Silent-Share: Decoupling Hidden Threshold Matching from Pairing Operations via Group-Valued Oblivious Key-Value Stores

Jie Zhang, *Student Member, IEEE*, Xiaohong Li, *Member, IEEE*,
Ruitao Feng, *Member, IEEE*, and Guangdong Bai, *Member, IEEE*

***Abstract*—Matchmaking encryption (ME) enables bilateral access control with private policies, but existing pairing-based constructions tie receiver-side authorization cost to the policy size. This is especially problematic when one party holds a large hidden policy while the other holds only a small attribute set.**

We present Silent-Share, a bilateral hidden-policy threshold access-control protocol that decouples policy representation from pairing-based authorization. The construction combines a one-sided hidden-threshold policy-based key encapsulation mechanism (PB-KEM) with a sparse group-valued oblivious key-value store (GOKVS). The GOKVS compactly encodes policy-dependent group elements, so a receiver holding attribute set $\mathcal{A}$ performs exactly $2|\mathcal{A}|$ pairings, independent of the policy size $|\mathcal{P}|$ and threshold d. Total decapsulation additionally incurs a hidden-threshold reconstruction cost, characterized separately. Two independent one-sided instances are composed and bound with AES-GCM to realize bilateral authorization.

We prove one-sided KEM confidentiality and policy hiding in the random-oracle model under a hidden common exponent assumption, and extend these guarantees to the bilateral composition. Our implementation on BN254 shows that, when the correct d-subset is provided, one-sided decapsulation for $|\mathcal{A}| = 10$ takes about 394 ms, dominated by pairing operations. The pairing-based authorization layer remains flat as $|\mathcal{P}|$ grows from 50 to 800, confirming the policy-size independence. The hidden-threshold reconstruction cost is reported separately and can dominate when $|\mathcal{A}|$ is large. Encapsulation is approximately 2–$3\times$ faster than fuzzy matchmaking encryption across the tested parameter range.

I. Introduction

Modern networked systems increasingly require access decisions to be made from private information held by both sides of an interaction [1, 2]. A requester may want a service only from an owner satisfying a private policy, while the owner may release data only to a requester satisfying a different private policy [3, 4]. This setting arises in privacy-preserving service discovery, selective data dissemination, and bilateral access control [2–10]. A natural cryptographic formulation is bilateral threshold matching: the requester R holds an attribute set $\mathcal{A}_R$ and a private policy $\mathcal{P}_R$ over the owner's attributes, while the owner O holds $\mathcal{A}_O$ and a private policy $\mathcal{P}_O$ over the requester's attributes. Access is granted if and only if

$$|\mathcal{A}_R \cap \mathcal{P}_O| \geq d \qquad \text{and} \qquad |\mathcal{A}_O \cap \mathcal{P}_R| \geq d.$$

The challenge is not merely to evaluate these predicates privately, but to make a successful match directly yield the cryptographic capability to decrypt protected data without revealing the underlying policies.

The unbalanced-policy bottleneck. Existing pairing-based matchmaking encryption and fuzzy variants couple receiver-side authorization work to the policy representation, so decryption cost can grow with $|\mathcal{P}|$ [1, 11–13]. This is particularly problematic when one party holds a large hidden policy while the other holds only a small attribute set, $|\mathcal{P}| \gg |\mathcal{A}|$: the receiver still pays cryptographic work proportional to the sender's policy, even though it only has a small attribute set to test against it [14–17].

A particularly relevant baseline is fuzzy identity-based matchmaking encryption (fuzzy IBME) [13], the only prior scheme that simultaneously supports bilateral access control, threshold matching, and partial policy hiding. In fuzzy IBME, the receiver must scan policy-dependent ciphertext elements to locate matching shares [2]; in an unbalanced setting ($|\mathcal{P}| \gg |\mathcal{A}|$), this scan is dominated by $|\mathcal{P}|$. Even with a known d-subset, the pairing-based reconstruction cost grows linearly with d (e.g., $0.50\,\text{s}$ at $d = 2$ to $2.48\,\text{s}$ at $d = 10$ on BN254). By contrast, Silent-Share performs $2|\mathcal{A}|$ pairings independent of $|\mathcal{P}|$ and d, with decryption time nearly constant at $1.19\,\text{s}$ across $d = 2, \ldots, 10$ in the same hint mode.

Design goal. We target the following systems-level objective:

Key Design Rationale

hide the policy, support threshold matching, bind the match to key release, and make receiver-side pairing work depend on $|\mathcal{A}|$ rather than $|\mathcal{P}|$.

Central observation and approach. The hidden policy and the authorization operation need not use the same representation. Existing OKVS constructions provide compact private matching representations [18–20], but conventional OKVS techniques are not designed to carry the correlated group-valued objects needed by pairing-based authorization [19–21]. We therefore separate the construction into two layers: a group-valued sparse OKVS that represents the sender's hidden policy, and a pairing-based authorization layer that operates only on the receiver's own attributes. This moves policy-size-dependent work to the encapsulation side and leaves the receiver with exactly $2|\mathcal{A}|$ pairings, independent of $|\mathcal{P}|$ and d.

This work is supported in part by the Beijing–Tianjin–Hebei Natural Science Foundation Joint Cooperation Program under Grant 25JJJJC0034, in part by the National Natural Science Foundation of China under Grant 62262073, 62332005, in part by the National Key Research and Development Program of China under Grant 2023YFB3107103.

Jie Zhang and Xiaohong Li are with the College of Intelligence and Computing, Tianjin University, Tianjin, China. (e-mail: {jackzhang, xiaohongli}@tju.edu.cn).

Ruitao Feng is with the Faculty of Science and Engineering, Southern Cross University, Australia (e-mail: ruitao.feng@scu.edu.au).

Guangdong Bai is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. (e-mail: g.bai@cityu.edu.hk).

Ruitao Feng and Guangdong Bai are the corresponding authors.

The full technical construction appears in Section V, and the GOKVS primitive is defined in Section III-E.

Why this is not PSI plus encryption. A generic PSI protocol can efficiently establish whether two sets satisfy a relation [22–24], but the result still needs to be bound to a cryptographic key without exposing additional information or reintroducing a second policy-scale authorization procedure [25–31]. In Silent-Share, the matching witness is embedded directly into the algebraic structure used for key recovery: the GOKVS hides the policy attributes, pairing cancellation extracts candidate shares using only receiver-held attributes, and threshold interpolation determines whether those shares jointly reconstruct the encapsulated secret. The one-sided primitive is non-interactive, and the bilateral protocol composes two such instances.

Our contributions. We make three main contributions.

1) **Hidden-policy threshold PB-KEM with policy-independent pairing authorization.** We construct a one-sided PB-KEM in which a receiver with attribute set $\mathcal{A}$ performs exactly $2|\mathcal{A}|$ pairing operations, independent of $|\mathcal{P}|$ and of the threshold d. Total decapsulation additionally incurs a hidden-threshold reconstruction cost over candidate subsets of the receiver's attributes, which we characterize separately.
2) **Group-valued sparse OKVS for pairing-compatible policy encoding.** We instantiate a sparse GOKVS whose values live directly in the pairing source group. Decoding touches only a constant number of buckets per queried attribute, reducing representation-layer cost from policy-dependent decoding to constant work per attribute.
3) **Silent-Share protocol and security analysis.** We present Silent-Share, which composes two independent one-sided instances and binds them with AES-GCM to realize bilateral threshold authorization. We analyze KEM confidentiality, policy privacy, and collusion resistance under a construction-specific hidden common exponent assumption in the random-oracle model, while making the unavoidable public leakage, including policy size and access decisions, explicit.

Main results and organization. Our implementation shows that the pairing component scales linearly with $|\mathcal{A}|$ while remaining flat as $|\mathcal{P}|$ grows, and that encapsulation achieves a 2–3$\times$ speedup over fuzzy matchmaking encryption across the tested policy sizes, and decapsulation is significantly faster when the receiver's attribute set is small. We also explicitly characterize the worst-case hidden-threshold reconstruction cost. Section II discusses related work; Section III gives preliminaries; Section IV defines the system model and security goals; Section V presents the construction; Section VI provides security results; Section VII evaluates performance; and Section VIII discusses limitations and future directions. Full details are deferred to the appendix.

II. Related Work

We organize prior work into two main lines and then explain why direct composition fails.

Matchmaking encryption and fuzzy variants. Matchmaking encryption (ME) [1] enables bilateral identity-based access control, but couples matching with encryption and supports only exact identity match [11, 12, 32]. Fuzzy IB-ME [13] extends ME to threshold attribute matching via Shamir secret sharing, and is therefore the closest prior work: it is the first to achieve threshold bilateral matching with partial policy privacy and offline discovery [2]. However, its matching-encryption coupling causes receiver-side pairing work to scale with the policy size, and the ciphertext contains policy-dependent components. Follow-up works add credentials, revocability, and multi-receiver support [15–17], but none resolves this coupling. Silent-Share inherits the threshold matching idea from fuzzy IB-ME but decouples policy representation from pairing-based authorization via a group-valued OKVS, so the receiver's pairing cost depends only on $|\mathcal{A}|$, not on $|\mathcal{P}|$.

TABLE I: Qualitative comparison with representative prior schemes.

Scheme	T1	T2	T3	T4	T5	T6	T7
Fuzzy IB-ME [13]	✓	✓*	✓	✗	✓	✗	✗
ME [1]	✓	✗	✗	✗	✓	✗	✗
OKVS-based PSI [18–20]	✗	✗	✓§	✓	✗	N/A	✗
Unbalanced PSI [22]	✗	✗	✓§	✓	✗	N/A	✗
Silent-Share (Ours)	✓	✓**	✓	✓	✓	✓	✓

✓: supported; ✗: not supported; N/A: not applicable.
T1: Bilateral access control; **T2:** Hidden policy (up to cardinality); **T3:** Threshold matching; **T4:** Unbalanced-friendly receiver cost; **T5:** Match-bound key release; **T6:** Receiver-side pairing cost independent of $|\mathcal{P}|$ and d; **T7:** Refreshable representation.
*Fuzzy IB-ME has partial policy hiding and reveals receiver attributes.
§Threshold PSI variants support threshold intersection but leak the intersection; hence they do not hide the policy.
**Silent-Share hides policies from unauthorized parties, but active participants learn their own access decision; see Section VI-G.

PSI and OKVS-based protocols. PSI protocols, including OKVS-based constructions [18–20], compute set intersections efficiently but are inherently interactive and do not deliver an encrypted payload. Threshold PSI variants [25, 29] still leak the intersection. Unbalanced PSI protocols [22–24] reduce dependence on the larger set, but the output is a set relation, not a decryption key. Silent-Share uses a group-valued OKVS non-interactively and masks the shares, preventing unauthorized membership tests and directly binding the matching witness to key recovery.

Why direct composition fails. A direct composition of PSI with encryption would either reveal the match bit, require additional rounds, or reintroduce work proportional to the larger set. In contrast, Silent-Share embeds the matching witness directly into the algebraic structure used for key recovery: the GOKVS hides the policy attributes, pairing cancellation extracts candidate shares using only receiver-held attributes, and threshold interpolation determines whether those shares jointly reconstruct the encapsulated secret. This is the key distinction captured in Table I.

III. Preliminaries

Notation. Let λ be the security parameter and p the prime order of the underlying groups. For a set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality; $|\mathcal{P}|$ and $|\mathcal{A}|$ denote the number of attributes

in a hidden policy and held by a user, respectively. Group operations are written multiplicatively: for a group element X and $x \in \mathbb{F}_p$, X^x denotes exponentiation. In the GOKVS abstraction, n denotes the number of encoded key-value pairs ($n = |\mathcal{P}|$ in our protocol), ρ the sparse row degree, α the storage load factor, and $M = \lceil \alpha |\mathcal{P}| \rceil$ the representation length.

A. Bilinear Groups

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of prime order p, equipped with an efficiently computable asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators. The pairing satisfies bilinearity and non-degeneracy: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{F}_p$, and $e(g_1, g_2)$ generates $\mathbb{G}_T$.

All group operations and pairing evaluations are assumed to be efficiently computable [33].

B. Random Oracle Model and KDF Instantiation

We model $H_{\mathsf{K}} : \mathbb{G}_T \to \{0,1\}^\lambda, H_{\mathsf{V}} : \mathbb{G}_T \to \{0,1\}^\lambda$, and $H_{\mathsf{x}} : \{0,1\}^* \to \mathbb{F}_p^*$ as independent random oracles. In the practical implementation, the independence of H_{K} and H_{V} is required because H_{K} derives the encapsulated key mask, whereas H_{V} authenticates the reconstructed pairing secret. Reusing the same function for both purposes would expose the XOR relation between the two values.

In the bilateral protocol, the final data key is derived using a key derivation function KDF. We instantiate KDF with HKDF [34] using SHA-256. In the security analysis, KDF is modeled as a random oracle; this is standard for protocols that use HKDF in practice.

C. Symmetric Encryption with AES-GCM

We use AES-GCM [35] as the symmetric encryption layer. For a symmetric key $K \in \{0,1\}^\lambda$, associated data A, a fresh nonce N, and plaintext M, encryption outputs

$$(C, T) \leftarrow \mathsf{AES\text{-}GCM.Enc}_K(M, A, N),$$

where C is the ciphertext and T is the authentication tag. Decryption $\mathsf{AES\text{-}GCM.Dec}_K(C, T, A, N)$ returns M or $\perp$ if authentication fails. AES-GCM provides confidentiality and ciphertext integrity. We use the notation $\tau = \mathsf{AES\text{-}GCM.Enc}_K(M, A, N)$ for the combined ciphertext/tag.

D. Hidden Common Exponent (HCE) Assumption

We define the HCE assumption as an adaptive experiment with a key-generation oracle. Let $H_{\mathsf{x}} : \{0,1\}^* \to \mathbb{F}_p^*$ be a random oracle. The HCE experiment for threshold d is parameterized by an attribute universe $\mathcal{U}$ and proceeds as follows.

a) Setup. The challenger samples $y \leftarrow \mathbb{F}_p^*$ and, for every $a \in \mathcal{U}$, samples $t_a \leftarrow \mathbb{F}_p^*$. It computes

$$Y = e(g_1, g_2)^y, \qquad T_a = g_2^{t_a},$$

and sets $msk = (y, \{t_a\}_{a \in \mathcal{U}})$,

$$pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, Y, \{T_a\}_{a \in \mathcal{U}}, H_{\mathsf{x}}).$$

The challenger gives pp to the adversary $\mathcal{B}$.

b) Adaptive key-generation queries. $\mathcal{B}$ may adaptively query a key-generation oracle $\mathsf{KeyGen}(msk, \cdot, \cdot)$ on identities u with attribute sets $\mathcal{A}_u$. Each identity can be queried at most once. For a query $(u, \mathcal{A}_u)$, the challenger samples a random polynomial

$$q_u(X) = y + \sum_{j=1}^{d-1} c_{u,j} X^j \in \mathbb{F}_p[X],$$

and, for every $a \in \mathcal{A}_u$, computes

$$x_a = H_{\mathsf{x}}(a), \qquad D_{u,a}^{(1)} = g_1^{q_u(x_a)/t_a}, \qquad D_{u,a}^{(2)} = g_1^{q_u(x_a)}.$$

It returns $SK_u = \left\{(a, x_a, D_{u,a}^{(1)}, D_{u,a}^{(2)})\right\}_{a \in \mathcal{A}_u}$.

c) Challenge. $\mathcal{B}$ outputs a challenge policy $\mathcal{P} \subseteq \mathcal{U}$ such that for every previously queried identity u, $|\mathcal{A}_u \cap \mathcal{P}| < d$. The challenger chooses $b \leftarrow \{0,1\}$.

If $b = 0$, it samples $s \leftarrow \mathbb{F}_p^*$ and, for every $a \in \mathcal{P}$, samples $r_a \leftarrow \mathbb{F}_p^*$, then returns

$$W = Y^s, \qquad X_a^{(1)} = T_a^{s+r_a}, \qquad X_a^{(2)} = g_2^{r_a}.$$

If $b = 1$, it samples $W \leftarrow \mathbb{G}_T$ uniformly and, for every $a \in \mathcal{P}$, samples independent uniform $X_a^{(1)}, X_a^{(2)} \leftarrow \mathbb{G}_2$.

