S}_i$ now chooses its input shares uniformly at random. The change is indistinguishable to an adversary, it holds that

$$\text{Adv}^{\text{GAME}_1}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) = \text{Adv}^{\text{GAME}_2}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n).$$

Game$_3$. GAME_3 is identical GAME_2, but we replace the Response query shares $(\text{qry}_i)_{i\in\text{Corr}}$ presented to the adversary, by shares chosen uniformly at random. As the secret sharing scheme is perfectly hiding, this change is indistinguishable to $\mathcal{A}$. It follows that

$$\text{Adv}^{\text{GAME}_2}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) = \text{Adv}^{\text{GAME}_3}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n).$$

In scheme GAME_3, all messages received by an adversary corrupting up to $n-1$ servers, including commitments, intermediate values of the MPC protocol and query shares, are independent of the private bit b. Therefore, it holds that

$$\text{Adv}^{\text{GAME}_3}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) = 0.$$

Combining the individual results, we find that

$$\text{Adv}^{\text{PRC-UL}}_{\mathcal{A},\Pi_{\text{PRC}}}(1^\lambda,n) = \text{Adv}^{\text{GAMEHID}}_{\mathcal{B},\text{Com}}(1^\lambda).$$

□

Finally, we show that our extended construction in §5.4 achieves database privacy.

Theorem 5.3 (PRC Database Privacy)**.** *Let Π_{auth} be an unforgeable authorization credential scheme and ZKP a knowledge-sound proof system. Our PRC construction $\Pi_{\text{PRC}db}$ from §5.4, instantiated with Π_{auth} and ZKP fulfills database privacy as per Definition C.4.*

Proof. We show that for any PPT adversary $\mathcal{A}$, there exists a negligible function negl, such that for all $\lambda, n \in \mathbb{N}$,

$$\text{Adv}^{\text{PRC-DBP}}_{\mathcal{A},\Pi_{\text{PRC}db},\Pi_{\text{auth}}}(1^\lambda,n) \leq \text{negl}(1^\lambda)$$

We prove the statement via a game-hop and a reduction to the unforgeability of the authorization credential scheme. Let $\mathcal{A}$ be a PPT adversary playing PRC-DBP, and define $\text{GAME}_0 = \text{PRC-DBP}\mathcal{A},\Pi_{\text{PRC}db},\Pi_{\text{auth}}$.

Game$_1$. We define GAME_1 to be identical to GAME_0, except for the following change. Instead of sending an authentication proof π_{auth} in each query for all servers, the client sends its authentication certificate in plain. For any adversary $\mathcal{A}$ in GAME_0, we can build an adversary $\mathcal{A}_1$ for GAME_1, that takes $\mathcal{A}$ as input and simulates GAME_0, forwarding the messages from its own challenger. If the employed proof system ZKP is knowledge-sound, there exists an extractor Ext, that given oracle access to a prover can extract a valid proof witness. $\mathcal{A}_1$ can run $\mathcal{A}$ up to the point where $\mathcal{A}$ generates a proof and store the execution state of the partial execution. $\mathcal{A}_1$ can then run Ext and provide it with oracle access to the partially executed adversary $\mathcal{A}$, emulating oracle access to an isolated prover. $\mathcal{A}_1$ can then send the output of Ext it in plain to its Response oracle. If $\mathcal{A}$ is successful, Ext is able to extract a valid witness with overwhelming probability. In this case, $\mathcal{A}_1$ perfectly simulates GAME_0 to $\mathcal{A}$ and can output the same bit as $\mathcal{A}$. If follows that

$$\begin{aligned}\text{Adv}^{\text{GAME}_0}_{\mathcal{A},\Pi_{\text{PRC}db},\Pi_{\text{auth}}}(1^\lambda,n) \leq & \text{Adv}^{\text{GAME}_1}_{\mathcal{A}_1,\Pi_{\text{PRC}db},\Pi_{\text{auth}}}(1^\lambda,n) \\ & + \text{negl}(1^\lambda)\end{aligned}$$

By definition (Definition C.4), the authorization credentials used are unforgeable, except with negligible probability. In construction $\Pi_{\text{PRC}db}$, the servers $\mathcal{S}_i$ check if the authorization credential matches the requested index. Therefore, adversary $\mathcal{A}_1$ in GAME_1 can only receive responses for indices it has queried, which exclude indices on which the two databases $\mathbf{D}_0$ and $\mathbf{D}_1$ differ. Each issued record certificate, concretely a signature on the committed record value, index, and auxiliary data, only depends on the database record of the queried index. Let $\mathcal{B}$ be an adversary playing AUTH-UF, the unforgeability game of $\Pi_{\text{PRC}db}$'s authentication certificates. Obtaining $\mathcal{A}_1$ as a subroutine, $\mathcal{B}$ can simulate GAME_1 by sampling the PRC setup (except for the authorization credentials) and forwarding all certificates for positions for which the adversary is authorized to $\mathcal{A}_1$. Since by construction, $\mathcal{A}_1$ can only obtain information about queried indices, if $\mathcal{A}_1$ is successful, there must be a query i to an index idx for which $\mathcal{A}_1$ cannot know the authorization credential. Using the QryExt algorithm on each O^{Response} query done by $\mathcal{A}_1$, $\mathcal{B}$ can extract the query index and randomness. If the authorization is valid with respect to an index for which $\mathcal{A}_1$ should not be authorized, $\mathcal{B}$ outputs the index - certificate pair as a forgery. As this is the only way in which $\mathcal{A}_1$ may win GAME_1, $\mathcal{B}$ wins iff $\mathcal{A}$ wins.

