Lightweight Symmetric-Key Hidden Vector Encryption without Pairings

Sikhar Patranabis and Debdeep Mukhopadhyay

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
sikhar.patranabis@iitkgp.ac.in, debdeep@cse.iitkgp.ernet.in

Abstract. Hidden vector encryption (HVE), introduced by Boneh and Waters in TCC'07, is an expressive sub-class of predicate encryption, that allows conjunctive, subset, range and comparison queries over encrypted data. All existing HVE constructions in the cryptographic literature use bilinear pairings over either composite order or prime order groups. In this paper, we address the open problem of constructing a lightweight symmetric-key HVE scheme that does not use bilinear pairings, but only efficient cryptographic primitives such as pseudo-random functions (PRFs) and block ciphers. The relevance of this problem stems from the implementation and performance overheads for bilinear pairings over composite/prime order groups, which are significantly larger than that for PRFs and block ciphers, in both software and hardware. While lightweight symmetric-key constructions exist for keyword search on encrypted data, we aim to expand the scope of such constructions to support a richer set of query predicates.

In this direction, we present the first lightweight symmetric-key HVE construction that does not use bilinear pairings. Our construction only uses a PRF and a PCPA-secure symmetric-key encryption algorithm, making it amenable to both hardware and software implementations in real-life resource-constrained environments. We prove the selective-simulation-security and adaptive-simulation-security of our construction in the standard model and ideal cipher model, respectively, against probabilistic polynomial-time adversaries that can make an unrestricted number of ciphertext generation and secret-key generation queries.

Keywords: Hidden Vector Encryption, Symmetric-Key, Simulation-Security, Searchable Encryption, Predicate Encryption

1 Introduction

Traditional predicate encryption schemes [1–3] in the public-key setting allow a single public-key to be associated with multiple secret-keys (also referred to as *tokens*), where each secret-key sk_f corresponds to a Boolean predicate $f : \Sigma \longrightarrow \{0, 1\}$ over a set of attributes Σ. Each of these secret-keys are derived from a single master-secret-key msk, which is known only to a trusted third party. A plaintext is defined an attribute-payload message pair $(I, M) \in \Sigma \times \mathcal{M}$, where $\mathcal{M}$ is the payload message space. Decryption works as follows: given a ciphertext C corresponding to a plaintext (I, M), a secret-key sk_f successfully decrypts C and recovers M if and only if $f(I) = 1$. If $f(I) = 0$, decryption returns the failure symbol $\perp$.

Symmetric-Key Predicate Encryption. Predicate encryption can be equivalently defined in the symmetric-key setting as well [4–6]. In this setting, there is no trusted third-party; the data owner herself possesses the master-secret-key msk, and uses it for *both* secret-key generation and encryption. In particular, an adversary does not have unrestricted access to an encryption oracle, since there is no publicly available encryption key. The semantics for decryption, however, are the same as in the public-key setting.

A Generic Framework for Searchable Encryption. Predicate encryption provides a generic framework for searchable encryption over a rich set of query predicates [1, 3]. Consider a scenario where a mail server receives a stream of messages encrypted with the recipient's public key, and is expected to selectively forward a message to the recipient's *urgent* folder if it contains certain keywords. Providing the mail server with the recipient's secret-key for decryption, although simple, is not a desirable solution from the point of view of security. Similarly, consider a credit card payment gateway that handles encrypted transactions, and is required to raise a flag if a certain transaction exceeds a threshold amount. Once again, storing the secret-key on the payment gateway leads to potential security vulnerabilities. A third example is where a data owner stores an encrypted database on the cloud, and requires an untrusted service provider to search for all records that satisfy a certain search predicate f. Predicate encryption aims to provide a generic solution to each of these problems. It allows generation of multiple secret-keys/tokens, that can be used by third parties to evaluate encrypted data *without the knowledge of the master-secret-key*. The plaintext data typically serves as the attribute I, while the payload message M could be some auxiliary information (for example, the recipient-folder name or the flag value in our aforementioned examples).

Identity-Based Encryption and Keyword Search. Identity-based encryption (IBE) [7–9] is the simplest sub-class of public-key predicate encryption. IBE supports a set of equality predicates of the form $f_{\mathsf{id}} : \Sigma \longrightarrow \{0, 1\}$ defined as $f_{\mathsf{id}}(x) = 1$ if and only if $x = \mathsf{id}$. The relationship between IBE and public-key encryption schemes supporting keyword search has been studied in [10, 9]. However, for many applications such as the ones described above, merely supporting keyword search on encrypted data is insufficient. This leads to the need for searchable encryption schemes supporting a more expressive class of predicates.

Hidden Vector Encryption (HVE). Hidden vector encryption (HVE), introduced by Boneh and Waters [1], is a more expressive class of predicate encryption, that supports, in addition to keyword search, conjunctive, subset, range and comparison queries over encrypted data. HVE is essentially a generalization of anonymous IBE [9] and was a precursor to inner-product encryption (IPE) [2, 3] that remains, till date, the most expressive class of predicate encryption schemes to be concretely realized. In HVE, each attribute I is a polynomial-length vector over an alphabet Σ, while a secret-key sk_f corresponds to a predicate vector f over the augmented alphabet $\Sigma_\star = \Sigma \cup \{\star\}$, containing the wildcard character $\star$. Decryption succeeds if the attribute I matches the predicate vector f in every coordinate that is not $\star$. An HVE system is said to be *efficient* if the overhead for ciphertexts and secret-keys is at most polynomial in the security parameter of the system. In addition, an HVE scheme is said to provide *attribute-hiding* security if an adversary, that can convince the system owner to generate ciphertexts and secret-keys corresponding to plaintexts and query predicates of its choice, learns nothing more than the outcomes of evaluating these secret-keys on the corresponding ciphertexts.

1.1 Background and Related Work

In this section, we review existing constructions for HVE in the cryptographic literature, and identify common characteristics for these constructions. We also look at more generic predicate encryption schemes such as inner-product encryption (IPE) that subsume HVE, and corresponding constructions in the literature. Finally, we enumerate the unsolved challenge with respect to HVE constructions that we address in this paper.

HVE Constructions from Bilinear Pairings. The first concrete HVE construction was proposed by Boneh and Waters [1]. Their construction is based on bilinear pairings over composite order groups, and is selectively indistinguishability-secure, under the restriction that the probabilistically

polynomial-time adversary can only make *non-matching* queries to the secret-key generation oracle. For a challenge pair of attribute I_0 and I_1, a non-matching query corresponds to a predicate f such that $f(I_0) = f(I_1) = 0$. Caro et al. [11] subsequently proposed an HVE construction, once again based on composite order bilinear pairings, that removes this restriction, while being adaptively secure. There also exist adaptively indistinguishability-secure HVE constructions based on prime order bilinear pairings; the most notable such construction being that by Park et al. [12]. Their HVE construction achieves constant overhead for secret-keys, and also requires only a constant number of pairing computations for decryption. Finally, while composite order pairings are usually less efficient that their prime-order counterparts, there exist generic techniques by Freeman [13] and, more recently, by Lewko [14] that allow instantiating composite order pairings over prime order bilinear groups.

To the best of our knowledge, the only existing simulation-secure HVE construction was proposed by Caro et al. in [15]. Their scheme is adaptively simulation-secure in the standard model, for a pre-bounded number of ciphertext queries, and an unbounded number of secret-key queries by a probabilistically polynomial-time algorithm. Their construction is again based on composite order bilinear groups. In summary, all existing HVE constructions in the literature are based on bilinear pairings over either composite order or prime order groups, and afford varying efficiency levels in terms of the number of group elements in the ciphertexts and secret-keys, as well as their encryption and decryption performances.

HVE from Inner-Product Encryption. HVE is naturally subsumed by inner-product encryption (IPE) [3]. Nearly all IPE constructions in the literature are based on either bilinear airings [3, 16–19], or on lattices [2, 20, 21]. In particular, there exist a number of symmetric-key IPE constructions [4, 22, 5, 6] that achieve strong indistinguishability-security guarantees under standard hardness assumptions. With regards to simulation-security, the general result of Gorbunov et al. [23] allows deriving simulation-secure constructions for a large class of functions subsuming both IPE and HVE; however, such generic constructions are often less efficient than constructions targeting specific functionalities.

1.2 Our Motivation: Symmetric-Key HVE Constructions without Pairings

The Open problem. In this paper, we address the open problem of constructing a lightweight symmetric-key HVE scheme that does not use bilinear pairings, but only efficient cryptographic primitives such as pseudo-random functions (PRFs) and block ciphers. The relevance of this problem stems from the implementation and performance overheads for bilinear pairings over composite/prime order groups, which are significantly larger than that for PRFs and block ciphers, in both software and hardware. We provide concrete evidence for the same in the forthcoming discussion.

Practical Validation. We choose bilinear pairing modules from the open-source library for pairing-based cryptography (PBC) [1], and compare the execution time for various group and pairing operations, with that for PRFs and block cipher modules chosen from the open-source cryptographic library *Libgcrypto* [2]. We present results for both prime and composite order bilinear pairing modules, which is compared against encryption/decryption timings for AES-128, and output generation timing for a PRF constructed by cascading a SHA-256 module with two AES-128 modules in parallel. Quite evidently, a single pairing operation requires around 10^4x to 10^5x more computation time than a block cipher encryption/decryption or a PRF operation. From an implementation perspective, these observations motivate the design of lightweight HVE schemes *that do not use pairings*. While such symmetric-key constructions exist specifically for single and multi-keyword search on encrypted data [24–27], we aim to expand the scope of lightweight constructions to support a richer set of query predicates. Hence, we choose HVE as our target class of predicate encryption schemes.