The challenger returns W, $\{(X_a^{(1)}, X_a^{(2)})\}_{a \in \mathcal{P}}$ to $\mathcal{B}$.

d) Post-challenge key queries. $\mathcal{B}$ may continue querying the key-generation oracle, subject to the restriction that every queried identity u satisfies $|\mathcal{A}_u \cap \mathcal{P}| < d$.

e) Output. $\mathcal{B}$ outputs a bit b'. Its advantage is $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{HCE}_d}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Definition 1 (Hidden Common Exponent Assumption)**.** *The* Hidden Common Exponent assumption *with threshold d holds if for every PPT adversary $\mathcal{B}$, the advantage $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{HCE}_d}(\lambda)$ is negligible.*

E. Group-Valued Oblivious Key-Value Store

We use a group-valued oblivious key-value store (GOKVS) as the policy representation layer. The abstraction is motivated by existing OKVS-based private matching techniques [18, 19, 22], but differs in that the stored values are group elements rather than field elements.

Let $\mathbb{G}$ be a cyclic group of prime order p. A GOKVS consists of two algorithms: $\mathsf{Encode}_{\mathbb{G}}$ and $\mathsf{Decode}_{\mathbb{G}}$.

Given a collection of key-value pairs

$$\mathcal{E} = \{(a_i, E_i)\}_{i=1}^n, \qquad E_i \in \mathbb{G},$$

the encoder outputs a vector $\mathbf{D} = (D_1, \ldots, D_M) \in \mathbb{G}^M$. The decoder takes a representation $\mathbf{D}$ and an attribute a and returns $\widehat{E}_a = \mathsf{Decode}_{\mathbb{G}}(\mathbf{D}, a)$.

We require the following properties.

Exact decoding. For every encoded pair (a_i, E_i), $\mathsf{Decode}_{\mathbb{G}}(\mathsf{Encode}_{\mathbb{G}}(\mathcal{E}), a_i) = E_i$.

Uniformity. Suppose that the encoded values $E_1, \ldots, E_n$ are independently sampled uniformly from $\mathbb{G}$ and that the encoder uses independent uniform internal randomness. If the encoding algorithm succeeds, then the resulting representation satisfies $\mathbf{D} \overset{d}{=} \mathbb{G}^M$. This property is used in our policy-hiding

proof to replace the encoded policy-dependent group elements by a policy-independent random representation.

The GOKVS abstraction does not prescribe a particular row representation or solving method. In particular, dense linear-algebraic constructions (as the baseline in Appendix B.) and sparse graph-based constructions (the detailed peeling-based encoding and decoding procedure is given in Appendix A) can both instantiate the interface. This separation allows the cryptographic construction and security argument to remain independent of a particular GOKVS realization.

F. Sparse-GOKVS Instantiation

We instantiate the GOKVS interface with a sparse, peeling-based construction designed for large hidden policies.

For each attribute a, a row-selection hash

$$H_{\mathsf{row}} : \{0,1\}^* \times [M]^\rho$$

with output interpreted as an ordered tuple of ρ distinct indices from $[M]$ is used to select the ρ buckets associated with a. The hash H_{row} is modeled as a random oracle in the security analysis and is independent of H_{x}, H_{K}, and H_{V}. The representation length is $M = \lceil \alpha n \rceil$, where ρ is a small constant and $\alpha > 1$ is the storage load factor. The resulting incidence structure can be viewed as a sparse ρ-uniform hypergraph: each policy attribute is associated with ρ buckets in the GOKVS representation.

The encoder first constructs the sparse incidence graph and performs peeling from buckets of degree one. If all policy attributes can be peeled, the encoder assigns the group values in reverse peeling order, while unassigned buckets are filled with independent random group elements. The decoder evaluates only the ρ buckets associated with its queried attribute.

For the concrete implementation evaluated in this paper, we use $\rho = 3, \alpha = 1.25$. Thus the representation contains $M = \lceil 1.25|\mathcal{P}| \rceil$ group elements. The empirical parameter selection is reported in Appendix F-B.

The sparse realization has expected linear encoding complexity in the number of policy attributes and constant decoding complexity per queried attribute, namely $T_{\mathsf{enc}} = O(\rho n)$ up to peeling retries and group-operation costs, and $T_{\mathsf{dec}} = O(\rho) = O(1)$. The latter property is important for Silent-Share because the requester probes the hidden policy only using its own attributes. Consequently, the number of group operations required for one attribute lookup does not grow with the policy size. Peeling failure is bounded in Section VI-A.

G. Policy-Based Key Encapsulation Mechanism

A policy-based key encapsulation mechanism (PB-KEM) is a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ with the following syntax.

- $\mathsf{Setup}(1^\lambda, \mathcal{U}, d) \to (\mathsf{pp}, \mathsf{msk})$ generates public parameters and master secret for attribute universe $\mathcal{U}$ and threshold d.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{uid}, \mathcal{A}_{\mathsf{uid}}) \to \mathsf{SK}_{\mathsf{uid}}$ generates a secret key for a registered identity with attribute set $\mathcal{A}_{\mathsf{uid}}$.
- $\mathsf{Encap}(\mathsf{pp}, \mathcal{P}) \to (\mathsf{CT}, \kappa)$ encapsulates a symmetric key under hidden policy $\mathcal{P}$.
- $\mathsf{Decap}(\mathsf{pp}, \mathsf{SK}_{\mathsf{uid}}, \mathsf{CT}) \to \kappa$ or $\perp$ recovers the encapsulated key.

Correctness requires that an honestly generated ciphertext under policy $\mathcal{P}$ can be decapsulated whenever $|\mathcal{A}_{\mathsf{uid}} \cap \mathcal{P}| \geq d$. The one-sided PB-KEM is the basic cryptographic building block of Silent-Share. The bilateral protocol instantiates two independent PB-KEM instances so that each party can enforce a private policy over the other party's attributes.

IV. System Model and Protocol Overview

We consider two parties: a *Requester* R and a *Data Owner* O. Each party holds two private objects: $R : (\mathcal{A}_R, \mathcal{P}_R)$, and $O : (\mathcal{A}_O, \mathcal{P}_O)$, where

- $\mathcal{A}_R \subseteq \mathcal{U}$ is the requester's own attribute set,
- $\mathcal{P}_R \subseteq \mathcal{U}$ is the requester's policy over the owner's attributes,
- $\mathcal{A}_O \subseteq \mathcal{U}$ is the owner's own attribute set,
- $\mathcal{P}_O \subseteq \mathcal{U}$ is the owner's policy over the requester's attributes.

The system is parameterized by a common threshold $d \in [1, |\mathcal{U}|]$. The *bilateral authorization predicate* is defined as

$$\mathrm{Acc}(R, O) = \mathbf{1}\left[|\mathcal{A}_R \cap \mathcal{P}_O| \geq d\right] \cdot \mathbf{1}\left[|\mathcal{A}_O \cap \mathcal{P}_R| \geq d\right].$$

That is, access is granted if and only if both parties satisfy the same threshold policy over the other party's attributes.

A. Protocol Overview

Silent-Share is built from two one-sided instances of a *hidden threshold KEM* primitive that share the same public parameters. Each instance handles one direction of the bilateral predicate.

- **Owner-side instance** $O \to R$: the owner generates

  $$(\mathsf{CT}_O, k_O) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_O).$$

  Only a requester with attributes $\mathcal{A}_R$ satisfying $|\mathcal{A}_R \cap \mathcal{P}_O| \geq d$ can recover k_O. This instance enforces the owner's policy over the requester's attributes.
- **Requester-side instance** $R \to O$: the requester generates

  $$(\mathsf{CT}_R, k_R) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_R).$$

  Only an owner with attributes $\mathcal{A}_O$ satisfying $|\mathcal{A}_O \cap \mathcal{P}_R| \geq d$ can recover k_R. This instance enforces the requester's policy over the owner's attributes.

After both parties have performed their local decapsulations, they combine the recovered secrets to derive the final data key. To ensure that the requester obtains the final key if and only if *both* predicates hold, the owner samples a fresh nonce N and sends the requester a symmetric encryption of k_R under k_O: $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$, where ctx is public context information and N is transmitted with the ciphertext. The requester first recovers k_O from CT_O, then decrypts τ using the same ctx and N to obtain k_R, and finally derives

$$K_{\mathrm{data}} = \mathsf{KDF}(k_O, k_R, \mathsf{ctx}),$$

where KDF is instantiated with HKDF-SHA256 as described in Section III-B.

The message flow is thus two-message, one round:

$$R \to O : \mathsf{CT}_R, \qquad O \to R : (\mathsf{CT}_O, \tau, N).$$

We emphasize that the underlying one-sided primitive is non-interactive. The bilateral composition requires one round of communication because the requester's policy $\mathcal{P}_R$ must be encapsulated into a ciphertext that the owner can decrypt.

B. Threat Model

We consider an adaptive key-query model with non-repeatable user registration: the adversary may obtain secret keys for arbitrary users, each registered at most once with a fixed attribute set. The adversary may corrupt multiple users and pool their keys. Collusion across different users is allowed and explicitly considered.

The adversary is probabilistic polynomial-time and has access to the public parameters, random oracles, and the key-generation oracle. The scheme should remain secure even if the adversary queries keys adaptively, before and after seeing the challenge ciphertext(s), subject to the authorization restrictions defined in the security experiments below.

C. Security Definitions

We define security separately for the one-sided primitive and the bilateral composition.

a) One-sided PB-KEM IND-CPA.: Let $\mathcal{B}$ be a PPT adversary. The experiment $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{kem-ind-cpa}}(\lambda)$ proceeds as follows.

a) The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{U}, d)$ and gives pp to $\mathcal{B}$.
b) $\mathcal{B}$ may adaptively query a key-generation oracle $\mathcal{O}_{\mathsf{KG}}$ on identities u with attribute sets $\mathcal{A}_u$. Each identity can be queried at most once.
c) $\mathcal{B}$ outputs a challenge policy $\mathcal{P}$ such that for every queried identity u, $|\mathcal{A}_u \cap \mathcal{P}| < d$.
d) The challenger samples $(\mathsf{CT}, \kappa_0) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P})$, samples $\kappa_1 \leftarrow \{0,1\}^\lambda$, chooses $b \leftarrow \{0,1\}$, and returns (CT, κ_b) to $\mathcal{B}$.
e) $\mathcal{B}$ may continue querying $\mathcal{O}_{\mathsf{KG}}$ subject to the same per-user restriction.
f) $\mathcal{B}$ outputs a bit b'.

The advantage is $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{kem-ind-cpa}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$. A PB-KEM is IND-CPA secure if this advantage is negligible for every PPT adversary.

b) One-sided Policy Hiding.: Let $\mathcal{B}$ be a PPT adversary. The experiment $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{policy-hide}}(\lambda)$ proceeds as follows.

a) The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{U}, d)$ and gives pp to $\mathcal{B}$.
b) $\mathcal{B}$ may adaptively query $\mathcal{O}_{\mathsf{KG}}$ as above.
c) $\mathcal{B}$ outputs two policies $\mathcal{P}_0, \mathcal{P}_1$ with $|\mathcal{P}_0| = |\mathcal{P}_1|$, subject to the restriction that for every queried identity u, $|\mathcal{A}_u \cap \mathcal{P}_0| < d$ and $|\mathcal{A}_u \cap \mathcal{P}_1| < d$.
d) The challenger chooses $b \leftarrow \{0,1\}$, runs $(\mathsf{CT}, \kappa) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_b)$, and returns CT to $\mathcal{B}$.
e) $\mathcal{B}$ may continue querying $\mathcal{O}_{\mathsf{KG}}$ subject to the same restriction.
f) $\mathcal{B}$ outputs a bit b'.

The advantage is $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{policy-hide}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$. A PB-KEM is policy-hiding up to policy size if this advantage is negligible for every PPT adversary.

c) Bilateral KEM experiment.: Let $\mathcal{A}$ be a PPT adversary. The experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bilateral-kem}}(\lambda)$ proceeds as follows.

a) The challenger runs $\mathsf{Setup}(1^\lambda, \mathcal{U}, d)$ and gives pp to $\mathcal{A}$.
b) $\mathcal{A}$ may adaptively query $\mathcal{O}_{\mathsf{KG}}$ on identities u, each at most once.
c) $\mathcal{A}$ outputs two challenge policies $\mathcal{P}_O, \mathcal{P}_R$. Let $\mathcal{A}_{\mathcal{A}}$ be the union of all attributes held by identities corrupted by $\mathcal{A}$. The challenge is invalid if

$$|\mathcal{A}_{\mathcal{A}} \cap \mathcal{P}_O| \geq d.$$

This condition is sufficient to exclude trivial wins: an adversary that satisfies the owner-side policy can recover k_O and then decrypt $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$, thereby obtaining both keys regardless of its relation to $\mathcal{P}_R$.
d) The challenger runs the bilateral protocol with $\mathcal{P}_O, \mathcal{P}_R$, obtaining K_{data} and the transcript $(\mathsf{CT}_O, \mathsf{CT}_R, \tau, N)$. It samples $K_0 = K_{\mathrm{data}}$, $K_1 \leftarrow \{0,1\}^\lambda$, chooses $b \leftarrow \{0,1\}$, and returns the transcript together with K_b to $\mathcal{A}$.
e) $\mathcal{A}$ may continue corrupting identities, subject to the same invalid-challenge condition.
f) $\mathcal{A}$ outputs a bit b'.

The advantage is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bilateral-kem}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

d) Bilateral KEM Security.: In the bilateral setting, the adversary can corrupt any set of users and may also act as either the requester or the owner, or both, subject to restrictions. We require that the final data key K_{data} is indistinguishable from random unless the adversary trivially has enough attributes to satisfy both sides of the predicate. Formally, for any adversary that has not obtained keys that collectively allow reconstruction of k_O and k_R for the challenge policies, the advantage in distinguishing K_{data} from random is negligible.

e) Bilateral Policy Hiding.: For an external observer that does not actively participate in the protocol, the two ciphertexts CT_O and CT_R hide the policies up to their cardinalities, under the one-sided policy-hiding guarantees.

For an active but *unauthorized* participant, the following predicate leakage is unavoidable. Let $\mathcal{A}_R$ be the requester's attribute set and $\mathcal{A}_O$ be the owner's attribute set. The requester can always learn

$$L_O(\mathcal{P}_O, \mathcal{A}_R) = \mathbf{1}[|\mathcal{A}_R \cap \mathcal{P}_O| \geq d],$$

by attempting to decapsulate CT_O. Similarly, the owner can always learn

$$L_R(\mathcal{P}_R, \mathcal{A}_O) = \mathbf{1}[|\mathcal{A}_O \cap \mathcal{P}_R| \geq d].$$

Thus, for an active but unauthorized participant, the bilateral policy-hiding guarantee is that nothing beyond

$$L(\mathcal{P}_O, \mathcal{P}_R, \mathcal{A}_R, \mathcal{A}_O) = (|\mathcal{P}_O|, |\mathcal{P}_R|, L_O(\mathcal{P}_O, \mathcal{A}_R), L_R(\mathcal{P}_R, \mathcal{A}_O))$$

is leaked by the ciphertexts and the protocol transcript.

For an authorized participant, a successful decapsulation inherently yields a matching d-subset and therefore reveals

those policy attributes. This information is outside the policy-hiding security model, because the policy-hiding experiment restricts the adversary to identities that do not satisfy the challenge policy.

V. Construction of Silent-Share

Figure 1 gives the compact protocol specification. The construction instantiates the PB-KEM syntax of Section III-G and combines it with the Sparse-GOKVS realization of Section III-F. Full algorithm-by-algorithm details and the correctness proof are given in Appendix D.

Design rationale. The GOKVS layer hides the policy attributes, while the pairing layer converts a successfully decoded policy entry into a threshold share bound to the decryptor's secret key. For an authorized attribute a, decoding followed by pairing cancellation yields $Z_a = e(g_1, g_2)^{s q_{\mathsf{uid}}(x_a)}$, where q_{uid} is a user-specific polynomial satisfying $q_{\mathsf{uid}}(0) = y$. Any d matching attributes from the same user reconstruct the common secret Y^s by Lagrange interpolation. Fewer than d matching attributes, or d attributes drawn from different users, leave the corresponding linear combination statistically masked by the random polynomials q_{uid}. This gives the three-state behavior summarized in Table II.

Bilateral access control. Silent-Share realizes bilateral threshold authorization by composing two independent one-sided instances of the same PB-KEM. The two instances use the same public parameters and the same threshold d, but different policies.

In the owner-side direction, the owner generates $(\mathsf{CT}_O, k_O) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_O)$. Only a requester whose attribute set $\mathcal{A}_R$ satisfies $|\mathcal{A}_R \cap \mathcal{P}_O| \geq d$ can recover k_O from CT_O. In the requester-side direction, the requester generates $(\mathsf{CT}_R, k_R) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_R)$. Only an owner whose attribute set $\mathcal{A}_O$ satisfies $|\mathcal{A}_O \cap \mathcal{P}_R| \geq d$ can recover k_R from CT_R.