It holds that

$$\mathrm{Adv}^{\mathrm{GAME}_1}_{\mathcal{A}_1,\Pi_{\mathsf{PRC}db},\Pi_{\mathsf{auth}}}(1^\lambda,n) \leq \mathrm{Adv}^{\text{AUTH-UF}}_{\mathcal{B},\Pi_{\mathsf{auth}}}(1^\lambda,n) \leq \mathsf{negl}(1^\lambda).$$

In summary, we find that

$$\begin{aligned}&\mathrm{Adv}^{\text{PRC-DBP}}_{\mathcal{A},\Pi_{\mathsf{PRC}db},\Pi_{\mathsf{auth}}}(1^\lambda,n)\\ \leq&\mathrm{Adv}^{\mathrm{GAME}_1}_{\mathcal{A}_1,\Pi_{\mathsf{PRC}db},\Pi_{\mathsf{auth}}}(1^\lambda,n)+\mathsf{negl}(1^\lambda)=\mathsf{negl}(1^\lambda),\end{aligned}$$

concluding the proof. □

D Security of our AC Revocation Scheme

In this section, we provide formal definitions of the core properties of anonymous credential revocation schemes and give extended formal proofs that our construction achieves them. In particular, we give a formal definition of *unforgeability*, *client unlinkability*, *revocation unlinkability*. We further extend the core definitions by database privacy in Appendix B. For brevity, we use the notation $\mathsf{getsublists}((x_i)_{i\in[n]},\mathcal{S})$ for a list $(x_i)_{i\in[n]}$ and set of lists $\mathcal{S}$ to denote the set of all lists that contain $(x_i)_{i\in[n]}$ as a subset, i.e., $\{S' \in \mathcal{S} \mid (x_i)_{i\in[n]} \subseteq S'\}$. In the following, we assume that adversaries implicitly keep a state throughout executions and use $\mathrm{Adv}^{\mathrm{GAME}}_{\mathcal{A},\Pi}$ to denote the advantage of adversary $\mathcal{A}$ in winning a security game GAME, instantiated with scheme Π. If clear from context, we omit Π as a parameter. Let params be the arguments of the security game. We define:

$$\mathrm{Adv}^{\mathrm{GAME}}_{\mathcal{A},\Pi}(\mathsf{params}) = \Pr[\mathrm{GAME}_{\mathcal{A},\Pi}(\mathsf{params})=1]$$

for games with computational win condition, and

$$\mathrm{Adv}^{\mathrm{GAME}}_{\mathcal{A},\Pi}(\mathsf{params}) = |\Pr[\mathrm{GAME}_{\mathcal{A},\Pi}(\mathsf{params})=1]| - \frac{1}{2}$$

for games with distinguishing win condition.

D.1 Properties of Anonymous Credentials

We extend classic anonymous credentials (AC) to support revocation. The security of our solution depends on the security of the underlying credential scheme. To present our security proofs, we provide game based definitions of unforgeability (in Figure 10) and unlinkability (in Figure 11) for classic anonymous credentials. Intuitively, unforgeability captures an adversaries ability to present a subset of credential attributes that it has not received a credential on before. This covers attacks in which the adversary attempts to combine attributes of different credentials in a single showing. Our unlinkability definition, for which we consider the adversary to corrupt the issuer and verifier, has the adversary distinguish between two credential presentations for different indices. This also captures multi-show unlinkability of single credentials, as the adversary is free to observe a number of self-generated presentations for each credential in advance.

GAME AC-UF$_{\mathcal{A},\mathsf{AC}}(1^\lambda)$:

1 : $(\mathsf{sk}_I,\mathsf{vk}_I) \leftarrow \mathsf{KGen}(1^\lambda)$
2 : $\mathcal{U} \leftarrow \emptyset$
3 : $(\mathsf{pt},(a_i)_{i\in\mathsf{Rvd}}) \leftarrow \mathcal{A}^{O^{\mathsf{Issue}}}(1^\lambda,\mathsf{vk}_I)$
4 : **return** $(\mathsf{getsublists}((a_i)_{i\in\mathsf{Rvd}},\mathcal{U})=\emptyset)$
$\wedge\ \mathsf{VerifyPresent}(\mathsf{vk}_I,\mathsf{pt},(a_i)_{i\in\mathsf{Rvd}})$

$O^{\mathsf{Issue}}((a_i)_{i\in[m]})$:

1 : $\mathcal{U} \leftarrow \mathcal{U}\cup\{(a_i)_{i\in[m]}\}$
2 : **return** $\mathsf{Issue}(\mathsf{sk}_I,(a_i)_{i\in[m]})$

Figure 10: Unforgeability game for an anonymous credential scheme AC (without revocation) as defined in §3. By λ, we denote the security parameter.

GAME AC-UL$_{\mathcal{A},\mathsf{AC}}(1^\lambda)$:

1 : $(\{\sigma_{\mathsf{cred},j},(a_{i,j})_{i\in[m]}\}_{j\in\{0,1\}},\mathsf{Rvd},\mathsf{vk}_I) \leftarrow \mathcal{A}(1^\lambda)$
2 : $b \xleftarrow{\$} \{0,1\}$
3 : **for** $i \in \{0,1\}$ **do** // b changes iteration order
4 : $\quad \mathsf{pt}_{i\oplus b} \leftarrow \mathsf{Present}(\sigma_{\mathsf{cred},i\oplus b},(a_{j,i\oplus b})_{j\in[m]},\mathsf{Rvd})$
5 : $b' \leftarrow \mathcal{A}(\mathsf{pt}_0,\mathsf{pt}_1)$
6 : **return** $(b'=b)\wedge\forall i\in\mathsf{Rvd}: a_{i,0}=a_{i,1}$

Figure 11: Unlinkability game for an anonymous credential scheme AC (without revocation) as defined in §3. By λ, we denote the security parameter.

D.2 Properties of AC Revocation Schemes

We now extend the property definitions of anonymous credential revocation schemes given in §4.3. In particular, we give game-based definitions of *unforgeability*, *client unlinkability* and *revocation unlinkability*. For unforgeability, we consider a threat model in which the client is malicious and up to $n-1$ RAs are semi-honest. The client unlinkability threat model considers up to $n-1$ RAs, colluding with a malicious verifier, while only $n-1$ RAs are malicious for the revocation unlinkability threat model.

