Conjunctive, Subset, and Range Queries on Encrypted Data

Dan Boneh[1] ⋆ and Brent Waters[2] ⋆⋆

[1] Stanford University, dabo@cs.stanford.edu
[2] SRI International, bwaters@csl.sri.com

Abstract. We construct public-key systems that support comparison queries ($x \geq a$) on encrypted data as well as more general queries such as subset queries ($x \in S$). Furthermore, these systems support arbitrary conjunctive queries ($P_1 \wedge \cdots \wedge P_\ell$) without leaking information on individual conjuncts. We present a general framework for constructing and analyzing public-key systems supporting queries on encrypted data.

1 Introduction

Queries on encrypted data are easiest to explain with an example. Consider a credit card payment gateway that observes a stream of encrypted transactions, say encrypted under Visa's public key. The gateway needs to flag all transactions satisfying a certain predicate P. Say, all transactions whose value is over \$1000. Storing Visa's secret key on the gateway is a bad idea for both security and privacy concerns. Instead, Visa wishes to give the gateway a token TK_P that enables the gateway to identify transactions satisfying P without learning anything else about these transactions. Of course, generating the token TK_P will require Visa's secret key.

As another example, consider a mail server that receives a stream of email messages encrypted under the recipients public key. If the email message satisfies a certain predicate P the mail server should forward the email to the recipient's pager. If the email satisfies some other predicate P' the server should just discard the email. Otherwise, the server should place the email in the recipient's inbox. The recipient does not want to give the mail server the full private key. Instead, she wants to give the server two tokens TK_P and $\mathrm{TK}_{P'}$ enabling the server to test for the predicates P and P' without learning any other information about the email.

Our goal is to build a public-key system that supports a rich set of query predicates. In our payment gateway example one can imagine comparison queries such as ($\mathsf{value} > 1000$) or even conjunctions such as ($\mathsf{value} > 1000$) and ($\mathsf{Transaction\ Time} > 5\mathrm{pm}$). The gateway should learn no information other than the value of the conjunctive predicate. In case a conjunction $P_1 \wedge P_2$ is false, the gateway

⋆ Supported by NSF and the Packard Foundation.
⋆⋆ Supported by NSF and U.S. Army Research Office under Research Grant No. W911NF-06-1-0316.

should not learn which of the two conjuncts P_1 or P_2 is false. In our second example involving a mail server one can imagine testing for subset queries such as ($\mathsf{sender} \in S$) where S is a set of email addresses. Conjunctive queries such as ($\mathsf{sender} \in S$) and ($\mathsf{subject} = \mathsf{urgent}$) also make sense. Perhaps in the distant future, when highly complex queries on encrypted data are possible, one can imagine running an anti-virus/anti-spam predicate on encrypted emails. The mail server learns nothing about incoming encrypted email other than its spam status.

Unfortunately, until now, only simple equality queries on encrypted data were possible. Song et al. [19] developed a mechanism for equality tests on data encrypted with a symmetric key system. Boneh et al. [8] constructed equality tests in the public-key settings.

Our results. We present a general framework for analyzing and constructing searchable public-key systems for various families of predicates. We then construct public-key systems that support comparison queries (such as greater-than) and general subset queries. We also support arbitrary conjunctions. We evaluate our results based on ciphertext size and token size. Let $T = \{1, 2, \ldots, n\}$ and suppose we encrypt a tuple $x = (x_1, \ldots, x_w) \in T^w$. Say x_1 is a transaction value, x_2 is a card expiration date, and so on. The following table summarizes our results at a high level.

Query Type		Source	Ciphertext Size	Token Size
Equality query: $(x_i = a)$	for any $a \in T$	[19, 17, 8, 1]	$O(1)$	$O(1)$
Comparison query: $(x_i \geq a)$	for any $a \in T$	[10, 12][3]	$O(\sqrt{n})$	$O(\sqrt{n})$
Subset query: $(x_i \in A)$	for any $A \subseteq T$	This paper	$O(n)$	$O(n)$
Equality conjunction: $(x_1 = a_1) \wedge \ldots \wedge (x_w = a_w)$		This paper	$O(w)$	$O(w)$
Comparison conjunction: $(x_1 \geq a_1) \wedge \ldots \wedge (x_w \geq a_w)$		This paper	$O(nw)$	$O(w)$
Subset conjunction: $(x_1 \in A_1) \wedge \ldots \wedge (x_w \in A_w)$		This paper	$O(nw)$	$O(nw)$

Here $(a_1, \ldots, a_w)$ is an arbitrary vector that defines a conjunctive equality or a comparison predicate. Similarly, $A_1, \ldots, A_w$ are *arbitrary* subsets of $\{1, \ldots, n\}$ that define a conjunctive subset query predicate. We emphasize that when a conjunction predicate is false, the system does not leak which of the w conjuncts caused it.

