IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 3, 29 pages.

https://doi.org/10.62056/akmpxrdja

Check for updates

Post-Quantum Access Control with Application to Secure Data Retrieval

Behzad Abdolmaleki[1], Hannes Blümel[2], Tianxiang Dai[3], Giacomo Fenzi[4], Homa Khajeh[5], Stefan Köpsell[2] and Maryam Zarezadeh[2]

[1] University of Sheffield, Sheffield, United Kingdom
[2] Barkhausen Institut, Dresden, Germany
[3] Lancaster University Leipzig, Leipzig, Germany
[4] EPFL, Lausanne, Switzerland
[5] Independent researcher, Isfahan, Iran

Abstract.
Servan-Schreiber et al. [IEEE S&P, 2023] presented a new notion called private access control lists (PACL) for function secret sharing (FSS), where the FSS evaluators can ensure that the FSS dealer is authorized to share the given function. Their construction relies on costly non-interactive secret-shared proofs and is not secure in the post-quantum setting. We give a construction of PACL from publicly verifiable secret sharing (PVSS) under the short integer solution (SIS) problem. Our construction adapts the Gentry et al.'s scheme [EUROCRYPT, 2022] for the post-quantum setting based on the learning with error (LWE) assumption aimed at ensuring that database access control policies are enforced by FSS evaluators, who verify that the FSS dealer is authorized to share the specified database query function. The benchmarks of our PACL show a trade-off between proving and verification efficiency, making the optimal choice dependent on application requirements. This construction has many applications for access control by applying FSS. We also present a secure data retrieval scheme using DPF-PACL for access control, which complements the PACL framework while providing independent utility.

Keywords: Access Control · Function Secret Sharing · Publicly Verifiable Secret Sharing · PACL

1 Introduction

The widespread adoption of distributed computing services, such as cloud computing, has significantly transformed how organizations manage and store data. However, this shift has also raised substantial concerns regarding the privacy and security of sensitive information, including financial records, medical histories, and personal data. While encryption is commonly applied to secure the contents of outsourced databases, it does not prevent service providers from accessing metadata, leaving a critical gap in data privacy protections. To address these privacy challenges, researchers have developed advanced protocols for secure data access and retrieval, including user authentication [DAT20, SZAJ20, RDV24], role-based access controls [ZVH11, TSCS+19, RPWG20, JBN+21], and detailed activity auditing [WWRL10, LCZL17]. These schemes enable users to access data stored on

E-mail: behzad.abdolmaleki@sheffield.ac.uk (Behzad Abdolmaleki), hannes.bluemel@barkhauseninstitut.org (Hannes Blümel), t.dai@lancaster.ac.uk (Tianxiang Dai), giacomo.fenzi@epfl.ch (Giacomo Fenzi), homa0khajeh@gmail.com (Homa Khajeh), stefan.koepsell@barkhauseninstitut.org (Stefan Köpsell), maryam.zarezadeh@barkhauseninstitut.org (Maryam Zarezadeh)

This work is licensed under a "CC BY 4.0" license.
Received: 2025-07-05 Accepted: 2025-09-02

servers without revealing specific data items being accessed. However, implementing these schemes involves a trade-off between protecting user privacy and allowing service providers to enforce security policies and maintain control over the data. To overcome this limitation, this study explores enforcing access control with privacy using function secret sharing (FSS) [BGI15]. FSS enables a dealer to distribute compact secret shares of a function to evaluators, allowing them to compute parts of the function's output on a shared input. Even if some evaluators collude, they can only infer the function's general class, not its specific details, preserving confidentiality while maintaining secure access. FSS is a foundational component in numerous privacy-oriented technologies, including systems for private databases [WYG+17, DRPS22], privacy-preserving machine learning models [RTPB20, LCC23, JMDL24], private clustering [FCS+24], searchable encryption [LZW+24], and private set intersection [DIL+22, Sun24]. These applications often process sensitive, user-specific data, and they frequently operate in environments where some users may attempt to bypass safeguards, making access control a crucial aspect. For instance, FSS enables users to interact with databases in a private manner, whether they are reading or writing data [CZLC20, BKKO20, GRS22] or query on a database [WYG+17, ZCD+24]. In these scenarios, access control mechanisms are essential to ensure that users cannot access or alter the data belonging to others, thus preventing unauthorized data exposure or tampering by potentially malicious actors.

Our Contributions. We introduce a post-quantum access control mechanism for secure data retrieval that ensures only authorized users can access specific database records. Our approach utilizes FSS in a black-box manner, enabling a function f to be split into shares that different servers can evaluate on an input x. These evaluations are then combined to reconstruct $f(x)$ while ensuring that neither the function itself nor its inputs are exposed. This method is particularly beneficial in multi-user environments where different users have varying access rights to specific computations. A key challenge in this setting is ensuring that the servers performing the evaluations can verify whether a client has the correct access rights to participate in the secret sharing process. At the same time, it is crucial to preserve access control privacy so that neither the function being evaluated nor the client's specific permissions are revealed to the servers. To address these challenges, we aim to design a secure access control mechanism for FSS, ensuring that only authorized clients can utilize the system while keeping their permissions confidential. We then apply this mechanism to secure data retrieval.

Our method builds upon the concept of private access control lists (PACL) introduced by Servan-Schreiber et al. [SSBYP23] for FSS. The PACL mechanism allows FSS evaluators to verify that the FSS dealer is authorized to distribute the specified function. We propose a PACL construction utilizing publicly verifiable secret sharing (PVSS) based on the short integer solution (SIS) problem. Our method builds on the PVSS scheme of Gentry et al. [GHL22], which we adapt for post-quantum security using the learning with errors (LWE) assumption. While [GHL22] indeed used LWE for post-quantum security, their PVSS design also relied on bulletproofs under the discrete logarithm (DL) assumption. They explicitly left as an open problem the construction of a fully post-quantum PVSS, suggesting exploration of lattice-based alternatives and studying the performance impact. Our work addresses this open problem by replacing DL-based bulletproofs with a STARK-family proof system (RISC Zero [BG23]) over the LWE assumption, re-engineering verification for PVW ciphertexts, and proving the resulting scheme's correctness, privacy, and soundness in the post-quantum model. This PVSS is then integrated into a PACL system which was not done in [GHL22].

In comparison with [SSBYP23], our approach does not rely on expensive secret-shared proofs and guarantees security in a post-quantum setting. Our PACL evaluation with various files has shown its practicality across different sizes, indicating that it is likely to perform well with large secret vectors as well. We also illustrate how our PACL design can

be used for secure data retrieval and present a secure data retrieval scheme with access control which may be of independent interest.

Technical Overview. We present a method for PACL for FSS in the post-quantum setting. Our approach uses FSS to distribute a function f among multiple evaluators in a privacy-preserving manner. Given an input x, the evaluators compute shares of $f(x)$ without learning any additional information about the function f itself. This property is particularly crucial for privacy-preserving access control, ensuring that access policies remain confidential while still being enforced correctly. The system consists of three main entities: a dealer, a set of evaluators, and users. The dealer is responsible for generating the FSS shares of the function and distributing them securely to the evaluators. The evaluators are servers that collectively enforce access control policies without having direct knowledge of the underlying function f. Users interact with the system by submitting queries to request access to certain resources or data. Since the evaluators do not possess full knowledge of f, they cannot infer sensitive information about the access control policy.

The access control process begins with the dealer generating secret shares of f using the FSS scheme and distributing them to the evaluators, ensuring that no single evaluator has complete knowledge of the function. A user then submits a query corresponding to an input x for which they seek access. This query is processed without revealing any sensitive details about the access policy. Each evaluator, using its share of the function, computes a share of $f(x)$ locally. Since these computations are performed independently on secret-shared data, no single evaluator gains complete knowledge of the function output. Finally, the individual shares computed by the evaluators are combined to reconstruct $f(x)$, and based on the reconstructed result, the system determines whether access should be granted or denied. By following this structured process, our approach ensures that access control decisions can be enforced without exposing the underlying policy rules to any single entity. The use of FSS guarantees that the function remains secret, preventing unauthorized parties from learning access criteria while still allowing legitimate users to receive correct access decisions.

The proposed access control method is based on PVSS and the distributed point function (DPF) [GI14], which is a special case of FSS where f belongs to the family of point functions. This approach ensures that only authorized users can access the data. We consider a set of access and verification keys to prove the access rights for the function under the verification keys. The verification keys are held by the evaluators, and the dealer holds the access key. The access key is shared between the evaluators by running secret sharing. Then, the user runs PVSS on the shared values and the evaluators audit the verification key and the proofs of the user from PVSS. Our construction extends the PVSS scheme of Gentry et al. [GHL22] for a post-quantum setting, relying on the LWE assumption.

We also utilize the proposed PACL for secure data retrieval based on FSS, leveraging FSS integrated with access control to ensure both security and controlled access to the data. The scheme is described in Fig. 1. In our scheme, the user owns the keyword w_u and n servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$ hold a shared database $\mathsf{DB} = \{(\mathsf{w}_1, \mathsf{data}_1), \ldots, (\mathsf{w}_v, \mathsf{data}_v)\}$ containing the keywords w_j and the related data data_j, $j \in [1, v]$. Each server S_k, $k \in [1, n]$ computes a deterministic finite automata (DFA) based on the keyword w_k. Then, S_k constructs a matrix representing the related DFA, garbles it, and sends it to a proxy server. The servers work together to enforce access control by performing the proposed PACL method. In other words, the user should own the access key that satisfies the access control list (ACL). If the previous step is successful, the proxy server, with its massive computational power and storage capacity, evaluates the garbled DFA matrix using the outputs of the proxy oblivious transfers (OTs) [CKKC13], which are executed based on the input keyword w_u from the user side. Finally, user can retrieve the most relevant data from the servers using the scores obtained from the results.

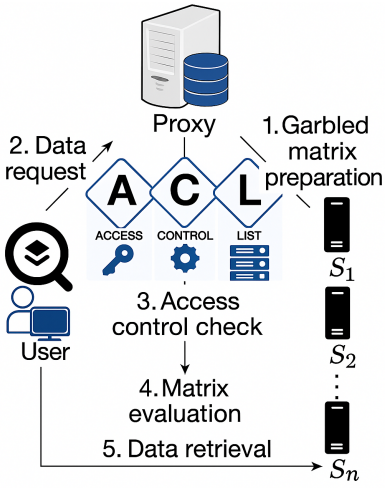


Figure 1: Scheme overview

The rest of the paper is organized as follows. Section 2 provides a description of the related work. Section 3 presents the primitives and the concepts used in our constructions. Next, the proposed PACL scheme is described in Section 4 and its application for data retrieval is detailed. The evaluation results of the proposed PACL scheme are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Access control encryption methods [TZMT17, KW17, WWC21] enhance data security by ensuring that only authorized users can access sensitive information through cryptographic techniques. However, despite their security benefits, efficiency remains a significant challenge, as many of these systems experience high computational overhead and increased latency during data retrieval. Some schemes [ABF+19, BFMP21, SYD21] focus on secret-sharing methods designed for specific access structures. These methods distribute shares of a secret among multiple parties, allowing only authorized groups to reconstruct the original information based on defined access policies. However, a significant challenge is managing the size of the secrets, as larger access structures can lead to inefficiencies in storage and transmission, which must be addressed for practical applications. [VSH22] proposes an access control solution for FSS, enabling users to anonymously write messages into mailboxes using DPFs. To prevent mailbox corruption by malicious users, access control is enforced via a secure computation protocol. However, this approach requires a large output range, causing high computational overhead for evaluators and additional communication with the dealer. Servan et al. [SSBYP23] introduces a method for implementing a private access control list for FSS. Their approach focuses on black-box DPF. That said, the scheme depends on computationally expensive secret-shared proofs, making it inefficient in practical applications. Additionally, the security of the scheme is not guaranteed in a post-quantum environment. Ji et al. [JZR23] proposed more fine-grained policy constraints for DPF. Their scheme allows an attribute-based access control that is secure based on the decisional Bilinear Diffie-Hellman assumption.

Functional encryption (FE) for access control over encrypted data has been explored

in several studies [ACGU20, DDH+24, NPP25]. However, existing FE methods often fall short, either being inefficient and impractical for real-world applications or inadvertently exposing sensitive user-specific information. Multi-party computation-based methods are also employed for access control. For example, schemes like [CDN09, CDNZ11, ACDN13, LLM+17] extend the OT protocol to enable users to access records for which they have the necessary credentials without revealing their identities. Nonetheless, as noted by Damodaran et al. [DR20], many proposed schemes lack support for policy updates. Additionally, in practical scenarios, multiple messages often share the same policy, yet existing schemes do not leverage this overlap to enhance storage efficiency. Furthermore, private set intersection techniques [CCZ24] can prevent the disclosure of sensitive information while facilitating authorized access. Zero-knowledge (ZK) proofs [MDHM18, XXHA18, LH21, LE24] offer additional security benefits, but scalability challenges may hinder the practicality of these approaches. Homomorphic encryption-based approaches, as noted in [LScY09, CHT13, HMSB16, Ver22, SA23], enable systems to store encrypted data in the cloud while allowing computations to be performed on this data without revealing plaintext to the service provider. Users possessing the appropriate decryption keys can locally decrypt the results according to access control policies. However, homomorphic encryption tends to be computationally intensive, especially when dealing with complex access policies and large datasets.