Explicit presentation notation. Our PresentNonRev algorithm involves multiple parties. We extend our definition from Section 4.3 to include the RAs' inputs as follows:

$$\mathsf{pt}/\perp,(\mathrm{View}_i)_{i\in[n]} \leftarrow \begin{pmatrix}\mathsf{PresentNonRev}_{\mathcal{C},\mathcal{R}}\\ (\sigma_{\mathsf{cred}},(a_i)_{i\in[m]},\mathsf{Rvd},\mathsf{vk}_{\mathcal{R}}),\\ (\mathsf{sk}_{\mathcal{R},i},\mathbf{D},\mathsf{vk}_{\mathcal{R}})_{i\in[n]}\end{pmatrix}$$

where $\mathcal{R}=(\mathcal{R}_i)_{i\in[n]}$ denotes the set of all RAs. The first argument line of PresentNonRev denotes the client's inputs, while

GAME REV-UF$_{\mathcal{A},\Pi}(1^\lambda, n)$:

1 : $(\mathsf{sk}_I, \mathsf{vk}_I) \leftarrow \mathsf{KGen}(1^\lambda)$

2 : $\left((\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}, \mathbf{D}\right) \leftarrow \mathsf{RevSetup}(n)$

3 : $\mathcal{U} \leftarrow \emptyset$

4 : $\mathbf{D}, \mathsf{Corr} \leftarrow \mathcal{A}^{O^{\mathsf{Issue}}}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}})$

5 : $(\mathsf{pt}, (a_i)_{i\in\mathsf{Rvd}}) \leftarrow \mathcal{A}^{O^{\mathsf{Issue}}, O^{\mathsf{Present}}}\left((\mathsf{sk}_{\mathcal{R},i})_{i\in\mathsf{Corr}}\right)$

6 : $\mathcal{U}' \leftarrow \mathsf{getsublists}((a_i)_{i\in\mathsf{Rvd}}, \mathcal{U})$

7 : **return** $\mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}, (a_i)_{i\in\mathsf{Rvd}})$

/ Either forged a credential or presented a revoked credential

$\wedge\,(\mathcal{U}' = \emptyset \vee \forall S \in \mathcal{U}' : \mathbf{D}[S[0]] = 0)$

$\wedge\,(|\mathsf{Corr}| < n)$

$O^{\mathsf{Issue}}((a_i)_{i\in[m]})$:

1 : $\mathcal{U} \leftarrow \mathcal{U} \cup \{(a_i)_{i\in[m]}\}$

2 : **return** $\mathsf{Issue}(\mathsf{sk}_I, (a_i)_{i\in[m]})$

$O^{\mathsf{Present}}_{\mathcal{A}}(\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd})$:

1 : $(\mathsf{pt}, (\mathsf{View}_i)_{i\in[n]}) \leftarrow \begin{pmatrix}\mathsf{PresentNonRev}_{\mathcal{A},\mathcal{R}}\\ (\sigma_{\mathsf{cred}}, (a_i)_{i\in[m]}, \mathsf{Rvd}, \mathsf{vk}_{\mathcal{R}}),\\ (\mathsf{sk}_{\mathcal{R},i}, \mathbf{D}, \mathsf{vk}_{\mathcal{R}})_{i\in[n]}\end{pmatrix}$

2 : **return** $\mathsf{pt}, \{\mathsf{View}_i\}_{i\in\mathsf{Corr}}$

Figure 12: Game-based definition of unforgeability for an anonymous credential revocation scheme Π (§4). By λ, we denote the security parameter, and by n the number of RA servers. In this setting, the client is malicious and up to $n-1$ RAs are semi-honest.

the each element of the second entry denotes the input of a RA $\mathcal{R}_i$. By View_i, we denote the internal view of RA $\mathcal{R}_i$ during the execution of PresentNonRev, including all received and sent messages. Explicitly setting View_i as the output of the protocol interaction lets us model semi-honest adversaries in our security games.

Definition D.1 (Unforgeability)**.** *An anonymous credential scheme with revocation Π is unforgeable, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all $\lambda, n \in \mathbb{N}$,*

$$Adv^{\text{REV-UF}}_{\mathcal{A},\Pi}(1^\lambda, n) \leq \mathsf{negl}(1^\lambda),$$

where REV-UF$_{\mathcal{A},\Pi}$ *is defined in Figure 12.*

Definition D.2 (Client Unlinkability)**.** *An anonymous credential revocation scheme Π fulfills client unlinkability, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all $\lambda, n \in \mathbb{N}$,*

$$Adv^{\text{REV-UL}^C}_{\mathcal{A},\Pi}(1^\lambda, n) \leq \mathsf{negl}(1^\lambda)$$

where REV-UL$^C_{\mathcal{A},\Pi}$ *is defined in Figure 13.*

Definition D.3 (Revocation Unlinkability)**.** *An anonymous credential revocation scheme Π fulfills revocation unlinkability, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl*, such that for all $\lambda, n \in \mathbb{N}$,*

$$Adv^{\text{REV-UL}}_{\mathcal{A},\Pi}(1^\lambda, n) \leq \mathsf{negl}(1^\lambda)$$

where REV-UL$^R_{\mathcal{A},\Pi}$ *is defined in Figure 13.*

Remark. Our definition of revocation unlinkability in Figure 13 can be strengthened further, by allowing the adversary to choose different sets of revealed attributes for the presentations. Our scheme can also be shown to fulfill this stronger notion, yet we omit this detail for clearer presentation.