[1] https://crypto.stanford.edu/pbc/

[2] https://www.gnupg.org/software/libgcrypt/index.html

Table 1: Performance Comparison on an Intel Core i5-4570 CPU(RAM: 4 Gb, Clock Frequency: 1.6 GHz)

Cryptographic Module	**Library**	**Specification**	**Operation**	**Time Required (in seconds)**
Composite-Order Bilinear Pairing	PBC	$E(\mathbb{F}_q) : y^2 = x^3 + x$ q: 1024 bit prime Order $\#(\mathbb{G}) = p_1p_2$: product of 512 bit primes	Group Multiplication	$t_{\mathbb{G},0} = 2.98 \times 10^{-4}$
			Group Exponentiation	$t_{\mathbb{G},1} = 6.31 \times 10^{-3}$
			Pairing Operations	$t_{\text{Pairing}} = 1.25 \times 10^{-1}$
Prime-Order Bilinear Pairing	PBC	$E(\mathbb{F}_q) : y^2 = x^3 + b$ q: 160 bit prime Order $\#(\mathbb{G}) = r$: 160 bit prime	Group Multiplication	$t_{\mathbb{G},0} = 4.02 \times 10^{-5}$
			Group Exponentiation	$t_{\mathbb{G},1} = 3.91 \times 10^{-4}$
			Pairing Operations	$t_{\text{Pairing}} = 2.40 \times 10^{-2}$
Block Cipher	Libgcrypt	AES-128	Encryption	$t_{\mathsf{SKE.Encrypt}} = 6.15 \times 10^{-7}$
			Decryption	$t_{\mathsf{SKE.Decrypt}} = 6.15 \times 10^{-7}$
Pseudo-Random Function	Libgcrypt	SHA-256 + AES-128 $\times 2$ (in parallel)	Pseudo-Random Output Generation	$t_{\mathsf{PRF}} = 2.32 \times 10^{-6}$

Table 2: Performance and Overhead Comparison of HVE Constructions: Width = l

HVE Scheme	**Overhead Complexity**		
	Master-Key Size	**Ciphertext Size**	**Secret-Key Size**
BW-HVE [1]	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$
KSW-HVE [3]	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$
CIP-HVE [11]	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$
CIJNPP-HVE [15]	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$
PLSL-HVE [12]	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(\lambda)$
Our HVE	$\mathcal{O}(\lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(l \cdot \lambda)$

(a) Spatial Overhead Comparison

HVE Scheme	**Time Complexity for Algorithms**		
	KeyGen	Enc	Dec
BW-HVE [1]	$\mathcal{O}(l \cdot t_{\mathbb{G}})$ [a]	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$	$\mathcal{O}(l \cdot t_{\text{Pairing}})$
KSW-HVE [3]	$\mathcal{O}(l \cdot t_{\mathbb{G}})$	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$	$\mathcal{O}(l \cdot t_{\text{Pairing}})$
CIP-HVE [11]	$\mathcal{O}(l \cdot t_{\mathbb{G}})$	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$	$\mathcal{O}(l \cdot t_{\text{Pairing}})$
CIJNPP-HVE [15]	$\mathcal{O}(l \cdot t_{\mathbb{G}})$	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$	$\mathcal{O}(l \cdot t_{\text{Pairing}})$
PLSL-HVE [12]	$\mathcal{O}(l \cdot t_{\mathbb{G}})$	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$	$\mathcal{O}(l \cdot t_{\mathbb{G}} + t_{\text{Pairing}})$
Our HVE	$\mathcal{O}(l \cdot t_{\mathsf{PRF}})$	$\mathcal{O}(l \cdot t_{\mathsf{PRF}} + t_{\mathsf{SKE.Encrypt}})$	$\mathcal{O}(l \cdot t_{\mathsf{XOR}} + t_{\mathsf{SKE.Decrypt}})$ [b]

(b) Timing Performance Comparison

[a] $t_{\mathbb{G}} = t_{\mathbb{G},0} + t_{\mathbb{G},1}$

[b] t_{XOR} denotes the time required to XOR two λ-bit quantities

1.3 Main Technical Results

In this paper, we present the first lightweight symmetric-key HVE construction that does not use bilinear pairings. Our construction only uses a PRF and a PCPA-secure symmetric-key encryption algorithm, making it amenable to both hardware and software implementations in real-life resource-constrained environments. We prove the selective-simulation-security and adaptive-simulation-security of our construction in the standard model and ideal cipher model, respectively, against probabilistic polynomial-time adversaries that can make an unrestricted number of ciphertext generation and secret-key generation queries.

Tables 2(a) and 2(b) present a comparison of the spatial overhead and timing performance of our proposed HVE construction with that of existing HVE constructions in the literature. While the spatial overhead complexity for our HVE scheme is comparable with that for existing constructions, it has superior timing performance owing to the fact that it avoids all pairing operations, which are more expensive than block cipher and PRF operations, as depicted in Table 1.

1.4 Organization of the Paper

The rest of the paper is organized as follows. Section 2 presents notations, preliminary background material and definitions for symmetric-key predicate encryption that are used throughout the rest of the paper. Section 3 formally defines symmetric-key hidden vector encryption and associated security notions. Section 4 presents our proposed SHVE construction without pairings, and presents the corresponding proofs of security. Finally, Section 5 concludes the paper and presents several open problems.

1.5 Notations Used

We write $x \xleftarrow{R} \chi$ to represent that an element x is sampled uniformly at random from a set $\mathcal{X}$. The output a of a deterministic algorithm $\mathcal{A}$ is denoted by $x \leftarrow \mathcal{A}$ and the output a' of a randomized algorithm $\mathcal{A}'$ is denoted by $x' \xleftarrow{R} \mathcal{A}'$. We refer to $\lambda \in \mathbb{N}$ as the security parameter, and denote by $\mathsf{exp}(\lambda)$, $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ any generic (unspecified) exponential function, polynomial function and negligible function in λ respectively. Note that a function $f : \mathbb{N} \rightarrow \mathbb{N}$ is said to be negligible in λ if for every positive polynomial p, $f(\lambda) < 1/p(\lambda)$ when λ is sufficiently large. Finally, for $a, b \in \mathbb{Z}$ such that $a \leq b$, we denote by $[a, b]$ the set of integers lying between a and b (both inclusive).

2 Background, Preliminaries and Definitions

This section presents basic cryptographic primitives and preliminary background material for symmetric-key predicate encryption that are used throughout the rest of the paper.

Pseudo-Random Function. A pseudo-random function (PRF) is a polynomial-time computable function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{n} \longrightarrow \{0,1\}^{m}$ such that for all polynomial-size algorithms $\mathcal{A}$,

$$\left| \Pr\left[\mathcal{A}^{\mathsf{PRF}(K,\cdot)} = 1 : K \xleftarrow{R} \{0,1\}^{\lambda}\right] - \Pr\left[\mathcal{A}^{g(\cdot)} = 1 : g \xleftarrow{R} \mathsf{Func}(n,m)\right] \right| \leq \mathsf{negl}(\lambda)$$

where the probabilities are taken over all possible choices of K and g.

Symmetric-Key Encryption. A symmetric-key encryption system SKE may be described as an ensemble of the following polynomial-time algorithms:

- $\mathsf{SKE.KeyGen}(1^{\lambda})$: A probabilistic algorithm that takes the security parameter λ as input and outputs a secret-key K.
- $\mathsf{SKE.Encrypt}(K, x)$: A deterministic algorithm that takes as input a key K and a plaintext x. Outputs a ciphertext c.
- $\mathsf{SKE.Decrypt}(K, c)$: A deterministic algorithm that takes as input a key K and a ciphertext c. Outputs the decrypted plaintext x.

In the standard model, we assume that SKE satisfies the security notion of pseudo-randomness against chosen plaintext attacks (PCPA), that guarantees that ciphertexts output by SKE are indistinguishable from the output of a pseudo-random permutation (PRP).

The Ideal Cipher Model. A more idealized model of computation for symmetric-key encryption schemes is the *ideal cipher model*, which is somewhat similar to the well-known random oracle model. While in the random oracle model, one has a publicly accessible random function, in the ideal cipher model, one has a publicly accessible ideal symmetric-key cipher. This ideal cipher is chosen uniformly at random among the family of all possible symmetric-key ciphers that support λ-bit keys, and n-bit plaintext/ciphertext pairs, for some $\lambda, n \in \mathbb{N}$. This is essentially equivalent to having a family of 2^λ independent permutations from $\{0,1\}^n$ to $\{0,1\}^n$. All parties, including the adversary can make both encryption and decryption queries to the ideal cipher, for any given key. Several existing cryptographic schemes have been proven secure in the ideal cipher model [28–30]. While it is possible to design artificial schemes that are secure in this model while being insecure for any concrete instantiations [31], a security proof in the ideal cipher model seems useful because it shows that a scheme is secure against generic attacks, that do not exploit any specific weaknesses of the underlying symmetric-key cipher [32]. The ideal cipher model has been proven to be equivalent to the random oracle model [33, 32]; in particular, one can construct an ideal cipher from a random oracle, and vice versa.

Symmetric-Key Predicate Encryption. A symmetric-key predicate encryption scheme for a class of predicates $\mathcal{F}$ over an attribute space Σ and a payload-message space $\mathcal{M}$ is a quadruple of probabilistic polynomial time algorithms $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, described as follows:

- $\Pi \cdot \mathsf{Setup}\left(1^\lambda\right)$: The setup algorithm takes as input the security parameter λ, and generates the master secret-key msk.
- $\Pi \cdot \mathsf{KeyGen}\left(\mathsf{msk}, f\right)$: The key-generation algorithm takes as input the master secret-key msk and a predicate $f \in \mathcal{F}$, and generates a secret-key sk_f corresponding to f.
- $\Pi \cdot \mathsf{Enc}\left(\mathsf{msk}, I, M\right)$: The encryption algorithm takes as input the master secret-key msk, an attribute $I \in \Sigma$ and a payload-message $M \in \mathcal{M}$, and outputs the corresponding ciphertext C.
- $\Pi \cdot \mathsf{Dec}\left(C, \mathsf{sk}_f\right)$: The decryption algorithm takes as input a ciphertext C and a secret-key sk_f, and outputs either a payload-message $M \in \mathcal{M}$ or the failure symbol $\perp$.