3 Preliminaries

Notation. Throughout the paper, we denote the security parameter by λ. We use bold upper-case letters like $\mathbf{M}$ to denote matrices, and bold lower-case letters like $\mathbf{v}$ to denote vectors. We use $x \xleftarrow{\$} \mathcal{D}$ to denote a random sample from a distribution $\mathcal{D}$. We denote a set of secret shares of x (function secret shares of f respectively) as $([x]_1, \ldots, [x]_n)$ $(([f]_1, \ldots, [f]_n)$ respectively) and $[x]_i$ (resp. $[f]_i$) as the ith secret share.

Secret sharing. Linear secret sharing [Sha79] is parameterized by a threshold $2 \leq t \leq n$ and consists of the algorithm $\mathsf{Share}_{(\mathbb{F},t,n)}$ which generates shares of a secret value in the field $\mathbb{F}$ such that any subset of at least t shares can be combined by the algorithm Recover to reveal the secret value. It should be mentioned that no subset of less than t shares obtains any information about the secret.

Function secret sharing (FSS)**.** FSS [BGI15] provides a way for additively secret-sharing a function f from a given function family F. FSS is a generalization of secret sharing which targets secret sharing a function rather than secret sharing a value. A (t, n)-FSS scheme for a function $f : \{0,1\}^n \to \mathbb{G}$ ($\mathbb{G}$ is an Abelian group) consists of the following algorithms. a) $(k_1, \ldots, k_n) \leftarrow \mathsf{Gen}(1^\lambda, f)$: inputs a security parameter λ and function f and outputs the evaluation keys $(k_1, \ldots, k_n)$. b) $[y]_i \leftarrow \mathsf{Eval}(k_i, x)$: inputs an evaluation key k_i and $x \in \{0,1\}^n$ and outputs secret share $[y]_i$.

Distributed point function (DPF)**.** DPF is a special case of FSS where F is the family of point functions, namely functions $f_{\alpha,\beta}$ that evaluate to β on the input α and to 0 on all other inputs.

Short Integer Solution (SIS). Let $q = q(\lambda)$, $n = n(\lambda)$ and $m = m(\lambda)$. We say that the SIS assumption holds if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ the following holds: $Pr[\mathbf{A} \cdot \mathbf{z} = \mathbf{0} \mid \mathbf{A} \leftarrow \mathcal{R}_q^{n \times m}, \mathbf{z} \leftarrow \mathcal{A}(\mathbf{A})] \leq \mathsf{negl}(\lambda)$.

Learning with error (LWE) problem. For a prime number q, the LWE problem with secret $\mathbf{s} \in \mathbb{Z}_q^k$ is defined as follows: given polynomially random, noisy linear equations in $\mathbf{s}$, find $\mathbf{s}$. More precisely, given $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + e_i)$, where $\mathbf{a}_i \in \mathbb{Z}_q^k$ is uniformly random, and e_i is drawn from a random distribution, the problem is to find $\mathbf{s}$. We denote m samples from the LWE distribution compactly as $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A} \in \mathbb{Z}_q^{m \times k}$ and $\mathbf{e}$ is sampled from

discrete Gaussian noise, or noise sampled uniformly from $[-b, b]$ where $b \ll q$ is a bound on the magnitude.

Proxy OT (POT). The notation of POT is defined by Naor et al. [NPS99] in which there are three parties: a sender that holds inputs $(\mathbf{x}_0, \mathbf{x}_1)$, a chooser that holds input b, and a proxy that learns $\mathbf{x}_b$ and nothing else at the end of the protocol. The details of a non-interactive POT [CKKC13] are described in following. A POT consists of the following algorithms [CKKC13] from the non-interactive key exchange (NIKE) protocol, which allows two parties to generate a shared key based on their respective public keys (without any direct interaction). In our scheme, n 1-out-of-256 OT should be run where n is the number of characters of the user's input. We use the construction of Naor and Pinkas [NP05] to efficiently reduce n invocations of the 1-out-of-256 OT to $8n$ invocations of 1-out-of-2 OT. So, $8n$ invocations of Proxy OT [CKKC13] are required. Each time, the parties need to run the setup algorithm to generate their key pairs. To reduce the key generation process, we designed a protocol for choosing the keys related to the input of the user. We can use batch oblivious pseudorandom function (batch OPRF) [KKRT16] instead of running $8n$ NIKE. In more detail, in the Proxy OT, the sender and the chooser obtain the outputs of the batch OPRF. We can assume that they are run in the offline phase, and the sender and the chooser use the outputs of the offline phase. We can rewrite the scheme [CKKC13] as follows. Assume that $t = 8n$ where n is the length of the user's input.

- *Offline:*
  - The sender and the chooser share the random values $\mathbf{r}_{b,1}, \ldots, \mathbf{r}_{b,t} \xleftarrow{\$} \{0,1\}^*$, $\pi_j \xleftarrow{\$} \{0,1\}$, $b \in [0,1], j \in [1,t]$.
  - $(\mathbf{z}_{b,1}, \ldots, \mathbf{z}_{b,t}) \leftarrow \mathsf{batch\ OPRF}(\mathbf{r}_{b,1}, \ldots, \mathbf{r}_{b,t})$
- *Online*
  - $\alpha_i \leftarrow \mathsf{Snd}(i, \mathbf{x}_{0,i}, \mathbf{x}_{1,i})$. In the i-th POT execution the sender, holding input $\mathbf{x}_{0,i}, \mathbf{x}_{1,i} \in \{0,1\}^\kappa$, runs this algorithm to generate a single encoded message α_i to be sent to the proxy. $\alpha_{\pi_i,i} = \mathbf{z}_{0,i} \oplus \mathbf{x}_{0,i}$, $\alpha_{1\oplus\pi_i,i} = \mathbf{z}_{1,i} \oplus \mathbf{x}_{1,i}$
  - $\beta_i \leftarrow \mathsf{Chs}(i, b_i)$. In the i-th POT protocol, the chooser, holding input $b_i \in \{0,1\}$, runs this algorithm to generate a single encoded message β_i to be sent to the proxy. $\beta_i \leftarrow (b_i \oplus \pi_i, \mathbf{z}_{b_i,i})$
  - $\mathbf{y}_i \coloneqq \mathsf{Prx}(i, \alpha_i, \beta_i)$. In the i-th POT protocol, the proxy runs this algorithm using the sender message α_i and the chooser message β_i, and computes the value $\mathbf{y}_i = \mathbf{x}_{b_i}$. Parse α_i as $(\alpha_{0,i}, \alpha_{1,i})$ and β_i as $(b'_i, \mathbf{z}'_i)$, $\mathbf{y}_i = \alpha_{b'_i,i} \oplus \mathbf{z}_{b'_i}$.

Deterministic finite automaton (DFA)**.** A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string. A DFA, denoted by the five-tuple $\Gamma = (\Sigma, Q, \sigma, q_0, F_q)$, consists of 1) Σ, a finite set of input symbols, 2) Q, a finite set of states, 3) σ, a transition function that takes as arguments a state and an input symbol and returns a state, 4) $q_0 \in Q$, a start state, 5) $F_q \subset Q$, a set of final or accepting states [HMU01].

3.1 Definitions

Definition 1. Let $\lambda \in \mathbb{N}$ represent a security parameter, and let $F : \{0,1\}^n \to \{0,1\}^*$ denote a family of functions, with $f_i \in F$. An access control list (ACL) scheme is defined by an access control list Λ_λ, which is parameterized by λ, that contains verification keys, along with an efficiently computable predicate CheckAccess $(\Lambda_\lambda, f_i, \mathsf{ak})$. This predicate outputs yes if and only if the access key ak satisfies a relation R in relation to the verification key corresponding to f_i in Λ_λ. For simplicity, we define $N := |F|$ and omit the subscript λ when it is evident from the context.

Definition 2. A private access control list (PACL) [SSBYP23] applied to FSS is initiated between a prover and a set of s verifiers. The prover holds an access key ak and the function f. The verifiers hold secret-shares $[f]_i$ and have the access control Λ for the function f. The verifiers check whether or not CheckAccess outputs yes, without learning f.

Definition 3. Let $f : \{0,1\}^n \to \{0,1\}^*$ be a function. Set $2 \leq t \leq s$ and let $(\mathsf{Gen}, \mathsf{Eval})$ instantiate a (t,s)-FSS scheme for f. A (t,s)-PACL scheme consists of the algorithms $\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Audit}$ and Verify as follows.

- $(\mathsf{ak}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$. Inputs a security parameter λ and a function f. The output is a pair of access and verification keys $(\mathsf{ak}, \mathsf{vk})$.
- $(\pi_1, \ldots, \pi_s) \leftarrow \mathsf{Prove}(f, \mathsf{ak})$. Inputs a function f and an access key ak, and outputs proofs of s shares.
- $\tau_i \leftarrow \mathsf{Audit}(\Lambda, [f]_i, \pi_i)$. Inputs access control $\Lambda = \mathsf{vk}$, function secret share $[f]_i$ of f and a proof of share π_i. The audit token τ_i is the output.
- $0/1 \leftarrow \mathsf{Verify}(\mathcal{T} = \{\tau_i \mid i \in I\})$. Inputs a set of at least t audit tokens indexed by the set $I \subseteq \{1, \ldots, s\}$ and outputs 0 or 1.

  The above functionality should satisfy these requirements [SSBYP23]:
  - **Completeness.** A (t,s)-PACL scheme is complete if for all subsets $I \subseteq \{1, \ldots, s\}$ with $|I| \geq t$ and $\Lambda = (\mathsf{vk})$ where vk is sampled according to KeyGen algorithm and for all secret shares $([f]_1, \ldots, [f]_s)$ of f sampled according to $\mathsf{Setup}(1^\lambda, f)$ algorithm, it holds that
  $$Pr\begin{bmatrix} (\pi_1, \ldots, \pi_s) \leftarrow \mathsf{Prove}(f, \mathsf{ak}); \\ \{\tau_i \leftarrow \mathsf{Audit}(\Lambda, [f]_i, \pi_i) \ | i \in I \} : \\ 0/1 \leftarrow \mathsf{Verify}(\mathcal{T} = \{\tau_i \mid i \in I\} = \\ \mathsf{CheckAccess}(\Lambda, f, \mathsf{ak})) \end{bmatrix} = 1$$
  - **Privacy.** For all subsets $I \subset \{1, \ldots, s\}$, $|I| < t$, define $J = \{1, \ldots, s\} \backslash I$ and $D_{I,J}$ to be the distribution over $\{(\pi_i, \tau_i^*) \mid i \in I\} \cup \{\tau_j \mid j \in J\}$ where π_i is sampled according to $\mathsf{Prove}(f, \mathsf{ak})$, τ_i^* is sampled arbitrary, and $\tau_j \leftarrow \mathsf{Audit}(\Lambda, [f]_j, \pi_j)$ for all $j \in J$. A (t,s)-PACL is private if there exists an efficient simulator Sim such that $D_{I,J} \approx_c \mathsf{Sim}(1^\lambda, I, \{\tau_i^* \mid i \in I\})$.
  - **Soundness.** There exists a negligible function negl such that for all efficient algorithms $\mathcal{A}$ and subsets $I \subseteq \{1, \ldots, s\}$ where $|I| \geq t$, $\mathsf{Pr}[\mathsf{soundness}_{(\mathsf{PACL},\mathcal{A},\mathsf{I})}(\lambda)] = \mathsf{yes} \leq \mathsf{negl}(\lambda)$, where $\mathsf{soundness}_{(\mathsf{PACL},\mathcal{A},\mathsf{I})}(\lambda)$ is defined in Fig. 2. In more detail, no efficient adversary $\mathcal{A}$ without knowledge of an access key for f_β, can forge a proof π that verifies with non-negligible probability.

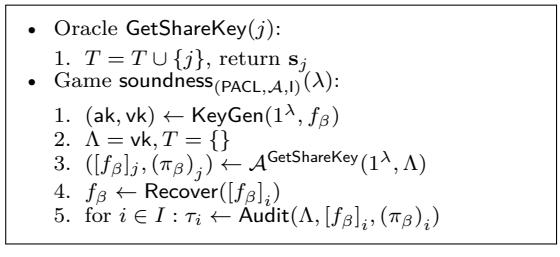


Figure 2: PACL soundness game [SSBYP23]

Definition 4. A publicly verifiable secret sharing scheme (PVSS) enables a dealer to share a secret among a committee of shareholders in a manner that allows everyone, not just the shareholders, to verify that the secret was shared correctly and assures that it is recoverable [Sta96]. A PVSS scheme consists of the following algorithms [CD24].

- Setup

  - $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{ip})$ outputs public parameters pp. The initial parameters ip include information about the number of parties, privacy and reconstruction thresholds and spaces of secrets and shares. The public parameters pp contain a description of spaces of private and public keys SK and PK and the relation $\mathsf{R}_{Key} \subseteq \mathsf{SK} \times \mathsf{PK}$ describing valid key pairs.
  - $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, generates a pair of private and public key, where $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{R}_{Key}$.
- Distribution
  - $((\mathsf{C}_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}) \leftarrow \mathsf{Dist}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, \mathbf{s})$ where $\mathbf{s} \in \mathcal{S}$ is a secret, outputs encrypted shares and a proof $\mathsf{Pf}_{\mathsf{Sh}}$ of sharing correctness.
- Distribution Verification
  - $0/1 \leftarrow \mathsf{VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i, \mathsf{C}_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}})$ determines whether the sharing is valid or not.
- Reconstruction
  - $(\mathsf{d}_i, \mathsf{Pf}_{\mathsf{Dec}_i}) \leftarrow \mathsf{DecShare}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{C}_i)$ outputs a decrypted share d_i and a proof $\mathsf{Pf}_{\mathsf{Dec}_i}$ of correct decryption.
  - $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{d}_i : i \in \mathcal{T}\})$ for some $\mathcal{T} \subseteq [n]$ outputs an element of the secret space $\mathbf{s}' \in \mathcal{S}$ or an error symbol $\perp$.
- Reconstruction Verification
  - $0/1 \leftarrow \mathsf{VerifyDec}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{C}_i, \mathsf{d}_i, \mathsf{Pf}_{\mathsf{Dec}_i})$ determines whether d_i is a valid decryption of C_i.
  - $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{d}_i : i \in \mathcal{T}\})$ for some $\mathcal{T} \subseteq [n]\}$ outputs an element of the secret space $s' \in \mathcal{S}$ or an error symbol $\perp$.