GAME REV-UL$^{C/R}_{\mathcal{A},\Pi}(1^\lambda, n)$:

1 : $(\mathbf{D}, \mathsf{Corr}, (\mathsf{sk}_{\mathcal{R},i})_{i\in[n]}, \mathsf{vk}_{\mathcal{R}}, \mathsf{vk}_I) \leftarrow \mathcal{A}(1^\lambda, n)$

2 : $\left(\{\sigma_j, (a_{i,j})_{i\in[m]}\}_{j\in\{0,1\}}, \mathsf{Rvd}\right) \leftarrow \mathcal{A}()$

3 : $b \xleftarrow{\$} \{0,1\}$

4 : **for** $j \in \{0,1\}$ **do** / b changes iteration order

5 : $(\mathsf{pt}_{j\oplus b}, \bot) \leftarrow \begin{pmatrix}\mathsf{PresentNonRev}_{\mathcal{C},\mathcal{A},(\mathcal{R}_i)_{i\notin\mathsf{Corr}}}\\ (\sigma_{j\oplus b}, (a_{i,j\oplus b})_{i\in[m]}, \mathsf{Rvd}, \mathsf{vk}_{\mathcal{R}}),\\ (\mathsf{sk}_{\mathcal{R},i}, \mathbf{D}, \mathsf{vk}_{\mathcal{R}})_{i\in[n]}\end{pmatrix}$

6 : $b' \leftarrow \mathcal{A}(\mathsf{pt}_0, \mathsf{pt}_1)$

7 : **return** $(b' = b) \wedge (|\mathsf{Corr}| < n)$

$\wedge\,(\mathsf{pt}_0, \mathsf{pt}_1 \neq \bot) \wedge \forall i \in \mathsf{Rvd} : a_{i,0} = a_{i,1}$

$\wedge\,(\mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}_0, (a_{0,i})_{i\in\mathsf{Rvd}})$

$= \mathsf{VerifyNonRev}(\mathsf{vk}_I, \mathsf{vk}_{\mathcal{R}}, \mathsf{pt}_1, (a_{1,i})_{i\in\mathsf{Rvd}}))$

Figure 13: Unlinkability games for an anonymous credential revocation scheme Π. By λ, we denote the security parameter, and by n the number of RA servers. In this setting, up to $n-1$ RAs are malicious. Unlike the revocation unlinkability game REV-UL^R, the client unlinkability game REV-UL^C includes the highlighted statements.

D.3 Security Proofs for our AC Revocation Construction

We now give formal proofs for the security of our anonymous credential revocation scheme Π_{rev}, as defined in §6. In particular, we prove unforgeability, client unlinkability and revocation unlinkability, via reduction to the security properties of our PRC construction, which we have analyzed in §C.3.

Theorem D.1 (Unforgeability)**.** *Our anonymous credential revocation scheme Π_{rev} in §6 is unforgeable, if the underlying credential scheme is unforgeable, the threshold signature and commitment schemes used in PRC are unforgeable and binding and the utilized proof system fulfills knowledge-soundness.*

Proof. We show that for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl, such that for all $1^\lambda, n \in \mathbb{N}$, it holds that:

$$\begin{aligned}\mathrm{Adv}^{\text{REV-UF}}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda,n) \leq\ & \mathrm{Adv}^{\text{AC-UF}}_{\mathcal{B},\mathsf{AC}}(1^\lambda,n) \\ &+ \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{C},\mathsf{TSig}}(1^\lambda) \\ &+ \mathrm{Adv}^{\text{GAMEBND}}_{\mathcal{D},\mathsf{Com}}(1^\lambda) + \mathsf{negl}(1^\lambda).\end{aligned}$$

Our anonymous credential revocation scheme Π_{rev} builds upon our PRC construction Π_{PRC}, presented in §5.2. By the unforgeability proof of Π_{PRC} in §C.3, for all PPT adversaries $\mathcal{E}$, there exist PPT adversaries $\mathcal{C}$ and $\mathcal{D}$, such that for all $\lambda, n \in \mathbb{N}$,

$$\begin{aligned}\mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{E},\Pi_{\mathsf{PRC}}}(1^\lambda,n) \leq\ & + \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{C},\mathsf{TSig}}(1^\lambda) \\ &+ \mathrm{Adv}^{\text{GAMEBND}}_{\mathcal{D},\mathsf{Com}}(1^\lambda).\end{aligned}$$

To prove the theorem, it remains to show that for all PPT adversaries $\mathcal{A}$ there exist PPT adversaries $\mathcal{B}$ and $\mathcal{E}$, such that for all $\lambda, n \in \mathbb{N}$,

$$\begin{aligned}\mathrm{Adv}^{\text{REV-UF}}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda,n) \leq\ & \mathrm{Adv}^{\text{AC-UF}}_{\mathcal{B},\mathsf{AC}}(1^\lambda,n) \\ &+ \mathrm{Adv}^{\text{PRC-UF}}_{\mathcal{E},\Pi_{\mathsf{PRC}}}(1^\lambda,n) + \mathsf{negl}(1^\lambda).\end{aligned}$$

We show this statement via a game-hop and a reduction to the unforgeability of the underlying PRC and credential schemes. Let $\text{GAME}_0 = \text{REV-UF}_{\mathcal{A},\Pi_{\mathsf{PRC}}}$.

Game$_1$. We define GAME_1 to be identical to GAME_0, but make the following modification: all presentation proofs are replaced by sending over the witness (hidden attributes, credential and non-revocation token) directly. If the underlying proof system is knowledge-sound, there exists an efficient extractor Ext, that given oracle access to a prover can extract the witness from a proof with overwhelming probability. An adversary $\mathcal{A}_1$ in GAME_1 can emulate oracle access to an isolated prover, by running $\mathcal{A}$ up to the point where $\mathcal{A}$ generates a proof and storing the execution state of the partial execution. $\mathcal{A}_1$ can then run Ext and provide it with oracle access to the partially executed adversary $\mathcal{A}$, sending the output of Ext it in plain to its $\mathsf{Response}$ oracle. As Ext is successful with overwhelming probability, it holds that

$$\mathrm{Adv}^{\text{GAME}_0}_{\mathcal{A},\Pi_{\mathsf{rev}}} \leq \mathrm{Adv}^{\text{GAME}_1}_{\mathcal{A}_1,\Pi_{\mathsf{rev}}} + \mathsf{negl}(1^\lambda).$$

Given $\mathcal{A}_1$, we now construct PPT adversaries $\mathcal{B}$ for AC-UF and $\mathcal{E}$ for PRC-UF.