Correctness. A symmetric-key predicate encryption scheme Π is said to be functionally correct if for any security parameter λ, for any predicate $f \in \mathcal{F}$, for any attribute $I \in \Sigma$ and any payload-message $M \in \mathcal{M}$, the following hold with probability at least $1 - \mathsf{negl}(\lambda)$:

1. If $f(I) = 1$, we have $\mathsf{Dec}\left(\mathsf{Enc}\left(\mathsf{msk}, I, M\right), \mathsf{KeyGen}\left(\mathsf{msk}, f\right)\right) = M$.
2. If $f(I) = 0$, we have $\mathsf{Dec}\left(\mathsf{Enc}\left(\mathsf{msk}, I, M\right), \mathsf{KeyGen}\left(\mathsf{msk}, f\right)\right) = \perp$.

where the probability is taken over the internal randomness of $\mathsf{Setup}, \mathsf{KeyGen}$, Enc, and Dec.

3 Symmetric-Key Hidden Vector Encryption (SHVE)

In this section, we formlly define a specific subclass of symmetric-key predicate encryption, namely symmetric-key hidden vector encryption (SHVE), that supports conjunctive query predicates. The basic framework for HVE in the public-key setting was first introduced by Boneh and Waters in [1]; the definitions introduced in this section are a natural adaptation of the same framework in the symmetric-key setting. Let Σ be an alphabet, and let $\Sigma_\star = \Sigma \cup \{\star\}$ be an *augmented* alphabet comprising of the wildcard character $\star$. Let λ be a security parameter, and let $l \leq \mathsf{poly}(\lambda)$ be a predefined constant. An SHVE scheme $\Pi_{\mathsf{SHVE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ comprises of attributes of the form $I = (x_1, \cdots, x_l) \in \Sigma^l$, and query predicates of the form $f = (y_1, \cdots, y_l) \in \Sigma_\star^l$, such that:

$$f(I) = \begin{cases} 1 \text{ if } \bigwedge_{j=1}^{j=l} \left(y_j = x_j \bigvee y_j = \star\right) \\ 0 \text{ otherwise} \end{cases}$$

Essentially, $f(I) = 1$ if and only if the attribute I matches the predicate vector f in every coordinate that is not $\star$. The constant l is denoted as the *width* of the SHVE scheme.

3.1 Security Definitions for SHVE

This section formally defines the security of an SHVE scheme. We begin by presenting the following auxiliary definitions for an SHVE scheme $\Pi_{\mathsf{SHVE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of width l defined over an alphabet Σ and a payload message space $\mathcal{M}$:

Definition 3.1 (Query History). *Let $(I, M) \in \Sigma^l \times \mathcal{M}$ be an attribute-message pair for Π_{SHVE}, and let $f_1, \cdots, f_Q \in \Sigma_\star^l$ be a set of query predicates. A query history $\mathcal{H}$ for Π_{SHVE} is a tuple of the form $((I, M), f_1, \cdots, f_Q)$.*

Definition 3.2 (Wildcard Pattern). *Given a query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$ for Π_{SHVE}, let $f_j = (y_{j,1}, \cdots, y_{j,l})$ for each $j \in [1, Q]$, where l is the width of Π_{SHVE}. The wildcard pattern $\alpha(\mathcal{H})$ is a set of boolean values $\{\alpha_{j,k}\}_{j \in [1,Q], k \in [1,l]}$ such that:*

$$\alpha_{j,k} = \begin{cases} 1 \text{ if } y_{j,k} = \star \\ 0 \text{ otherwise} \end{cases}$$

Definition 3.3 (Query Pattern). *Given a query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$ for Π_{SHVE}, let $f_j = (y_{j,1}, \cdots, y_{j,l})$ for each $j \in [1, Q]$, where l is the width of Π_{SHVE}. The query pattern $\delta(\mathcal{H})$ is a set of boolean values $\{\delta_{i,j,k}\}_{i,j \in [1,Q], k \in [1,l]}$ such that:*

$$\delta_{i,j,k} = \begin{cases} 1 \text{ if } y_{i,k} = y_{j,k} \\ 0 \text{ otherwise} \end{cases}$$

Definition 3.4 (Decryption Pattern). *Given a query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$ for Π_{SHVE}, let sk_{f_j} be the secret-key corresponding to the predicate f_j, where $j \in [1, Q]$. Also, let C denote a ciphertext obtained upon encryption of the plaintext (I, M). The decryption pattern $\chi(\mathcal{H})$ is a set of values $\{\chi_1, \cdots, \chi_Q\}$, such that, for $j \in [1, Q]$, χ_j is the output of decrypting C using sk_{f_j}.*

The Leakage Pattern for SHVE. Consider an adversary against an SHVE scheme that can obtain the ciphertext C corresponding to a plaintext (I, M), and has access to a secret-key generation oracle which it can query with polynomially many query vectors $f_1, \cdots, f_Q$. Quite evidently, the decryption pattern $\chi(\mathcal{H})$ corresponding to the query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$ is a trivial leakage, since the adversary can easily observe the outcome of decrypting the ciphertext C using the secret-keys $\{\mathsf{sk}_{f_j}\}_{j \in [1,Q]}$ generated by the oracle. In addition, for an SHVE scheme with a deterministic secret-key generation algorithm, as presented in this paper, the query pattern $\delta(\mathcal{H})$ is also leaked [26]. Finally, the wildcard pattern is also a trivial leakage in most existing HVE schemes in the literature [1, 11, 12]. Constructing HVE schemes that hide the wildcard pattern is an open problem [34], and is not the main focus of this paper. Consequently, in this paper, we consider the *allowable leakage pattern* for an SHVE scheme corresponding to a query history $\mathcal{H}$ to be the tuple $(\alpha(\mathcal{H}), \delta(\mathcal{H}), \chi(\mathcal{H}))$. In other words, these leakages are trivial and do not constitute a threat to the security of an SHVE scheme.

We now formally define the security notions for SHVE in the simulation-based setting. The definition comprises of a pair of experiments - a *real experiment* and a *simulation experiment*. In the real experiment, a probabilistic polynomial time algorithm $\mathcal{A}$ interacts with a challenger who knows the master-secret-key for the SHVE scheme. The algorithm $\mathcal{A}$ makes polynomially many ciphertext generation and secret-key generation queries to the challenger, based on a query history comprising of plaintexts and predicates chosen by $\mathcal{A}$. In the simulation experiment, the role of the challenger is played by a probabilistic polynomial time simulator algorithm $\mathcal{S}$ that only has access to the leakage pattern corresponding to the query history chosen by $\mathcal{A}$, and *not the master-secret-key*. The simulation-security of the SHVE scheme is defined in terms of the computational indistinguishability of the views of algorithm $\mathcal{A}$ in the two experiments.

Adaptively Simulation-Secure SHVE. We first define the adaptive notion of simulation-security for SHVE. For a security parameter $\lambda \in \mathbb{N}$, an SHVE scheme $\Pi_{\mathsf{SHVE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, and a non-uniform probabilistic polynomial-time algorithm $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{Q'+Q-1})$ (where $Q', Q \leq \mathsf{poly}(\lambda)$), we define the experiment $\mathsf{Expt}^{\mathrm{ada,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ as follows:

1. $\mathsf{msk} \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{Setup}\left(1^\lambda\right)$
2. $\left((I_1, M_1), \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_0\left(1^\lambda\right)$.
3. $C_1 \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{Enc}\left(\mathsf{msk}, I, M\right)$.
4. For $i \in [2, Q']$:
   (a) $\left((I_i, M_i), \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{i-1}\left(C_1, \cdots, C_{i-1}, \mathsf{state}_{\mathcal{A}}\right)$
   (b) $C_i \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{Enc}\left(\mathsf{msk}, I_i, M_i\right)$
5. $\left(f_1, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{Q'}\left(C_1, \cdots, C_{Q'}, \mathsf{state}_{\mathcal{A}}\right)$
6. $\mathsf{sk}_{f_1} \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{KeyGen}\left(\mathsf{msk}, f_1\right)$
7. For $j \in [2, Q]$:
   (a) $\left(f_j, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{Q'+j-1}\left(C_1, \cdots, C_{Q'}, \mathsf{sk}_{f_1}, \cdots, \mathsf{sk}_{f_{j-1}}, \mathsf{state}_{\mathcal{A}}\right)$
   (b) $\mathsf{sk}_{f_j} \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{KeyGen}\left(\mathsf{msk}, f_j\right)$
8. Output $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right)$

Observe that the algorithm $\mathcal{A}$ can specify the query vectors $f_1, \cdots, f_Q$ adaptively, after seeing the secret-keys corresponding to the previously queried vectors. Also, note that the variable $\mathsf{state}_{\mathcal{A}}$ captures the updated state of the algorithm $\mathcal{A}$ at various stages during the experiment.

Now, let $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_{Q'+Q-1})$ be a non-uniform probabilistic polynomial-time simulator, and let $\mathcal{H}_{i,j}$ denote the query history $\left((I_i, M_i), f_1, \cdots, f_j\right)$ for $i \in [1, Q']$ and $j \in [1, Q]$. We define a second experiment $\mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$ (where $\mathcal{A}$ is the same non-uniform probabilistic poly-time algorithm as described above) as follows:

1. $\left((I_1, M_1), \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_0\left(1^\lambda\right)$.
2. $\left(C_1, \mathsf{state}_{\mathcal{S}}\right) \xleftarrow{R} \mathcal{S}_0\left(1^\lambda\right)$
3. For $i \in [2, Q']$:
   (a) $\left((I_i, M_i), \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{i-1}\left(C_1, \cdots, C_{i-1}, \mathsf{state}_{\mathcal{A}}\right)$
   (b) $\left(C_i, \mathsf{state}_{\mathcal{S}}\right) \xleftarrow{R} \mathcal{S}_{i-1}\left(\mathsf{state}_{\mathcal{S}}\right)$
4. $\left(f_1, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{Q'}\left(C, \mathsf{state}_{\mathcal{A}}\right)$
5. $\left(\mathsf{sk}_{f_1}, \mathsf{state}_{\mathcal{S}}\right) \xleftarrow{R} \mathcal{S}_{Q'}\left(\{\alpha\left(\mathcal{H}_{i,1}\right), \delta\left(\mathcal{H}_{i,1}\right), \chi\left(\mathcal{H}_{i,1}\right)\}_{i\in[1,Q']}, \mathsf{state}_{\mathcal{S}}\right)$
6. For $j \in [2, Q]$:
   (a) $\left(f_j, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}_{Q'+j-1}\left(C, \mathsf{sk}_{f1}, \cdots, \mathsf{sk}_{f_j}, \mathsf{state}_{\mathcal{A}}\right)$
   (b) $\left(\mathsf{sk}_{f_j}, \mathsf{state}_{\mathcal{S}}\right) \xleftarrow{R} \mathcal{S}_{Q'+j-1}\left(\{\alpha\left(\mathcal{H}_{i,j}\right), \delta\left(\mathcal{H}_{i,j}\right), \chi\left(\mathcal{H}_{i,j}\right)\}_{i\in[1,Q']}, \mathsf{state}_{\mathcal{S}}\right)$
7. Output $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right)$

Once again, the variable $\mathsf{state}_{\mathcal{S}}$ captures the updated state of the simulator $\mathcal{S}$ at various stages during the experiment.