Security properties of a PVSS [CD24]:

Correctness with t-reconstruction. This requirement guarantees that if all participants act honestly, all proofs will be validated, and any group of at least t participants can reconstruct the secret using their shares. Each participant must first decrypt their share, after which they collectively apply the reconstruction algorithm Rec.

Verifiability. This property ensures that if the verification procedures VerifySharing and VerifyDec are passed, then the key pairs are properly constructed, the set of encrypted shares represents a correct sharing of the secret, and the shares have been accurately decrypted.

4 Post-Quantum Access Control scheme

In this section, we propose a post-quantum PACL based on the SIS assumption for FSS. Our model and definitions for post-quantum PACL for FSS are derived from [SSBYP23]. This scheme is a requirement for several existing applications of access control for FSS [SSBYP23, ECZB21, NSSD22] and demands a stringent set of efficiency requirements that closely align with the goals of FSS. To achieve this goal, we initially built a non-interactive PVSS scheme from the LWE assumption. Subsequently, using PVSS, we construct our post-quantum PACL scheme. The DPF scheme used in our work is adopted from existing literature, specifically from [SSBYP23], which introduces a method for implementing a PACL for FSS with a focus on black-box DPF. However, their approach relies on computationally intensive secret-shared proofs, leading to inefficiencies in practical applications. Additionally, the scheme does not provide security guarantees in a post-quantum setting. Our key contribution is the integration of this DPF-based scheme within a post-quantum PACL framework, utilizing the SIS and LWE assumptions to enhance security and ensure resilience against quantum adversaries.

In Section 4.1, we first revisit the PVSS construction by Gentry *et al.* [GHL22], then describe a non-interactive post-quantum PVSS, constructed upon Gentry et al.'s work,

leveraging the LWE assumption. Then, we introduce our post-quantum PACL in Section 4.2. In the end, Section 4.3 describes the application of the proposed PACL for data retrieval.

4.1 Post-Quantum Non-Interactive PVSS

To build a post-quantum non-interactive PVSS, we initially consider the PVSS scheme by Gentry et al. [GHL22], which relies on both the discrete logarithm (DL) and LWE assumptions. Our work enhances this scheme by eliminating its dependence on the DL assumption and instead leveraging the LWE assumption. This modification ensures post-quantum security, making the scheme resilient against quantum attacks. We then transition it into a construction solely based on the LWE assumption.

Gentry et al. [GHL22] presented a non-interactive PVSS scheme, which we refer to as GHL-PVSS, that utilizes the Peikert-Vaikuntanathan-Waters (PVW) encryption scheme in a multi-receiver setting. We will now recall a brief description of the PVW encryption scheme as outlined in [PVW08]. Let $\mathbf{A}$ be a public random matrix. Each party i sets $\mathbf{B}_i = \mathbf{S}_i \cdot \mathbf{A} + \mathbf{e}_i$ as its public key, where the matrix $\mathbf{S}_i$ is secret. The parties' public keys are collected into a matrix $\mathbf{B}$. The public key of the PVSS system is $\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}$. The encryption of message $\mathbf{m} \in \mathbb{Z}_q^k$ is $\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \cdot \mathbf{r} + \begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{m}\end{bmatrix} = \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}$ where $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ are small vectors. A member from the committee will apply PVSS scheme to encrypt k re-shares of this share to the next committee with k members, $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{Z}_q^k, \mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{k\times m}, \mathbf{r} \in \mathbb{Z}_q^m$. GHL-PVSS uses bulletproofs [BBB+18] to get compact proofs of correct encryption/decryption of shares. In particular, bulletproofs enable efficient exact range proofs, allowing one to show that the vectors used within the key generation and encryption process are indeed short. Gentry *et al.* [GHL22] identified the construction of a post-quantum PVSS scheme as an open problem, encouraging further exploration of possible alternatives to their bulletproofs variant, specifically those based on lattice assumptions [CLM23]. Our contribution to PVSS is the integration of a protocol from the STARK [BBHR19] family into PVSS framework [GHL22], which results in improved proof sizes. The details are provided in the following.

A Non-Interactive PVSS from LWE. We adopt the concept of GHL-PVSS [GHL22] and incorporate a post-quantum proof system to verify the correctness of the encryption and decryption of the associated shares in the PVSS scheme. Specifically, we employ a ZK proof system from the STARK [BBHR19] family of protocols, known as the RISC Zero protocol [BG23] (denoted by RISC0), which enables a shareholder to prove that incoming and outgoing PVW ciphertexts accurately encrypt re-shares associated with their share. In other words, we adopt RISC0 since it offers better proof sizes and tooling while maintaining post-quantum security under the LWE assumption.

For the sake of readability, we present our proposed post-quantum non-interactive PVSS, so-called GHL$^+$ scheme as described in Fig. 3. As we noted before, the bulletproof proof system, as used in [GHL22], relies on the hardness of the DL assumptions, and is thus not secure against a quantum adversary. While there exist lattice variants of bulletproofs [BLNS20], the consensus is that they do not lead to concretely efficient proof systems. Another line of work that achieves post-quantum security is that initiated by [BCS16], which is secure in the quantum random oracle model [CMS19]. Our proof system of choice was the RISC Zero proof system, which is based on the STARK [BBHR19] family of protocols. We did consider recent linear-time proof systems such as Brakedown [GLS+23] and Orion [XZS22], but ultimately decided for a proof system with better proof sizes and whose tooling would facilitate our implementation.

- $\mathsf{GHL}^+.\mathsf{Setup}(1^\lambda, \mathsf{ip})$:
  - $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{ip})$ outputs public parameters pp. Specify a set of pairwise distinct points $\{\alpha_1, \ldots, \alpha_m, \gamma_1, \ldots, \gamma_n\} \subset \mathbb{Z}_q$. Let $\mathsf{pp} = (q, k, t, n, (\alpha_j)_{j\in[m]}, , (\gamma_i)_{i\in[n]})$. The initial parameters ip contain information about the number of parties, privacy and reconstruction thresholds and spaces of secrets and shares.
  - $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{PVW.KeyGen}(\mathsf{pp}, \mathsf{id})$, where sk and pk are the secret and public key of PVW encryption scheme.
  - $\mathsf{pp}_{\mathsf{RISC0}} \leftarrow \mathsf{RISC0.KeyGen}(\mathsf{pp}, \mathsf{id})$, where $\mathsf{pp}_{\mathsf{RISC0}}$ is the public parameter of the RISC zero protocol.
- $\mathsf{GHL}^+.\mathsf{Dist}(\mathsf{sk}_i, \mathbf{s})$:
  - $([\mathbf{s}]_1, \cdots, [\mathbf{s}]_n) \leftarrow \mathsf{Secret.Share}(\mathbf{s})$ where $\mathbf{s} \in \mathcal{S}$ is a secret.
  - $(\mathsf{C}_i)_{i\in[n]} \leftarrow \mathsf{PVW.Enc}(\mathsf{pk}_i, [\mathbf{s}]_i)$ where C_i is a PVW ciphertext of the shares $[\mathbf{s}]_i$.
  - $(\pi_{\mathsf{i,RISC0}})_{i\in[n]} \leftarrow \mathsf{RISC0.Prover}(\mathsf{pp}_{\mathsf{RISC0}}, [\mathbf{s}]_i, \mathbf{s}, \mathsf{C}_i)$ where $\pi_{\mathsf{i,RISC0}}$ is a RISC zero protocol proof of sharing correctness.
- $\mathsf{GHL}^+.\mathsf{VerifySharing}(\mathsf{pp}, \mathsf{pp}_{\mathsf{RISC0}}, (\mathsf{pk}_i, \mathsf{C}_i, \pi_{\mathsf{i,RISC0}})_{i\in[n]})$:
  - $0/1 \leftarrow \mathsf{RISC0.Verify}(\mathsf{pp}, \mathsf{pp}_{\mathsf{RISC0}}, (\mathsf{pk}_i, \mathsf{C}_i, \pi_{\mathsf{i,RISC0}})_{i\in[n]})$ (checks C_i are well-formed and indeed encrypt the shares)
- $\mathsf{GHL}^+.\mathsf{DecShare}(\mathsf{pp}, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{C}_i)$:
  - $\mathsf{d}_i \leftarrow \mathsf{PVW.Dec}(\mathsf{sk}_i, \mathsf{C}_i)$ outputs a decrypted share.
  - $\pi_{\mathsf{Dec_i,RISC0}} \leftarrow \mathsf{RISC0.Prover}(\mathsf{pp}_{\mathsf{RISC0}}, \mathsf{d}_i, \mathsf{C}_i)$: outputs a proof of correct decryption.
  - $\mathsf{Rec}(\mathsf{pp}, \{\mathsf{d}_i : i \in \mathcal{T}\})$: for some $\mathcal{T} \subseteq [n]$ outputs an element of the secret space $\mathbf{s}' \in \mathcal{S}$ or an error $\perp$.
- $\mathsf{GHL}^+.\mathsf{VerifyDec}(\mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{C}_i, \mathsf{d}_i, \pi_{\mathsf{Dec_i,RISC0}})$:
  - $0/1 \leftarrow \mathsf{RISC0.Verfiy}(\mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{C}_i, \mathsf{d}_i, \pi_{\mathsf{Dec_i,RISC0}})$ as a verdict on whether d_i a valid decryption of C_i.

Figure 3: LWE-based GHL^+ scheme

Theorem 1. *Let* Π_{RISC0} *be the* RISC *Zero protocol. Let* PVW *be an encryption scheme based on LWE assumption. Then, the proposed* GHL^+ *scheme is a non-interactive* PVSS.

Proof. We show that the GHL^+ scheme guarantees the security properties from Definition 4. Let λ be the security parameter and $q > 2^\lambda$ prime. Let m (size of the secret), t (privacy threshold) and n (number of parties). We prove that GHL^+ scheme is a correct PVSS with $t + m$-reconstruction. If all parties in $[n]$ honestly create keys, then for all i we have pk_i for some sk_i. If the dealer is honest the values C_i are computed correctly where $p(X)$ is of degree $t + m - 1$ and $p(\alpha_j) = s_j$ are the coordinates of the secret, a set of pairwise distinct points $\{\alpha_1, \ldots, \alpha_m\}$. Then clearly DecShare, when honestly applied to C_i, computes d_i. Therefore given a set $\mathcal{T}$ of parties of size at least $t + m$ who correctly decrypted their shares, and a subset $\mathcal{T}'$ of exactly $t + m$ parties, we have that the reconstructed values are $\Pi_{i\in\mathcal{T}'} p(\gamma_i)L_i(\alpha_j) = p(\alpha_j) = s_j$, for each $j \in [m]$, by Lagrange interpolation, where L_i is a polynomial degree at most $n - 1$. On the other hand, if RISC0 is a proof with soundness error negligible in λ then GHL^+ has verifiability of sharing distribution. Also, if RISC0 with soundness error negligible in λ then GHL^+ scheme has verifiability of share decryption. The proof is trivial as the statements proved by the ZK proofs exactly guarantee correct sharing distribution and share decryption, respectively. □

4.2 The Proposed Post-Quantum PACL

Now, we introduce our construction of post-quantum PACL tailored for DPFs. Our DPF-PACL construction for the matching predicate extends the approach employed by Newman *et al.* [NSSD22] and Schreiber *et al.* [SSBYP23], which is based on the LWE and SIS assumptions, both presumed to be post-quantum secure.

Construction. In Fig. 4, we present the proposed construction for a DPF-PACL with CheckAccess instantiated for the match predicate described in Section 3. Our construction relies mainly on the following primitives: (i) a post-quantum non-interactive verifiable secret sharing (GHL^+ scheme in Fig. 3) and (ii) a DPFs scheme (e.g., DPFs by Gilboa-Ishai [GI14]). Roughly speaking, the construction leverages the DPFs to locally select shares of the i-th verification key in Λ. Two key facts enable this process: (1) all verifiers possess $\Lambda = \mathbf{As}$, and (2) the FSS key k_i, encoding a DPF. The verifiers first evaluate the $[y_x]_i \leftarrow \mathsf{DPF.Eval}(k_{x,i})$, $x \in [1, z], i \in [1, n]$, and then compute the product as $\mathbf{B}_i \leftarrow \Sigma_{x=1}^{z}(\mathbf{As}_x) \cdot [y_x]_i$ (Audit phase). To verify knowledge of $\mathbf{s}$, utilizing the GHL^+ scheme, the prover distributes additive secret shares of $\mathsf{ak} = -\mathbf{s}$ to the verifiers (described in the Prove phase). In our scheme, the secret value $\mathbf{s}$ is determined based on the number of files to which the prover has access. In more detail, if we consider z files and the prover has access only to files f_2, f_7, f_{12}, he/she gets the sum of $\mathbf{s}_2$, $\mathbf{s}_7$ and $\mathbf{s}_{12}$, i.e., $\mathsf{ak} = -(\mathbf{s}_2 + \mathbf{s}_7 + \mathbf{s}_{12})$ but the verification key is vk. Each verifier computes $\tau_i := \mathbf{b}_i + \mathbf{A}[\mathbf{s}]_i$ using the Audit algorithm and reveals τ_i in a verifiable manner (using the GHL^+.DecShare algorithm of the PVSS) to all other verifiers. All verifiers proceed to check that $|\Sigma_i \tau_i| \overset{?}{\approx} \mathbf{0}$ (described in the Verify phase).