If $\mathcal{A}_1$ wins GAME_1, it must have produced a valid presentation pt, for some index idx. At least one of two win conditions holds: (1) The forgery reveals an attribute set that was never issued ($\mathcal{U}' = \emptyset$), (2) the presented credential is revoked, yet the token verifies ($\forall S \in \mathcal{U}' : \mathbf{D}[S[0]] = 0$).

If $\mathcal{A}_1$ produces a type (1) forgery, we can construct a PPT adversary $\mathcal{B}$ for AC-UF as follows: When provided with vk_I from its challenger, $\mathcal{B}$ forwards it to $\mathcal{A}_1$ and runs $\mathsf{RevSetup}$. The reduction forwards any O^{Issue} queries by $\mathcal{A}_1$ to its own challenger and returns the credential to $\mathcal{A}_1$. The presentation oracle is handled locally by $\mathcal{B}$, simulating the servers in the PRC response protocol, which is enabled by $\mathcal{B}$'s knowledge of all RA keys. $\mathcal{A}_1$ outputs a tuple $(\sigma_{\mathsf{rec}}, \mathsf{idx}, \mathsf{rec}, \mathsf{aux}, \mathsf{qst}, \sigma_{\mathsf{cred}}, (a_i)_{i\in[m]})$ as its forgery, where σ_{cred} is a forgery on message $(a_i)_{i\in[m]}$ in $\mathcal{B}$'s AC-UF game. For type (1) forgeries, $\mathcal{A}_1$ has forged a valid presentation proof for AC, on a novel set of attributes.

As any interaction with O^{Present} only produces tokens for the correct database value, an adversary producing a type (2) forgery must have forged a PRC token. In detail, an adversary $\mathcal{E}$ for PRC-UF might sample its own issuer key, before obtaining the server secret keys from its challenger. Using its knowledge of sk_I, $\mathcal{E}$ can trivially answer O^{Issue} queries, while an O^{Present} query is forwarded to its own O^{Present} oracle, before combining the response with the attribute vector and credential. If $\mathcal{A}_1$ produces a type (2) forgery, the presentation is valid, yet $\mathbf{D}[S[0]] = 0$ for all $S \in \mathcal{U}'$. As $\mathcal{U}'$ only contains attribute sets issued by the $\mathsf{Present}$ oracle, this presentation constitutes a PRC forgery as per Definition C.2.

Using the results for both types of forgeries and a union bound, we find that

$$\begin{aligned}\mathrm{Adv}^{\text{REV-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda) \leq\ & \mathrm{Adv}^{\text{REV-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda) \cdot (\Pr[\text{type (1) forgery}]) \\ &+ \mathrm{Adv}^{\text{REV-UF}}_{\mathcal{A},\Pi_{\mathsf{PRC}}}(1^\lambda) \cdot (\Pr[\text{type (2) forgery}]) \\ &\leq \mathrm{Adv}^{\text{AC-UF}}_{\mathcal{B},\mathsf{AC}}(1^\lambda) \\ &+ \mathrm{Adv}_{\mathcal{C},\mathsf{TSig}}\text{EUF-CMA}(1^\lambda) \\ &+ \mathrm{Adv}^{\text{GAMEBND}}_{\mathcal{D},\mathsf{Com}}(1^\lambda) + \mathsf{negl}(1^\lambda).\end{aligned}$$

□

A remark on zk-credentials and AC unlinkability. The credential schemes we use to instantiate our scheme all fall in the category of *zk-credentials*, i.e. credentials that adhere to the framework of building anonymous credentials from a commitment scheme and an efficient zero-knowledge proof system. Naturally, credentials of this type allow for efficient proofs over the credential and work well with the signed commitment provided by the PRC servers. To be more explicit (and support other types of credentials), we separate proof system and anonymous credential in our definitions. Our construction Π_{rev} then makes non-black-box use the credential scheme, in the sense that it proves a relation between the PRC token and the credential itself, not its presentation proof. Therefore, the following theorems depend only on the zero-knowledge property of the proof system used, not on the unlinkability of the

AC scheme, as that is defined with respect to the presentation, not the credential. It is important to note however, that for all zk-credentials, i.e. all natural instantiations of the scheme, the associated proof system is zero-knowledge iff the credential scheme is unlinkable. For zk-credentials, unlinkability is therefore very much required.

Theorem D.2 (Client Unlinkability)**.** *Our anonymous credential revocation scheme* Π_{rev} *in §6 fulfills client unlinkability in the* $\mathcal{F}_{\mathsf{CRG}}$*-hybrid model (see §C.3), if the proof system fulfills zero-knowledge, and the commitment scheme is hiding.*

Proof. We show, that for all PPT adversaries $\mathcal{A}$, there exists a PPT adversaries $\mathcal{B}$ and a negligible function negl, such that for all $\lambda, n \in \mathbb{N}$:

$$\mathrm{Adv}^{\text{REV-UL}^C}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GAMEHID}}_{\mathcal{B},\mathsf{Com}}(1^\lambda) + \mathsf{negl}(1^\lambda).$$

Our credential revocation construction Π_{rev} builds on top of our PRC scheme Π_{PRC} (§5.2). As shown in §C.3, for each adversary $\mathcal{C}$ for $\text{PRC-UL}_{\mathcal{C},\Pi_{\mathsf{PRC}}}$, there exists an adversary $\mathcal{B}$ for $\text{GAMEHID}_{\mathcal{B},\mathsf{Com}}$, such that for all $\lambda, n \in \mathbb{N}$

$$\mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GAMEHID}}_{\mathcal{B},\mathsf{Com}}(1^\lambda).$$

It remains to show that

$$\mathrm{Adv}^{\text{REV-UL}^C}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n) + \mathsf{negl}(1^\lambda).$$

For client unlinkability, we assume that the verifier is colluding with the adversarial RA, of which up to $n-1$ parties are compromised. We show the statement via a game-hop, in which we simulate the employed zero-knowledge proofs. Finally, we reduce the unlinkability of the modified game to that of PRC. Let $\mathcal{A}$ be a PPT adversary with non-negligible advantage in REV-UL^C. Further, we let $\text{GAME}_0 = \text{REV-UL}^C_{\mathcal{A},\Pi_{\mathsf{rev}}}$.