Definition 3.5 (Adaptively Simulation-Secure SHVE). *An SHVE scheme Π_{SHVE} is said to be adaptively simulation-secure if for any non-uniform probabilistic polynomial-time algorithm $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{Q'+Q-1})$, there exists a non-uniform probabilistic polynomial time simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_{Q'+Q-1})$, such that the ensemble distribution:*

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$$

and the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$$

are computationally indistinguishable.

Selectively Simulation-Secure SHVE. We also define the weaker, selective notion of simulation security for SHVE where the adversary must specify the entire query history non-adaptively at the beginning of both the real and simulation experiments. For a security parameter $\lambda \in \mathbb{N}$, an SHVE scheme $\Pi_{\mathsf{SHVE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, and a uniform probabilistic polynomial-time algorithm $\mathcal{A}$, we define the experiment $\mathsf{Expt}^{\mathrm{sel,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ as follows:

1. $\mathsf{msk} \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{Setup}\left(1^\lambda\right)$
2. $\left((I_1, M_1), \cdots, (I_{Q'}, M_{Q'}), f_1, \cdots, f_Q, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}\left(1^\lambda\right)$
3. For $i \in [1, Q']$, $C_i \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{Enc}\left(\mathsf{msk}, I_i, M_i\right)$
4. For $j \in [1, Q]$, $\mathsf{sk}_{f_j} \xleftarrow{R} \Pi_{\mathsf{SHVE}} \cdot \mathsf{KeyGen}\left(\mathsf{msk}, f_j\right)$
5. Output $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right)$

where $Q', Q \leq \mathsf{poly}(\lambda)$. Once again, the variable $\mathsf{state}_{\mathcal{A}}$ captures the updated state of the algorithm $\mathcal{A}$ at various stages during the experiment. Observe that in this experiment, the algorithm $\mathcal{A}$ must specify the query vectors $f_1, \cdots, f_Q$ non-adaptively, before seeing the secret-keys corresponding to any of them.

Now, let $\mathcal{S}$ be a uniform probabilistic polynomial-time simulator. Also, recall that $\mathcal{H}_{i,j}$ denotes the query history $\left((I_i, M_i), f_1, \cdots, f_j\right)$ for $i \in [1, Q']$ and $j \in [1, Q]$. We define a second experiment $\mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$ (where $\mathcal{A}$ is the same non-uniform probabilistic poly-time algorithm as described above) as follows:

1. $\left((I_1, M_1), \cdots, (I_{Q'}, M_{Q'}), f_1, \cdots, f_Q, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathcal{A}\left(1^\lambda\right)$
2. $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}\right) \xleftarrow{R} \mathcal{S}\left(1^\lambda, \{\alpha\left(\mathcal{H}_{i,Q}\right), \delta\left(\mathcal{H}_{i,Q}\right), \chi\left(\mathcal{H}_{i,Q}\right)\}_{i\in[1,Q']}\right)$
3. Output $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right)$

Definition 3.6 (Selectively Simulation-Secure SHVE). *An SHVE scheme Π_{SHVE} is said to be selectively simulation-secure if for any uniform probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a uniform probabilistic polynomial time simulator $\mathcal{S}$, such that the ensemble distribution:*

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$$

and the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$$

are computationally indistinguishable.

4 Our SHVE Construction without Pairings

In this section, we present our SHVE construction $\Pi_{\mathsf{SHVE\text{-}1}}$ in the standard model. Unlike existing HVE constructions in the literature, our construction does not use bilinear pairings. Instead it ues the following cryptographic primitives:

- A pseudo-random function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{\lambda+\log\lambda} \longrightarrow \{0,1\}^\lambda$.
- A PCPA-secure symmetric-key encryption scheme SKE with both the key-space and the plaintext-space being $\{0,1\}^\lambda$.

where $\lambda \in \mathbb{N}$ is the security parameter. This makes our construction more lightweight and efficiently implementable than its existing pairing-based counterparts.

4.1 Construction Details

The detailed construction for our proposed SHVE scheme $\Pi_{\mathsf{SHVE\text{-}1}}$ of width $l < \mathsf{poly}(\lambda)$, defined over an alphabet $\Sigma = \{0,1\}^\lambda$ and a message space $\mathcal{M} \subset \{0,1\}^\lambda$, is presented next. For simplicity of presentation, we initially assume that $\mathcal{M}$ is a small subset of $\{0,1\}^\lambda$; in particular, $|\mathcal{M}| < 2^\lambda/\mathsf{poly}(\lambda)$. Subsequently, we present a discussion on how this restriction may be relaxed in our construction.

- $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{Setup}\left(1^\lambda\right)$: On input the security parameter λ, the setup algorithm uniformly samples $\mathsf{msk} \xleftarrow{R} \{0,1\}^\lambda$, and outputs the same.

- $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{KeyGen}\left(\mathsf{msk}, f\right)$: The key-generation algorithm takes as input the master secret-key msk and a predicate $f = (y_1, \cdots, y_l)$, such that $y_j \in \{0,1\}^\lambda \cup \{\star\}$, for each $j \in [1,l]$. The algorithm generates a secret-key:
$$\mathsf{sk}_f = \left(\left(d_1, b_1\right), \cdots, \left(d_l, b_l\right)\right)$$
such that for each $k \in [1,l]$, we have $d_k = \mathsf{PRF}\left(\mathsf{msk}, y_k||k\right)$, while $b_k = 1$ if $y_k = \star$, and 0 otherwise. The secret-key sk_f is produced as the output.

- $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{Enc}\left(\mathsf{msk}, I, M\right)$: The encryption algorithm takes as input the master secret-key msk, an attribute $I = (x_1, \cdots, x_l) \in \{0,1\}^l$ and a payload-message $M \in \mathcal{M}$. It uniformly samples $K \xleftarrow{R} \mathsf{SKE.KeyGen}(1^\lambda)$, and breaks it into l random shares $K_1, \cdots, K_l$ such that:
$$K = K_1 \oplus K_2 \oplus \cdots \oplus K_l$$
It then outputs the ciphertext $C = \left(\{c_{k,0}, c_{k,1}\}_{k\in[1,l]}, c_{l+1}\right)$ such that:
$$\begin{aligned} c_{k,0} &= \mathsf{PRF}\left(\mathsf{msk}, x_k||k\right) \oplus K_k \text{ for } k \in [1,l] \\ c_{k,1} &= \mathsf{PRF}\left(\mathsf{msk}, \star||j\right) \oplus K_k \text{ for } k \in [1,l] \\ c_{l+1} &= \mathsf{SKE.Encrypt}\left(K, M\right) \end{aligned}$$

- $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{Dec}\left(C, \mathsf{sk}_f\right)$: The decryption algorithm takes as input a ciphertext $C = \left(\{c_{k,0}, c_{k,1}\}_{k\in[1,l]}, c_{l+1}\right)$ and a secret-key $\mathsf{sk}_f = \left(\left(d_1, b_1\right), \cdots, \left(d_l, b_l\right)\right)$. It first computes:
$$K' = \left(c_{1,b_1} \oplus d_1\right) \oplus \left(c_{2,b_2} \oplus d_2\right) \oplus \cdots \oplus \left(c_{l,b_l} \oplus d_l\right)$$
and then $M' = \mathsf{SKE.Decrypt}\left(K', c_{l+1}\right)$. If $M' \in \mathcal{M}$, it outputs M', else it outputs the failure symbol $\perp$.

Correctness. Consider a ciphertext $C = \left(\{c_{k,0}, c_{k,1}\}_{k\in[1,l]}, c_{l+1}\right)$ corresponding to an attribute $I = (x_1, \cdots, x_l)$ and a secret-key $\mathsf{sk}_f = \left(\left(d_1, b_1\right), \cdots, \left(d_l, b_l\right)\right)$ corresponding to a query-predicate $f = (y_1, \cdots, y_l)$. The following observations establish the correctness of $\Pi_{\mathsf{SHVE\text{-}1}}$:

1. If $f(I) = 1$, for each $k \in [1,l]$, we have either $y_k = x_k$, or $y_k = \star$. It is straightforward to observe that in such a case, for each $k \in [1,l]$, if $b_k = 0$, then $c_{k,0} = d_k \oplus K_k$, while if $b_k = 1$, then $c_{k,1} = d_k \oplus K_k$. Hence, we have:
$$\begin{aligned} K' &= \left(c_{1,b_1} \oplus d_1\right) \oplus \left(c_{2,b_2} \oplus d_2\right) \oplus \cdots \oplus \left(c_{l,b_l} \oplus d_l\right) \\ &= K_1 \oplus K_2 \oplus \cdots \oplus K_l \\ &= K \end{aligned}$$
which in turn implies that $M' = \mathsf{SKE.Decrypt}\left(K, c_{l+1}\right) = M$. Thus, if $f(I) = 1$, the payload message is recovered correctly.

2. If $f(I) = 0$, then there must exist some $k \in [1, l]$ such that $y_k \neq x_k$ and $y_k \neq \star$. Consequently, it follows that $b_k = 0$, and $c_{k,0} = d_k \oplus K'_k$, for some uniformly random $K'_k \neq K_k$. This, in turn, implies that $K' = K \oplus K_k \oplus K'_k$, hence the recovered message $M' = \mathsf{SKE.Decrypt}\,(K', c_{l+1})$ is a uniformly random string in $\{0,1\}^\lambda$. Since the payload message space $\mathcal{M}$ is assumed to be small, the probability that a uniformly random M' lies in $\mathcal{M}$ is negligible. In other words, if $f(I) = 0$, the decryption algorithm returns $\perp$ with overwhelmingly large probability.