Prior to these results the best systems for comparison and subset queries were the trivial brute-force systems that we discuss in Section 3. For comparison queries these systems generate a ciphertext of size $O(n^w)$ and for subset queries they generate a ciphertext of size $O(2^{nw})$. Note that even without conjunction,

[3] Both papers [10, 12] focus on traitor tracing, but as we show in the full version of our paper [11], their approach directly gives a comparison searching system without conjunctions.

namely for $w = 1$, our subset query construction generates ciphertexts that are exponentially shorter than the best known previous solution ($O(n)$ vs. $O(2^n)$).

The main tool used in these constructions is a new primitive we call *Hidden Vector Encryption* or HVE for short. This primitive can be viewed as an extreme generalization of Anonymous Identity Based Encryption (AnonIBE) [8, 1, 13]. We show how HVE implies all the results in the table.

A natural question is to look for public key systems that support larger classes of predicates, such as regular expressions. Ultimately, one would like a public-key system that supports searches for any predicate computable by a poly-size circuit. Presently, this appears to be a difficult open problem.

Related work. Equality tests on encrypted data were considered in [19, 8]. Equality searches on an encrypted audit log were proposed in [20]. Equality tests in the symmetric key settings are closely related to oblivious RAM techniques [17, 14]. Equality tests in the public key settings are closely related to Anonymous Identity Based Encryption (AnonIBE) [8, 1, 13]. Conjunctive equality queries were first studied in [15]. Equality searches on streaming data that hide the requested predicate were discussed in [18] and [4]. Efficient equality searches in databases were recently presented in [2]. Bethencourt et al. [3] recently gave a construction for efficient range queries in a weaker security model. That is, when the encrypted index falls in the specified range, the search token reveals the index.

2 Definitions

We begin by defining a general framework for queries on encrypted data. Let Σ be a finite set of binary strings. A predicate P over Σ is a function $P : \Sigma \to \{0, 1\}$. We say that $I \in \Sigma$ satisfies the predicate if $P(I) = 1$.

2.1 Searchable encryption

Let Φ be a set of predicates over Σ. A **Φ-searchable** public key system comprises of the following algorithms:

Setup(λ) A probabilistic algorithm that takes as input a security parameter and outputs a public key PK and secret key SK.

Encrypt(PK, I, M) Encrypts the plaintext pair (I, M) using the public key PK. We view $I \in \Sigma$ as the searchable field, called an **index**, and $M \in \mathcal{M}$ as the data.

GenToken$(\mathrm{SK}, \langle P \rangle)$ Takes as input a secret key SK and the description of a predicate $P \in \Phi$. It outputs a token TK_P.

Query(TK, C) Takes a token TK for some predicate $P \in \Phi$ as input and a ciphertext C. It outputs a message $M \in \mathcal{M}$ or $\perp$. Roughly speaking, if C is an encryption of (I, M) then the algorithm outputs M when $P(I) = 1$ and outputs $\perp$ otherwise. The precise requirement is captured in the query correctness property below.

Correctness. The system must satisfy the following **correctness property**:

– **Query correctness**: For all $(I, M) \in \Sigma \times \mathcal{M}$ and all predicates $P \in \Phi$:

Let $(\mathrm{PK}, \mathrm{SK}) \xleftarrow{R} \mathit{Setup}(\lambda)$, $C \xleftarrow{R} \mathit{Encrypt}(\mathrm{PK}, I, M)$,
and $\mathrm{TK} \xleftarrow{R} \mathit{GenToken}(\mathrm{SK}, \langle P \rangle)$.

If $P(I) = 1$ then $\mathit{Query}(\mathrm{TK}, C) = M$.

If $P(I) = 0$ then $\Pr[\mathit{Query}(\mathrm{TK}, C) = \perp] > 1 - \epsilon(\lambda)$ where $\epsilon(\lambda)$ is a negligible function.