Game$_1$**.** We define GAME_1 to be identical to GAME_0, but with the following change: We let the challenger simulate the zero-knowledge proofs used as part of the presentation. Therefore, both pt_0 and pt_1 are now independent of the witness used to generate them, including the credential σ_{cred} and all $(a_i)_{i \notin \mathsf{Rvd}}$. Notably, the presentations still reveal the PRC token σ_{rec}. If the employed proof system is zero-knowledge, adversary $\mathcal{A}$ can only distinguish between real and simulated proofs with negligible probability. If follows that

$$\mathrm{Adv}^{\text{GAME}_0}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GAME}_1}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) + \mathsf{negl}(1^\lambda).$$

Given an adversary $\mathcal{A}$ in GAME_1, we construct an adversary $\mathcal{C}$ for the unlinkability of PRC as follows: $\mathcal{C}$ obtains the database, corruption set and RA keys from $\mathcal{A}$ and forwards them to its challenger. Upon receiving the attribute set from $\mathcal{A}$, it forwards the two indices $a_{0,0}$ and $a_{0,1}$ to its challenger. To simulate a PresentNonRev protocol interaction in GAME_1, $\mathcal{C}$ forwards all messages belonging to Present that are sent and received by $\mathcal{A}$ to its challenger. The remaining presentation proof of the credential scheme may be simulated by $\mathcal{C}$. The combination of PRC record certificate and simulated proofs constitutes a valid presentation that $\mathcal{C}$ can output to $\mathcal{A}$. $\mathcal{C}$ can output the same bit as $\mathcal{A}$ and is successful iff $\mathcal{A}$ is. It follows that

$$\mathrm{Adv}^{\text{GAME}_1}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) = \mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n),$$

which concludes the proof. □

Theorem D.3 (Revocation Unlinkability)**.** *When instantiated with a hiding commitment scheme, our anonymous credential revocation scheme in §6 fulfills revocation unlinkability in the* $\mathcal{F}_{\mathsf{CRG}}$*-hybrid model (§C.3).*

Proof. We show, that for all PPT adversaries $\mathcal{A}$, there exists a PPT adversaries $\mathcal{B}$, such that for all $\lambda, n \in \mathbb{N}$:

$$\mathrm{Adv}^{\text{REV-UL}^R}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GAMEHID}}_{\mathcal{B},\mathsf{Com}}(1^\lambda).$$

Our credential revocation construction Π_{rev} builds on top of our PRC scheme Π_{PRC} (§5.2). As shown in §C.3, for each adversary $\mathcal{C}$ for $\text{PRC-UL}_{\mathcal{C},\Pi_{\mathsf{PRC}}}$, there exists an adversary $\mathcal{B}$ for $\text{GAMEHID}_{\mathcal{B},\mathsf{Com}}$, such that for all $\lambda, n \in \mathbb{N}$

$$\mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{GAMEHID}}_{\mathcal{B},\mathsf{Com}}(1^\lambda).$$

It remains to show that

$$\mathrm{Adv}^{\text{REV-UL}^R}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) \leq \mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n).$$

In contrast to client unlinkability, the revocation unlinkability challenger does not supply the adversary $\mathcal{A}$ with the final presentations. Therefore, we can use $\mathcal{A}$ reduce to PRC-UL directly. We build an adversary $\mathcal{C}$ for the unlinkability of PRC as follows: $\mathcal{C}$ obtains the database, corruption set and RA keys from $\mathcal{A}$ and forwards them to its challenger. Upon receiving the attribute set from $\mathcal{A}$, it forwards the two indices $a_{0,0}$ and $a_{0,1}$ to its challenger. To simulate a PresentNonRev protocol interaction in GAME_1, $\mathcal{C}$ forwards all messages belonging to Present that are sent and received by $\mathcal{A}$ to its challenger. This way, $\mathcal{C}$ perfectly simulates REV-UL^R to $\mathcal{A}$. Since $\mathcal{A}$ only obtains information dependent on the secret bit b via the PresentNonRev interaction, $\mathcal{C}$ can output the same bit as $\mathcal{A}$ and be successful iff $\mathcal{A}$ is. It follows that

$$\mathrm{Adv}^{\text{REV-UL}^R}_{\mathcal{A},\Pi_{\mathsf{rev}}}(1^\lambda, n) = \mathrm{Adv}^{\text{PRC-UL}}_{\mathcal{C},\Pi_{\mathsf{PRC}}}(1^\lambda, n),$$

concluding the proof. □

E Pre-processing and correlated randomness generation

Efficient SMC protocols heavily rely on preprocessing to reduce their latency and online phase cost. We use Beaver triples [7] for performing AND operations and daBits [47, 92]

for Boolean to arithmetic conversions. The majority of MPC protocols have expensive preprocessing phases requiring GiBs of communication per execution. However, our protocol only requires sub-linear preprocessing material. Specifically, in our largest setting with binary credential statuses ($l = 1$) and a billion credentials ($N = 10^9$), generating $\sim 3\sqrt{N}$ Beaver triples and $\lg(N) + l$ daBits require a transfer of 13.5 KiB for all servers combined when generated via a dealer.

Memory. The pre-processing material required for each query is only 37 KiB small. The client is not involved in the MPC computation and pre-processing material is independent of the querier; thus, servers do not need to fetch client dependent material. Reserving 1 GiB of RAM and pre-loading material in the background supports responding to over 28k queries. We assume that correlation material is available in RAM.