Relaxing the Restriction on $|\mathcal{M}|$. Note that our proposed SHVE scheme $\Pi_{\mathsf{SHVE\text{-}1}}$ can be easily modified to remove the aforementioned restriction on the size of the payload message space $\mathcal{M}$. In particular, the ciphertext C may be augmented to also contain an additional component $H(K)$, where H is a collision-resistant hash function and K is the secret-key used to encrypt the payload message M. The decryption procedure is modified accordingly: upon computing $K' = (c_{1,b_1} \oplus d_1) \oplus \cdots \oplus (c_{l,b_l} \oplus d_l)$ as described above, the decryptor checks if $H(K) = H(K')$. If yes, the decryptor returns the payload message $M' = \mathsf{SKE.Decrypt}\,(K', c_{l+1})$. Else, it returns $\perp$. The collision resistance of H ensures that the event $H(K') = H(K)$ for $K' \neq K$ occurs with negligible probability. The modified construction can thus accommodate the unrestricted payload message space $\mathcal{M} = \{0,1\}^\lambda$.

4.2 Selective Simulation-Security of Our Proposed Scheme in the Standard Model

We state and prove the following theorem for the selective security of $\Pi_{\mathsf{SHVE\text{-}1}}$:

Theorem 4.1 *The SHVE scheme $\Pi_{\mathsf{SHVE\text{-}1}}$ is selectively simulation-secure in the standard model under the assumption that* PRF *is a pseudo-random function and* SKE *is a PCPA-secure symmetric-key encryption scheme.*

Proof. We present the detailed proof of security. In particular, we show that for any uniform probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a uniform probabilistic polynomial time simulator $\mathcal{S}$, such that the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$$

and the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$$

are computationally indistinguishable, where the experiments $\mathsf{Expt}^{\mathrm{sel,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$ are as described in Section 3.1. In particular, we construct the simulator $\mathcal{S}$ in the experiment $\mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$. We first consider the simplified scenario where $\mathcal{A}$ makes $Q' = 1$ ciphertext queries, and $Q \leq \mathsf{poly}(\lambda)$ secret-key queries. We then present a generalized argument to show that the simulator algorithm can be easily extended to also answer polynomially many ciphertext queries.

The Inputs to the Simulator $\mathcal{S}$. Since $Q' = 1$, we assume that the algorithm $\mathcal{A}_0$ queries the simulator with a query history of the form $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$. The simulator $\mathcal{S}$ receives as advice the security parameter 1^λ, along with the tuple $(\alpha(\mathcal{H}), \delta(\mathcal{H}), \chi(\mathcal{H}))$, that is, the wildcard pattern, the query pattern and the decryption pattern, respectively, corresponding to $\mathcal{H}$. For clarity of presentation, we recall the details for $\alpha(\mathcal{H})$, $\delta(\mathcal{H})$ and $\chi(\mathcal{H})$ below:

- In the query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$, let $f_j = (y_{j,1}, \cdots, y_{j,l})$ for $j \in [1, Q]$, where l is the width of Π_{SHVE}. The wildcard pattern $\alpha(\mathcal{H})$ received by $\mathcal{S}$ is a set of boolean values $\{\alpha_{j,k}\}_{j\in[1,Q],k\in[1,l]}$ such that:

$$\alpha_{i,j} = \begin{cases} 1 \text{ if } y_{j,k} = \star \\ 0 \text{ otherwise} \end{cases}$$

- In the query history $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$, let $f_j = (y_{j,1}, \cdots, y_{j,l})$ for $j \in [1, Q]$, where l is the width of Π_{SHVE}. The query pattern $\delta(\mathcal{H})$ received by $\mathcal{S}$ is a set of boolean values $\{\delta_{i,j,k}\}_{i,j \in [1,Q], k \in [1,l]}$ such that:
$$\delta_{i,j,k} = \begin{cases} 1 \text{ if } y_{i,k} = y_{j,k} \\ 0 \text{ otherwise} \end{cases}$$
- Let sk_{f_j} be the secret-key corresponding to f_j in $\mathcal{H} = ((I, M), f_1, \cdots, f_Q)$, where $j \in [1, Q]$. Also, let C denote a ciphertext obtained upon encryption of the plaintext (I, M). The decryption pattern $\chi(\mathcal{H})$ received by $\mathcal{S}$ is a set of values $\{\chi_1, \cdots, \chi_Q\}$, such that, for $j \in [1, Q]$, χ_j is the output of decrypting C using sk_{f_j}. Quite evidently, if $f_j(I) = 1$, χ_j is M, else $\chi_j = \perp$.

Simulating the Secret-Keys. The simulator $\mathcal{S}$ now generates the secret-keys $\{\mathsf{sk}_{f_j}\}_{j \in [1,Q]}$ using the procedure described in Phase-1 of Algorithm 1. Note that since the secret-key generation algorithm $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{KeyGen}.$ is deterministic, the simulator must ensure that the secret-keys generated are consistent with the inter-relations between the corresponding predicate vectors, as well as the locations of the wildcard characters. This is achieved in Algorithm 1 using the wildcard pattern and the query pattern. In particular, the values $A_1, A_2, \cdots, A_l$ are used to simulate $\mathsf{PRF}(\mathsf{msk}, \star||k)$ for $k \in [1, l]$, while the consistency check Line 7 ensures that, if two predicate vectors f_i and f_j match in the k^{th} coordinate, then the corresponding secret-keys sk_{f_i} and sk_{f_j} also match in the k^{th} coordinate. This is in accordance with the deterministic nature of $\Pi_{\mathsf{SHVE\text{-}1}} \cdot \mathsf{KeyGen}$.

Simulating the Ciphertext. The final step for the simulator $\mathcal{S}$ is to simulate the ciphertext C. Note that the plaintext (I, M) is not known to $\mathcal{S}$; the only available information are the various leakage patterns, and the secret-keys simulated as described above. As we demonstrate next, this information is enough to simulate C:

- Observe that if for some $j \in [1, Q]$, $f_j(I) = 1$, then the attribute I must match f_j in all coordinates that do not contain wildcard characters. In particular, suppose $I = (x_1, \cdots, x_l)$ and $f_j = (y_{j,1}, \cdots, y_{j,l})$, such that $f_j(I) = 1$. If for some $k \in [1, l]$, $y_{j,k} \neq \star$, it must be the case that $x_k = y_{j,k}$. Additionally, the information that $f_j(I) = 1$ can be easily inferred by $\mathcal{S}$ from the fact that $\chi_j \neq \perp$; while the information of whether the coordinate $y_{j,k} \neq \mathit{star}$ in f_j is available from the wildcard pattern entry $\alpha_{j,k}$. This information allows $\mathcal{S}$ to simulate the ciphertext component $c_{k,0}$ for $k \in [1, l]$ as follows:

  1. $\mathcal{S}$ first checks for an $f_j = (y_{j,1}, \cdots, y_{j,l})$ such that such that $f_j(I) = 1$ and $y_{j,k} \neq \star$ (see Step 22 of Algorithm 1). If such a matching f_j is found, $\mathcal{S}$ uses the corresponding secret-key component $d_{j,k}$ from Phase-1 to simulate $c_{k,0}$. This ensures that decrypting C using sk_{f_j} correctly recovers the payload message M.
  2. If no such matching f_j is found for a given k, it must be the case that for all $j \in [1, Q]$, either $f_j(I) = 0$ or, $f_j(I) = 1$ but the corresponding component $y_{j,k} = \star$. In either of these scenarios, the output of decrypting C using sk_{f_j} is independent of $c_{k,0}$; consequently, the simulator randomly samples $c_{k,0}$.

- The next step is to simulate the components $c_{k,1}$, for $k \in [1, l]$. As already mentioned, the values $A_1, A_2, \cdots, A_l$ are used by $\mathcal{S}$ to simulate $\mathsf{PRF}(\mathsf{msk}, \star||k)$, and hence, $c_{k,1}$, for $k \in [1, l]$.

- The final step is to simulate the ciphertext component c_{l+1}. Recall that the decryption pattern is a set of values $\{\chi_1, \cdots, \chi_Q\}$, such that, for $j \in [1, Q]$, χ_j is the output of decrypting C using sk_{f_j}. If, for some $j \in [1, Q]$, $f_j(I) = 1$, then χ_j reveals M to $\mathcal{S}$, which is used to generate the component $c_{l+1} = \mathsf{SKE.Encrypt}(K, M)$ as in the real world. This ensures that decrypting C using sk_{f_j}, the payload message M is recovered correctly. If, on the other hand, $f_j(I) = 0$ for each $j \in [1, Q]$, decrypting C with any of the secret-keys $\{\mathsf{sk}_{f_j}\}_{j \in [1,Q]}$ anyways returns $\perp$. Hence, in such a scenario, $\mathcal{S}$ uses a uniformly random M' for simulating c_{l+1}.