Lemma 1. *If adversary $\mathcal{A}$ can win the* PACL *soundness game described in Fig. 2 for our post-quantum* PACL *scheme with non-negligible probability $\gamma(\lambda)$ for a function f_β^* which is not sampled from $\mathcal{F}_{\mathsf{DPF}}$ and proof π^*, then there exists an adversary $\mathcal{A}'$ that wins the* PACL *soundness game with probability $\gamma(\lambda)$, such that $\mathcal{A}'$ outputs $f_\beta \in \mathcal{F}_{\mathsf{DPF}}$ and π.*

Proof. A property of our DPF-PACL scheme is the ability to aggregate proofs across different DPFs and access control lists. In more detail, for any integer k that is polynomial in the security parameter λ and family of point functions F, it holds the following property. Let Λ be an ACL (see Definition 1) for the family F and let $f_1, \ldots, f_k \in F$ have associated access keys $\mathsf{ak}_1, \ldots, \mathsf{ak}_k \in \Lambda$, then $\mathbf{s}' := \sum_{i=1}^{k} \mathsf{ak}_i$ is an access key for $f'(x) := \sum_{i=1}^{k} f_i(x)$. Consider an efficient $\mathcal{A}$ that outputs a point function f_β^* and π^*. So, it holds that $f_\beta^* = \sum_{j \in S} c_j f_j$ which $Y \subseteq \{0, 1, \ldots, z\}$, each f_j is a point function, and c_j's are arbitrary non-zero scales in $\mathbb{Z}_p$. We construct an adversary $\mathcal{A}'$ that breaks the PACL soundness game with a point function f_β as follows. First, run $\mathcal{A}$ to get function f_β^*. Then compute $f_\beta := f_\beta^* - \sum_{j \in Y, j \neq \gamma} c_j \alpha_j$, note that $\mathcal{A}'$ is allowed to query for all α_j provided $j \neq \gamma$. Finally, output f_β and π. It must hold that $\gamma \in Y$, otherwise $\mathcal{A}$ does not succeed since it queried all the necessary access keys ak_j for $j \in Y$. Based on the aggregation property of our construction, it follows that f_β is a point function and π is a valid access proof for f_β. Hence, $\mathcal{A}'$ succeeds with the same probability. □

Theorem 2. *Assume GHL^+ scheme is a non-interactive publicly verifiable secret sharing from LWE assumption, see Fig. 3. Let* DPF *be a distributed point function, let $\mathbb{Z}_q$ denote the ring of integers modulo q in which the SIS problem is assumed to be computationally hard. Then, the proposed* PACL *in Fig. 4 satisfies the completeness, efficiency, soundness, and privacy properties of Definition 3 as defined in Section 3.*

Proof. • **Completeness**: Suppose that we have z files. If the requirements are satisfied PACL can be passed as follows. In Audit step, $\mathbf{B}_i$ is computed as $\mathbf{B}_i \leftarrow \Sigma_{x=1}^{z}(\mathbf{As}_x) \cdot [y_x]_i = \mathbf{A}\Sigma_{x=1}^{z}\mathbf{s}_x \cdot [y_x]_i$. Also, the verifiers compute $\tau_i = \mathbf{B}_i + \mathbf{A}[s_j]_i$. Verify phase will be done by computing $\Sigma\tau_i = \Sigma_{i=1}\mathbf{A}\Sigma_{x=1}^{z}\mathbf{s}_x \cdot [y_x]_i + \Sigma_{i=1}\mathbf{A}[s_j]_i = \mathbf{A}\Sigma_{i=1}\Sigma_{x=1}^{z}\mathbf{s}_x \cdot [y_x]_i + \Sigma_{i=1}\mathbf{A}[s_j]_i = \mathbf{A}\Sigma_{x=1}^{z}\mathbf{s}_x\Sigma_{i=1}[y_x]_i + \Sigma_{i=1}\mathbf{A}[s_j] = \mathbf{A}\Sigma_{x=1}^{z}\mathbf{s}_x + \Sigma_{i=1}\mathbf{A}[s_j]_i = \mathbf{As} - \mathbf{As}$.

- Setup:
  - $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pp})_{i\in[n]} \leftarrow \mathsf{GHL}^+.\mathsf{Setup}(1^\lambda, \mathsf{ip})$
  - Return $(k_{x,1} \ldots, k_{x,n}, (\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pp})_{i\in[n]})$
- $\mathsf{KeyGen}(1^\lambda, f_x) : x \in [1, z]$
  - $\mathbf{s}_x \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{A} \in \mathbb{Z}_q^{k\times m}$
  - $\mathsf{ak} = -(\mathbf{s}_1, \ldots, \mathbf{s}_z)$, access to z files $f_1, \ldots, f_z$, $\mathsf{vk}_x = \mathbf{A}\mathbf{s}_x$
  - Return $(\mathsf{ak}, \mathsf{vk}_1, \ldots, \mathsf{vk}_z)$
- $\mathsf{Prove}(f, \mathsf{ak})$:
  - $([s_j]_1, \ldots, [s_j]_n) \leftarrow \mathsf{Share}_{(\mathbb{F}_p,t,n)}(-s_j)$, $j \in [1, m]$, $\mathbf{s} = \Sigma_{x\in[z]}\mathbf{s}_x = (s_1, \ldots, s_m)$
  - $(\mathsf{C}_i^j, \pi^j_{\mathsf{i},\mathsf{RISC0}})_{i\in[n]} \leftarrow \mathsf{GHL}^+.\mathsf{Dist}(\mathsf{pp}, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{ak}, \mathsf{pk}_i, [s_j]_i)$
  - Return $(\mathsf{C}_i^j, \pi^j_{\mathsf{i},\mathsf{RISC0}})_{i\in[n]}$
- $\mathsf{Audit}(\Lambda, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pp}, ([f_x]_i, \mathsf{sk}_i, \mathsf{pk}_i, \mathsf{C}_i^j, \pi^j_{\mathsf{i},\mathsf{RISC0}})_{i\in[n]})$:
  - $0/1 \leftarrow \mathsf{GHL}^+.\mathsf{VerifySharing}(\mathsf{pp}, \mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{C}_i^j, \pi^j_{\mathsf{i},\mathsf{RISC0}})$
  - parse $\Lambda = (\mathbf{A}\mathbf{s}_1, \ldots, \mathbf{A}\mathbf{s}_z)$ and $[f_x]_i = k_{x,i}, x \in [1, z], i \in [1, n]$
  - $[y_x]_i \leftarrow \mathsf{DPF.Eval}(k_{x,i})$, $x \in [1, z], i \in [1, n]$
  - $\mathbf{B}_i \leftarrow \Sigma_{x=1}^z(\mathbf{A}\mathbf{s}_x) \cdot [y_x]_i$
  - $([s_j]_i, \pi^j_{\mathsf{Dec_i},\mathsf{RISC0}}) \leftarrow \mathsf{GHL}^+.\mathsf{DecShare}(\mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{C}_i^j)$.
  - $\tau_i = \mathbf{B}_i + \mathbf{A}[s_j]_i$
  - Return $(\tau_i, \mathsf{d}_i^j := [s_j]_i, \pi^j_{\mathsf{Dec_i},\mathsf{RISC0}})$
- $\mathsf{Verify}(\mathcal{T}, \mathsf{pp}_{\mathsf{RISC0}}, (\mathsf{pk}_i, \mathsf{C}_i^j, \mathsf{d}_i^j, \pi^j_{\mathsf{Dec_i},\mathsf{RISC0}})_{i\in[n]})$:
  - $0/1 \leftarrow \mathsf{GHL}^+.\mathsf{VerifyDec}(\mathsf{pp}_{\mathsf{RISC0}}, \mathsf{pk}_i, \mathsf{C}_i^j, \mathsf{d}_i^j, \pi^j_{\mathsf{Dec_i},\mathsf{RISC0}})$
  - parse $\mathcal{T} := \{\tau_1, \ldots, \tau_t\}$
  - Check whether $\Sigma_i\tau_i \stackrel{?}{\approx} 0$, and return 1 if successful.
  - Return 1 if the both checking output 1. Otherwise 0.

Figure 4: Post-Quantum DPF-PACL for Match Predicates.

- **Privacy**: The simulator Sim on input $(1^\lambda, I, \{\tau_i^* | i \in I\})$ for any subset of $t-1$ verifiers acts as follows. Sim sets $J = \{1, \ldots, s\}\backslash I$. Next, Sim shares the vector $\mathbf{0}$,$([\mathbf{0}]_1, \ldots, [\mathbf{0}]_s) \leftarrow \mathsf{Share}_{\mathbb{F}_p,t,s}(\mathbf{0})$, and computes the proofs of the shares by $(\mathsf{C}_i, \mathsf{Pf}_{\mathsf{Sh}_i})_{i\in[n]} \leftarrow \mathsf{GHL}^+.\mathsf{Dist}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, ([\mathbf{0}]_i)_{i\in[n]})$. Then, Sim sets $\tau_k = \mathbf{A}[0]_k$ for all $k \in I \cup J$ and outputs $\{(\mathsf{Pf}_{\mathsf{Sh}_i}, \tau_i) \mid i \in I\} \cup \{\tau_j \mid j \in J\}$. In the real world, the audit tokens are computed using the output of the DPF and they are computationally-hiding additive secret shares of $\mathbf{A}[\mathbf{0}]$ while in the ideal execution Sim outputs information-theoretically hiding additive shares. So, the difference between the real and ideal executions is this point. If an efficient distinguisher can distinguish the two views, that means that the FSS scheme is not secure which is a contradiction.

- **Soundness**: Suppose there exists an efficient prover $\mathcal{A}$ and a non-negligible function γ so that for all $I \subseteq \{1, \ldots, s\}$, $|I| \geq t$, $\Pr\left[\mathsf{soundness}_{(\mathsf{PACL},\mathcal{A},\mathsf{I})}(\lambda)\right] = \mathsf{yes} \geq \gamma(\lambda)$. Based on Lemma 1, suppose that f_β is output by $\mathcal{A}$ in Fig. 2. Then adversary $\mathcal{A}'$ proceeds as follows. On input $\mathbf{y} = \mathbf{A}\mathbf{s}$, selects $\beta' \xleftarrow{\$} \{1, \ldots, n\}$ and $(\alpha_1, \ldots, \alpha_n) \xleftarrow{\$} (\mathbb{Z}_q^m)^n$ as shares and computes $\mathbf{A}\alpha_1, \ldots, \mathbf{A}\alpha_n$ but replace $\mathbf{A}\alpha_{\beta'}$ with $\mathbf{y}$. In the soundness game, the adversary $\mathcal{A}'$ runs $\mathcal{A}^{\mathsf{GetShareKey}}(1^\lambda, \Lambda)$ and responses to each query with α_j such that

if $j \neq \beta'$ aborts, and adds j to T. If $\beta = \beta'$ outputs $\mathbf{s}$ and sets $\mathsf{ak} = -\mathbf{s}$, otherwise outputs fail. The probability that $\beta = \beta'$ is $\frac{1}{n}$ and so $\mathcal{A}'$ succeeds with probability at least $\frac{1}{n} \cdot \gamma(\lambda)$ which is non-negligible. Hence $\mathcal{A}'$ successfully can solve the SIS in $\mathbb{Z}_q$ which is a contradiction.

□

4.3 Secure Data Retrieval with Access Control

As an independent contribution, we employ our proposed DPF-PACL approach to achieve secure data retrieval, using DPF integrated with access control to ensure both security and controlled access to the data. The proposed scheme is denoted by $\pi_{\mathsf{DR-ACL}}$ and described in Fig. 5, while Fig. 1 provides a high-level overview of the system architecture and the interaction among the entities. Suppose that n servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$ own a shared data base $\mathsf{DB} = \{(\mathsf{w}_1, \mathsf{data}_1), \ldots, (\mathsf{w}_v, \mathsf{data}_v)\}$, where w_j represents the keyword and data_j, $j \in [1, v]$ denotes the related data. The user owns the keyword w_u and would like to retrieve the most relevant data based on their keyword. For this, the steps are as follows. **Step 1: garbling the DFA matrix.** the server S_k runs the algorithm described in Fig. 6 to garble a matrix representing the DFA for the keywords w_j, $j \in [1, v]$ and sends the resultant matrix to the proxy. The steps of Fig. 6 are described below.