To cryptographically enforce that the requester obtains the final data key if and only if *both* predicates hold, the owner samples a fresh nonce N, computes $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$, and sends (CT_O, τ, N) to the requester. The owner can compute τ only if it has already recovered k_R from CT_R, which is possible exactly when the owner satisfies the requester's policy. The requester can decrypt τ only if it has recovered k_O from CT_O, which is possible exactly when the requester satisfies the owner's policy. Thus successful recovery of k_R by the requester simultaneously proves both directional matches. Both parties then derive $K_{\mathrm{data}} = \mathsf{HKDF}(k_O \| k_R, \mathsf{ctx}, \lambda)$.

This composition is illustrated in Phase 3 of Figure 1. It requires one round of communication: the requester first sends CT_R to the owner, and the owner replies with (CT_O, τ, N). The one-sided primitive itself remains non-interactive.

VI. Security Analysis

We analyze the one-sided PB-KEM and the bilateral composition under the security definitions of Section IV-C. Full proofs appear in Appendix E.

A. Security-Relevant Bad Events

Let Q_{x} denote the total number of distinct attributes appearing in all key-generation and challenge-policy queries, and let Q_{row} denote the total number of attributes on which H_{row} is evaluated during encoding and decoding queries. Define $\mathsf{Bad}_{\mathsf{x}}$ as the event that $H_{\mathsf{x}}(a) = H_{\mathsf{x}}(a')$ for two distinct attributes $a \neq a'$. Since H_{x} is modeled as a random oracle into $\mathbb{F}_p^*$,

$$\Pr[\mathsf{Bad}_{\mathsf{x}}] \leq \frac{\binom{Q_{\mathsf{x}}}{2}}{p-1}.$$

Throughout the security proof we condition on $\neg\mathsf{Bad}_{\mathsf{x}}$. All interpolation points used by a single key are distinct.

The Sparse-GOKVS encoder may fail if peeling does not succeed for a particular hash assignment. Let $\mathsf{Bad}_{\mathsf{peel}}$ denote the event that the encoder fails for the challenge policy. For the chosen parameters $\rho = 3$ and $\alpha = 1.25$, random 3-uniform hypergraph theory suggests that peeling succeeds with overwhelming probability as $|\mathcal{P}|$ grows. Our experiments in Appendix F-B confirm success for all tested policy sizes. In the security analysis we treat $\Pr[\mathsf{Bad}_{\mathsf{peel}}]$ as an explicit implementation failure probability, which is negligible when $|\mathcal{P}|$ is sufficiently large. This is a negligible function of $|\mathcal{P}|$. If the scheme is instantiated with $|\mathcal{P}| = \Omega(\lambda)$, then it is also negligible in the security parameter; otherwise it is kept as an explicit additive term in the reduction.

A final negligible event concerns accidental collisions in the verification hash H_{V}. If a decryptor evaluates at most Q_{cand} candidate subsets, the probability that an incorrect candidate $\Omega' \neq Y^s$ satisfies $H_{\mathsf{V}}(\Omega') = H_{\mathsf{V}}(Y^s)$ is at most

$$\frac{Q_{\mathsf{cand}}}{2^\lambda}.$$

B. One-Sided KEM Confidentiality

Let q_K, q_V, and q_x denote the number of queries made by the adversary to H_{K}, H_{V}, and H_{x}, respectively. We assume $q = \max\{q_K, q_V, q_x\}$.

Theorem 2 (One-Sided KEM-IND-CPA Security)**.** *Let $\mathcal{A}$ be a PPT adversary against the one-sided Silent-Share PB-KEM. Under the HCE_d assumption in the random oracle model, there exists a PPT adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{kem-ind-cpa}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{HCE}_d}_{\mathcal{B}}(\lambda) + \frac{q_K + q_V}{p-1} + \frac{1}{2^\lambda} + \Pr[\mathsf{Bad}_{\mathsf{x}}] + \Pr[\mathsf{Bad}_{\mathsf{peel}}].$$

Proof sketch. The reduction $\mathcal{B}$ receives the HCE public parameters and forwards adaptive key-generation queries to the HCE challenger's KeyGen oracle. For the challenge policy $\mathcal{P}$, it submits $\mathcal{P}$ and receives either the real HCE transcript or a random transcript.

In the real branch, encoding the received group elements produces a ciphertext distributed exactly as in the real encapsulation algorithm. In the random branch, Lemma 9 ensures that both GOKVS vectors are uniform and independent of the challenge policy. Conditioned on the random target W not being previously queried to H_{K} or H_{V}, the values E_0 and C are uniform and independent of the challenge key. Thus the adversary's view is independent of the internal challenge bit.

Phase 0: System Setup & Key Generation (Authority)

0a Choose bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with generators g_1, g_2 and pairing e. Sample master exponent $y \leftarrow \mathbb{F}_p^*$; for every attribute $a \in \mathcal{U}$, sample $t_a \leftarrow \mathbb{F}_p^*$.

0b Compute $Y = e(g_1, g_2)^y$ and $T_a = g_2^{t_a}$. Set public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, Y, \{T_a\}, H_{\mathsf{K}}, H_{\mathsf{V}}, H_{\mathsf{x}})$, and master secret $\mathsf{msk} = (y, \{t_a\})$.

0c For user uid with attribute set $\mathcal{A}_{\mathsf{uid}}$, sample polynomial $q_{\mathsf{uid}}(X) = y + \sum_{j=1}^{d-1} c_{\mathsf{uid},j} X^j$. For every $a \in \mathcal{A}_{\mathsf{uid}}$, let $x_a = H_{\mathsf{x}}(a)$ and compute $D^{(1)}_{\mathsf{uid},a} = g_1^{q_{\mathsf{uid}}(x_a)/t_a}$, $D^{(2)}_{\mathsf{uid},a} = g_1^{q_{\mathsf{uid}}(x_a)}$. Return $\mathsf{SK}_{\mathsf{uid}} = \{(a, x_a, D^{(1)}_{\mathsf{uid},a}, D^{(2)}_{\mathsf{uid},a})\}$.

Phase 1: One-Sided Encapsulation (Sender, offline)

(1) Choose $\kappa \leftarrow \{0,1\}^\lambda$ and $s \leftarrow \mathbb{F}_p^*$. Compute $Y^s = e(g_1, g_2)^{ys}$.

(2) Set $E_0 = \kappa \oplus H_{\mathsf{K}}(Y^s)$ and $C = H_{\mathsf{V}}(Y^s)$.

(3) For every policy attribute $\pi \in \mathcal{P}$, sample $r_\pi \leftarrow \mathbb{F}_p^*$ and compute $E^{(1)}_\pi = T_\pi^{s+r_\pi}$, $E^{(2)}_\pi = g_2^{r_\pi}$.

(4) Encode $\mathbf{D}^{(1)} \leftarrow \mathsf{Encode}_{\mathbb{G}_2}(\{(\pi, E^{(1)}_\pi)\})$ and $\mathbf{D}^{(2)} \leftarrow \mathsf{Encode}_{\mathbb{G}_2}(\{(\pi, E^{(2)}_\pi)\})$ using Sparse-GOKVS with the same salt and identical row positions, but with independent randomness for all unassigned buckets in the two encodings.

(5) Output $\mathsf{CT} = (E_0, C, \mathbf{D}^{(1)}, \mathbf{D}^{(2)})$.

Phase 2: One-Sided Decapsulation (Receiver, online)

(6) For each $a \in \mathcal{A}_{\mathsf{uid}}$, decode $\widehat{E}^{(1)}_a = \mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}^{(1)}, a)$ and $\widehat{E}^{(2)}_a = \mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}^{(2)}, a)$.

(7) Compute $Z_a = e(D^{(1)}_{\mathsf{uid},a}, \widehat{E}^{(1)}_a) \cdot e(D^{(2)}_{\mathsf{uid},a}, \widehat{E}^{(2)}_a)^{-1}$. For matching attribute a, this equals $e(g_1, g_2)^{s q_{\mathsf{uid}}(x_a)}$.

(8) For each candidate d-subset $\mathcal{S} \subseteq \mathcal{A}_{\mathsf{uid}}$, compute $\Omega_{\mathcal{S}} = \prod_{a \in \mathcal{S}} Z_a^{\Delta_{a,\mathcal{S}}(0)}$. Accept if $H_{\mathsf{V}}(\Omega_{\mathcal{S}}) = C$; then $\kappa = E_0 \oplus H_{\mathsf{K}}(\Omega_{\mathcal{S}})$.

Phase 3: Bilateral Composition (Online, one round)

(9) Requester R generates $(\mathsf{CT}_R, k_R) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_R)$ and sends CT_R to owner O.

(10) Owner O generates $(\mathsf{CT}_O, k_O) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_O)$; recovers k_R from CT_R; defines $\mathsf{ctx} = H(\mathsf{domain}, \mathsf{CT}_R, \mathsf{CT}_O, \mathsf{sid})$, where sid is a session identifier; samples a fresh nonce N; computes $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$; sends (CT_O, τ, N) to R.

(11) Requester recovers k_O from CT_O, recomputes ctx from the same transcript, then decrypts $k_R = \mathsf{AES\text{-}GCM.Dec}_{k_O}(\tau, \mathsf{ctx}, N)$; both parties derive $K_{\mathrm{data}} = \mathsf{HKDF}(k_O \| k_R, \mathsf{ctx})$.

Correctness. For any d matching attributes from the same user, exact Sparse-GOKVS decoding yields $Z_a = e(g_1, g_2)^{s q_u(x_a)}$; Lagrange interpolation reconstructs Y^s, so $H_{\mathsf{V}}(Y^s) = C$ and $E_0 \oplus H_{\mathsf{K}}(Y^s) = \kappa$.

Fig. 1: Compact protocol specification of Silent-Share. Full algorithm-by-algorithm details appear in Appendix D.

The full reduction and event analysis appear in Appendix E-C. □

C. One-Sided Policy Hiding

Theorem 3 (One-Sided Policy Hiding)**.** *Under HCE_d in the random oracle model, the one-sided Silent-Share ciphertext hides the policy up to its cardinality. Formally, for every PPT adversary $\mathcal{A}$ making at most q_K, q_V, q_x queries to the respective random oracles, there exist PPT adversaries $\mathcal{B}_0, \mathcal{B}_1$ such that*

$$\mathsf{Adv}^{\mathsf{policy-hide}}_{\mathcal{A}} \le \mathsf{Adv}^{\mathsf{HCE}_d}_{\mathcal{B}_0} + \mathsf{Adv}^{\mathsf{HCE}_d}_{\mathcal{B}_1} + 2 \cdot \frac{q_K + q_V}{p - 1} + 2\Pr[\mathsf{Bad}_{\mathsf{x}}] + 2\Pr[\mathsf{Bad}_{\mathsf{peel}}].$$

Proof sketch. For each challenge policy $\mathcal{P}_b$, we construct an HCE adversary that produces a ciphertext from either the real HCE transcript or a random transcript. In the random branch, the GOKVS uniformity lemma and random oracle programming make the entire ciphertext independent of $\mathcal{P}_b$. A triangle inequality between the two hybrids gives the claim. In the random branch, Lemma 9 ensures that both GOKVS vectors are uniform and independent of the policy. Consequently, for any candidate subset, the reconstructed value is uniformly distributed in $\mathbb{G}_T$ and independent of Y^s. The verification equation thus provides no additional distinguishing signal beyond the random ciphertext distribution. The full proof appears in Appendix E-D. □

D. Bilateral KEM Security

Recall that the bilateral protocol derives $K_{\mathrm{data}} = \mathsf{HKDF}(k_O \| k_R, \mathsf{ctx})$, where k_O is encapsulated under $\mathcal{P}_O$ and k_R is encapsulated under $\mathcal{P}_R$. Two one-sided instances are independent. The owner samples a fresh nonce N and sends $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$ together with N.

Theorem 4 (Bilateral KEM Security)**.** *Assume the one-sided Silent-Share PB-KEM is IND-CPA secure, AES-GCM is semantically secure, and HKDF is modeled as a random oracle. Then the bilateral protocol is KEM-secure: for any PPT adversary $\mathcal{A}$ that cannot trivially reconstruct both k_O and k_R for the challenge policies, the advantage in distinguishing K_{data} from a random string is negligible. Specifically,*

$$\mathsf{Adv}^{\mathsf{bilateral-kem}}_{\mathcal{A}} \le \mathsf{Adv}^{\mathsf{kem-ind-cpa}}_1 + \mathsf{Adv}^{\mathsf{kem-ind-cpa}}_2 + \mathsf{Adv}_{\mathsf{AES\text{-}GCM}} + \mathsf{negl}(\lambda).$$

Throughout this hybrid proof, we restrict attention to adversaries that do not satisfy the owner-side challenge policy $\mathcal{P}_O$, i.e.,

$$|\mathcal{A}_{\mathcal{A}} \cap \mathcal{P}_O| < d.$$

By the definition of the bilateral KEM experiment, challenges with $|\mathcal{A}_{\mathcal{A}} \cap \mathcal{P}_O| \geq d$ are excluded. Under this restriction the adversary cannot recover k_O from CT_O, so the owner-side one-sided IND-CPA replacement is valid. The proof does not rely on the adversary being unable to recover k_R alone; the requester-side key is handled separately in a later hybrid transition.

The two one-sided instances share the same public parameters but use independent encapsulation randomness and independent keys. The hybrid argument below applies each one-sided IND-CPA reduction sequentially; since the reductions do not reuse randomness across the two instances, the shared setup does not affect the simulation.

Proof sketch. The proof uses the following hybrid order.
Game 0. Real bilateral protocol.
Game 1. Replace k_O with k_O^*, while keeping k_R real. This transition is justified by the IND-CPA security of the owner-side one-sided PB-KEM.
Game 2. Replace the AES-GCM plaintext with a random string of the same length. This transition is justified by AES-GCM semantic security under the uniform key k_O^*.
Game 3. Replace k_R with k_R^*. This transition is justified by the IND-CPA security of the requester-side one-sided PB-KEM.
Final. In Game 3, k_O^*, k_R^* are uniform and independent, and τ is independent of both. Because HKDF is modeled as a random oracle, the final data key is indistinguishable from random unless the adversary queries the KDF on exactly $(k_O^*, k_R^*, \mathsf{ctx})$, which occurs with negligible probability.

The complete hybrid proof appears in Appendix E-E. □

E. Bilateral Policy Hiding

a) Bilateral policy-hiding experiment.: Let $\mathcal{A}$ be a PPT adversary. The challenger runs Setup, gives pp to $\mathcal{A}$, and allows adaptive key-generation queries. $\mathcal{A}$ outputs two policy pairs $(\mathcal{P}_O^0, \mathcal{P}_R^0)$ and $(\mathcal{P}_O^1, \mathcal{P}_R^1)$ such that $|\mathcal{P}_O^0| = |\mathcal{P}_O^1|$ and $|\mathcal{P}_R^0| = |\mathcal{P}_R^1|$. If $\mathcal{A}$ acts as an active participant with attribute set $\mathcal{A}^*$, the policies must additionally satisfy

$$L_O(\mathcal{P}_O^0, \mathcal{A}^*) = L_O(\mathcal{P}_O^1, \mathcal{A}^*) \quad \text{and} \quad L_R(\mathcal{P}_R^0, \mathcal{A}^*) = L_R(\mathcal{P}_R^1, \mathcal{A}^*).$$

The challenger picks $b \leftarrow \{0,1\}$, runs the bilateral protocol with $(\mathcal{P}_O^b, \mathcal{P}_R^b)$, and returns the transcript to $\mathcal{A}$. The adversary outputs b'. The advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bilateral-ph}} = |\Pr[b' = b] - \tfrac{1}{2}|.$$

Theorem 5 (Bilateral Policy Hiding with Predicate Leakage)**.** *Assume the one-sided Silent-Share PB-KEM is policy-hiding. Then the bilateral protocol satisfies the following guarantees.*

- *For an external observer that does not actively participate, the ciphertexts $\mathsf{CT}_O, \mathsf{CT}_R$ hide the policies up to their cardinalities.*
- *For an active but unauthorized participant with attribute set $\mathcal{A}^*$, the ciphertexts leak nothing beyond the leakage function $L(\mathcal{P}_O, \mathcal{P}_R, \mathcal{A}^*) = \big(|\mathcal{P}_O|, |\mathcal{P}_R|, L_O(\mathcal{P}_O, \mathcal{A}^*), L_R(\mathcal{P}_R, \mathcal{A}^*)\big)$, where $L_O(\mathcal{P}_O, \mathcal{A}^*) = \mathbf{1}[|\mathcal{A}^* \cap \mathcal{P}_O| \geq d]$ and $L_R(\mathcal{P}_R, \mathcal{A}^*) = \mathbf{1}[|\mathcal{A}^* \cap \mathcal{P}_R| \geq d]$.*

Formally, for any PPT adversary satisfying the above experiment,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bilateral-ph}} \leq \mathsf{Adv}_{O}^{\mathsf{policy-hide}} + \mathsf{Adv}_{R}^{\mathsf{policy-hide}} + \mathsf{negl}(\lambda),$$

where the two terms denote the policy-hiding advantages of the two one-sided instances.