Algorithm 1 Simulator S for Single Ciphertext Query and Q Secret-key Queries

Input: $\left(1^{\lambda}, \alpha\left(\mathcal{H}\right), \delta\left(\mathcal{H}\right), \chi\left(\mathcal{H}\right)\right)$, where $\mathcal{H} = \left(\left(I, M\right), f_1, \cdots, f_Q\right)$
Output: $\left(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}\right)$

Choose $A_1, A_2, \cdots, A_l \xleftarrow{R} \{0,1\}^{\lambda}$

Phase 1 - Simulation of Secret-Keys

Initialize a $Q \times l$ matrix d to empty
for each $j \in [1, Q]$ **do**
  for each $k \in [1, l]$ **do**
    if $\alpha_{j,k} = 1$ **then**
      $d_{j,k} \leftarrow A_k$
    else if there exists $i \in [1, j-1]$ such that $\delta_{i,j,k} = 1$ **then**
      $d_{j,k} \leftarrow d_{i,k}$
    else
      $d_{j,k} \xleftarrow{R} \{0,1\}^{\lambda}$
    end if
  end for
  $\mathsf{sk}_{f_j} \leftarrow \left(\left(d_{j,1}, \alpha_{j,1}\right), \cdots, \left(d_{j,l}, \alpha_{j,l}\right)\right)$
end for

Phase 2 - Simulation of Ciphertext

if there exists $j \in [1, Q]$ such that $\chi_j \neq \perp$ **then**
  $M \leftarrow \chi_j$
else
  $M \xleftarrow{R} \{0,1\}^{\lambda}$
end if
Uniformly sample $K \xleftarrow{R} \mathsf{SKE.KeyGen}(1^{\lambda})$, and break it into l random shares $K_1, \cdots, K_l$
for each $k \in [1, l]$ **do**
  if there exists $j \in [1, Q]$ such that $\chi_j \neq \perp$ and $\alpha_{j,k} = 0$ **then**
    $c_{k,0} \leftarrow d_{j,k} \oplus K_k$
  else
    $c_{k,0} \xleftarrow{R} \{0,1\}^{\lambda}$
  end if
  $c_{k,1} \leftarrow A_k \oplus K_k$
end for
$c_{l+1} \leftarrow \mathsf{SKE.Encrypt}\left(K, M\right)$
$C \leftarrow \left(\{c_{k,0}, c_{k,1}\}_{k\in[1,l]}, c_{l+1}\right)$

Final Output

Return $\left(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}\right)$

Functional Correctness of the Simulation. The consistency properties adhered to by $\mathcal{S}$ when generating the ciphertext C and the secret-keys $\{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}$ suffice to ensure that decrypting C using these secret-keys yields the desired outputs. In particular, for each $j \in [1, Q]$ such that $f_j(I) = 1$, decryption will successfully recover the payload message M, while in the remaining cases, decryption will return $\perp$ with overwhelmingly large probability.

The Indistinguishability Argument. It remains to argue that a probabilistic polynomial-time distinguisher $\mathcal{D}$ cannot computationally distinguish $\left(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ from $\left(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{sel,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$. We present the following arguments to establish this indistinguishability property:

- Since the state variable $\mathsf{state}_{\mathcal{A}}$ does not include the master-secret-key msk with all but negligible probability, the secret-keys generated in the real mode of the experiment using the pseudo-random function PRF are indistinguishable from the uniformly random secret-keys generated by the simulator $\mathcal{S}$. Otherwise, one could distinguish between the output of PRF and a uniformly random string of size λ without knowing the corresponding secret-key msk. By the same argument, the components $\{c_{k,0}\}_{k\in[1,l]}$ and $\{c_{k,1}\}_{k\in[1,l]}$ of the ciphertext C in the real experiment must also be indistinguishable from those generated by the simulator $\mathcal{S}$.

- Finally, if there exists at least one $j \in [1, Q]$ such that $f_j(I) = 1$, $\mathcal{S}$ gains the knowledge of the payload message M, which is used to generate the component $c_{l+1} = \mathsf{SKE.Encrypt}\,(K, M)$ as in the real world. In this case, indistinguishability holds trivially. If, on the other hand, $f_j(I) = 0$ for each $j \in [1, Q]$, decrypting C with any of the secret-keys $\{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}$ does not reveal the secret-key K used to encrypt M. Now, since the state variable $\mathsf{state}_{\mathcal{A}}$ does not include K with all but negligible probability, the PCPA security of SKE guarantees the indistinguishability of c_{l+1} in the real and simulation experiments.

Generalization to Polynomially many Ciphertext Queries. The simulator $\mathcal{S}$ can be extended to address polynomially many ciphertext queries of the form $\{(I_i, M_i)\}_{i\in[1,Q']}$. The detailed simulation algorithm for the generalization is presented in Appendix A; we provide a brief overview here. *The simulator $\mathcal{S}$ essentially repeats the same ciphertext simulation phase in* Algorithm 1 *for each such query.* As per our security definition, the simulator $\mathcal{S}$ receives as input the decryption pattern corresponding to each ciphertext query (I_i, M_i). Hence, it can either infer the corresponding payload message M_i from the decryption pattern, or randomly sample the same, as in Steps 16 through 19 of Algorithm 1. Additionally, the components for C_i are chosen consistently with the secret-keys $\{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}$, following Steps 22 through 26 of Algorithm 1. This ensures that, if $f_j(I_i) = 1$ for some $j \in [1, Q]$, then decrypting C_i using sk_{f_j} correctly recovers M_i. Also, the ciphertext components corresponding to the wildcard characters are simulated consistently as in Step 27 of Algorithm 1. Finally, since a different random K_i will be used to mask the components of each ciphertext C_i (see Step 20 of Algorithm 1), the simulation of two ciphertexts C_i and $C_{i'}$ can proceed independent of whether the corresponding attributes I_i and $I_{i'}$ match in one or more components. . This completes the proof of Theorem 4.1. □

4.3 Adaptive Simulation-Security of Our Proposed Scheme in the Ideal Cipher Model

In the previous subsection, we have established the selective security of our proposed scheme $\Pi_{\mathsf{SHVE\text{-}1}}$ in the standard model. Interestingly, the same scheme may be proved to be adaptively secure, albeit in the ideal cipher model. We briefly explain the proof intuition, followed by the detailed proof.

Proof Intuition: Using the Ideal Cipher Model. We provide a brief intuition as to why an adaptive proof of security is possible in the ideal cipher model, and not in the standard model. In the adaptive experiment $\mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$, the simulator $\mathcal{S}_0$ must generate the ciphertext C, without possessing any information about the corresponding plaintext (I, M) chosen by the algorithm $\mathcal{A}$. For a functionally correct simulation in the standard model, the simulator must correctly guess M, which occurs with negligible probability. A wrong guess, on the other hand, would result in a ciphertext C that is not well-formed with respect to M, which can be readily inferred by decrypting C using a secret-key sk_{f_j} generated by the simulator at a later stage of the experiment, such that $f_j(I) = 1$. Hence, so long as the block cipher is assumed to be only PCPA-secure, it seems difficult to prove the adaptive security of our scheme.

However, in the ideal cipher model, the simulator can control the mapping between a plaintext and a ciphertext for a given key. Consequently, the ciphertext C may be initially generated uniformly randomly by the simulator $\mathcal{S}_0$ without knowing M. If a subsequent key-generation query made by the algorithm $\mathcal{A}$ corresponds to a predicate f_j such that $f_j(I) = 1$, the corresponding simulator algorithm $\mathcal{S}_j$ infers M from the decryption pattern, and creates the appropriate plaintext-ciphertext mapping to ensure that C is well-formed with respect to M. The only concern is that prior to knowing M, the simulator might fail to ascertain the appropriate response to certain encryption/decryption queries from the algorithm $\mathcal{A}$, in which case it must abort. We formally prove the probability of this abortion to be negligible in the security parameter λ.

The Formal Proof. We state and prove the following theorem:

Theorem 4.2 *The SHVE scheme* $\Pi_{\mathsf{SHVE\text{-}1}}$ *is adaptively simulation-secure in the ideal cipher model under the assumption that* PRF *is a pseudo-random function and* SKE *is modeled as an ideal symmetric-key encryption scheme.*

Proof. We present the detailed proof of security. In particular, we show that for any non-uniform probabilistic polynomial-time algorithm $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{Q'+Q-1})$, there exists a non-uniform probabilistic polynomial time simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_{Q'+Q-1})$, such that the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$$

and the ensemble distribution:

$$\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\right) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$$

are computationally indistinguishable, where the experiments $\mathsf{Expt}^{\mathrm{ada,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$ are as described in Section 3.1. In particular, we construct the simulator $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_{Q'+Q-1})$. Once again, we first consider the simplified scenario where $\mathcal{A}$ makes a single ciphertext query of the form (I, M), and $Q \leq \mathsf{poly}(\lambda)$ secret-key queries for $f_1, \cdots, f_Q$. We then present a generalized argument to show that the simulator algorithm can be easily extended to also answer polynomially many ciphertext queries. Note that in the forthcoming discussion, we use the notation $\mathcal{H}$ to denote the query history $((I, M), f_1, \cdots, f_Q)$, and $\mathcal{H}_j$ to denote the partial query history $((I, M), f_1, \cdots, f_j)$, for $j \in [1, Q]$.

- $\mathcal{S}_0\left(1^\lambda\right)$: Since $Q' = 1$, the simulator $\mathcal{S}_0$ needs to generate a single ciphertext C corresponding to the plaintext query (I, M). It samples $A_1, A_2, \cdots, A_l \xleftarrow{R} \{0,1\}^\lambda$ and $B_1, B_2 \cdots, B_l \xleftarrow{R} \{0,1\}^\lambda$. It also samples $K \xleftarrow{R} \{0,1\}^\lambda$ and breaks it into l random shares $K_1, \cdots, K_l$. It then sets the following:

$$\begin{aligned} c_{k,0} &\leftarrow B_k \oplus K_k \text{ for } k \in [1, l] \\ c_{k,1} &\leftarrow A_k \oplus K_k \text{ for } k \in [1, l] \\ c_{l+1} &\xleftarrow{R} \{0,1\}^\lambda \end{aligned}$$

As in the proof of selective security, the values $A_1, A_2, \cdots, A_l$ are used to simulate the deterministic values corresponding to $\mathsf{PRF}(\mathsf{msk}, \star||k)$ for $k \in [1, l]$. Additionally, the values $B_1, B_2, \cdots, B_l$ are used to simulate $\mathsf{PRF}(\mathsf{msk}, x_k||k)$ for $k \in [1, l]$, where x_k is the k^{th} component of the attribute I chosen by algorithm $\mathcal{A}$. Finally, since $\mathcal{S}_0$ has no information about the payload message M chosen by the algorithm $\mathcal{A}$, it uniformly samples c_{l+1}. The rationale behind this step will be explained subsequently.