- **Step 1: garbling the DFA matrix.** The server S_k, $k \in [1, n]$, constructs a garbled DFA matrix on the keyword w_k by running the algorithm described in Fig. 6, and sends the resulting matrix to the proxy server.
- **Step 2: access control check.** The servers collectively enforce access control by running the proposed DPF-PACL described in Fig. 4. If this step is successful, the next step will be executed; otherwise, halt.
- **Step 3: evaluating the matrix.** The proxy evaluates the garbled matrix on the input of the user and sends the result to user; the protocol is described in Fig. 7.
- **Step 4: retrieve the data.** Next, the user can use the scores obtained from the result to retrieve the most relevant data from the servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$; the algorithm is described in Fig. 8.

Figure 5: Data Retrieval with Access Control, $\pi_{\mathsf{DR-ACL}}$

- *Constructing the matrix.* Inspired by [MNSS12, NSMS14], the DFA will be expressed as a matrix, after which each server S_k, $k \in [1, n]$, will permute and encrypt it before sending the garbled version to the proxy server. Each server S_k computes a DFA from the keyword w_k, $k \in [1, n]$. Then, the DFA should be represented by a matrix $\mathbf{M}$. To achieve this, we apply the proposed method in [NSMS14]. In more detail, for a state q_i, we use the notion of a character group [NSMS14] to denote the set of characters in Σ that for them there exists a transition from q_i to the same destination state q_j. In other words, c and d are in the same character group if $\sigma(q_i, c) = \sigma(q_i, d) = q_j$. The set of character groups of a state q_i is denoted by $\mathbf{ch}_i$ and ch_{max} is the number of unique character groups.

- *Generating random pads.* The server $\mathsf{S}_k, k \in [1, n]$ generates random pads $\mathbf{r}_{\mathbf{0}}^{\mathbf{j}}$, $j \in [1, |\mathbf{ch_0}|]$ for the start state q_0. For other state q_j, inp random pads are generated, denoted by $\mathbf{pad}_i^v$, $v = 1, \ldots, |\mathbf{ps}_i|$. $\mathbf{ps}_i$ denotes the states for which a transition exists from them to the state q_i. Suppose that inp is the maximum number of previous states for any state.

- *Generating random keys.* The parties agree on a Proxy OT denoted by $\mathsf{POT}(\mathsf{K}'_1, \ldots, \mathsf{K}'_{|\Sigma|})$ where $\mathsf{K}'_1, \ldots, \mathsf{K}'_{|\Sigma|}$ are the keys which are constructed by each server, Proxy OT is

This algorithm is run by the server S_k

- Input: keyword w_k
- Output: the garbled matrix $\mathbf{GM}^k$, the vectors $\mathbf{gm}_{0,k}$, $\mathbf{a}_0^k$ and the vectors $\{\mathbf{ga}_i^k\}_{i\in[|Q|]}$
- *Constructing the matrix*, $i \in [0, |Q|-1]$
  - $\mathbf{m_i} \leftarrow \sigma(i, \mathbf{ch}_i)$, $\mathbf{m_i} = (m_i(1), \ldots, m_i(|\mathbf{ch}_i|))$
  - $m_i(0) \leftarrow 1$ if $i \in F$, else $m_i(0) \leftarrow 0$
  - $\mathbf{m}_{|\mathbf{Q}|} \leftarrow \mathbf{m_0}$
  - $m_i(j) \leftarrow |Q|$ if $m_i(j) = 0$, $j \in [1, |\mathbf{ch}_i|]$, $i \geq 1$
- *Random pads*, $i \in [0, |Q|]$, $j \in [1, |\mathbf{ch}_i|]$
  - *generating random strings*
    - $s \leftarrow m_i(j)$
    - if $s \neq 0$: $s' \leftarrow \mathsf{PER}(s)$, $\mathbf{r}^{\mathbf{i}-\mathbf{s}'} \xleftarrow{\$} \{0,1\}^\kappa$
    - if $s = 0$: $\mathbf{r}^{\mathbf{i}-\mathbf{0}} \xleftarrow{\$} \{0,1\}^\kappa$
  - *computing previous random pads*, $i \in [1, |Q|]$, $\mathbf{ps_i} = (ps_i(1), \ldots, ps_i(|\mathbf{ps_i}|))$
    - if $ps_i(v) = 0$: $\mathbf{pd}_i^v \leftarrow \mathbf{r}^{0-i}$, $v = 1$ to $|\mathbf{ps_i}|$
    - if $ps_i(v) \neq 0$: $s'' \leftarrow \mathsf{PER}(ps_i(v))$, $\mathbf{pd}_i^v \leftarrow \mathbf{r}^{s''-i}$
    - $\mathbf{pd}_i^z \xleftarrow{\$} \{0,1\}^\kappa$, $z = |\mathbf{ps_i}| + 1$ to *inp*
- *Generating random keys*
  - $\mathsf{K}_i^j \xleftarrow{\$} \{0,1\}^{\kappa'}, i \in [0, |Q|], j \in [1, |\mathbf{ch}_i|]$
  - $t = 0, m \in [0, |\Sigma|], y_m \in \Sigma$
  - if $y_m \in \mathbf{ch}_i^j, i \in [0, |Q|], j \in |\mathbf{ch}_i|$:
    - $t++, \mathsf{K'}_m^t \leftarrow \mathsf{K}_i^j$
  - $\mathsf{K'}_m \leftarrow \mathsf{K'}_m || \mathsf{K}_m^{z(b)}, b \in [1, t], \mathbf{z} \leftarrow \text{permute}(1, \ldots, t)$
  - $l = \kappa' \times ch_{max}$, if $|\mathsf{K'}_m| < l : \mathbf{k} \xleftarrow{\$} \{0,1\}^{\kappa' \times (ch_{max} - |\mathsf{K'}_m|)}, \mathsf{K'}_m \leftarrow \mathsf{K'}_m || \mathbf{k}$
- *Garbling the matrix*
  - *Start state:* $i = 0$, $j \in [1, |\mathbf{ch}_i|]$
    - $s \leftarrow m_0(j)$,
    - if $s \neq 0$: $s' \leftarrow \mathsf{PER}(s)$, $\mathbf{m'_0}^j \leftarrow (s' || \mathbf{r}^{\mathbf{0}-\mathbf{s}'} || 0^{\kappa + \lg|Q|}) \oplus \mathsf{K}_i^j$
    - if $s = 0$: $\mathbf{r}^{\mathbf{0}-\mathbf{0}} \xleftarrow{\$} \{0,1\}^\kappa$, $\mathbf{m'_0}^j \leftarrow (s || \mathbf{r}^{\mathbf{0}-\mathbf{0}} || 0^{\kappa + \lg|Q|}) \oplus \mathsf{K}_i^j$
    - $\mathbf{m'_0}^z \leftarrow \{0,1\}^{\kappa'}$, $z = |\mathbf{ch}_i| + 1$ to *out*
    - $\mathbf{a}_0^k \leftarrow (r_0^k)_2$ if $m_0(0) = 1$, else $\mathbf{a}_0^k \leftarrow (r_0^k + 1)_2$, where r_0^k is a random value with κ bits in binary format $(r_0^k)_2$
  - *Other states:* $i = 1$ to $|Q|$, $j = 1$ to $|\mathbf{ch}_i|$
    - $s \leftarrow m_i(j)$,
    - $s' \leftarrow \mathsf{PER}(s)$
    - *Input edges for any state:* $v = 1$ to *inp*
      - $\mathbf{t} \leftarrow \mathsf{perm}(i, j)$
      - $\mathbf{mm_i}^{j,v} \leftarrow (t(v) || s' || \mathbf{r}^{\mathbf{i}-\mathbf{s}'} || 0^\kappa) \oplus \mathsf{PRG}(\mathbf{pd_i}^{t(v)}) \oplus \mathsf{K}_i^j$
    - $\mathbf{mm}_i^{z,1} \leftarrow \{0,1\}^{\kappa'}, \ldots, \mathbf{mm}_i^{z,inp} \leftarrow \{0,1\}^{\kappa'}$, $z = |\mathbf{ch}_i| + 1$ to *out*
    - $\mathbf{a}_i^k \leftarrow (r_i^k)_2$ if $m_i(0) = 1$, else $\mathbf{a}_i^k \leftarrow (r_i^k + 1)_2$, where r_i^k is a random value with κ bits in binary format $(r_i^k)_2$
- *Permuting and constructing the matrix*, $i \in [1, |Q|]$
  - $\mathbf{gm}_{\mathsf{PER}(i),k}^{1,t} \leftarrow \mathbf{mm}_i^{1,t}, \ldots, \mathbf{gm}_{\mathsf{PER}(i)}^{out,t} \leftarrow \mathbf{mm}_i^{out,t}$, $t = 1$ to *inp*
  - $\mathbf{GM}^k \leftarrow \{\mathbf{gm}_{i,k}\}_{i\in[|Q|]}$, $\mathbf{gm}_{0,k} \leftarrow \mathbf{m}_0^{'}$, $\mathbf{ga}_{\mathsf{PER}(i)}^k \leftarrow \mathbf{a}_i^k$

Figure 6: Constructing the garbled DFA matrix

described in Section 3. The server S_k assigns the keys to each character group. Then, for any character $y_i \in \Sigma$, S_k concatenates the keys of character groups of which y_i is a member of them.

- *Computing the garbled matrix.* Finally, $\mathsf{S}_k, k \in [1, n]$ constructs the garbled DFA matrix through the following steps. **a)** S_k generates a random vector for the start state and a matrix for $|Q|$ states using randomly chosen pads, where $\xleftarrow{\$}$ denotes the assignment of a randomly selected element. Then S_k generates a permuted set of states for $\{0, \ldots, |Q|\}$ using the permutation function $\mathsf{PER}(s) \rightarrow s'$ and inputs a state $s \in \{1, \ldots, |Q|\}$, and outputs a unique state $s' \in \{1, \ldots, |Q|\}$. **b)** Next S_k garbles the DFA matrix according to the following algorithm. K_j's are the random keys which are used in OTs. Note that $\mathsf{PRG} : \{0,1\}^{\kappa} \rightarrow \{0,1\}^{\kappa'}$ is a secure pseudorandom generator (PRG). Assume that $\kappa' = 2\kappa + 2\log|Q|$, and let *out* denote the maximum number of outgoing edges for a state of DFA. For any state, the next state is denoted by s, and the permuted next state is obtained using PER. Also, $\mathsf{perm}(i, j) \rightarrow s'$ takes the state i and j, the index of the j-th character group, as inputs and outputs a vector that is a random permutation of values $(1, \ldots, inp)$. Note that, PER and perm are used to hide the structure of the DFA on the keyword. A random value r_i^k is used to determine whether a state is accepting or not. If state i is accepting, the binary format of r_i^k is assigned to $\mathbf{a}_i^k$; otherwise, the binary format of $r_i^k + 1$ is assigned. Any element of the garbled matrix is the result of XORing the concatenation of t', $\mathsf{PER}(s)$, a random pad, and κ zeros with the corresponding keys.

Step 2: access control check. The servers collaboratively enforce access control by executing the proposed DPF-PACL protocol, as outlined in Fig. 4. If this process does not succeed, the operation will halt; otherwise, the next step will proceed. **Step 3: evaluating the matrix.** For each keyword, the server S_k sends the random value $r^k = \Sigma\, r_i^k$ to the user and the garbled matrix to the proxy server. Also n Proxy OTs, as described in Section 3, are run between proxy, the server S_k, and user to evaluate the user's input $\mathsf{w_u}$ on the garbled matrix. This phase is described in Fig. 7. Finally, the proxy server computes the matched score score_j, $j \in [1, v]$ and send it to the user. **Step 4: retrieve the data.** then, user retrieves the data with the maximum value of scores and sends a request to the servers $\mathsf{S_1}, \ldots, \mathsf{S_n}$ as it is described in Fig. 8. The user computes the scores and assumes that they would like to retrieve $\mathsf{data_{index}}$. The user uses a DPF $f : \{0,1\}^{\kappa} \rightarrow \mathbb{Z}_2$ where $[f_x]_k \leftarrow \mathsf{DPF.Eval}(\mathsf{k}_{x,k})$, $x \in [1, v], k \in [1, n]$; note that $\mathsf{out} = \bigoplus_{k=1}^{n} [f_x]_k$ equals 1 if $x = \mathsf{index}$, and 0 otherwise. Each server $\mathsf{S}_k, k \in [1, n]$ runs DPF on the data and sends the result to user to obtain the corresponding value.

Theorem 3. *In the* OT*-hybrid model, given a computationally secure* PRG*, the scheme* $\pi_{\mathsf{DR-ACL}}$ *is secure in the presence of a semi-honest setting.*

Proof. The proof is based on the POT hybrid model and we consider three cases in which an adversary corrupts S_k, proxy and user in the following.