Proof sketch. The two ciphertexts are generated using independent encapsulation randomness under the same public parameters. By the one-sided policy-hiding theorem, each ciphertext hides its policy up to size under the corresponding leakage condition. The AES-GCM tag τ is computed from the recovered keys and does not reveal additional policy information beyond the access decision already implied by successful or failed decapsulation. A standard hybrid argument yields the bound. The complete proof appears in Appendix E-F. □

F. Collusion Resistance

The three-state algebraic structure established in Lemmas 7 and 8 prevents cross-user collusion. For any set of ℓ attributes contributed by one or more users:

- $\ell < d$: any linear combination remains statistically uniform, so the master secret y and the encapsulated key are hidden.
- $\ell = d$, all from the same user: Lagrange interpolation recovers Y^s.
- $\ell = d$, from at least two users: the mixed-user combination remains uniform, so collusion cannot pool shares to reach the threshold.

This property is summarized in Table II and is used in the security proofs. In the bilateral setting, both directions must individually resist collusion; since the two one-sided instances are independent and each enjoys the three-state property, the bilateral protocol inherits the same guarantee.

G. Security Boundary and Leakage

The policy-hiding property protects policies against adversaries whose collective matching capability remains below the threshold in each direction. The public leakage is limited to the policy sizes $|\mathcal{P}_O|$ and $|\mathcal{P}_R|$, which is inherent because the GOKVS dimension depends on the number of encoded attributes.

Access-decision leakage. Silent-Share does not hide the access decision itself. An active participant who is able to decapsulate one side learns that its own attribute set satisfies the corresponding policy. This is inherent in bilateral threshold access control: the protocol output is a key that can be verified by the participant, and a successful or failed decapsulation reveals the predicate. Our policy-hiding theorem therefore explicitly includes this predicate leakage rather than claiming only cardinality leakage.

Implementation-level leakage. The current decapsulation algorithm enumerates candidate d-subsets and stops upon first verification success. The number of candidates tried before

success may depend on the ordering of the receiver's attribute set and the position of the matching subset, and could leak timing or random-oracle query count information beyond the access bit. Our cryptographic model does not cover such implementation-level side channels. A fixed-budget decapsulation that evaluates all $\binom{|\mathcal{A}|}{d}$ candidates in a fixed order eliminates this leakage at the cost of worst-case reconstruction time.

Why we avoid public membership filters. One might be tempted to append a Bloom filter or similar structure for $\mathcal{P}$ to reduce the subset-search overhead in decapsulation [2]. However, any public data structure that supports membership queries for $\mathcal{P}$ would immediately break the strong policy-hiding property: an adversary holding a single attribute a could test whether $a \in \mathcal{P}$, even without satisfying the threshold. We therefore deliberately do not include such filters in the base construction. If a weaker privacy guarantee is acceptable, e.g., hiding policies from external observers but not from attribute holders, then a separate variant with a membership filter could be considered; we leave this to future work and do not claim it here.

VII. Implementation and Evaluation

We evaluate Silent-Share to answer three high-level questions:

- **Q1: Does the design decouple pairing-based authorization from policy size and threshold?**
- **Q2: Does the sparse GOKVS achieve its promised asymptotic gains?**
- **Q3: How does the protocol perform compared to fuzzy matchmaking encryption in unbalanced and balanced settings?**

We first describe experimental setup, then present the one-sided measurements, followed by the comparative study, and finally summarize the findings. Additional microbenchmarks and bilateral scalability results are reported in Appendix F.

A. Experimental Setup

All experiments are conducted on an Intel i7-11700 (2.50 GHz, 16 cores) with 32 GB RAM, Ubuntu 20.04.1 LTS, Python 3.6.13, Charm-Crypto 0.50, and PBC 0.5.14. The bilinear groups are instantiated with the asymmetric BN254 curve, which provides approximately 100-bit security. Although this is below the 128-bit security level often assumed in formal analyses, it is sufficient for evaluating algorithmic scaling and relative performance. A production deployment would instantiate the scheme with a 128-bit-secure curve (e.g., BLS12-381) without changing the asymptotic behavior. Random oracles H_{K}, H_{V}, and H_{x} are realized with SHA-256 using domain separation. The protocol uses the Sparse-GOKVS realization with $\rho = 3$ and $\alpha = 1.25$. Each data point is the average of 5 independent runs.

The Fuzzy IB-ME baseline [13] is ported from the original JPBC Type-A implementation to the BN254 curve used in our evaluation platform, while preserving the algebraic operations and parameter roles of the published scheme. All baseline measurements are conducted in the same hardware/software environment and include cryptographic operations and serialization. For decryption, we consider the case where the correct d-subset is known a priori (the "hint" mode), which isolates the reconstruction cost; the subset-search overhead is discussed separately in Appendix F-G.

B. Theoretical Cost Model

For a policy $\mathcal{P}$, Silent-Share encapsulation generates two group elements per attribute and encodes two Sparse-GOKVS structures. The measured encapsulation time grows linearly with $|\mathcal{P}|$ at approximately 3.1 ms per attribute (averaged over $|\mathcal{P}| = 50$ to 800), consistent with the cost of two $\mathbb{G}_2$ exponentiations and sparse encoding overhead.

Given a receiver attribute set $\mathcal{A}$, the online decapsulation cost is

$$T^{\mathrm{SS}}_{\mathrm{dec}}(|\mathcal{A}|, d) = 2|\mathcal{A}|\, T_{\mathrm{pair}} + 2\rho|\mathcal{A}|\, T_{\mathrm{mult},G_2} + N_{\mathrm{trial}}\, d\, T_{\mathrm{exp},G_T},$$

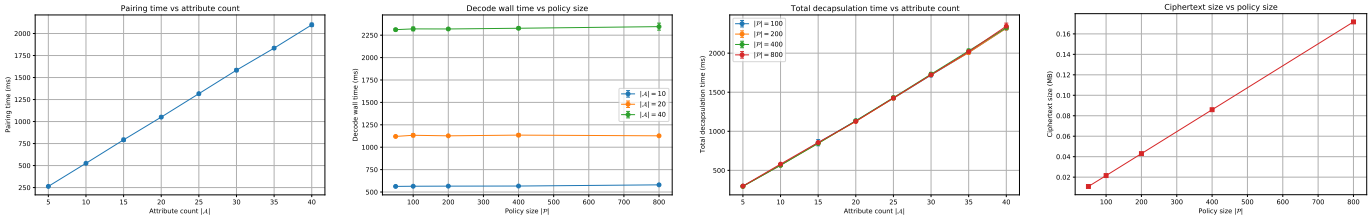
where N_{trial} is the number of threshold subsets enumerated before verification succeeds. In the best case $N_{\mathrm{trial}} = 1$; in the worst case $N_{\mathrm{trial}} = \binom{|\mathcal{A}|}{d}$. The first two terms constitute the pairing-based authorization layer and are independent of $|\mathcal{P}|$ and d. Third term is hidden-threshold reconstruction cost, which depends on $|\mathcal{A}|$, d, and is characterized separately.

Communication overhead model. Silent-Share ciphertext consists of the masked key E_0, the verification hash C, and two Sparse-GOKVS vectors $\mathbf{D}^{(1)}, \mathbf{D}^{(2)}$, each of length $M = \lceil 1.25|\mathcal{P}| \rceil$. Thus its size is $|\mathsf{CT}_{\mathrm{SS}}| = 2\lambda + 2M \cdot \ell_{\mathbb{G}_2}$, where $\ell_{\mathbb{G}_2}$ is the serialized length of one $\mathbb{G}_2$ element. The decryption key for a receiver with attribute set $\mathcal{A}$ contains two $\mathbb{G}_1$ elements per attribute, so $|\mathsf{SK}_{\mathrm{SS}}| = 2|\mathcal{A}| \cdot \ell_{\mathbb{G}_1}$. The public parameters contain one $\mathbb{G}_2$ element per attribute in the universe, so $|\mathsf{PP}| = O(|\mathcal{U}| \cdot \ell_{\mathbb{G}_2})$. These formulas give the linear dependencies observed in Figure 3(b) and (d), and explain why Silent-Share maintains smaller ciphertexts and keys than fuzzy IB-ME under the same parameters.

C. Decoupling Pairing-Based Authorization from Policy Size and Threshold

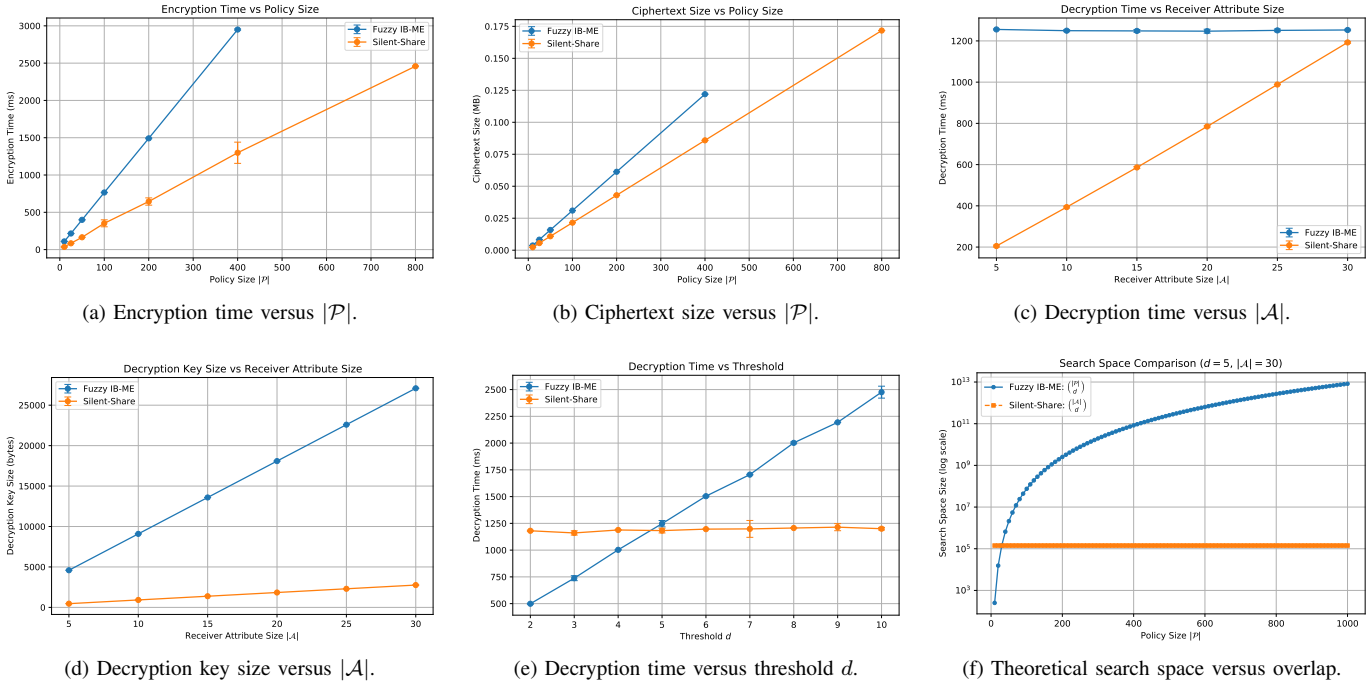
Figure 2 presents four views of the one-sided primitive. The top-left panel shows that pairing time grows linearly with $|\mathcal{A}|$, with a measured slope of approximately 39.5 ms per attribute (consistent with two pairings per attribute). The top-right panel shows that the online decapsulation time is flat in $|\mathcal{P}|$: for $|\mathcal{A}| = 10$, it stays around 394 ms across all policy sizes from 50 to 800, confirming that Sparse-GOKVS decoding and pairing costs do not grow with the policy size. The bottom-left panel overlays total decapsulation time for $|\mathcal{P}| = 100, 200, 400, 800$; the curves overlap almost perfectly, demonstrating that the online decapsulation cost is independent of $|\mathcal{P}|$ for a fixed $(|\mathcal{A}|, d)$ and under the correct-subset hint condition. For $|\mathcal{A}| = 10$, the total time is approximately 394 ms regardless of the policy size.

Finding: Silent-Share successfully decouples pairing-based authorization from the policy size and threshold, validating Q1. The hidden-threshold reconstruction cost is not policy-dependent and is reported separately in Appendix F.



(a) Pairing time versus $|\mathcal{A}|$. (b) Decoding time versus $|\mathcal{P}|$. (c) Total decapsulation versus $|\mathcal{A}|$. (d) Ciphertext size versus $|\mathcal{P}|$.

Fig. 2: One-sided Silent-Share measurements on BN254.



(a) Encryption time versus $|\mathcal{P}|$. (b) Ciphertext size versus $|\mathcal{P}|$. (c) Decryption time versus $|\mathcal{A}|$.

(d) Decryption key size versus $|\mathcal{A}|$. (e) Decryption time versus threshold d. (f) Theoretical search space versus overlap.

Fig. 3: Performance comparison between Silent-Share and fuzzy IB-ME on BN254.

D. Performance Comparison with Fuzzy IB-ME

Figure 3 summarizes the comparison across six dimensions.

Encryption and ciphertext size. Figures 3(a) and (b) show that Silent-Share encapsulation is consistently faster and produces smaller ciphertexts than fuzzy IB-ME for the same policy size. At $|\mathcal{P}| = 50$, Silent-Share requires 165 ms and 11.4 KB, whereas fuzzy IB-ME requires 399 ms and 16.6 KB. The gap widens as $|\mathcal{P}|$ grows: at $|\mathcal{P}| = 800$, Silent-Share encapsulates in about 2.46 s with a ciphertext of 180 KB. For fuzzy IB-ME, the largest directly measured policy size is $|\mathcal{P}| = 400$, where it requires 2.95 s. Across the tested range ($|\mathcal{P}| \leq 400$ for fuzzy IB-ME), Silent-Share encryption is 2–3$\times$ faster and about 30% smaller. Across the tested range, Silent-Share encryption is 2–3$\times$ faster and about 30% smaller.

Decryption and key material. Figures 3(c) and (d) compare decryption time and decryption key size as functions of the receiver's attribute set size $|\mathcal{A}|$. Silent-Share's decryption time grows linearly with $|\mathcal{A}|$, at approximately 39 ms per attribute on BN254, dominated by the two required pairings per attribute. Fuzzy IB-ME's decryption time remains nearly constant around 1.25 s because its per-candidate reconstruction cost depends on d, not $|\mathcal{A}|$. Consequently, Silent-Share is substantially faster for small $|\mathcal{A}|$: at $|\mathcal{A}| = 5$, it requires only 205 ms versus 1256 ms for fuzzy IB-ME; at $|\mathcal{A}| = 10$, 394 ms versus 1250 ms. When $|\mathcal{A}|$ reaches 30, the linear pairing cost of Silent-Share approaches that of fuzzy IB-ME, with both around 1.2 s. For larger $|\mathcal{A}|$, Silent-Share may surpass fuzzy IB-ME, which is consistent with its design targeting small attribute sets. This tradeoff is consistent with the design goal: Silent-Share targets unbalanced settings with a small attribute set on the receiver side. Figure 3(d) shows that Silent-Share's decryption key is an order of magnitude smaller: for $|\mathcal{A}| = 30$, Silent-Share's key is 2.76 KB versus 27.1 KB for fuzzy IB-ME.

Threshold sensitivity. Figure 3(e) isolates the effect of the threshold d. Fuzzy IB-ME decryption time grows linearly with d, from 0.50 s at $d = 2$ to 2.48 s at $d = 10$. In contrast, under the experimental condition that the correct d-subset is provided first, Silent-Share decapsulation time is approximately 1.19 s for the tested threshold range $d = 2, \ldots, 10$. The pairing

component is independent of d, while the hidden-threshold reconstruction component contributes d exponentiations in $\mathbb{G}_T$. For the parameter choices used here, the reconstruction component is small relative to the pairing component, so the total time appears nearly flat. The worst-case reconstruction cost, without a correct-subset hint, is reported separately in Appendix F-D. As a result, for $d \geq 6$, Silent-Share is faster, and the advantage increases with larger thresholds. This property is particularly relevant for high-threshold policies, where fuzzy IB-ME's reconstruction cost becomes prohibitive.