Generation. We use a dealer to generate correlations. The dealer exchanges symmetric keys with the parties in the beginning. To generate Beaver triples $(a, b, c = a \wedge b)$, the dealer uses a keyed PRF with the symmetric keys to generate shares of $n-1$ parties without having to transfer this material. For the last party, the dealer still uses a keyed PRF to randomly select (a_n, b_n) and only computes and sends the final share $c_n = \bigoplus_{i\in[n-1]} c_i \oplus (\bigoplus_{i\in[n]} a_i \wedge \bigoplus_{i\in[n]} b_i)$, requiring 1-bit of transfer per AND triples regardless of the number of parties. We note that keeping triples in RAM requires 3-bits. To generate daBits $(b \in \{0,1\}, a \in \mathbb{Z}_q = b)$, we follow a similar approach of using a keyed PRF to generate $n-1$ shares and send $(b_n \xleftarrow{\$} \{0,1\}, a_n = (\bigoplus b_i - \sum a_i) \bmod q)$ to the last party. Sending and storing a daBit requires $(\lg q + 1) = 257$ bits regardless of the number of parties.

Alternative generations. Generating correlated randomness involves several trade-offs, including cost (computation and communication), trust assumptions (information-theoretic or computational), parties (honest majority, dishonest majority, dealer-based, or TEE-based), and adversary model (passive or active). The choice of how preprocessing material should be generated should be decided based on the deployment scenario. We provide a summary of existing approaches, but comparing them is beyond the scope of this paper.

Our solution requires only a few correlations per query, but we have to support a high number of queries. As preprocessing generates correlations far in advance, we can produce them in large batches for a high number of queries, allowing high batching parameters and thus low amortized cost. A common preprocessing approach is using additive homomorphic encryption [84] or somewhat homomorphic encryption [18, 19] to generate correlations [38, 39, 69]. Cut-and-choose techniques can further ensure correctness and security [38, 47], and our setting naturally supports very large batches that improve efficiency. Another approach is using oblivious transfer to generate correlations [68]. Advances such as OT extensions [64], SoftSpoken [93], or Silent OT [13] significantly lower the communication cost. Furthermore, pseudorandom

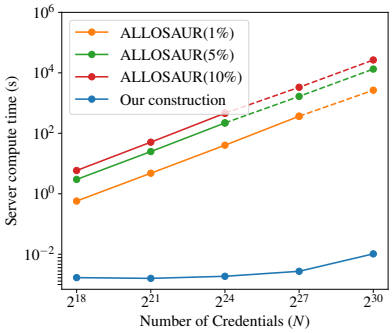

Figure 14: Server computation time for different numbers of credentials and varying revocation rates for our revocation construction vs. ALLOSAUR.

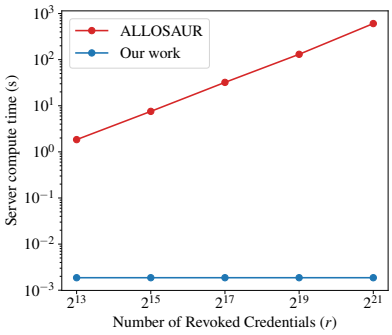

Figure 15: Server computation time for a fixed number of 10M credentials and different numbers of revoked credentials for our revocation construction vs. ALLOSAUR.

Correlation Generators (PCG) [12, 13, 89] offer communication sublinear in the number of correlations at the price of more expensive computation. Leveraging zk-FLIOPs [10] enables lifting the security of PCGs to a malicious setting while retaining the sublinear cost [14, 17]. Finally, instead of relying on public key operations, one can use a trusted execution environment to replace the role of the dealer. As dealer's interactions are limited to the preprocessing, the TEE is only needed for an initialization ceremony, after which the state of TEE can be securely wiped, thus limiting the scope of hardware attacks.

F Comparison with ALLOSAUR

The ALLOSAUR scheme [65] proposes a revocation mechanism in a threshold setting. They utilize MPC to let clients retrieve an updated witness for an accumulator, which accumulates all non-revoked credentials. Conceptually, privately obtaining the up-to-date accumulator witness by the MPC servers can be considered functionally similar to obtaining a non-revocation token as done in our work. While in our

system updating the revocation database is trivial (i.e., just changing a bit in a private table), updating the accumulator in ALLOSAUR incurs linear work in the number of total credentials for the accumulator managers. Every update also forces clients to refresh their witnesses, which should be performed regularly at fixed intervals to maintain anonymity. To do so, clients first send a secret-shared query to the servers, and then retrieve update information which they can recombine to update their witness locally. However, the client and server computation and communication cost scales with the number of revocations since the client last updated its witness.

We compare ALLOSAUR and our solution with a RAs consisting of 2 servers in terms of server computation time with varying numbers of credentials and revocation rates. While ALLOSAUR costs depend on the number of revocations, our scheme only depends on the number of credentials. Note that for ALLOSAUR, we ignore the linear cost for each credential revocation itself and only consider the cost for the witness update. Considering 1 billion credentials with a conservative revocation rate of 1%, the server computation time for a witness update in ALLOSAUR would require approximately 2663 seconds, while our solution only needs 10.294 ms server computation time. As we were unable to run ALLOSAUR for 1 billion credentials, we extrapolated the server computation time and visualize extrapolated numbers as dashed lines in Figure 14. We compare the server computation time of our solution with ALLOSAUR for varying numbers of credentials and different revocation rates for ALLOSAUR in Figure 14. Additionally, in Figure 15, we show the server computation time for fixed number of credentials and varying revocation rates.

G Artifact Appendix

G.1 Abstract

We provide an artifact containing a proof-of-concept implementation of the Private Record Certification (PRC) protocol described in the paper. We also provide Docker files which automate the process of benchmarking our solution and generating the performance plots in the paper.

G.2 Description & Requirements

G.2.1 Security, privacy, and ethical concerns

This artifact is non-destructive and only collects performance measurement of its execution. There are no security, privacy, or ethical concerns.

G.2.2 How to access

The artifact is available on https://zenodo.org/records/20433787.

G.2.3 Hardware dependencies

This artifact requires an x86_64 CPU with AVX2 support. This includes most modern Intel and AMD CPUs, but excludes ARM-based platforms, including Apple Silicon.