- user *is corrupted.* Suppose that a PPT adversary $\mathcal{A}$ controls user. Sim emulates $\mathcal{F}_{\mathsf{POT}}$ and obtains the keys used in POT's execution. Then, Sim garbles the matrix constructed from the server's keyword. The simulator inherits the properties of the underlying POT protocol, and therefore, the adversary is unable to distinguish between the ideal world and real-world executions.
- proxy *is corrupted.* Suppose that a PPT adversary $\mathcal{A}$ controls proxy, and the simulator Sim behaves as follows, simulating the view of $\mathcal{A}$. Sim emulates the functionality $\mathcal{F}_{\mathsf{POT}}$ and gets the input of the server S_k. Then, Sim runs $\mathcal{A}$ on the input w_k, which $\mathcal{A}$ forwards after receiving it from S_k in the real execution. Next, Sim generates ch_{max} random keys for running OTs on w_k and sends them to the trusted authority. Sim sends the server's input, w_k, to the trusted authority and receives back the evaluation of DFA on it, with the set of corresponding states denoted by Γ_{w_k}. Next, Sim generates

This algorithm is run by the proxy server
- Input: the garbled matrix GM^k, the vectors $\mathbf{gm}_{0,k}$, $\mathbf{a}_0^k$ and $\{\mathbf{ga}_i^k\}_{i\in[|Q|]}$
- Output: random value $\mathbf{score}_k$

Steps for any $z \in [1, |\mathbf{w}_u|]$ where $\mathbf{w}_u$ is input of the user
- parse $\mathbf{GM}^k$ as $\{\mathbf{gm}_{i,k}\}_{i\in[|Q|]}$
- $\mathsf{K}''_1, \ldots, \mathsf{K}''_{ch_{max}} \leftarrow \mathsf{POT}(\mathsf{K}'_1, \ldots, \mathsf{K}'_{|\Sigma|})$ (see Section 3)
- **Start state:** $(i = 0)$, $j \in [1, out]$
  - $state||\mathbf{pad}||\mathbf{taill} \leftarrow \mathbf{gm}_{0,k} \oplus \mathsf{K}''_v$, $v \in [1, ch_{max}]$
    - if $tail = \mathbf{0}^{\kappa+\lg|Q|}$ then
      - $t' \leftarrow 0$
      - $\mathbf{score}_k \leftarrow \mathbf{a}_0^k$
      - $i \leftarrow state$
      - $\mathbf{pd_i} \leftarrow \mathbf{pad}$
    - end if
- **Other states:**
  - if $t' = 0$ then
    - $t \in [1, inp]$
    - $t'||state||\mathbf{pad}||\mathbf{tail} \leftarrow \mathbf{gm}_i^{j,t} \oplus \mathsf{K}''_v \oplus \mathsf{PRG}(\mathbf{pd_i})$, $v \in [1, ch_{max}]$
    - if $\mathbf{tail} = \mathbf{0}^{\kappa}$ then
      - $\mathbf{score}_k \leftarrow \mathbf{score}_k + \mathbf{ga}_i^k$
      - $i \leftarrow state$
      - $\mathbf{pd_i} \leftarrow \mathbf{pad}$
    - end if
  - else if $t' \neq 0$ then
    - $t \leftarrow t'$
    - $t'||state||\mathbf{pad}||\mathbf{tail} \leftarrow \mathbf{gm}_i^{j,t} \oplus \mathsf{K}''_v \oplus \mathsf{PRG}(\mathbf{pd_i})$, $v \in [1, ch_{max}]$
    - if $\mathbf{tail} = \mathbf{0}^{\kappa}$ then
      - $\mathbf{score}_k \leftarrow \mathbf{score}_k + \mathbf{ga}_i^k$
      - $i \leftarrow state$
      - $\mathbf{pd_i} \leftarrow \mathbf{pad}$
    - end if
  - end if

Figure 7: Evaluating the garbled matrix

This algorithm is run by the user and servers; note that $k \in [1, n]$ and $j \in [1.v]$. Let n servers, denoted $\mathsf{S}_1, \ldots, \mathsf{S}_n$, each store a shared database $\mathsf{DB} = \{(\mathsf{w}_1, \mathsf{data}_1), \ldots, (\mathsf{w}_v, \mathsf{data}_v)\}$, where w_x corresponds to the keyword and data_x (for $x \in [1, v]$) represents the related data.
- user computes $\mathsf{score}_k(j) = r^k(j) - (\mathbf{score}_k(j))_{10}$, where $(\mathbf{score}_k(j))_{10}$ denotes the decimal representation of $\mathbf{score}_k(j)$
- user sets $\mathsf{index} = \mathsf{inx} \mid \mathsf{score}_{\mathsf{inx}} = \min_{j=1}^{v}(\mathsf{score}_k(j))$
- each server S_k sets $[f_x]_k \leftarrow \mathsf{DPF.Eval}(\mathsf{k}_{x,k})$, $x \in [1, v], k \in [1, n]$
- each server S_k sends back the $\mathsf{ans_k} = \bigoplus_{x=1}^{v} \mathsf{data}_x[f_x]_k$, $k \in [1, n]$
- user obtains $\mathsf{data}_{\mathsf{index}} \leftarrow \bigoplus_{k=1}^{n} \mathsf{ans}_k$

Figure 8: Data Retrieval

the garbled DFA matrix using the following steps for any character of the input and forwards it to $\mathcal{A}$. Let n denote the length of w_k and $\kappa' = 2\kappa + 2\log|Q|$.

– *Initialize:* $\{s_1, s_2, \ldots, s_{nk}\} \xleftarrow{\$} \{1, \ldots, |Q|\}$, $start \leftarrow 0, i \leftarrow start$, $t \xleftarrow{\$} \{1, \ldots, inp\}$

– *Start state:*

* $s' \xleftarrow{\$} \{s_1, s_2, \ldots, s_{nk}\}$, $\mathbf{r} \xleftarrow{\$} \{0,1\}^{\kappa}$, $\{\mathsf{K}_1, \mathsf{K}_2, \ldots, \mathsf{K}_{ch_{max}}\} \xleftarrow{\$} \{0,1\}^{\kappa'}$; generate ch_{max} random keys
* $col \xleftarrow{\$} \{1, \ldots, out\}$, $\mathbf{m'_0}^{col} \leftarrow (s' \| \mathbf{r} \| 0^{\kappa + \log|Q|}) \oplus \mathsf{K}_{col}$
* $\mathbf{m'_0}^{j} \xleftarrow{\$} 0^{\kappa'}$, $j = 1, \ldots, |Q|$, $j \neq col$, $\mathbf{a_0} \xleftarrow{\$} \{0,1\}^{\kappa}$, $t' \xleftarrow{\$} \{1, \ldots, inp\}$, $i \leftarrow s'$, $\mathbf{pd}_i{}^{t'} \leftarrow \mathbf{r}$

– *Other states:*

* $i \leftarrow s'$, $\mathbf{pd}_i{}^{t'} \leftarrow \mathbf{r}$, $col \xleftarrow{\$} \{1, \ldots, out\}$
* $\mathbf{gm_i}^{col} \leftarrow (t' \| s' \| \mathbf{r} \| 0^{\kappa}) \oplus \mathsf{K}_{col} \oplus \mathsf{PRG}(\mathbf{pd}_i{}^{t})$
* $\mathbf{gm_i}^{j,1} \xleftarrow{\$} 0^{\kappa'}, \mathbf{gm_i}^{j,2} \xleftarrow{\$} 0^{\kappa'}, \ldots, \mathbf{gm_i}^{j,inp} \xleftarrow{\$} 0^{\kappa'}$, $j = 1, \ldots, |Q|$, $j \neq col$
* $\mathbf{a_i} \xleftarrow{\$} \{0,1\}^{\kappa}$, $t \leftarrow t'$, $i \leftarrow s'$, $\mathbf{pd}_i{}^{t'} \leftarrow \mathbf{r}$

We now prove that $\mathcal{A}$ cannot distinguish between its view during the real execution and its interaction with Sim. Suppose that a distinguisher $\mathcal{D}$ distinguishes the distributions of the real and ideal world transcripts with non-negligible probability. Let $\mathbf{gm}_i$ be in the i-th element of in the garbled matrix. Relying on a hybrid argument, we define the hybrid distributions $H_i \stackrel{\mathsf{def}}{=} (\mathbf{gm}_{1,re}, \ldots, \mathbf{gm}_{i,re}, \mathbf{gm}_{i+1,id}, \ldots, \mathbf{gm}_{z,id})$, $i = 0, \ldots, nk$ $z = |Q|out$. Note that H_0 is the view of $\mathcal{A}$ in the real execution and H_{nk} is his interaction with Sim. Hence, $\mathcal{D}$ distinguishes H_{i-1} from H_i for some i in polynomial time with non-negligible probability. But H_{i-1} and H_i are generated in the same way. Moreover, random strings for all values in row i are uniformly chosen except for one value that appears on the transit path of input. Thus, due to the security of the PRG used in the matrix construction, this contradicts the security of PRG. Hence we can argue that $H_{i-1} \stackrel{c}{\approx} H_i$. So, the view of $\mathcal{A}$ when interacting with Sim is the same as H_{nk}.

- *The server* S_k *is corrupted.* The only message from user occurs during the POT protocol execution. user's privacy against S_k follows from the privacy of the POT protocol used.

□

4.3.1 Analysis

Complexity Analysis. Let $|\Sigma|$ denote the size of the alphabet being used, let $|Q|$ denote the number of states of DFA and let n_u denote the size of user's input. The symmetric computation is PRG invocations. The asymmetric commutation is for OTs. We have taken advantage of the concept of $\mathsf{Proxy\ OT}$ [CKKC13] in our scheme to improve efficiency. The communication overhead of our scheme consists of OT communication and garbled DFA matrix transfer. In our scheme, the proxy obtains the keys by running a few 1-out-of-2 OTs and XOR operations. The parties should run $\mathsf{Proxy\ OT}$ (see Section 3) such that in the offline phase, the server S_k computes the related key for OT and sends the encoded messages to proxy. So, in the online phase any user can only send a message to proxy which is encoded format of his/her input. Also, user computes $O(n)$ PRGs to compute the matched result. For communication, the server transmits $O(|Q|out \cdot inp)$ bits, note that Q is the number of states of DFA, out and inp denote the maximum number of output and input edges in DFA graph representation, respectively. For each cell, the server performs one PRG G invocation. On the other hand, the client invokes one PRG for each row of the garbled matrix.

Efficiency Comparison. We use a method similar to [MNSS12, NSMS14], which garbles a matrix representing a DFA for secure evaluation. In [MNSS12], a server garbles an

$n \times |Q|$ matrix and sends it to a client, who uses OTs to ungarble a path and extract the output. While [MNSS12] struggles with high complexity for ASCII alphabets, [NSMS14] improves efficiency in such cases. In [NSMS14], a DFA matrix is constructed to compare a keyword with the user's input. In our scheme, the garbled DFA matrix is used for secure data retrieval with access control, where the user's input is evaluated against a set of keywords related to a shared database to retrieve relevant data without revealing the input or the database contents. We compare the two schemes without considering or applying OT extension for either. Let n be the size of the user input. Assume κ is the security parameter and n is the user input size. In both schemes, OTs and PRGs are the two main operations. In [NSMS14] the client and user require $O(n)$ OT computations, the server requires $O(n|Q|)$ PRG computations, and the user requires $O(n)$ PRG computations. We use the construction of [NP05] to efficiently reduce n invocations of the 1-out-of-256 OT to approximately $8n$ invocations of 1-out-of-2 OTs. In our scheme, these 1-out-of-2 OTs are constructed based on XOR operations and [NSMS14] requires exponentiations to run these OTs. The communication complexity is $O(|\Sigma||Q|n\kappa)$ for [MNSS12], $O(out|Q|n\kappa)$ for [NSMS14], and $O(out \cdot inp|Q|\kappa)$ for our scheme. Ignoring OT extensions, [MNSS12] and [NSMS14] require $O(n)$ exponentiations and $O(n|Q|)$ PRGs for the server, while the client performs $O(n)$ exponentiations and PRGs. In contrast, our scheme reduces server computation to $O(\kappa)$ exponentiations and $O(|Q| \cdot out \cdot inp)$ PRGs, with the client computing $O(\kappa)$ exponentiations and $O(n)$ PRGs.

5 Evaluation

Comparison to Previous Work. We chose to use a hash-based succinct argument as our post-quantum argument instead of a lattice-based argument for two reasons. Natural replacements of Bulletproofs, such as lattice-based Bulletproof [CLM23] have argument sizes that are concretely extremely large. Newer schemes such as Greyhound [NS24] and CMNW [CMNW24] offer better proof sizes but lack readily usable implementations. Within the landscape of hash-based succinct arguments, we chose a STARK-based [BBHR19] system over alternatives [GLS+23, XZS22, ACFY24] because of the concretely smaller proof sizes and the mature tooling, which helped our implementation efforts. The motivation for using STARK [BBHR19] over lattice-based bulletproofs [CLM23] stems from two main issues with lattice-based arguments: first, existing schemes produce extremely large proofs, to the extent that [CLM23] does not even compute the argument size, and second, there are no feasible implementations. Although newer schemes like Greyhound [NS24] and CMNW [CMNW24] offer better proof sizes, they still lack readily usable implementations. In comparison, both Brakedown [GLS+23] and Orion [XZS22] produce proofs that are excessively large, ranging from 1 to 10 MB, which makes them impractical for scenarios that require smaller proofs. Additionally, the tooling for these schemes is subpar, meaning that their implementation would have been time-consuming. These challenges highlight the need for a more efficient and feasible approach, such as STARK [BBHR19].

Concurrent Work. In concurrent work, Abdolmaleki et al. [AFKN25] propose a post-quantum PVSS scheme based on a lattice-based SNARK. While their approach is theoretically interesting, our scheme includes an implementation and, aside from potentially enabling straight-line extraction, the STARK-based construction does not appear to offer significant practical advantages.