Search space. Figure 3(f) plots the expected number of candidate d-subsets that must be tested before finding a valid one, as a function of the overlap $|\mathcal{A} \cap \mathcal{P}|$. Both schemes face the same theoretical search factor, which depends only on the combinatorial parameters and not on the cryptographic construction. The difference lies in the cost per candidate: Silent-Share requires only $2|\mathcal{A}|$ pairings per candidate and no additional policy-dependent work, whereas fuzzy IB-ME requires $13d$ pairings and multiple exponentiations. Thus, when the receiver's attribute set is small and the threshold is high, Silent-Share's total online cost remains significantly lower even under identical search overhead. A detailed theoretical decomposition of fuzzy IB-ME's cost is given in Appendix F-G.

E. Summary of Evaluation

Q1 is answered affirmatively: pairing-based authorization cost is independent of $|\mathcal{P}|$ and d, depending only on $|\mathcal{A}|$. The total decapsulation additionally includes a hidden-threshold reconstruction cost, whose worst-case behavior is reported in Appendix F.

Q2 is confirmed by the dense-to-sparse microbenchmarks in Appendix F: sparse encoding is linear and decoding is constant per attribute, whereas dense encoding is quadratic.

Q3 shows that Silent-Share outperforms fuzzy IB-ME in encryption and ciphertext size across all tested parameters, and in decryption when $|\mathcal{A}|$ is small to moderate ($|\mathcal{A}| \leq 15$). This matches the intended unbalanced policy scenario.

VIII. Discussion

A. On the Hidden Common Exponent Assumption

The HCE_d assumption captures exactly the algebraic transcript that is produced by the encapsulation algorithm before the GOKVS encoding layer is applied. It asserts that $W,\ \{(X_a^{(1)}, X_a^{(2)})\}_{a\in\mathcal{P}}$ is pseudorandom even in the presence of adaptive user secret keys that do not individually meet the challenge threshold.

This is not a restatement of the KEM security theorem. The KEM proof additionally uses the GOKVS uniformity lemma, random-oracle programming, and the threshold reconstruction lemmas to move from the HCE transcript to the full ciphertext and key. The HCE assumption itself says nothing about GOKVS, policy hiding, or bilateral composition.

We do not currently know how to reduce HCE_d to standard static assumptions such as SXDH or DBDH. The main difficulty is that the HCE challenge contains correlated pairs

$$X_a^{(1)} = T_a^{s+r_a}, \qquad X_a^{(2)} = g_2^{r_a}$$

that are not directly captured by one- or two-element bilinear assumptions. The HCE_d assumption is construction-specific and captures the correlated common-exponent transcript generated by our policy-hiding layer. Security is proved in the random-oracle model under the Hidden Common Exponent (HCE) assumption defined in Section III-D. The HCE assumption is construction-specific; at present we do not have a reduction to a standard static assumption such as SXDH or DBDH. Appendix C provides a generic-group sanity check showing that the assumption does not fail for elementary algebraic reasons. This sanity check is not a formal proof and should not be interpreted as a security reduction. It is included only to document why the assumption does not fail for elementary structural reasons.

B. Hidden Threshold Reconstruction as a Decoding Problem

The decapsulation algorithm must recover the common secret Y^s from the set $\{(x_a, Z_a)\}_{a\in\mathcal{A}}$ without knowing which d attributes are members of $\mathcal{P}$. For each member $\pi \in \mathcal{P}$, we have

$$Z_\pi = e(g_1, g_2)^{sq_u(x_\pi)} = G^{sq_u(x_\pi)},$$

where $G = e(g_1, g_2)$. Since $q_u(x)$ is a degree-$(d-1)$ polynomial, the set of member points lies on a group-valued polynomial

$$F_u(x) = A_0 A_1^x A_2^{x^2} \cdots A_{d-1}^{x^{d-1}}, \qquad A_0 = Y^s.$$

Non-member points decode to independently random elements of $\mathbb{G}_T$ by the GOKVS uniformity property. Thus the receiver faces a *group-valued Reed–Solomon decoding* problem with errors: find any d points that are consistent with a degree-$(d-1)$ polynomial and recover the constant term $F(0) = Y^s$.

This viewpoint suggests that the current exhaustive subset enumeration is not merely an implementation choice but a manifestation of the underlying decoding problem. In future work, it would be interesting to design a privacy-compatible list decoder for this group-valued setting, or to specialize known Reed–Solomon list-decoding techniques to the case where the inlier set may be as small as the threshold d and the outlier set is arbitrary.

C. Refreshability and Long-Lived Policies

Because the GOKVS representation is randomized, the encryptor can refresh the two vectors $\mathbf{D}^{(1)}$ and $\mathbf{D}^{(2)}$ without changing the encapsulated key Y^s. This is useful for long-lived policies: the public representation can be re-randomized while the underlying policy and encapsulated key remain unchanged.

We emphasize that this is a representation-refresh mechanism, not an unlinkability mechanism. The values $E_0 = \kappa \oplus H_{\mathsf{K}}(Y^s)$, and $C = H_{\mathsf{V}}(Y^s)$ remain unchanged across refreshes, so two refreshed ciphertexts of the same encapsulated key are still linkable through (E_0, C). We do not claim otherwise.

D. Limitations

The current construction has several limitations. First, security is conditional on the HCE assumption; a standard-model or falsifiable assumption would be preferable. Second, threshold reconstruction is exponential in the worst case, so the protocol is best suited to scenarios with small $|\mathcal{A}|$ or d close to $|\mathcal{A}|$. Third, the bilateral protocol requires one round of interaction and reveals the outcome of each directional decapsulation to the participating party; we do not hide the access decision itself. Finally, our implementation uses an asymmetric BN254 pairing group and a centralized authority [36]; decentralized or multi-authority extensions are left to future work.

IX. Conclusion and Future Work

We have presented **Silent-Share**, a bilateral hidden-policy threshold access-control protocol that decouples policy representation from pairing-based authorization. Combining a group-valued oblivious key-value store (GOKVS) with a policy-based key encapsulation mechanism (PB-KEM), Silent-Share achieves two key properties. First, the pairing-based authorization layer requires exactly $2|\mathcal{A}|$ pairings, independent of $|\mathcal{P}|$ and d; the sparse GOKVS realization further reduces representation-layer decoding to $O(\rho)$ group multiplications per attribute, while total decapsulation additionally incurs a hidden-threshold reconstruction cost (characterized separately). Second, composing two independent one-sided instances with an AES-GCM-based binding layer realizes bilateral matchmaking semantics, granting access if and only if both parties' threshold policies are satisfied.

Experiments on BN254 confirm the design goals: pairing scales at ≈ 39.5 ms per attribute, and one-sided decapsulation for $|\mathcal{A}| = 10$ remains ≈ 394 ms whether $|\mathcal{P}|$ is 50 or 800 in the favorable reconstruction configuration. The bilateral protocol scales linearly with the sum of the two policy sizes and is dominated by offline encapsulation.

Future directions include replacing the HCE assumption with a standard or falsifiable one, developing privacy-compatible list-decoding for group-valued Reed–Solomon codes to mitigate the exponential reconstruction cost, extending to post-quantum or lattice-based constructions, and decentralizing the key authority via distributed key generation.

We believe Silent-Share opens a promising path toward practical, privacy-preserving access control for unbalanced hidden-policy settings.

REFERENCES

[1] G. Ateniese, D. Francati, D. Nuñez, and D. Venturi, "Match me if you can: Matchmaking encryption and its applications," *J. Cryptol.*, vol. 34, no. 3, p. 16, 2021.

[2] J. Zhang, X. Li, S. Xu, H. Wu, R. Feng, and G. Bai, "PriME-Deal: privacy-preserving bilateral data trading with efficient matchmaking and auditable fair exchange on blockchain," *CoRR*, vol. abs/2606.11539, 2026.

[3] Y. Yang, R. H. Deng, G. Yang, Y. Li, H. Pang, M. Huang, R. Shi, and J. Weng, "Prisrv: Privacy-enhanced and highly usable service discovery in wireless communications," in *31st Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2024.

[4] Y. Yang, G. Yang, Y. Li, P. Wu, R. Shi, M. Huang, J. Weng, H. Pang, and R. H. Deng, "Prisrv+: Privacy and usability-enhanced wireless service discovery with fast and expressive matchmaking encryption," in *33rd Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2026.

[5] M. Wen, M. Xiao, W. Li, and B. Xiao, "A sanitizable and bilateral access control scheme based on blockchain," *IEEE Internet Things J.*, vol. 12, no. 24, pp. 53 900–53 913, 2025.

[6] L. Guo, H. Lei, and W. Yau, "One-time rewritable blockchain with traitor tracing and bilateral access control," in *IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2022, Wuhan, China, December 9-11, 2022*. IEEE, 2022, pp. 905–912.

[7] J. Zhang, X. Li, M. Zhang, R. Feng, S. Xu, Z. Hou, and G. Bai, "QAE-BAC: achieving quantifiable anonymity and efficiency in blockchain-based access control with attribute," *IEEE Internet Things J.*, vol. 13, no. 15, pp. 35 109–35 120, 2026.

[8] H. Fang, L. Xu, G. Nan, D. Zheng, H. Zhao, and X. Wang, "Accountable distributed access control with privacy preservation for blockchain-enabled internet of things systems: A zero-trust security scheme," *IEEE Internet Things J.*, vol. 12, no. 11, pp. 17 936–17 947, 2025.

[9] M. Zhang, X. Li, J. Zhang, Z. Hou, G. Bai, and R. Feng, "Swiftguard: Enhanced privacy and efficiency in blockchain-based fine-grained access control for cross-domain healthcare collaboration," in *28th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2025, Compiegne, France, May 5-7, 2025*. IEEE, 2025, pp. 1863–1868.

[10] J. Zhang, X. Li, R. Feng, S. Xu, Z. Hou, H. Wu, and G. Bai, "From isolation to integration: A reputation-backed auditable model for cohort data sharing," *IEEE Trans. Dependable Secur. Comput.*, vol. 23, no. 1, pp. 637–654, 2026.

[11] J. Chen, Y. Li, J. Wen, and J. Weng, "Identity-based matchmaking encryption from standard assumptions," in *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, ser. Lecture Notes in Computer Science. Springer, 2022, pp. 394–422.

[12] J. Sun, G. Xu, T. Zhang, X. Yang, M. Alazab, and R. H. Deng, "Privacy-aware and security-enhanced efficient matchmaking encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4345–4360, 2023.

[13] A. Wu, W. Luo, J. Weng, A. Yang, and J. Wen, "Fuzzy identity-based matchmaking encryption and its application," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 5592–5607, 2023.

[14] D. Francati, A. Guidi, L. Russo, and D. Venturi, "Identity-based matchmaking encryption without random oracles," in *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, ser. Lecture Notes in Computer Science, vol. 13143. Springer, 2021, pp. 415–435.

[15] N. Yang, C. Tang, and D. He, "A lightweight certificateless multi-user matchmaking encryption for mobile devices: Enhancing security and performance," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 251–264, 2024.

[16] J. Yan, Y. Ling, J. Chen, and H. Qian, "Museme: Multi-user secure and efficient matchmaking encryption for mobile devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 9084–9098, 2025.

[17] J. Yan, J. Chen, H. Qian, J. Ning, and D. He, "Decentralized multi-authority accurate matchmaking encryption scheme for mobile social networks," *IEEE Trans. Mob. Comput.*, vol. 25, no. 3, pp. 4381–4392, 2026.

[18] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science. Springer, 2021, pp. 395–425.

[19] A. Bienstock, S. Patel, J. Y. Seo, and K. Yeo, "Near-optimal oblivious key-value stores for efficient psi, PSU and volume-hiding multi-maps," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. USENIX Association, 2023, pp. 301–318.

[20] S. Raghuraman and P. Rindal, "Blazing fast PSI from improved OKVS and subfield VOLE," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 2505–2517.

[21] K. Han, S. Kim, B. Lee, and Y. Son, "Revisiting okvs-based OPRF and PSI: cryptanalysis and better construction," *IACR Cryptol. ePrint Arch.*, vol. 2024, p. 1989, 2024.

[22] M. Hao, W. Liu, L. Peng, H. Li, C. Zhang, H. Chen, and T. Zhang, "Unbalanced circuit-psi from oblivious key-value retrieval," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[23] X. Yang, L. Cai, Y. Wang, K. Yin, L. Sun, and J. Hu, "Efficient unbalanced quorum PSI from homomorphic encryption," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024*. ACM, 2024.

[24] R. A. Mahdavi, N. Lukas, F. Ebrahimianghazani, T. Humphries, B. Kacsmar, J. A. Premkumar, X. Li, S. Oya, E. Amjadian, and F. Kerschbaum, "PEPSI: practically efficient private set intersection in the unbalanced setting," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[25] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty PSI and extensions to circuit/quorum PSI," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. ACM, 2021, pp. 1182–1204.

[26] M. Wu, T. H. Yuen, and K. Y. Chan, "O-ring and k-star: Efficient multi-party private set intersection," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[27] D. Bui, G. Garimella, P. Miao, and V. L. P. Pham, "New framework for structure-aware PSI from distributed function secret sharing," in *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part V*, ser. Lecture Notes in Computer Science. Springer, 2025, pp. 294–326.

[28] M. Dong, Y. Chen, C. Zhang, Y. Bai, and Y. Cao, "Multi-party private set operations from predicative zero-sharing," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025, pp. 4079–4093.

[29] L. Piske, J. Singh, N. Trieu, V. Kolesnikov, and V. Zikas, "Distance-aware OT with application to fuzzy PSI," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025, pp. 4679–4691.

[30] C. Dang, X. Zhou, and B. Liang, "Efficient fuzzy PSI based on prefix representation," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025, pp. 2204–2218.

[31] A. Davidson, A. Deo, and L. T. Thibault, "Pool: A practical ot-based OPRF from learning with rounding," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. ACM, 2025, pp. 1038–1052.

[32] H. Wang, K. Chen, Q. Xie, and Q. Meng, "Post-quantum secure identity-based matchmaking encryption," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 1, pp. 833–844, 2025.

[33] H. Mao, J. Zhang, Y. Zhang, and X. Li, "A cross-domain data sharing scheme based on federated blockchain," in *Theoretical Aspects of Software Engineering - 19th International Symposium, TASE 2025, Limassol, Cyprus, July 14-16, 2025, Proceedings*, ser. Lecture Notes in Computer Science, P. Rümmer and Z. Wu, Eds., vol. 15841. Springer, 2025, pp. 285–302.

[34] H. Krawczyk and P. Eronen, "Hmac-based extract-and-expand key derivation function (HKDF)," *RFC*, vol. 5869, pp. 1–14, 2010.

[35] J. Zhang, X. Li, R. Feng, S. Xu, Y. Lv, Z. Hou, and G. Bai, "Medichainaudit: Encrypted medical data auditing on blockchain with fuzzy deduplication and secure sharing," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–17, 2026.

[36] J. Zhang, X. Li, M. Zheng, R. Feng, S. Xu, Z. Hou, and G. Bai, "Verifuzzy: A dynamic verifiable fuzzy search service framework for encrypted cloud data," *IEEE Trans. Serv. Comput.*, vol. 19, no. 1, pp. 780–793, 2026.

APPENDIX A
ADDITIONAL DETAILS ON SPARSE GOKVS ENCODING

For each attribute a, the sparse row function selects $\rho = 3$ distinct positions $j_{a,1}, j_{a,2}, j_{a,3} \in [M]$, where $M = \lceil \alpha |\mathcal{P}| \rceil$, and all row coefficients are fixed to 1. The encoder builds a ρ-uniform hypergraph in which each policy attribute is an edge and each GOKVS bucket is a vertex. Peeling repeatedly removes vertices of degree one, recording the order. If all policy attributes are peeled, the encoder assigns bucket values in reverse peeling order: when a policy attribute π is the only unassigned neighbor of a bucket j, the bucket value is set as

$$D_j = E_\pi \cdot \left(\prod_{\ell \in \mathcal{C}} D_\ell^{-1} \right),$$

where $\mathcal{C}$ is the set of already assigned neighboring buckets. Unassigned buckets are filled with independent uniform group elements.

For an encoded attribute a, decoding computes

$$\mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}, a) = \prod_{k=1}^{\rho} D_{j_{a,k}},$$

which requires exactly ρ group multiplications and no exponentiations. Exact decoding follows from the reverse peeling assignment: at the time a bucket is assigned, all other neighbors of the corresponding attribute have already been assigned, so the product of the three bucket values equals E_π.

Salt handling. In the protocol, the encoder samples a uniformly random salt and uses it to derive the row positions. If peeling fails for this salt, the encoder outputs $\perp$ and the encapsulation aborts; no retry is performed. This ensures the distribution of the salt does not depend on the policy.

Uniformity. If the encoded values $E_1, \ldots, E_{|\mathcal{P}|}$ are independent and uniform in $\mathbb{G}$, then the resulting vector $\mathbf{D}$ is uniform in $\mathbb{G}^M$. This follows because unassigned buckets are uniform by construction, and each assigned bucket is formed by multiplying a uniform value with group elements that are independent of it. This property is used in the security proof to switch to a policy-independent random representation.