Our benchmark evaluates up to 8 PRC servers. We measure single-core performance and restrict each server to one core. To simplify the process, we run all servers on the same machine with a simulated network. Therefore, we recommend running the full benchmark on a machine with at least 8 cores. If fewer cores are available, multiple servers will share cores, which may increase latency. The RAM footprint is low. A machine with 8 GiB of RAM should be sufficient for running the full 8-server benchmark.

G.2.4 Software dependencies

We provide a Docker container, based on Debian, to facilitate running the artifact. We require Docker to be installed on the device. We only provide instructions on how to run the container on a Linux machine, but it is possible to run it on different operating systems.

G.2.5 Benchmarks

None.

G.3 Set-up

G.3.1 Installation

To install the artifact, download it or clone the corresponding Git repository. The artifact requires Docker; if Docker is not already installed, follow the official installation instructions.[2] In the artifact's top-level directory, build the container:

```
sudo docker build -t prc .
```

G.3.2 Basic Test

This artifact is built in Rust, and we use `cargo test` for basic tests. You can use the following command to run all tests in the container. We expect that all tests should pass.

```
sudo docker run \
 --rm \
 -v "$(pwd)/measurements:/AnonCred/measurements" \
 prc \
 cargo test
```

G.4 Evaluation workflow

G.4.1 Major Claims

This artifact reproduces the server-side PRC measurements (cf. Figure 3). The benchmark directly measures latency, com-

[2] https://docs.docker.com/get-docker/

putation cost, and communication cost and estimates the operation's monetary cost based on the AWS price model.

(C1): Sub-second latency at billion-credential scale. PRC achieves sub-second latency even for databases with over one billion credentials. The latency is dominated by the simulated network delay. We also note that when RTT is 20 ms, the latency is under 100 ms. This is proven by the latency measures provided in `latency.pdf` plot as part of experiment (E1).

(C2): Linear server computation. The computation cost of PRC servers grows linearly with the number of credentials. This is proven by the server computation cost plotted in `comp_comm.pdf` generated in experiment (E1).

(C3): Sublinear server communication. The communication cost of PRC servers grows sublinearly with the number of credentials; more precisely, it grows as $O(\sqrt{N})$. This is proven by server communication cost plotted in `comp_comm.pdf` generated in experiment (E1).

(C4): Efficient monetary cost at scale. The monetary cost grows linearly with the number of credentials, and the protocol remains efficient even when processing millions of PRC queries. The estimated monetary cost is computed from measured resource consumption using AWS pricing. Our pricing model is as follows: one CPU core, less than 2 GiB RAM, $0.04 per compute hour, free inbound transfer, and $0.09 per GiB outbound. Our cost efficiency is proven by the operation cost plotted in `costs.pdf` generated in experiment (E1).

The paper also makes the following client-side claims, which are not directly shown in the produced figures.

(C5): Client cost is independent of the number of revocations. In our protocol, revoking a credential only requires changing a value in the servers' database. There is no corresponding client-side update, and client queries are agnostic to the changes in the servers' database. The code shows that any change in revocation status is fully server-side, with no impact or notification to clients.

(C6): Client cost is logarithmic to the number of credentials. In theory, the client cost is logarithmic in the number of credentials because the record index size grows with the database size. In practice, we represent indices as a single integer where a `u32` supports up to four billion records. Therefore, within the evaluated range, the client cost is effectively independent of the number of credentials.

The paper compares the cost of supporting anonymous-credential revocation using our PRC construction against a naive PIR-based approach and ALLOSAUR. For the naive PIR baseline, the cost corresponds to performing a single PIR query over a database of n records, each of size 96 B. For ALLOSAUR, we use the implementation as provided, without modification.

Comparison with related work Because these comparisons evaluate unmodified external implementations—Hintless PIR[3], DPF-based PIR[4], and ALLOSAUR[5]—we do not include their source code in this artifact. A database with one billion 96 B entries requires 96 GB of storage, which is beyond the scalability of existing PIR implementations. We therefore extrapolate costs for settings in which prior systems fail to run.
We describe how we ran the external implementations, the raw measurements we obtained, and the extrapolation method used beyond their scalability limits. Since the external projects already provide their own installation and usage instructions, we do not include detailed setup instructions or automation for those packages.

G.4.2 Experiments

(E1): [human-interaction: minimal, compute-time: 3:30 min, peak memory: 1 GiB]: Run the automated evaluation.
How to: The evaluation is fully automated. After installing Docker and building the image per set-up instructions, run the following command:

```
sudo docker run \
 --cap-add=NET_ADMIN \
 --rm \
 -v "$(pwd)/measurements:/AnonCred/measurements" \
 prc \
 bash /AnonCred/bench_run_all.sh
```

Results: This script runs PRC protocols with $\{2,4,8\}$ PRC servers on a single machine with a simulated network—setting an RTT of $\{20\,\text{ms}, 40\,\text{ms}, 80\,\text{ms}\}$ via `tc` command—and measures the latency, server computation, and communication cost of running a PRC query based on the number of records held by the server. We run each experiment 5 times, record the performance measurement, and plot the costs. After the benchmark finishes, the following plots `latency.pdf`, `costs.pdf`, and `comp_comm.pdf` will be generated in the `measurements/` directory. These three plots correspond to Figure 3 in the paper and support our claims.

The PRC measurements reported in the paper are measure on a system running Debian 13 with an AMD Ryzen AI 7 PRO 350 CPU and 64 GB RAM.

G.5 Notes on Reusability

We provide detailed instructions on how to build the artifact from the source, manually run the server with customizable parameters, and describe the artifact structure and subcomponents that may be of independent interest, such as SIMD-aligned bit-arrays, our custom MPC engine, conversions, and PIR construction in the artifact readme. If you are interested in using/reusing our prototype, please refer to the latest version of the repository.

[3] https://github.com/google/hintless_pir
[4] https://github.com/google/distributed_point_functions
[5] https://github.com/sam-jaques/allosaurust

G.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.