5.1 Experiment Setup

Our implementation can be found at [pac25]. We used C++ for our main PVSS protocol. For the ZK proofs, we used the risc0 $v2.1.0$ framework [ris24] for the STARK version and

the *bulletproofs* [1] framework for the Bulletproof version [GHL22], both in Rust. Our DPF implementation was integrated from [Cit21] and used as it is. Our application consists of a single program creating 3 threads (user, data server 1, data server 2). They all run on one machine and use network communication to exchange data between each other. We implement the PACL similarly to that of [SSBYP23], while involving the LWE-based GHL^+ scheme for secret sharing. In addition, we compare the STARK-based ZK proofs with Bulletproofs under the same constraints. For our benchmarks, we run our program on a server with two Intel Xeon Gold 5317 CPUs with 24 cores @ 3.00 GHz and 128 GB of RAM.

Parameters The performance evaluation was conducted across the following parameter space, with default values used as baselines for individual parameter studies:

- Number of files (n): $\{64, 128, 256, \mathbf{512}, 1024\}$
- Number of secrets (m): $\{2, 4, 8, 16, \mathbf{32}, 64\}$
- PVW redundancy (l): $\{10, \mathbf{20}, 30, 40, 50\}$
- LWE redundancy (k): $\{1, \mathbf{2}, 3, 4, 5\}$

The default configuration ($n = 512, m = 32, l = 20, k = 2$) represents a typical deployment scenario and serves as the baseline for parameter sensitivity analysis.

5.2 End-to-End Comparison

Table 1: Performance results of DPF-PACL

Scheme	PACL (ms)	ZK (ms)
[SSBYP23] (Schnorr)	13	44
[GHL22] (Bulletproof)	8	2,511
Ours (STARK)	8	114,571

To evaluate our approach against existing work, we benchmark the PIR-PACL implementation from [SSBYP23] [2] using parameters $db_size = 512$ and $item_size = 128$[3] for fair comparison with our default configuration. Table 1 separates PACL and ZK proof times to demonstrate the efficiency of the PVSS approach, excluding ZK. The results demonstrate that PACL overhead is negligible compared to ZK proof costs, making ZK performance the primary focus of our following analysis.

5.3 Benchmarks of ZK Proofs

This subsection presents a comprehensive performance comparison between two prominent ZK proof systems for our Post-Quantum PACL use case: Bulletproofs (BP) and STARKs. The evaluation focuses on their application in PVSS. Four primary operations were evaluated: Bulletproof and STARK proving/verifying for both encryption and decryption across various system parameters.

5.3.1 Results and Analysis

Proving Performance Figure 9 presents the proving performance analysis across different parameter configurations. The results reveal significant differences between Bulletproofs and STARKs.

[1] https://github.com/dalek-cryptography/bulletproofs at commit 464acb7
[2] https://github.com/sachaservan/pacl at commit 2b1c6ee
[3] Our secret is 4-byte long; 32 secrets total 128 bytes.

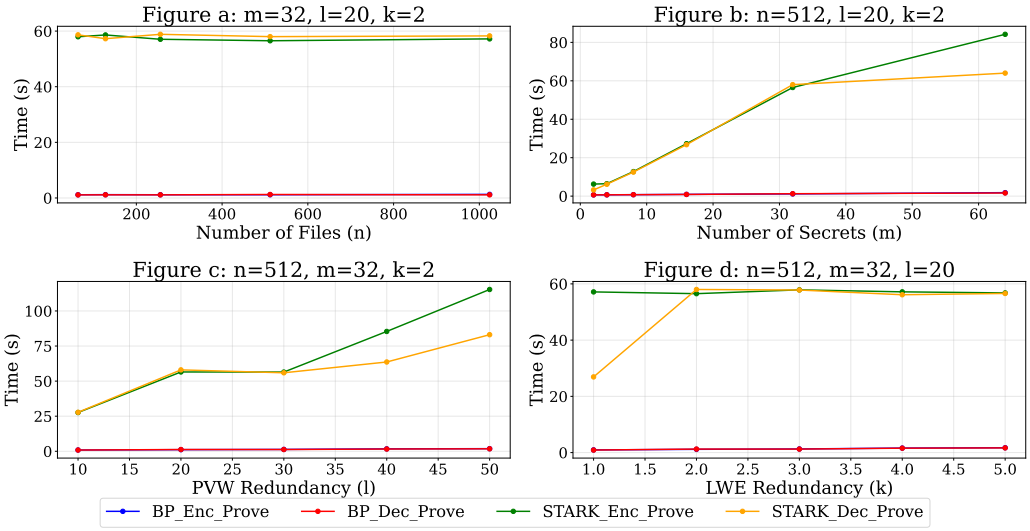


Figure 9: ZK Proof Performance

Bulletproof methods achieve proving times of approximately 1-2 seconds, while STARK methods require 50-80 seconds, making STARKs approximately 50× slower than Bulletproofs for proving operations. This substantial performance gap represents the primary distinguishing factor between the two approaches.

Parameter sensitivity analysis shows that secret count ($m = 2 \rightarrow 64$) has the most significant impact on performance with linear growth, while file count ($n = 64 \rightarrow 1024$) and LWE redundancy ($k = 1 \rightarrow 5$) maintain stable performance. PVW redundancy ($l = 10 \rightarrow 50$) shows moderate performance increases. Within each proof system family, encryption and decryption proving operations exhibit similar performance characteristics.

Verification Performance Figure 10 presents the verification performance analysis, demonstrating an inverse relationship to proving performance.

Bulletproof verification requires approximately $\sim 50-110\,ms$, while STARK verification achieves faster times of $\sim 20-45\,ms$, making STARKs approximately $2-3\times$ faster than Bulletproofs for verification operations.

As shown in Fig. 10, verification times exhibit more modest scaling with parameter variations compared to proving times, with both systems maintaining reasonable verification performance across the tested parameter space. This scalability characteristic ensures that verification remains efficient regardless of the specific parameter configuration used.

Comparative Analysis The performance characteristics illustrated in Figs. 9 and 10 reveal a fundamental trade-off between proving and verification efficiency. While bulletproofs demonstrate superior proving performance, STARKs excel in verification speed. This complementary relationship suggests that the optimal choice depends on the specific application requirements and the expected ratio of proving to verification operations.

Proof size We have provided a comparison in Table 2. The primary motivation for our approach is not to outperform Bulletproofs in proof size-rather, it is to achieve post-quantum security while avoiding the massive size of other post-quantum systems like Brakedown [GLS+23] or Orion [XZS22], and the inefficiency of [SSBYP23]'s secret-shared proofs (which are $O(m)$ and can be much larger still).

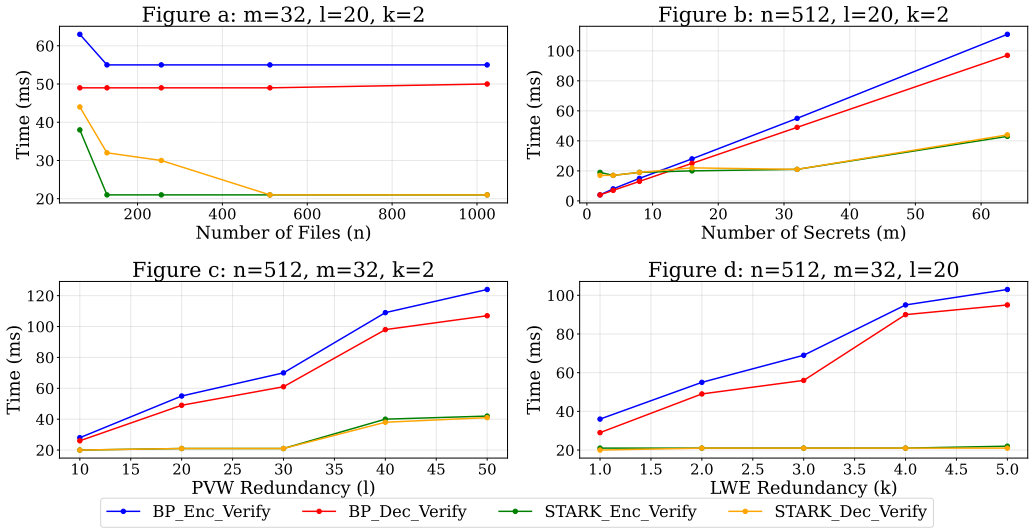


Figure 10: ZK Verification Performance

Table 2: Asymptotic proof size. Here, m is the vector/constraint length in Bulletproof checks; T is the execution-trace length of the PVSS-check program (STARK); t is the audit threshold / number of participating verifiers; λ is the security parameter.

Scheme / Proof type	**Underlying system**	**Asymptotic proof size**
[GHL22]: sharing correctness	Bulletproofs (DL-based)	$O(\log m)$ group elements
[GHL22]: decryption correctness	Bulletproofs (DL-based)	$O(\log m)$ group elements
[SSBYP23]: authorization proof	SPoSS (Schnorr-over-shares)	$O(\lambda)$ per verifier (proof share & audit token)
Ours: sharing correctness	STARK (RISC Zero)	$O(\text{polylog}\, T)$ field elements
Ours: decryption correctness	STARK (RISC Zero)	$O(\text{polylog}\, T)$ field elements
Total in PACL (our work)	STARK + per-verifier auth	$O(\text{polylog}\, T) + O(t\lambda)$

Proof-size discussion. A trade-off in our design is proof size versus post-quantum security. Classical Bulletproofs (as used in [GHL22]) yield succinct proofs with $O(\log m)$ *group* elements and excellent concrete sizes, but they rely on discrete-log assumptions and are thus not post-quantum secure. In contrast, our PACL adopts a STARK-family system (RISC Zero) to obtain post-quantum security; the resulting receipts scale $O(\text{polylog}\, T)$ in field elements, but carry larger constants due to commitment layers and FRI [BSBHR18] queries. Consequently, for comparable parameters, our post-quantum proofs are typically larger than Bulletproofs (by design), while still markedly more compact than other post-quantum options (e.g., certain hash-based systems with multi-megabyte proofs) and asymptotically superior to the generic SNIP-based proofs [CGB17] in [SSBYP23] that grow $O(m)$. This choice strikes a practical balance: we forgo the smallest (classical) proofs in exchange for concrete, implementable post-quantum security with proof sizes that remain manageable for our target access-control workloads.

6 Conclusion

In this work, we presented a novel construction of private access control lists (PACL) based on PVSS under the SIS problem, ensuring post-quantum security through the LWE assumption. Our approach enforces database access control policies via FSS evaluators,

who can verify the dealer's authorization to share specific functions. Additionally, we proposed a secure data retrieval scheme that incorporates DPF-PACL for enhanced access control. This scheme is not only complementary to the PACL framework, but it also offers a standalone utility for secure and controlled data access.

Acknowledgements

The authors appreciate Geoffroy Couteau's valuable and profound comments. Giacomo Fenzi is partially supported by the Ethereum Foundation. The authors from the Barkhausen Institut received support from the German Federal Ministry of Research, Technology, and Space through the SEMECO project (Grant No. 03ZU1210AA). They are additionally funded by resources allocated through the Saxonian State Parliament in Germany.

References

[ABF+19] Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 441–471. Springer, 2019. doi:10.1007/978-3-030-17659-4_15.

[ACDN13] Masayuki Abe, Jan Camenisch, Maria Dubovitskaya, and Ryo Nishimaki. Universally composable adaptive oblivious transfer (with access control) from standard assumptions. In *Proceedings of the 2013 ACM workshop on Digital identity management*, pages 1–12, 2013. doi:10.1145/2517881.2517883.

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. LNCS, pages 380–413, August 2024. doi:10.1007/978-3-031-68403-6_12.

[ACGU20] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 467–497. Springer, 2020. doi:10.1007/978-3-030-64840-4_16.

[AFKN25] Behzad Abdolmaleki, Mohammad Foroutani, Shahram Khazaei, and Sajjad Nasirzadeh. A generic framework for practical lattice-based non-interactive publicly verifiable secret sharing. *Cryptology ePrint Archive*, 2025. URL: https://eprint.iacr.org/2025/901.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732, August 2019. doi:10.1007/978-3-030-26954-8_23.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, October / November 2016. doi:10.1007/978-3-662-53644-5_2.

[BFMP21] Amos Beimel, Oriol Farràs, Yuval Mintz, and Naty Peter. Linear secret-sharing schemes for forbidden graph access structures. *IEEE Transactions on Information Theory*, 68(3):2083–2100, 2021. doi:10.1109/TIT.2021.3132917.

[BG23] Jeremy Bruestle and Paul Gafni. Risc zero zkvm: Scalable, transparent arguments of risc-v integrity. *Draft. July*, 29, 2023. URL: https://dev.risczero.com/proof-system-in-detail.pdf.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer, 2015. doi:10.1007/978-3-662-46803-6_12.

[BKKO20] Paul Bunn, Jonathan Katz, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient 3-party distributed oram. In *Security and Cryptography for Networks: 12th International Conference, SCN 2020, Amalfi, Italy, September 14–16, 2020, Proceedings 12*, pages 215–232. Springer, 2020. doi:10.1007/978-3-030-57990-6_11.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469, August 2020. doi:10.1007/978-3-030-56880-1_16.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *45th international colloquium on automata, languages, and programming (icalp 2018)*, pages 14–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.14.

[CCZ24] Rongsheng Cai, Lanxiang Chen, and Yizhao Zhu. Using private set intersection to achieve privacy-preserving authorization for iot systems. *Journal of Information Security and Applications*, 83:103759, 2024. doi:10.1016/j.jisa.2024.103759.

[CD24] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 216–248. Springer, 2024. doi:10.1007/3-540-68339-9_17.

[CDN09] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 131–140, 2009. doi:10.1145/1653662.1653679.