APPENDIX B
DENSE VANDERMONDE GOKVS BASELINE

Baseline only. This section describes a dense Vandermonde realization of the GOKVS interface that we implement solely as a performance baseline to quantify the benefit of the sparse representation. It is not part of the Silent-Share construction.

For an attribute a, let $x_a = H_{\mathsf{x}}(a)$ and define the dense row

$$\mathsf{row}(a) = (1, x_a, x_a^2, \ldots, x_a^{M-1}).$$

For a policy containing n attributes, the encoder solves the resulting dense linear system over the group. Correctness follows from the invertibility of the corresponding Vandermonde submatrix when the hashed attribute points are distinct.

However, the dense realization requires $\Theta(n^2)$ algebraic operations for encoding and $\Theta(n)$ group operations for each decode query. It is therefore unsuitable for large policies in our target unbalanced setting. We use this realization only as a baseline to isolate the benefit of the sparse representation; it is not the final implementation of Silent-Share.

APPENDIX C
GENERIC-GROUP ANALYSIS OF HCE

We provide a generic bilinear group sanity check for the HCE_d assumption. The analysis is not a reduction from a static assumption, but it shows that the assumption does not fail for trivial algebraic reasons.

Consider a generic bilinear group model in which group elements are represented by symbolic variables, and the adversary can only perform group operations and pairing evaluations through oracles. The random oracle H_{x} is modeled by assigning a random field element to each distinct attribute label.

The HCE challenge for a policy $\mathcal{P}$ consists of

$$W = Y^s, \qquad X_a^{(1)} = T_a^{s+r_a}, \qquad X_a^{(2)} = g_2^{r_a}$$

for every $a \in \mathcal{P}$. The adversary also obtains user secret keys for identities u with attribute sets $\mathcal{A}_u$ satisfying

$$|\mathcal{A}_u \cap \mathcal{P}| < d.$$

For each such user, the key contains

$$D_{u,a}^{(1)} = g_1^{q_u(x_a)/t_a}, \qquad D_{u,a}^{(2)} = g_1^{q_u(x_a)}$$

for $a \in \mathcal{A}_u$.

Goal. To distinguish the real challenge from a random one, the adversary must construct a nontrivial pairing relation involving the hidden exponents s and $\{r_a\}$. In particular, it must combine group elements from the challenge with key elements from its queried users to form a group element whose exponent is identical to sy in the real world, but not in the random world.

Algebraic constraints. For a queried user u, each attribute $a \in \mathcal{A}_u$ gives the adversary two pairs:

$$D_{u,a}^{(1)} \quad \text{paired with} \quad X_a^{(1)}$$

and

$$D_{u,a}^{(2)} \quad \text{paired with} \quad X_a^{(2)}.$$

Pairing these yields

$$e(D_{u,a}^{(1)}, X_a^{(1)}) = g_T^{q_u(x_a)(s+r_a)},$$

and

$$e(D_{u,a}^{(2)}, X_a^{(2)}) = g_T^{q_u(x_a) r_a}.$$

Dividing the first by the second gives

$$g_T^{q_u(x_a)s}.$$

Why collusion cannot reconstruct Y^s. To obtain $g_T^{sy} = Y^s$, the adversary must interpolate d distinct values $q_u(x_a)$ for the same user u. Since each queried user satisfies $|\mathcal{A}_u \cap \mathcal{P}| < d$, no single user provides d matching attributes. Combining values from different users does not help because the polynomials

q_u are independent and the coefficients $c_{u,j}$ are random and hidden. Any linear combination across different users contains at least one random coefficient that remains independent of the challenge exponents, so the resulting group element is uniformly random in $\mathbb{G}_T$.

Thus, in the generic group model with random oracle H_{x}, the adversary cannot construct a group element whose exponent equals sy unless it trivially violates the threshold restriction. This appendix presents an informal generic-group sanity check for the HCE_d assumption. It is not a formal proof and does not constitute a security reduction. Its purpose is limited to documenting that the assumption is not obviously breakable by simple algebraic combinations in a symbolic group model.

APPENDIX D
DETAILED PROTOCOL SPECIFICATION AND CORRECTNESS

This appendix gives the algorithm-by-algorithm specification of the one-sided Silent-Share PB-KEM and the bilateral composition. The notation follows Section III; the compact version is shown in Figure 1.

A. System Setup

The authority samples a master exponent $y \leftarrow \mathbb{F}_p^*$ and, for every attribute $a \in \mathcal{U}$, samples $t_a \leftarrow \mathbb{F}_p^*$. It computes $Y = e(g_1, g_2)^y$ and $T_a = g_2^{t_a}$.

The public parameters are

$$\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e, Y, \{T_a\}_{a\in\mathcal{U}}, H_{\mathsf{K}}, H_{\mathsf{V}}, H_{\mathsf{x}}),$$

and the master secret is

$$\mathsf{msk} = (y, \{t_a\}_{a\in\mathcal{U}}).$$

B. Key Generation

For a registered user uid with attribute set $\mathcal{A}_{\mathsf{uid}}$, the authority samples a random polynomial

$$q_{\mathsf{uid}}(X) = y + \sum_{j=1}^{d-1} c_{\mathsf{uid},j} X^j,$$

where $c_{\mathsf{uid},j} \leftarrow \mathbb{F}_p$. Hence, $q_{\mathsf{uid}}(0) = y$.

For every attribute $a \in \mathcal{A}_{\mathsf{uid}}$, let $x_a = H_{\mathsf{x}}(a)$. The authority computes

$$D^{(1)}_{\mathsf{uid},a} = g_1^{q_{\mathsf{uid}}(x_a)/t_a}$$

and

$$D^{(2)}_{\mathsf{uid},a} = g_1^{q_{\mathsf{uid}}(x_a)}.$$

The resulting secret key is

$$\mathsf{SK}_{\mathsf{uid}} = \left\{ \left(a, x_a, D^{(1)}_{\mathsf{uid},a}, D^{(2)}_{\mathsf{uid},a} \right) \right\}_{a\in\mathcal{A}_{\mathsf{uid}}}.$$

Each identity is registered at most once, and all attributes issued to the same identity are derived from the same polynomial q_{uid}. This binding is essential: the threshold reconstruction property holds only when the required shares originate from the same user-specific polynomial, and it prevents collusion attacks that combine shares from different users.

C. Sparse-GOKVS Encoding

For a policy

$$\mathcal{P} = \{\pi_1, \ldots, \pi_{|\mathcal{P}|}\},$$

the encryptor first determines the Sparse-GOKVS representation length $M = \lceil \alpha|\mathcal{P}| \rceil$. For each policy attribute π, the sparse row function selects ρ distinct positions in $[M]$:

$$\mathsf{Pos}(\pi) = \{j_{\pi,1}, \ldots, j_{\pi,\rho}\}.$$

The row positions are derived deterministically from H_{x} together with a public encoding salt.

For each $\pi \in \mathcal{P}$, the encryptor prepares a group value $E_\pi \in \mathbb{G}_2$. The collection

$$\mathcal{E} = \{(\pi, E_\pi)\}_{\pi\in\mathcal{P}}$$

is then encoded as

$$\mathbf{D} \leftarrow \mathsf{Encode}_{\mathbb{G}_2}(\mathcal{E}).$$

The encoder constructs the corresponding sparse incidence graph and applies peeling. Whenever a bucket has degree one, the corresponding policy entry can be removed from the residual graph and recorded in the peeling order. If all $|\mathcal{P}|$ entries are peeled, the group-valued representation is constructed in reverse peeling order. Unassigned buckets are filled with independent random elements of $\mathbb{G}_2$.

If peeling fails, the encoder samples a fresh salt and repeats the encoding procedure. The implementation uses a fixed maximum number of attempts; failure after this bound results in $\perp$.

The crucial property for Silent-Share is that decoding an attribute requires only the ρ positions associated with that attribute:

$$\mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}, \pi) = \prod_{k=1}^{\rho} D_{j_{\pi,k}},$$

because all row coefficients are fixed to 1. Thus decoding requires exactly ρ group multiplications and is independent of $|\mathcal{P}|$.

D. Encapsulation

The encryptor chooses $\kappa \leftarrow \{0,1\}^\lambda$ and samples $s \leftarrow \mathbb{F}_p^*$. It computes the common pairing secret $Y^s = e(g_1, g_2)^{ys}$.

The encapsulated key is masked as

$$E_0 = \kappa \oplus H_{\mathsf{K}}(Y^s),$$

and the reconstruction value is authenticated by

$$C = H_{\mathsf{V}}(Y^s).$$

For every policy attribute $\pi \in \mathcal{P}$, where $\pi = a \in \mathcal{U}$, the $T_\pi = T_a = g_2^{t_a} = g_2^{t_\pi}$, the encryptor samples $r_\pi \leftarrow \mathbb{F}_p^*$ and computes

$$E^{(1)}_\pi = T_\pi^{s+r_\pi} = g_2^{t_\pi(s+r_\pi)}$$

and

$$E^{(2)}_\pi = g_2^{r_\pi}.$$

The two collections

$$\mathcal{E}^{(1)} = \left\{ (\pi, E^{(1)}_\pi) \right\}_{\pi\in\mathcal{P}}$$

and

$$\mathcal{E}^{(2)} = \left\{(\pi, E_\pi^{(2)})\right\}_{\pi\in\mathcal{P}}$$

are encoded independently as

$$\mathbf{D}^{(1)} \leftarrow \mathsf{Encode}_{\mathbb{G}_2}\left(\mathcal{E}^{(1)}\right)$$

and

$$\mathbf{D}^{(2)} \leftarrow \mathsf{Encode}_{\mathbb{G}_2}\left(\mathcal{E}^{(2)}\right),$$

using the same salt and the same row positions.

The ciphertext is

$$\mathsf{CT} = \left(E_0, C, \mathbf{D}^{(1)}, \mathbf{D}^{(2)}\right).$$

The policy itself is not included in the ciphertext. The public salt, row degree ρ, and representation length M can be included as public metadata, because they do not leak policy attributes.

E. Decapsulation

Given a secret key $\mathsf{SK}_{\mathsf{uid}}$ and a ciphertext CT, the decryptor first probes the two Sparse-GOKVS representations using its own attributes.

For each $a \in \mathcal{A}_{\mathsf{uid}}$, it obtains

$$\widehat{E}_a^{(1)} = \mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}^{(1)}, a)$$

and

$$\widehat{E}_a^{(2)} = \mathsf{Decode}_{\mathbb{G}_2}(\mathbf{D}^{(2)}, a).$$

The decryptor then computes

$$Z_a = e\left(D_{\mathsf{uid},a}^{(1)}, \widehat{E}_a^{(1)}\right) \cdot e\left(D_{\mathsf{uid},a}^{(2)}, \widehat{E}_a^{(2)}\right)^{-1}.$$

For an attribute a that belongs to both $\mathcal{A}_{\mathsf{uid}}$ and $\mathcal{P}$, Sparse-GOKVS exact decoding gives

$$\widehat{E}_a^{(1)} = g_2^{t_a(s+r_a)}$$

and

$$\widehat{E}_a^{(2)} = g_2^{r_a}.$$

Therefore,

$$Z_a = e(g_1, g_2)^{s q_{\mathsf{uid}}(x_a)}.$$

Let $\mathcal{S} \subseteq \mathcal{A}_{\mathsf{uid}}$ be a candidate subset of size d. The decryptor computes

$$\Omega_{\mathcal{S}} = \prod_{a\in\mathcal{S}} Z_a^{\Delta_{a,\mathcal{S}}(0)},$$

where $\Delta_{a,\mathcal{S}}(0)$ is the Lagrange coefficient for interpolation at zero.

The candidate is accepted if

$$H_{\mathsf{V}}(\Omega_{\mathcal{S}}) = C.$$

Upon a successful verification, the decryptor derives

$$\kappa = E_0 \oplus H_{\mathsf{K}}(\Omega_{\mathcal{S}}).$$

If no candidate satisfies the verification equation, the algorithm outputs $\perp$.

F. Correctness

Suppose that

$$|\mathcal{A}_{\mathsf{uid}} \cap \mathcal{P}| \geq d.$$

Then there exists a subset

$$\mathcal{S}^* \subseteq \mathcal{A}_{\mathsf{uid}} \cap \mathcal{P}$$

with $|\mathcal{S}^*| = d$.

By the exact-decoding property of Sparse-GOKVS, for every $\pi \in \mathcal{S}^*$,

$$\widehat{E}_\pi^{(1)} = g_2^{t_\pi(s+r_\pi)} \qquad \text{and} \qquad \widehat{E}_\pi^{(2)} = g_2^{r_\pi}.$$

Hence,

$$Z_\pi = e(g_1, g_2)^{s q_{\mathsf{uid}}(x_\pi)}.$$

By Lagrange interpolation,

$$\sum_{\pi\in\mathcal{S}^*} q_{\mathsf{uid}}(x_\pi)\Delta_{\pi,\mathcal{S}^*}(0) = q_{\mathsf{uid}}(0) = y.$$

Consequently,

$$\Omega_{\mathcal{S}^*} = \prod_{\pi\in\mathcal{S}^*} Z_\pi^{\Delta_{\pi,\mathcal{S}^*}(0)} = e(g_1, g_2)^{s\sum_{\pi\in\mathcal{S}^*} q_{\mathsf{uid}}(x_\pi)\Delta_{\pi,\mathcal{S}^*}(0)} = e(g_1, g_2)^{sy} = Y^s.$$

Therefore,

$$H_{\mathsf{V}}(\Omega_{\mathcal{S}^*}) = H_{\mathsf{V}}(Y^s) = C,$$

and the decryptor obtains

$$\kappa = E_0 \oplus H_{\mathsf{K}}(\Omega_{\mathcal{S}^*}) = \kappa \oplus H_{\mathsf{K}}(Y^s) \oplus H_{\mathsf{K}}(Y^s) = \kappa.$$

Thus every honestly authorized user successfully decapsulates except with negligible probability due to the verification-hash collision of an incorrect candidate.

G. Bilateral Composition

The bilateral protocol uses two independent one-sided instances and one AES-GCM encryption.

1) The requester generates $(\mathsf{CT}_R, k_R) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_R)$ and sends CT_R to the owner.
2) The owner generates $(\mathsf{CT}_O, k_O) \leftarrow \mathsf{Encap}(\mathsf{pp}, \mathcal{P}_O)$, decapsulates CT_R to obtain k_R, samples a fresh nonce N, computes

$$\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N),$$

and sends (CT_O, τ, N) to the requester.
3) The requester decapsulates CT_O to obtain k_O, decrypts

$$k_R = \mathsf{AES\text{-}GCM.Dec}_{k_O}(\tau, \mathsf{ctx}, N),$$

and both parties derive $K_{\mathrm{data}} = \mathsf{HKDF}(k_O \| k_R, \mathsf{ctx}, \lambda)$.

Correctness follows directly from the one-sided correctness property. If the owner satisfies the requester's policy, it recovers k_R and can construct τ. If the requester satisfies the owner's policy, it recovers k_O and can decrypt τ to obtain k_R. Therefore both parties derive the same K_{data} if and only if both directional predicates hold.

TABLE II: Algebraic reconstruction states for a set of matching attributes.

Number of attributes	User composition	Reconstructed value
$\ell < d$	any	uniform in $\mathbb{G}_T$
$\ell = d$	all same user	Y^s
$\ell = d$	at least two users	uniform in $\mathbb{G}_T$

H. Instantiation Details

For concreteness, we specify the standard instantiations used in our implementation and implicitly assumed in the security analysis.

Key derivation function. The final data key is derived using HKDF [34] with SHA-256:

$$K_{\text{data}} = \mathsf{HKDF}(k_O \| k_R, \mathsf{ctx}, \lambda)\,,$$

where $k_O \| k_R$ is the concatenation of the two recovered one-sided keys and λ is the output length. HKDF is used because it is standardized, widely available, and designed to accept high-entropy input keying material of arbitrary length.

Symmetric encryption. The symmetric encryption layer is instantiated with AES-GCM. The owner samples a fresh nonce N, computes

$$\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N),$$

where k_R is the plaintext, k_O is the 128-bit or 256-bit symmetric key, N is the fresh nonce, and ctx is the associated data. The nonce N is transmitted together with τ. The requester decrypts using

$$k_R = \mathsf{AES\text{-}GCM.Dec}_{k_O}(\tau, \mathsf{ctx}, N).$$

AES-GCM provides both confidentiality and authenticity, so a successful decryption also ensures that τ was generated by a party possessing k_O and the correct plaintext k_R.