Note that since SKE is modeled as an ideal cipher in this proof, at any point of time, the algorithm $\mathcal{A}$ is allowed to make encryption/decryption queries using plaintext-key/ciphertext-key pairs of its choice. The simulator $\mathcal{S}_0$ initializes a table T, referred to as the SKE-table, comprising of tuples of the form $(k_i, m_i, c_i) \in \{0,1\}^\lambda \times \{0,1\}^\lambda \times \{0,1\}^\lambda$. When $\mathcal{A}$ issues an encryption-query of the form (k, m) (equivalently, a decryption query of the form (k, c)), $\mathcal{S}_0$ looks up the table T for a matching tuple. If such a tuple is found, it responds with the corresponding ciphertext (equivalently, plaintext) entry. Otherwise, if $k \neq K$, it uniformly samples a random string in $\{0,1\}^\lambda$, returns the same to $\mathcal{A}$, and adds the resulting triplet as a new entry to the SKE-table.

If $k = K$, $\mathcal{S}_0$ must abort. Note that the tuple (K, M, c_{l+1}) should ideally exist in the SKE-table T; however, $\mathcal{S}_0$ has no information about the payload message M chosen by the algorithm $\mathcal{A}$ at this point. Upon encountering an encryption query of the form (K, m), $\mathcal{S}_0$ can correctly guess if $M = m$ with only negligible probability; hence it cannot ascertain if the corresponding ciphertext c to be returned should be c_{l+1} or a random string in $\{0,1\}^\lambda$. A similar argument may be made for the case of a decryption query as well.

Finally, $\mathcal{S}_0$ outputs the simulated ciphertext $C = \left(\{c_{k,0}, c_{k,1}\}_{k\in[1,l]}, c_{l+1}\right)$ and the state variable $\mathsf{state}_{\mathcal{S}} = \left(\{A_k, B_k\}_{k\in[1,l]}, K, c_{l+1}, T\right)$.

- $\mathcal{S}_1\left(\alpha\left(\mathcal{H}_1\right), \delta\left(\mathcal{H}_1\right), \chi\left(\mathcal{H}_1\right), \mathsf{state}_{\mathcal{S}}\right)$: Recall that the notation $\mathcal{H}_1$ is used to denote the query history $\left((I, M), f_1\right)$. Let $f_1 = (y_{1,1}, y_{2,1} \cdots, y_{l,1})$ and $\mathsf{state}_{\mathcal{S}} = \left(\{A_k, B_k\}_{k\in[1,l]}, K, c_{l+1}, T\right)$. $\mathcal{S}_1$ sets the secret-key $\mathsf{sk}_{f_1} = \left((d_{1,1}, \alpha_{1,1}), \cdots, (d_{1,l}, \alpha_{1,l})\right)$, such that for each $k \in [1, l]$, :

$$d_{1,k} = \begin{cases} A_k & \text{if } \alpha_{1,k} = \star \\ B_k & \text{if } \alpha_{1,k} \neq \star \text{ and } \chi_1 \neq \perp \\ r \xleftarrow{R} \{0,1\}^\lambda & \text{otherwise} \end{cases}$$

The generation of $d_{1,k}$ follows the same logic as discussed in the proof of selective security. The coordinates of f_1 that contain the wildcard character $\star$ are readily inferred from the wildcard pattern α and set accordingly. If $\chi_1 \neq \perp$, that is, $f_1(I) = 1$, the predicate vector f_1 must match the attribute I in all components that are not $\perp$; hence they are generated consistently with the ciphertext to allow correct decryption. The remaining components are generated at random since they play no role in decryption irrespective of χ_1.

Observe that unlike $\mathcal{S}_0$, $\mathcal{S}_1$ potentially learns the payload message M from the decryption pattern $\chi\left(\mathcal{H}_1\right)$. In particular, if $\chi_1 \neq \perp$, that is, $f_1(I) = 1$, $\mathcal{S}_1$ sets $M = \chi_1$ and adds the tuple (K, M, c_{l+1}) to the SKE-table T. It answers encryption and decryption queries in the same way as $\mathcal{S}_0$, except that if the entry (K, M, c_{l+1}) has already been added to T, it no longer needs to abort on any encryption/decryption query. Finally, $\mathcal{S}_1$ outputs the secret-key sk_{f_1} and the updated state variable $\mathsf{state}_{\mathcal{S}} = \left(\{A_k, B_k\}_{k\in[1,l]}, K, c_{l+1}, T\right)$, containing the augmented SKE-table T.

- $\mathcal{S}_j\left(\alpha\left(\mathcal{H}_j\right), \delta\left(\mathcal{H}_j\right), \chi\left(\mathcal{H}_j\right), \mathsf{state}_{\mathcal{S}}\right)$ for $j \in [2, Q]$: Again, let $f_j = (y_{1,j}, y_{2,j} \cdots, y_{l,j})$ and $\mathsf{state}_{\mathcal{S}} = \left(\{A_k, B_k\}_{k\in[1,l]}, K, c_{l+1}, T\right)$. $\mathcal{S}_j$ sets the secret-key $\mathsf{sk}_{f_j} = \left((d_{j,1}, \alpha_{j,1}), \cdots, (d_{j,l}, \alpha_{j,l})\right)$, such that

for each $k \in [1, l]$, :

$$d_{j,k} = \begin{cases} A_k & \text{if } \alpha_{j,k} = \star \\ B_k & \text{if } \alpha_{j,k} \neq \star \text{ and } \chi_j = 1 \\ d_{i,k} & \text{if there exists } i \in [1, j-1] \text{ such that } \delta_{i,j,k} = 1 \\ r \xleftarrow{R} \{0,1\}^\lambda & \text{otherwise} \end{cases}$$

The generation of the secret-key components addresses the same consistency requirements as in the case of $\mathcal{S}_1$. An additional check based on the query pattern δ is made in view of the deterministic nature of the secret-key generation algorithm $KeyGen$ in the real world experiment. Once again, if $\chi_j \neq \perp$ and the tuple (K, M, c_{l+1}) is not already present in T, $\mathcal{S}_j$ sets $M = \chi_j$ and adds the tuple (K, M, c_{l+1}) to T. All encryption and decryption queries are answered following the same procedure as $\mathcal{S}_1$. Finally, $\mathcal{S}_j$ outputs the secret-key sk_{f_j} and the augmented state variable $\mathsf{state}_{\mathcal{S}} = \big(\{A_k, B_k\}_{k\in[1,l]}, K, c_{l+1}, T\big)$, containing the augmented SKE-table T.

Functional Correctness of the Simulation. As in the proof of selective security, the consistency properties adhered to by $\mathcal{S}_0, \cdots, \mathcal{S}_Q$ when generating the ciphertext C and the secret-keys $\{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}$, as well as maintaining the SKE-table T, suffice to ensure that decrypting C using these secret-keys yields the desired outputs. In particular, for each $j \in [1, Q]$ such that $f_j(I) = 1$, decryption will successfully recover the payload message M, while in the remaining cases, decryption will return $\perp$ with overwhelmingly large probability.

Indistinguishability Argument. In the absence of an abortion, a probabilistic polynomial-time distinguisher $\mathcal{D}$ cannot computationally distinguish $\big(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\big) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,real}}_{\Pi_{\mathsf{SHVE}},\mathcal{A}}(\lambda)$ from $\big(C, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}, \mathsf{state}_{\mathcal{A}}\big) \xleftarrow{R} \mathsf{Expt}^{\mathrm{ada,sim}}_{\Pi_{\mathsf{SHVE}},\mathcal{A},\mathcal{S}}(\lambda)$. As in the proof of selective security, since the state variable $\mathsf{state}_{\mathcal{A}}$ does not include the master-secret-key msk with all but negligible probability, the secret-keys as well as the components $\{c_{k,0}\}_{k\in[1,l]}$ and $\{c_{k,1}\}_{k\in[1,l]}$ of the ciphertext C in the real experiment must be indistinguishable from those generated by the simulator algorithms $\mathcal{S}_0, \cdots, \mathcal{S}_Q$. Otherwise, one could distinguish between the output of PRF and a uniformly random string of size λ without knowing the corresponding secret-key msk. Finally, if there exists at least one $j \in [1, Q]$ such that $f_j(I) = 1$, the corresponding simulator $\mathcal{S}_j$ gains the knowledge of the payload message M, and adds the tuple (K, M, c_{l+1}) to the SKE-table T. This ensures that decrypting C with sk_{f_j} recovers M, as in the real experiment. If, on the other hand, $f_j(I) = 0$ for each $j \in [1, Q]$, decrypting C with any of the secret-keys $\{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}$ does not reveal the secret-key K used to encrypt M. Now, since the state variable $\mathsf{state}_{\mathcal{A}}$ does not include K with all but negligible probability, and c_{l+1} is sampled uniformly at random, the indistinguishability of c_{l+1} in the real and simulation experiments is guaranteed.

Bounding the Probability of Abortion. It remains to bound the probability that any of the simulators $\mathcal{S}_0, \cdots, \mathcal{S}_Q$ has to abort. Observe that abortion can only happen if the algorithm $\mathcal{A}$ makes an encryption query of the form (K, m) or a decryption query of the form (K, c), where K is the secret-key used to encrypt M in the ciphertext C. In addition, such a query must be made *before* a secret-query is made for a predicate f_j such that $f_j(I) = 1$; otherwise the corresponding simulator $\mathcal{S}_j$ learns M from the decryption pattern and no longer needs to abort. However, the algorithm $\mathcal{A}$ also does not know the secret-key K; indeed, it can recover K via decryption *only after* receiving a secret-key sk_{f_j} such that $f_j(I) = 1$. Prior to this, it must guess K to force any of the simulator algorithms to abort, which happens with negligible probability.