[CDNZ11] Jan Camenisch, Maria Dubovitskaya, Gregory Neven, and Gregory M Zaverucha. Oblivious transfer with hidden access control policies. In *Public Key Cryptography–PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings 14*, pages 192–209. Springer, 2011. doi:10.1007/978-3-642-19379-8_12.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 259–282, 2017. URL:

https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs.

[CHT13] Michael Clear, Arthur Hughes, and Hitesh Tewari. Homomorphic encryption with access policies: Characterization and new constructions. In *International Conference on Cryptology in Africa*, pages 61–87. Springer, 2013. doi:10.1007/978-3-642-38553-7_4.

[Cit21] C++ dpf-pir library, 2021. URL: https://github.com/dkales/dpf-cpp.

[CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *Theory of Cryptography Conference*, pages 499–518. Springer, 2013. doi:10.1007/978-3-642-36594-2_28.

[CLM23] Valerio Cini, Russell WF Lai, and Giulio Malavolta. Lattice-based succinct arguments from vanishing polynomials. In *Annual International Cryptology Conference*, pages 72–105. Springer, 2023. doi:10.1007/978-3-031-38545-2_3.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: post-quantum security, fast verification and transparent setup. In *Annual International Cryptology Conference*, pages 207–242. Springer, 2024. doi:10.1007/978-3-031-68403-6_7.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29, December 2019. doi:10.1007/978-3-030-36033-7_1.

[CZLC20] Zhihong Chen, Bo Zhao, Hai Lin, and Lin Chen. Etoram: A more efficient oram for secure computation. *IEEE Open Journal of the Computer Society*, 1:285–294, 2020. doi:10.1109/OJCS.2020.3032020.

[DAT20] Bakkiam David Deebak and Fadi Al-Turjman. Smart mutual authentication protocol for cloud based medical healthcare systems using internet of medical things. *IEEE Journal on Selected Areas in Communications*, 39(2):346–360, 2020. doi:10.1109/JSAC.2020.3020599.

[DDH+24] Uddipana Dowerah, Subhranil Dutta, Frank Hartmann, Aikaterini Mitrokotsa, Sayantan Mukherjee, and Tapas Pal. Sacfe: Secure access control in functional encryption with unbounded data. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 860–882. IEEE, 2024. doi:10.1109/EuroSP60621.2024.00053.

[DIL+22] Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. Streaming and unbalanced psi from function secret sharing. In *International Conference on Security and Cryptography for Networks*, pages 564–587. Springer, 2022. doi:10.1007/978-3-031-14791-3_25.

[DR20] Aditya Damodaran and Alfredo Rial. Unlinkable updatable databases and oblivious transfer with access control. In *Australasian Conference on Information Security and Privacy*, pages 584–604. Springer, 2020. doi:10.1007/978-3-030-55304-3_30.

[DRPS22] Emma Dauterman, Mayank Rathee, Raluca Ada Popa, and Ion Stoica. Waldo: A private time-series database from function secret sharing. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2450–2468. IEEE, 2022. doi:10.1109/SP46214.2022.9833611.

[ECZB21] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 1775–1792. USENIX Association, August 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/eskandarian.

[FCS+24] Jiaxuan Fu, Ke Cheng, Anxiao Song, Yuheng Xia, Zhao Chang, and Yulong Shen. Fss-dbscan: Outsourced private density-based clustering via function secret sharing. *IEEE Transactions on Information Forensics and Security*, 2024. doi:10.1109/TIFS.2024.3446233.

[GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 458–487. Springer, 2022. doi:10.1007/978-3-031-06944-4_16.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 640–658. Springer, 2014. doi:10.1007/978-3-642-55220-5_35.

[GLS+23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226, August 2023. doi:10.1007/978-3-031-38545-2_7.

[GRS22] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In *Annual International Cryptology Conference*, pages 323–352. Springer, 2022. doi:10.1007/978-3-031-15802-5_12.

[HMSB16] Kamlesh Kumar Hingwe and S Mary Saira Bhanu. Hierarchical role-based access control with homomorphic encryption for database as a service. In *Proceedings of International Conference on ICT for Sustainable Development: ICT4SD 2015 Volume 2*, pages 437–448. Springer, 2016. doi:10.1007/978-981-10-0135-2_43.

[HMU01] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001. doi:10.1145/568438.568455.

[JBN+21] T Jayasankar, RM Bhavadharini, NR Nagarajanz, G Mani, and S Ramesh. Securing medical data using extended role based access control model and two fish algorithms on cloud platform. *European Journal of Molecular and Clinical Medicine*, 8(1):1075–1090, 2021. URL: https://api.semanticscholar.org/CorpusID:231798340.

[JMDL24] Yufan Jiang, Fei Mei, Tianxiang Dai, and Yong Li. Sigbdt: Large-scale gradient boosting decision tree training via function secret sharing. In *Proceedings of*

the 19th ACM Asia Conference on Computer and Communications Security, pages 274–288, 2024. doi:10.1145/3634737.3657024.

[JZR23] Keyu Ji, Bingsheng Zhang, and Kui Ren. Fine-grained policy constraints for distributed point function. *Cryptology ePrint Archive*, 2023. URL: https://eprint.iacr.org/2023/1774.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829, 2016. doi:10.1145/2976749.2978381.

[KW17] Sam Kim and David J Wu. Access control encryption for general policies from standard assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 471–501. Springer, 2017. doi:10.1007/978-3-319-70694-8_17.

[LCC23] Jing Liu, Jamie Cui, and Cen Chen. Online efficient secure logistic regression based on function secret sharing. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1597–1606, 2023. doi:10.1145/3583780.3614998.

[LCZL17] Song Li, Jie Cui, Hong Zhong, and Lu Liu. Public auditing with privacy protection in a multi-user model of cloud-assisted body sensor networks. *Sensors*, 17(5):1032, 2017. doi:10.3390/s17051032.

[LE24] Wassila Lalouani and Lloyd Emokpae. Lightweight zero knowledge proof-based multi access control schema for smart telehealth system. In *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6. IEEE, 2024. doi:10.1109/SmartNets61466.2024.10577736.

[LH21] Han Liu and Dezhi Han. Non-interactive zero knowledge proof based access control in information-centric internet of things. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 617–631. Springer, 2021. doi:10.1007/978-3-030-95388-1_41.

[LLM+17] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Adaptive oblivious transfer with access control from lattice assumptions. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 533–563. Springer, 2017. doi:10.1007/978-3-319-70694-8_19.

[LScY09] Ying Luo, S Cheung Sen-ching, and Shuiming Ye. Anonymous biometric access control based on homomorphic encryption. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1046–1049. IEEE, 2009. doi:10.1109/ICME.2009.5202677.

[LZW+24] Jianhao Li, Rui Zhang, Jiabei Wang, Yansen Xin, and Wenhan Xu. Nemo: Practical distributed boolean queries with minimal leakage. *IEEE Transactions on Information Forensics and Security*, 2024. doi:10.1109/TIFS.2024.3351433.

[MDHM18] Lukas Malina, Petr Dzurenda, Jan Hajny, and Zdenek Martinasek. Secure and efficient two-factor zero-knowledge authentication solution for access control systems. *Computers & Security*, 77:500–513, 2018. doi:10.1016/j.cose.2018.05.006.

[MNSS12] Payman Mohassel, Salman Niksefat, Saeed Sadeghian, and Babak Sadeghiyan. An efficient protocol for oblivious dfa evaluation and applications. In *Topics in Cryptology–CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27–March 2, 2012. Proceedings*, pages 398–415. Springer, 2012. doi:10.1007/978-3-642-27954-6_25.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18:1–35, 2005. doi:10.1007/s00145-004-0102-6.

[NPP25] Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with public inputs and strong security. In *IACR International Conference on Public-Key Cryptography*, pages 68–101. Springer, 2025. doi:10.1007/978-3-031-91826-1_3.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999. doi:10.1145/336992.337028.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In *Annual International Cryptology Conference*, pages 243–275. Springer, 2024. doi:10.1007/978-3-031-68403-6_8.

[NSMS14] Salman Niksefat, Babak Sadeghiyan, Payman Mohassel, and Saeed Sadeghian. Zids: A privacy-preserving intrusion detection system using secure two-party computation protocols. *The Computer Journal*, 57(4):494–509, 2014. doi:10.1093/comjnl/bxt019.

[NSSD22] Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. Spectrum: High-bandwidth anonymous broadcast. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 229–248, 2022. URL: https://www.usenix.org/conference/nsdi22/presentation/newman.

[pac25] Post-quantum access control implementation, 2025. URL: https://github.com/Barkhausen-Institut/PQACL.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Annual international cryptology conference*, pages 554–571. Springer, 2008. doi:10.1007/978-3-540-85174-5_31.

[RDV24] K Rajeshkumar, S Dhanasekaran, and V Vasudevan. A novel three-factor authentication and optimal mapreduce frameworks for secure medical big data transmission over the cloud with shaxecc. *Multimedia Tools and Applications*, pages 1–29, 2024. doi:10.1007/s11042-024-18147-6.

[ris24] Risc0, 2024. URL: https://github.com/risc0/risc0.

[RPWG20] Mamoon Rashid, Shabir Ahmad Parah, Aabid Rashid Wani, and Sachin Kumar Gupta. Securing e-health iot data on cloud systems using novel extended role based access control model. *Internet of Things (IoT) Concepts and Applications*, pages 473–489, 2020. doi:10.1007/978-3-030-37468-6_25.

[RTPB20] Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis Bach. Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *arXiv preprint arXiv:2006.04593*, 2020. doi:10.2478/popets-2022-0015.

[SA23] Urvashi Rahul Saxena and Taj Alam. Role-based access using partial homomorphic encryption for securing cloud data. *International Journal of System Assurance Engineering and Management*, 14(3):950–966, 2023. doi:10.1007/s13198-023-01896-2.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

[SSBYP23] Sacha Servan-Schreiber, Simon Beyzerov, Eli Yablon, and Hyojae Park. Private access control for function secret sharing. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 809–828. IEEE, 2023. doi:10.1109/SP46215.2023.10179295.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 190–199. Springer, 1996. doi:10.1007/3-540-68339-9_17.

[Sun24] Zhen Sun. Efficient multiparty private set intersection protocol based on function secret sharing. In *International Conference on Computer Network Security and Software Engineering (CNSSE 2024)*, volume 13175, pages 20–25. SPIE, 2024. doi:10.1117/12.3031902.

[SYD21] Vipin Singh Sehrawat, Foo Yee Yeo, and Yvo Desmedt. Extremal set theory and lwe based access structure hiding verifiable secret sharing with malicious-majority and free verification. *Theoretical Computer Science*, 886:106–138, 2021. doi:10.1016/j.tcs.2021.07.022.

[SZAJ20] Kashish A Shakil, Farhana J Zareen, Mansaf Alam, and Suraiya Jabin. Bamhealthcloud: A biometric authentication and data management system for healthcare data in cloud. *Journal of King Saud University-Computer and Information Sciences*, 32(1):57–64, 2020. doi:10.1016/j.jksuci.2017.07.001.

[TSCS+19] Akshay Tembhare, S Sibi Chakkaravarthy, D Sangeetha, V Vaidehi, and M Venkata Rathnam. Role-based policy to maintain privacy of patient health records in cloud. *The Journal of Supercomputing*, 75:5866–5881, 2019. doi:10.1007/s11227-019-02887-6.

[TZMT17] Gaosheng Tan, Rui Zhang, Hui Ma, and Yang Tao. Access control encryption based on lwe. In *Proceedings of the 4th ACM International Workshop on ASIA Public-Key Cryptography*, pages 43–50, 2017. doi:10.1145/3055504.3055509.

[Ver22] Neha Verma. Access control-based cloud storage using role-fully homomorphic encryption scheme. *Research Journal of Computer Systems and Engineering*, 3(1):78–83, 2022. doi:10.52710/rjcse.46.

[VSH22] Adithya Vadapalli, Kyle Storrier, and Ryan Henry. Sabre: Sender-anonymous messaging with fast audits. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1953–1970. IEEE, 2022. doi:10.1109/SP46214.2022.9833601.

[WWC21] Xiuhua Wang, Harry WH Wong, and Sherman SM Chow. Access control encryption from group encryption. In *International Conference on Applied Cryptography and Network Security*, pages 417–441. Springer, 2021. doi:10.1007/978-3-030-78372-3_16.

[WWRL10] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *2010 proceedings ieee infocom*, pages 1–9. Ieee, 2010. doi:10.1109/INFCOM.2010.5462173.

[WYG+17] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 299–313, 2017. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/wang-frank.

[XXHA18] Cheng Xu, Jianliang Xu, Haibo Hu, and Man Ho Au. When query authentication meets fine-grained access control: A zero-knowledge approach. In *Proceedings of the 2018 International Conference on Management of Data*, pages 147–162, 2018. doi:10.1145/3183713.3183741.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In *Annual International Cryptology Conference*, pages 299–328. Springer, 2022. doi:10.1007/978-3-031-15985-5_11.

[ZCD+24] Yichen Zang, Chengjun Cai, Wentao Dong, Lei Xu, and Cong Wang. Hiddentor: Toward a user-centric and private query system for tor bridgedb. In *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, pages 1260–1271. IEEE, 2024. doi:10.1109/ICDCS60910.2024.00119.

[ZVH11] Lan Zhou, Vijay Varadharajan, and Michael Hitchens. Enforcing role-based access control for secure data storage in the cloud. *The Computer Journal*, 54(10):1675–1687, 2011. doi:10.1093/comjnl/bxr080.