Random oracle compatibility. In the security proof, H_{K}, H_{V}, and KDF are modeled as random oracles. In practice we instantiate H_{K} and H_{V} with SHA-256 using domain separation, and KDF with HKDF-SHA256. This is a standard ideal-model assumption; the gap between the random-oracle model and the concrete instantiation is not known to be exploitable for these standard hash-based constructions in the setting considered here.

APPENDIX E
FULL SECURITY PROOFS

This appendix contains the complete formal proofs for the security theorems stated in Section VI. Notation and assumptions follow Sections III and IV-C. The reconstruction states is shown in Table II.

A. Algebraic Reconstruction Lemmas

Lemma 6 (Same-User Threshold Reconstruction)**.** *Let u be a registered user with secret polynomial $q_u(x) = y + \sum_{j=1}^{d-1} c_{u,j} x^j$, where $q_u(0) = y$. Let $\mathcal{S} = \{a_1, \ldots, a_d\}$ be d distinct attributes belonging to the same user's attribute set. For every $a_i \in \mathcal{S}$, let $Z_{a_i} = e(g_1, g_2)^{s q_u(x_{a_i})}$. Then*

$$\prod_{i=1}^{d} Z_{a_i}^{\Delta_{a_i,\mathcal{S}}(0)} = Y^s.$$

Proof. By Lagrange interpolation,

$$\sum_{i=1}^{d} q_u(x_{a_i}) \Delta_{a_i,\mathcal{S}}(0) = q_u(0) = y.$$

Consequently,

$$\prod_{i=1}^{d} Z_{a_i}^{\Delta_{a_i,\mathcal{S}}(0)} = e(g_1, g_2)^{s \sum_{i=1}^{d} q_u(x_{a_i}) \Delta_{a_i,\mathcal{S}}(0)} = e(g_1, g_2)^{sy} = Y^s.$$

□

Lemma 7 (Sub-threshold Linear Combination Masking)**.** *Let $t < d$, and let $\mathcal{S} = \{(a_i, u_i)\}_{i=1}^{t}$ be a collection of t distinct attributes together with their registered users. Let $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_p$ be arbitrary coefficients such that $\sum_{i=1}^{t} \alpha_i = 1$. Then the linear combination*

$$L_{\mathcal{S}} = \sum_{i=1}^{t} \alpha_i q_{u_i}(x_{a_i})$$

is uniformly distributed over $\mathbb{F}_p$, except with probability bounded by the attribute-hash collision event $\mathsf{Bad}_{\mathsf{x}}$. Consequently, for every nonzero s, $e(g_1, g_2)^{s L_{\mathcal{S}}}$ is uniformly distributed in $\mathbb{G}_T$.

Proof. Write $q_u(x) = y + \sum_{j=1}^{d-1} c_{u,j} x^j$. Then

$$L_{\mathcal{S}} = y \sum_{i=1}^{t} \alpha_i + \sum_{u} \sum_{j=1}^{d-1} c_{u,j} \left(\sum_{i:u_i=u} \alpha_i x_{a_i}^j \right) = y + \sum_{u} \sum_{j=1}^{d-1} c_{u,j} \beta_{u,j},$$

where $\beta_{u,j} = \sum_{i:u_i=u} \alpha_i x_{a_i}^j$. If all $\beta_{u,j}$ were zero, then for each user u with $k_u \le t < d$ attributes, the Vandermonde matrix $(x_{a_i}^j)_{j \in [k_u], i:u_i=u}$ would force all α_i with $u_i = u$ to be zero, contradicting $\sum_i \alpha_i = 1$. Hence there exists $u^\star$ and $j^\star$ with $\beta_{u^\star,j^\star} \neq 0$. Since $c_{u^\star,j^\star}$ is uniform and independent, $L_{\mathcal{S}}$ is uniform. □

Lemma 8 (Mixed-User d-Tuple Masking)**.** *Let $\mathcal{S} = \{(a_i, u_i)\}_{i=1}^{d}$ be a collection of d distinct attributes together with their registered users, and suppose that not all u_i are equal, i.e., the attributes come from at least two different users. Let $\Delta_i = \Delta_{a_i,\mathcal{S}}(0)$ be the Lagrange coefficients for interpolation at 0. Then the linear combination*

$$L_{\mathcal{S}} = \sum_{i=1}^{d} \Delta_i q_{u_i}(x_{a_i})$$

is uniformly distributed over $\mathbb{F}_p$, except with probability bounded by $\mathsf{Bad}_{\mathsf{x}}$. Consequently, for every nonzero s, $e(g_1, g_2)^{s L_{\mathcal{S}}}$ is uniformly distributed in $\mathbb{G}_T$.

Proof. The proof is similar to Lemma 7. Write

$$L_{\mathcal{S}} = y \sum_{i=1}^{d} \Delta_i + \sum_{u} \sum_{j=1}^{d-1} c_{u,j} \left(\sum_{i:u_i=u} \Delta_i x_{a_i}^j \right).$$

Since $\sum_{i=1}^{d} \Delta_i = 1$, the first term is y. If all coefficients $\beta_{u,j}$ were zero, then for a user u with $k_u < d$ attributes, the $k_u \times k_u$ Vandermonde matrix would force all Δ_i with $u_i = u$ to be zero, implying $\sum_i \Delta_i = 0$, a contradiction. Hence some $\beta_{u^\star,j^\star} \neq 0$, and uniformity follows from the uniform random coefficient $c_{u^\star,j^\star}$. □

B. GOKVS Uniformity Lemma

Lemma 9 (Uniformity of Sparse-GOKVS Encoding)**.** *Let $\mathcal{P} = \{a_1, \ldots, a_{|\mathcal{P}|}\}$ be a policy for which the Sparse-GOKVS encoder succeeds. Suppose the associated values $E_1, \ldots, E_{|\mathcal{P}|}$ are sampled independently and uniformly from a group $\mathbb{G}$. Then the resulting vector*

$$\mathbf{D} \leftarrow \mathsf{Encode}_{\mathbb{G}}(\{(a_i, E_i)\}_{i=1}^{|\mathcal{P}|})$$

is uniformly distributed over $\mathbb{G}^M$, where $M = \lceil \alpha |\mathcal{P}| \rceil$.

Proof. By the peeling construction, the encoder first identifies a peeling order and fills all buckets not touched by peeling with independent uniform group elements. For each peeled bucket, its value is set to

$$D_j = E_\pi \cdot \left(\prod_{\ell \in \mathcal{C}} D_\ell^{-1} \right),$$

where $\mathcal{C}$ is the set of already-assigned neighboring buckets and E_π is uniform and independent of all previous bucket values. Since multiplication by a uniform group element is a bijection, each peeled bucket is uniformly distributed given the previous buckets. Therefore, the joint distribution of all M buckets is uniform over $\mathbb{G}^M$. Exact decoding follows from the peeling construction. □

C. One-Sided KEM-IND-CPA: Full Proof

Full proof of Theorem 2. We construct an adversary $\mathcal{B}$ against HCE_d.

Simulation before the challenge. $\mathcal{B}$ receives the public parameters $Y, \{T_a\}_{a\in\mathcal{U}}$ from its HCE challenger and gives them to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a key-generation query $(u, \mathcal{A}_u)$, $\mathcal{B}$ forwards the query to its HCE challenger's key-generation oracle and returns the resulting key to $\mathcal{A}$. Thus the joint distribution of all user keys is identical in the two experiments.

$\mathcal{B}$ simulates the random oracles H_{K} and H_{V} using lazy sampling. Queries to H_{x} are forwarded to the HCE challenger, so that the values $x_a = H_{\mathsf{x}}(a)$ used in key generation and in the GOKVS encoding are consistent with the HCE experiment.

Challenge generation. Eventually $\mathcal{A}$ outputs an admissible challenge policy $\mathcal{P}$. $\mathcal{B}$ submits the same policy to the HCE challenger and receives

$$W_b, \qquad \{(X^{(1)}_{a,b}, X^{(2)}_{a,b})\}_{a\in\mathcal{P}}.$$

The reduction samples $\kappa \leftarrow \{0,1\}^\lambda$. If W_b has not been previously queried to H_{V}, $\mathcal{B}$ chooses $C = H_{\mathsf{V}}(W_b)$ uniformly at random and records it. If W_b has not been previously queried to H_{K}, $\mathcal{B}$ chooses $R = H_{\mathsf{K}}(W_b)$ uniformly at random and records it. It sets

$$E_0 = \kappa \oplus R.$$

It computes

$$\mathbf{D}^{(1)} = \mathsf{Encode}_{\mathbb{G}_2}\left(\{(a, X^{(1)}_{a,b})\}_{a\in\mathcal{P}}\right)$$

and

$$\mathbf{D}^{(2)} = \mathsf{Encode}_{\mathbb{G}_2}\left(\{(a, X^{(2)}_{a,b})\}_{a\in\mathcal{P}}\right),$$

using fresh independent encoder randomness and the Sparse-GOKVS realization of Section III-F.

The resulting challenge ciphertext is $\mathsf{CT} = (E_0, C, \mathbf{D}^{(1)}, \mathbf{D}^{(2)})$.

$\mathcal{B}$ chooses an internal uniform bit $c \in \{0,1\}$. If $c = 0$, it gives (CT, κ) to $\mathcal{A}$; if $c = 1$, it gives (CT, κ_R) for a fresh uniform $\kappa_R \in \{0,1\}^\lambda$. It then answers post-challenge key queries as before and outputs whatever $\mathcal{A}$ outputs.

Real HCE branch. Suppose $b = 0$. Then $W_0 = Y^s$ and $X^{(1)}_{a,0} = T_a^{s+r_a}$, $X^{(2)}_{a,0} = g_2^{r_a}$. Thus the generated GOKVS structures are distributed exactly as in the real Encapsulation algorithm, while $R = H_{\mathsf{K}}(Y^s)$ and $C = H_{\mathsf{V}}(Y^s)$ by the programmed oracles. Hence the resulting challenge ciphertext and key pair are distributed exactly as in the real KEM game with internal bit c.

Random HCE branch. Suppose $b = 1$. Then $W_1 \leftarrow \mathbb{G}_T$ and all $X^{(1)}_{a,1}, X^{(2)}_{a,1}$ are independent uniform elements of $\mathbb{G}_2$. By Lemma 9,

$$\mathbf{D}^{(1)}, \mathbf{D}^{(2)} \overset{d}{=} \mathbb{G}_2^M.$$

These vectors are independent of κ.

Let Bad_K be the event that W_1 was previously queried to H_{K}, and Bad_V be the event that W_1 was previously queried to H_{V}. Since W_1 is uniform in $\mathbb{G}_T$ and independent of the adversary's view, the probability of either event is at most $(q_K + q_V)/(p-1)$. Conditioned on $\neg\mathsf{Bad}_K \wedge \neg\mathsf{Bad}_V$, $H_{\mathsf{K}}(W_1)$ and $H_{\mathsf{V}}(W_1)$ are fresh uniform strings independent of κ. Therefore (E_0, C) is uniformly distributed over $\{0,1\}^{2\lambda}$ and independent of κ.

The additional term $2^{-\lambda}$ accounts for the probability that $\kappa_R = \kappa$ when $c = 1$. Finally, $\Pr[\mathsf{Bad}_{\mathsf{x}}]$ and $\Pr[\mathsf{Bad}_{\mathsf{peel}}]$ are bounded in Section VI-A.

Combining the real and random branches via the HCE assumption yields the claimed bound. □

D. One-Sided Policy Hiding: Full Proof

Full proof of Theorem 3. Let $\mathcal{P}_0, \mathcal{P}_1$ be the two challenge policies with $|\mathcal{P}_0| = |\mathcal{P}_1|$. For each $b \in \{0,1\}$, construct an HCE adversary $\mathcal{B}_b$ that submits $\mathcal{P}_b$ as its challenge policy and receives either the real transcript or the random transcript.

The reduction generates a ciphertext exactly as in the KEM proof: it samples κ, computes $C = H_{\mathsf{V}}(W)$, programs $H_{\mathsf{K}}(W) = R$, sets $E_0 = \kappa \oplus R$, and Sparse-GOKVS encodes the received group pairs.

In the real branch, the ciphertext is distributed as a true Silent-Share ciphertext for $\mathcal{P}_b$.

In the random branch, by Lemma 9 both $\mathbf{D}^{(1)}$ and $\mathbf{D}^{(2)}$ are uniform over $\mathbb{G}_2^M$. Conditioned on W not having been previously queried to H_{K} or H_{V}, the pair (E_0, C) is uniform and independent of the policy. Therefore the entire ciphertext is independent of $\mathcal{P}_b$.

Let RndCT denote the policy-independent random ciphertext distribution. Then

$$\mathsf{CT}(\mathcal{P}_b) \approx \mathsf{RndCT}$$

with distinguishing advantage bounded by the HCE advantage plus the bad-event probabilities. By the triangle inequality, the distance between $\mathsf{CT}(\mathcal{P}_0)$ and $\mathsf{CT}(\mathcal{P}_1)$ is at most the sum of the two hybrid advantages, giving the claimed bound.

Since $|\mathcal{P}_0| = |\mathcal{P}_1|$, both ciphertexts have the same GOKVS dimension M, so the only leakage is the public policy size. □

E. Bilateral KEM Security: Full Hybrid Proof

Full proof of Theorem 4. We proceed by a sequence of games.

Game 0. This is the real bilateral protocol. The adversary sees $(\mathsf{CT}_R, \mathsf{CT}_O, \tau, N)$ and receives K_{data} as the challenge key.

Game 1. Replace the owner-side key k_O with a uniformly random string k_O^*, while keeping the requester-side key k_R unchanged. That is, the challenger still generates CT_O as a valid ciphertext, but uses k_O^* and the same nonce N in the AES-GCM encryption and in the final HKDF:

$$\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O^*}(k_R, \mathsf{ctx}, N), \qquad K_{\mathrm{data}} = \mathsf{HKDF}(k_O^* \| k_R, \mathsf{ctx}).$$

By the IND-CPA security of the owner-side one-sided PB-KEM, Game 0 and Game 1 are indistinguishable.

Game 2. Replace the AES-GCM plaintext with a random string of the same length as k_R. That is, sample $\widehat{k}_R \leftarrow \{0,1\}^{|k_R|}$ and set

$$\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O^*}(\widehat{k}_R, \mathsf{ctx}, N).$$

The final key still uses the real k_R:

$$K_{\mathrm{data}} = \mathsf{HKDF}(k_O^* \| k_R, \mathsf{ctx}).$$

Since k_O^* is uniformly random and independent of the view of $\mathcal{A}$, AES-GCM semantic security implies that Game 1 and Game 2 are indistinguishable.

Game 3. Replace the requester-side key k_R with a uniformly random string k_R^*. The AES-GCM ciphertext remains

$$\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O^*}(\widehat{k}_R, \mathsf{ctx}, N),$$

which is independent of k_R. The final key becomes

$$K_{\mathrm{data}} = \mathsf{HKDF}(k_O^* \| k_R^*, \mathsf{ctx}).$$

By the IND-CPA security of the requester-side one-sided PB-KEM, Game 2 and Game 3 are indistinguishable.

Final step. In Game 3, k_O^* and k_R^* are uniform and independent, and τ is independent of both. Since HKDF is modeled as a random oracle, the output

$$K_{\mathrm{data}} = \mathsf{HKDF}(k_O^* \| k_R^*, \mathsf{ctx})$$

is indistinguishable from a uniform random string unless the adversary queries KDF on exactly $(k_O^* \| k_R^*, \mathsf{ctx})$. Because the adversary's view contains no information about k_O^* or k_R^*, the probability of such a query is negligible.

Combining the transitions yields the bound in the theorem statement. □

TABLE III: Peeling success rate of Sparse-GOKVS over 20 independent trials (BN254).

		$\alpha = m/n$					
n	ρ	1.15	1.20	1.25	1.30	1.35	1.40
200	2	0	0	0	0	0	0
200	3	0	0	**1**	1	1	1
200	4	0	0	0	**1**	1	1
300	2	0	0	0	0	0	0
300	3	0	**1**	1	1	1	1
300	4	0	0	0	**1**	1	1
400	2	0	0	0	0	0	0
400	3	**1**	1	1	1	1	1
400	4	0	0	**1**	1	1	1

F. Bilateral Policy Hiding: Full Proof

Full proof of Theorem 5. The two ciphertexts are generated using independent encapsulation randomness under the same public parameters. By the one-sided policy-hiding theorem, each ciphertext individually hides its policy up to size. In the bilateral policy-hiding experiment, the adversary's output policies must satisfy the leakage consistency condition; that is, for the active participant's attribute set $\mathcal{A}^*$, the predicates $L_O(\mathcal{P}_O^0, \mathcal{A}^*)$ and $L_O(\mathcal{P}_O^1, \mathcal{A}^*)$ are identical, and similarly for L_R. Hence, the adversary cannot distinguish the two challenge worlds by observing its own access decision.