Generalization to Polynomially many Ciphertext Queries. We now argue that the simulator $\mathcal{S}$ can be extended to address polynomially many ciphertext queries of the form $\{(I_i, M_i)\}_{i\in[1,Q']}$. The simulators $\mathcal{S}_0$ through $\mathcal{S}_{Q'-1}$ essentially follow the same procedure as $\mathcal{S}_0$ in the aforementioned

description to simulate the ciphertexts C_1 through $C_{Q'}$. Note that none of the corresponding payload messages $M_1, \cdots, M_{Q'}$ are known to the respective simulators; consequently, each of the final ciphertext components $c_{i,l+1}$ for $i \in [1, Q']$ are initially sampled uniformly at random. The secret-key simulation phase also proceeds as described above, in consistence with the previously generated ciphertexts. As and when the decryption pattern reveals a given payload message M_i, the look-up table T for the ideal cipher SKE is updated accordingly with the tuple $(K_i, M_i, c_{i,l+1})$. The probability of abortion on an encryption/decryption query to the ideal cipher can be similarly bounded to be negligible in λ; in particular, since $Q' \leq \mathsf{poly}(\lambda)$, the probability that the adversary correctly guesses any K_i for $i \in [1, Q']$ without an appropriate secret-key sk_{f_j} such that $f_j(I_i) = 1$, is negligible in λ.

There, however, remains a final issue to be addressed. Consider a scenario in which two attribute vectors I_i and $I_{i'}$ queried by $\mathcal{A}$ match in some coordinate $k \in [1, l]$, but the corresponding PRF outputs $B_{i,k}$ and $B_{i',k}$ are simulated as distinct values. This happens with high probability since, in the ciphertext generation phase, the simulator has no prior information about the plaintext attributes. This could lead to an ambiguity during the subsequent secret-key generation phase, specifically if the algorithm $\mathcal{A}$ queries with an f_j such that $f_j(I) = f_j(I') = 1$. This is tackled as follows: without loss of generality, assume that $i < i'$. The simulator preserves the ciphertext C_i as is, and generates sk_{f_j} to be consistent with C_i. The other ciphertext $C_{i'}$, is now made semantically consistent with both C_i and sk_{f_j} by re-adjusting the corresponding mask components that were XOR-ed with the PRF outputs. In particular, the mask share $K_{i',k}$ is now changed to $K'_{i',k} = B_{i,k} \oplus B_{i',k} \oplus K_{i',k}$. However, this now implies that the payload message $M_{i'}$ should be encrypted using a different overall secret-key $K'_{i'} = K_{i'} \oplus K_{i',k} \oplus K'_{i',k}$. Since SKE is modeled as an ideal cipher, this can again be made consistent by replacing the tuple $(K_{i'}, M_{i'}, c_{i',l+1})$ with $(K'_{i'}, M_{i'}, c_{i',l+1})$ in the SKE-table T; provided a tuple of the form $(K'_{i'}, M_{i'}, \cdot)$ does not already exist. Finally, as already argued before, the probability that the adversary $\mathcal{A}$ has already made an encryption/decryption query with either $K_{i'}$ or $K'_{i'}$ before receiving sk_{f_j} is negligible in λ. This concludes the proof of Theorem 4.2. □

5 Conclusion and Open Problems

Hidden vector encryption (HVE), introduced by Boneh and Waters in [1], is an expressive sub-class of predicate encryption, that allows conjunctive, subset, range and comparison queries over encrypted data. All existing HVE constructions in the cryptographic literature use bilinear pairings over either composite order or prime order groups. In this paper, we addressed the open problem of constructing a lightweight symmetric-key HVE scheme that does not use bilinear pairings, but only efficient cryptographic primitives such as pseudo-random functions (PRFs) and block ciphers. The relevance of this problem stems from the implementation and performance overheads for bilinear pairings over composite/prime order groups, which are significantly larger than that for PRFs and block ciphers, in both software and hardware. While lightweight symmetric-key constructions exist for keyword search on encrypted data, we aimed to expand the scope of such constructions to support a richer set of query predicates. In this direction, we presented the first lightweight symmetric-key HVE construction that does not use bilinear pairings. Our construction only uses a PRF and a PCPA-secure symmetric-key encryption algorithm, making it amenable to both hardware and software implementations in real-life resource-constrained environments. We proved the selective-simulation-security and adaptive-simulation-security of our construction in the standard model and ideal cipher model, respectively, against probabilistic polynomial-time adversary can make an unrestricted number of ciphertext generation and secret-key generation queries. The following open problems arise out of our work:

- An interesting open problem is to design a lightweight SHVE scheme that is adaptively simulation-secure/enhanced-simulation-secure in the standard model. While known impossibility results do not rule out such a construction [35, 36], especially if the number of ciphertext queries were restricted, it seems challenging to achieve such a construction in practice.

- Another interesting problem is to achieve a lightweight SHVE scheme with short secret-keys and/or ciphertexts. This would lead to implementations with even lesser implementation/performance overhead. Unfortunately, discarding pairings implies sacrificing the nice mathematical properties of bilinear groups, which makes it difficult to combine multiple secret-key/ciphertext components into a single component of constant overhead.

References

1. Dan Boneh and Brent Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 535–554, 2007.
2. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional Encryption for Inner Product Predicates from Learning with Errors. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 21–40, 2011.
3. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *J. Cryptology*, 26(2):191–224, 2013.
4. Emily Shen, Elaine Shi, and Brent Waters. Predicate Privacy in Encryption Systems. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 457–473, 2009.
5. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 470–491, 2015.
6. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 164–195, 2016.
7. Craig Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 445–464, 2006.
8. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456. Springer, 2005.
9. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. *J. Cryptology*, 21(3):350–391, 2008.
10. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522, 2004.
11. Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure hidden vector encryption. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 102–121, 2012.
12. Jong Hwan Park, Kwangsu Lee, Willy Susilo, and Dong Hoon Lee. Fully secure hidden vector encryption under standard assumptions. *Information Sciences*, 232:188–207, 2013.
13. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Eurocrypt*, volume 6110, pages 44–61. Springer, 2010.
14. Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 318–335. Springer, 2012.
15. Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam ONeill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology–CRYPTO 2013*, pages 519–535. Springer, 2013.
16. Allison B Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Eurocrypt*, volume 6110, pages 62–91. Springer, 2010.

17. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *Eurocrypt*, volume 7237, pages 591–608. Springer, 2012.
18. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. *Advances in Cryptology–ASIACRYPT 2012*, pages 349–366, 2012.
19. Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. *Designs, Codes and Cryptography*, 77(2-3):725–771, 2015.
20. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 733–751, 2015.
21. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 333–362, 2016.
22. Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. Function private functional encryption and property preserving encryption : New definitions and positive results. *IACR Cryptology ePrint Archive*, 2013:744, 2013.
23. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, volume 7417, pages 162–179. Springer, 2012.
24. Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 442–455, 2005.
25. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55, 2000.
26. Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 79–88, 2006.
27. David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 353–373, 2013.
28. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from pgv. In *Crypto*, volume 2442, pages 320–335. Springer, 2002.
29. Anand Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In *Advances in CryptologyCRYPTO 2000*, pages 359–375. Springer, 2000.
30. Joe Kilian and Phillip Rogaway. How to protect des against exhaustive key search (an analysis of desx). *Journal of Cryptology*, 14(1):17–35, 2001.
31. John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *FSE*, volume 6, pages 328–340. Springer, 2006.
32. Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 1–20, 2008.
33. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 430–448, 2005.
34. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-Private Subspace-Membership Encryption and Its Applications. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 255–275, 2013.
35. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 253–273, 2011.
36. Mihir Bellare and Adam ONeill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *International Conference on Cryptology and Network Security*, pages 218–234. Springer, 2013.

A Generalized Simulator in the Proof of Theorem 4.1

Algorithm 2 is the generalized form of the simulator algorithm 1 that answers $Q' \leq \mathsf{poly}(\lambda)$ ciphertext queries in addition to $Q \leq \mathsf{poly}(\lambda)$ secret-key queries.

Algorithm 2 Generalized Simulator S for Q' Ciphertext Queries and Q Secret-key Queries

Input: $\left(1^\lambda, \{\alpha\left(\mathcal{H}_{i,Q}\right), \delta\left(\mathcal{H}_{i,Q}\right), \chi\left(\mathcal{H}_{i,Q}\right)\}_{i\in[1,Q']}\right)$
Output: $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}\right)$

Choose $A_1, A_2, \cdots, A_l \xleftarrow{R} \{0,1\}^\lambda$

Phase 1 - Simulation of Secret-Keys

Initialize a $Q \times l$ matrix d to empty
for each $j \in [1, Q]$ **do**
  for each $k \in [1, l]$ **do**
    if $\alpha_{j,k} = 1$ **then**
      $d_{j,k} \leftarrow A_k$
    else if there exists $i \in [1, j-1]$ such that $\delta_{i,j,k} = 1$ **then**
      $d_{j,k} \leftarrow d_{i,k}$
    else
      $d_{j,k} \xleftarrow{R} \{0,1\}^\lambda$
    end if
  end for
  $\mathsf{sk}_{f_j} \leftarrow \left((d_{j,1}, \alpha_{j,1}), \cdots, (d_{j,l}, \alpha_{j,l})\right)$
end for

Phase 2 - Simulation of Ciphertexts

for each $i \in [1, Q']$ **do**
  if there exists $j \in [1, Q]$ such that $\chi_{i,j} \neq \perp$ **then**
    $M_i \leftarrow \chi_{i,j}$
  else
    $M_i \xleftarrow{R} \{0,1\}^\lambda$
  end if
  Uniformly sample $K_i \xleftarrow{R} \mathsf{SKE.KeyGen}(1^\lambda)$, and break it into l random shares $K_{i,1}, \cdots, K_{i,l}$
  for each $k \in [1, l]$ **do**
    if there exists $j \in [1, Q]$ such that $\chi_{i,j} \neq \perp$ and $\alpha_{j,k} = 0$ **then**
      $c_{i,k,0} \leftarrow d_{j,k} \oplus K_{i,k}$
    else
      $c_{i,k,0} \xleftarrow{R} \{0,1\}^\lambda$
    end if
    $c_{i,k,1} \leftarrow A_k \oplus K_{i,k}$
  end for
  $c_{i,l+1} \leftarrow \mathsf{SKE.Encrypt}\left(K_i, M_i\right)$
  $C_i \leftarrow \left(\{c_{i,k,0}, c_{i,k,1}\}_{k\in[1,l]}, c_{i,l+1}\right)$
end for

Final Output

Return $\left(\{C_i\}_{i\in[1,Q']}, \{\mathsf{sk}_{f_j}\}_{j\in[1,Q]}\right)$