For an external observer, the only additional protocol value is $\tau = \mathsf{AES\text{-}GCM.Enc}_{k_O}(k_R, \mathsf{ctx}, N)$. AES-GCM does not reveal its plaintext or the policies beyond the access-decision leakage already implied by successful or failed decapsulation. Thus the external view leaks only the two policy sizes and no policy attributes. A standard hybrid argument over the two policies yields the claimed bound. □

Appendix F
Additional Evaluation Details

This appendix reports additional experiments supporting the evaluation in Section VII. All measurements are conducted on the BN254 curve under the same environment described in Section VII-A.

A. Microbenchmarks: Sparse GOKVS vs Dense GOKVS

Figure 4 compares encoding and decoding time between the dense Vandermonde baseline and Sparse-GOKVS. The dense baseline is described in Appendix B. On BN254, the dense encoding grows quadratically, requiring 4.75 s for $|\mathcal{P}| = 50$ and 1078 s for $|\mathcal{P}| = 800$. Sparse-GOKVS encoding remains linear: 34.3 ms at $|\mathcal{P}| = 50$, 434.4 ms at $|\mathcal{P}| = 800$. For single-attribute decoding, dense decoding grows linearly from 95.5 ms at $|\mathcal{P}| = 50$ to 1364.2 ms at $|\mathcal{P}| = 800$, whereas Sparse-GOKVS decoding stays constant at about 0.01 ms per attribute.

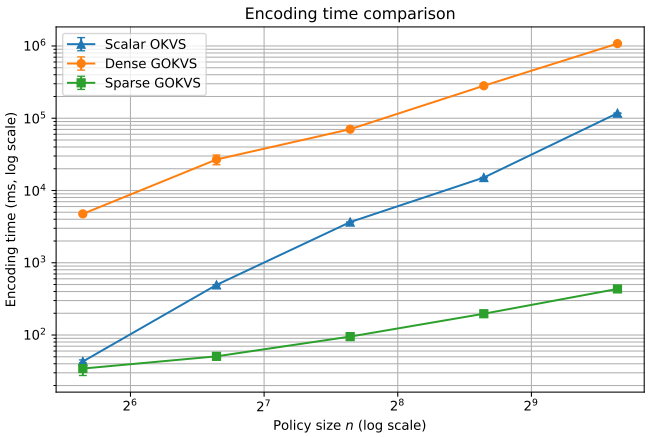
Finding: The sparse representation is essential for large policies, reducing encoding from quadratic to linear and decoding from linear to constant.

B. Parameter Selection for Sparse-GOKVS

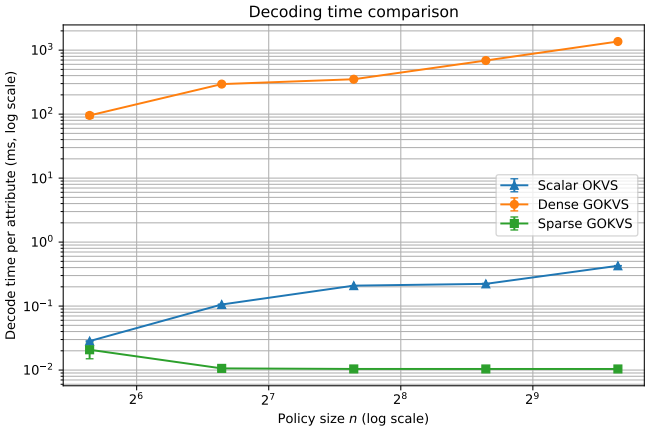
We determine the row sparsity ρ and the bucket-to-item ratio $\alpha = m/n$ through a parameter sensitivity analysis on BN254. We vary $\rho \in \{2, 3, 4\}$ and $\alpha \in \{1.15, 1.20, 1.25, 1.30, 1.35, 1.40\}$, and measure the peeling success rate over 20 independent trials for policy sizes $n \in \{200, 300, 400\}$.

Table III reports the success rate for each configuration. A value of 1 indicates that all 20 trials succeeded, while 0 indicates that all trials failed. The entries in **bold** mark the smallest α at which a given ρ first achieves a 100% success rate for that policy size.

From the success table, $\rho = 2$ is ruled out because it fails for all tested α. Between $\rho = 3$ and $\rho = 4$, $\rho = 3$ achieves 100% success with a smaller bucket ratio ($\alpha = 1.25$ for $n = 200, 300$, and $\alpha = 1.15$ for $n = 400$) compared with $\rho = 4$ (which requires $\alpha \geq 1.30$ for $n = 200, 300$).



(a) Encoding time.



(b) Decoding time per attribute.

Fig. 4: Dense Vandermonde GOKVS vs. Sparse-GOKVS on BN254.

Figure 5 compares encoding time for the representative policy size $n = 300$. We choose $n = 300$ because it lies in the middle of the tested range and clearly exhibits the timing gap between $\rho = 3$ and $\rho = 4$ in their valid parameter regions. The figure shows that $\rho = 3$ consistently achieves lower encoding time than $\rho = 4$ at the same α, while $\alpha = 1.25$ provides a good balance between reliability and ciphertext size.

Based on these measurements, we select $\rho = 3$ and $\alpha = 1.25$ as the default parameters for the Sparse-GOKVS instantiation used in Silent-Share. This choice achieves 100% empirical success over all tested policy sizes and keeps the representation overhead moderate.

Peeling failure handling. If peeling fails for a given hash salt, the encoder samples a fresh salt and retries. In all our parameter-selection experiments, the selected configuration ($\rho = 3, \alpha = 1.25$) achieved success on the first attempt for every tested policy size, so no retry was actually required. For larger policy sizes or adversarial inputs, the encoder may need to resample; the implementation records the number of retries and aborts only after a fixed bound.

Theoretical peeling bound. For a random 3-uniform hypergraph with $M = \alpha n$ vertices and n edges, the peeling process succeeds with overwhelming probability when α is below the peelability threshold. The value $\alpha = 1.25$ selected above is below this threshold and is confirmed empirically in Table III.

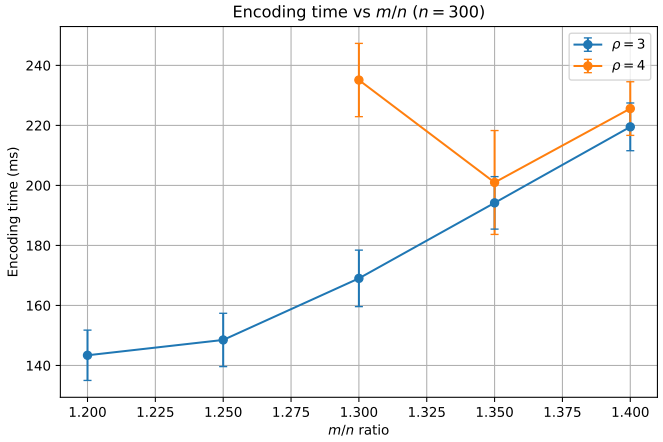


Fig. 5: Encoding time of Sparse-GOKVS for $n = 300$ as a function of α and row sparsity ρ.

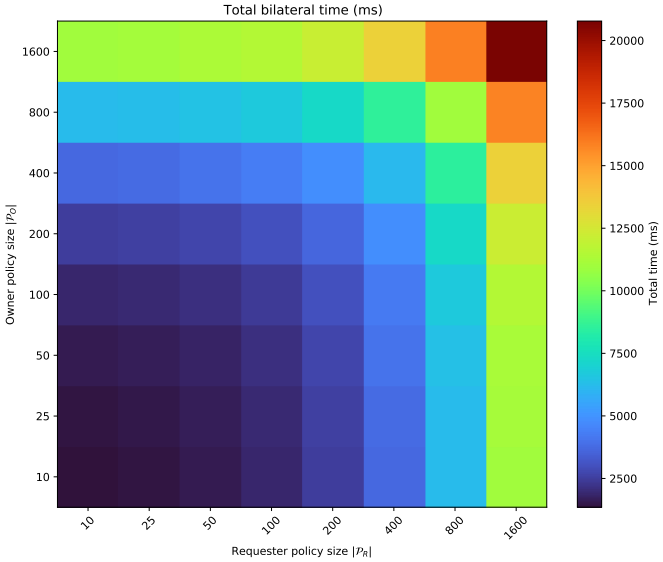


Fig. 6: Total bilateral protocol time (ms) on BN254.

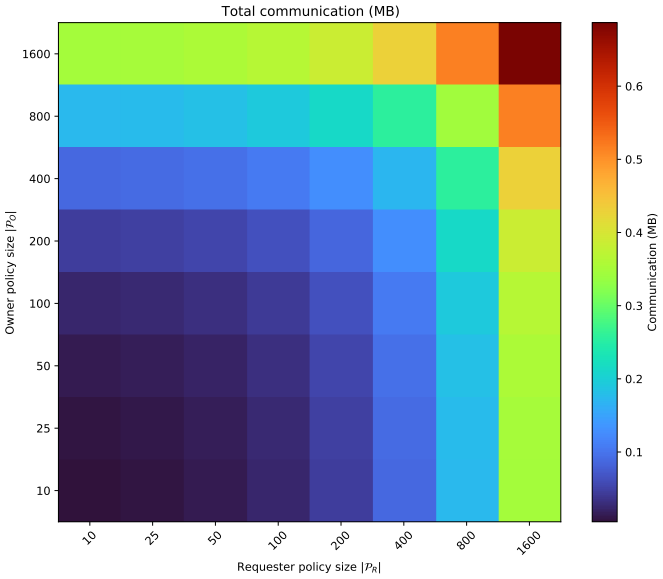


Fig. 7: Total bilateral communication (MB).

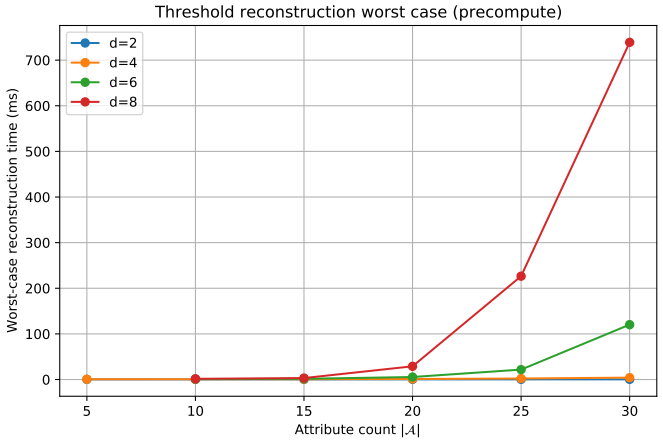


Fig. 8: Worst-case threshold reconstruction time (precompute mode).

C. Bilateral Protocol Scalability

We evaluate the full bilateral protocol with $|\mathcal{P}_O|$ and $|\mathcal{P}_R|$ ranging from 10 to 1600, and $|\mathcal{A}_O| = |\mathcal{A}_R| = 10$, $d = 5$. Figure 6 presents a heatmap of the average total time. The time is dominated by the two GOKVS encodings and the four decapsulation operations (two per party). For example, $(|\mathcal{P}_O|, |\mathcal{P}_R|) = (10, 10)$ completes in 1.35 s, $(400, 400)$ in 6.14 s, and $(1600, 1600)$ in 20.78 s. Figure 7 shows the corresponding total communication (sum of ciphertext sizes and the AES-GCM wrapper), which grows linearly with the sum of the two policy sizes; at $(1600, 1600)$ it is approximately 0.69 MB.

D. Threshold Reconstruction Overhead

The worst-case threshold reconstruction cost is measured by enumerating all $\binom{|\mathcal{A}|}{d}$ candidate subsets before finding the correct one (or exhausting all possibilities). Figure 8 plots the worst-case reconstruction time (precompute mode) for $d \in \{2, 4, 6, 8\}$ and $|\mathcal{A}| \in \{5, \ldots, 30\}$. The exponential growth is evident: for $d = 4$, $|\mathcal{A}| = 30$ the worst-case time reaches several seconds, and for larger d it becomes prohibitive. This cost is inherent to the hidden-threshold subset search and is discussed further in Section VIII.

E. Throughput Summary

Table IV lists the encapsulation and decapsulation throughput for selected parameter combinations on BN254, derived from the corresponding time measurements in the redesigned fair comparison. These values illustrate the tradeoffs discussed in the main text.

F. Comparison Plots in Main Text

The detailed performance comparison between Silent-Share and fuzzy IB-ME is presented in Figure 3 of the main text, which consolidates the six most relevant dimensions. The redesigned fair benchmark ensures that both schemes are measured under identical parameters and with the correct-subset hint mode. Additional per-parameter plots (e.g., encryption time for each $|\mathcal{A}|$ separately) are omitted here to avoid redundancy; the aggregated trends are fully captured by the main figure and by the data files released with the artifact.

G. Subset Search and Theoretical Complexity of Fuzzy IB-ME

We provide a theoretical decomposition of fuzzy IB-ME decryption cost to interpret the experimental results. The online cost can be

TABLE IV: Throughput comparison (operations per second) on BN254.

Parameter	Encapsulation (by $\lvert\mathcal{P}\rvert$)		Decapsulation (by $\lvert\mathcal{A}\rvert$)	
	Silent-Share	Fuzzy IB-ME	Silent-Share	Fuzzy IB-ME
50 / 5	6.05	2.51	4.88	0.80
100 / 10	2.84	1.31	2.54	0.80
200 / 20	1.55	0.67	1.27	0.80
400 / 30	0.77	0.34	0.84	0.80

split into (i) a per-candidate reconstruction cost, and (ii) the expected number of candidate subsets that must be tested before a valid one is found.

Per-candidate reconstruction. Given a candidate set $W_A \subseteq \mathcal{A}$ and a candidate set $W_B \subseteq \mathcal{B}$ of size d, the fuzzy IB-ME decryption algorithm computes several pairing products. In our BN254 implementation, each such candidate evaluation requires exactly $13d$ pairing operations and a constant number of exponentiations and multiplications. Thus the per-candidate cost can be modeled as

$$T_{\text{cand}} = 13d\,T_{\text{pair}} + c_{\exp}\,d\,T_{\exp,\mathbb{G}_T} + O(d),$$

where T_{pair} is the cost of one pairing evaluation and $T_{\exp,\mathbb{G}_T}$ is the cost of one exponentiation in the target group. This model explains the linear growth of fuzzy IB-ME decryption time with d observed in Figure 3(e).

Subset-search overhead. When the correct d-subset is not known a priori, the receiver must enumerate candidate subsets of its attribute set until a valid one is found. Let $n = |\mathcal{A}|$ and let $n' = |\mathcal{A} \cap \mathcal{P}|$ be the number of attributes that actually match the sender's policy. Assuming the receiver tests d-subsets uniformly at random, the probability that a random subset is fully contained in the intersection is

$$p_r = \frac{\binom{n'}{d}}{\binom{n}{d}}.$$

The expected number of trials until the first success is therefore

$$E[N_{\text{trials}}] = \frac{1}{p_r} = \frac{\binom{n}{d}}{\binom{n'}{d}}.$$

When $n' = d$, the search terminates immediately; when $n' \ll n$, the search factor grows combinatorially. This is precisely the behavior shown in Figure 3(f).

Total expected online cost. Combining the two components, the expected total decryption time of fuzzy IB-ME in the absence of a hint can be approximated as

$$T_{\text{total}} \approx E[N_{\text{trials}}] \cdot T_{\text{cand}} = \frac{\binom{n}{d}}{\binom{n'}{d}} \cdot \left(13d\,T_{\text{pair}} + c_{\exp}\,d\,T_{\exp,\mathbb{G}_T}\right).$$

In contrast, Silent-Share's pairing-based authorization cost depends only on $|\mathcal{A}|$, not on d or $|\mathcal{P}|$, and its per-candidate verification requires only $2|\mathcal{A}|$ pairings plus Lagrange interpolation. Thus, under identical search overhead, Silent-Share's total cost grows more slowly with d when $|\mathcal{A}|$ is small, which aligns with the experimental results.

Communication complexity. The ciphertext size of fuzzy IB-ME is dominated by the lists $\{C_{1,i}\}_{i\in P_B}$, $\{C_{2,i}\}_{i\in S_A}$, $\{C_{3,i}\}_{i\in S_A}$, $\{C_{4,i}\}_{i\in S_A}$, $\{C_{5,i}\}_{i\in S_A}$. Each list contains one group element per attribute, yielding a total of $|\mathcal{P}| + 4|\mathcal{A}|$ group elements plus constant overhead. In contrast, Silent-Share's ciphertext contains exactly $2M = 2\lceil 1.25|\mathcal{P}|\rceil$ group elements plus two hash outputs. The measured slopes in Figure 3(b) are consistent with this model.