Suppose that given a ciphertext $C \leftarrow \mathit{Encrypt}(\mathrm{PK}, I, M)$ we are only interested in testing whether a predicate $P(I)$ is satisfied. In this case the message space $\mathcal{M}$ can be set to a singleton, say $\mathcal{M} = \{\mathsf{true}\}$. Algorithm $\mathit{Query}(\mathrm{TK}, C)$ will return true when $P(I) = 1$ and $\perp$ otherwise. A larger message space $\mathcal{M}$ is useful if TK is intended to unlock some $M \in \mathcal{M}$ whenever the predicate $P(I) = 1$. For example, when the transaction value is over \$1000 we may want the payment gateway to obtain more information about the transaction. Otherwise, the gateway should learn nothing.

Notice that a Φ-searchable system does not provide a *Decrypt* algorithm that uses SK to decrypt a ciphertext C and outputs (I, M). One can always add this capability by also encrypting (I, M) under a standard public key system. There is no need for the searchable system to explicitly provide this capability.

An example – comparison queries. Before defining security, we first give a motivating example using comparison queries. Let $\Sigma = \{1, \dots, n\}$ for some integer n. For $\sigma \in \{1, \dots, n\}$ let P_σ be the following comparison predicate:

$$P_\sigma(x) = \begin{cases} 1 & \text{if } x \geq \sigma, \\ 0 & \text{otherwise} \end{cases}$$

Let $\Phi_n = \{P_1, \dots, P_n\}$ be the set of all n comparison predicates. Suppose the adversary has the tokens for predicates $P_{\sigma_1}, P_{\sigma_2}, \dots, P_{\sigma_w}$ where $\sigma_1 < \sigma_2 < \cdots < \sigma_w$. Lets x, y, z be some integers as in Figure 1. Clearly the adversary can distinguish $\mathit{Encrypt}(\mathrm{PK}, x, m)$ from $\mathit{Encrypt}(\mathrm{PK}, y, m)$ using the token for the predicate P_{σ_2}. However, the adversary should not be able to distinguish $\mathit{Encrypt}(\mathrm{PK}, y, m)$ from $\mathit{Encrypt}(\mathrm{PK}, z, m)$. Indeed, separating an encryption of y from an encryption of z is information that should not be exposed by the tokens at the adversary's disposal. Our definition of security captures this property using the general framework.

2.2 Security

We define security of a Φ-searchable system $\mathcal{E}$ using a **query security game** that captures the intuition that tokens TK reveal no unintended information about the plaintext. The game gives the adversary a number of tokens and requires that the adversary cannot use these tokens to deduce unintended information. The game proceeds as follows:

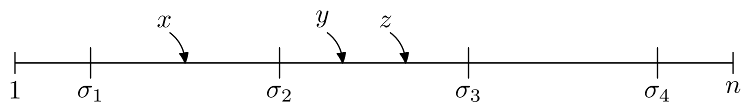

Fig. 1. Tokens for $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ given to the adversary

- **Setup.** The challenger runs *Setup*(λ) and gives the adversary PK.
- **Query phase 1.** The adversary adaptively outputs descriptions of predicates $P_1, P_2, \ldots, P_{q_1} \in \Phi$. The challenger responds with the corresponding tokens $\mathrm{TK}_j \leftarrow \mathit{GenToken}(\mathrm{SK}, \langle P_j \rangle)$. We refer to such queries as **predicate queries**.
- **Challenge.** The adversary outputs two pairs (I_0, M_0) and (I_1, M_1) subject to two restrictions:
  - First, $P_j(I_0) = P_j(I_1)$ for all $j = 1, 2, \ldots, q_1$.
  - Second, if $M_0 \neq M_1$ then $P_j(I_0) = P_j(I_1) = 0$ for all $j = 1, 2, \ldots, q_1$.

  The challenger flips a coin $\beta \in \{0,1\}$ and gives $C_* \xleftarrow{R} \mathit{Encrypt}(\mathrm{PK}, I_\beta, M_\beta)$ to the adversary.

  The two restrictions ensure that the tokens given to the adversary do not trivially break the challenge. The first restriction ensures that tokens given to the adversary do not directly distinguish I_0 from I_1. The second restriction ensures that the tokens do not directly distinguish M_0 from M_1.
- **Query phase 2.** The adversary continues to adaptively request tokens for predicates $P_{q_1+1}, \ldots, P_q \in \Phi$, subject to the two restrictions above. The challenger responds with the corresponding tokens $\mathrm{TK}_j \leftarrow \mathit{GenToken}(\mathrm{SK}, \langle P_j \rangle)$.
- **Guess** The adversary returns a guess $\beta' \in \{0,1\}$ of β.

We define the advantage of adversary $\mathcal{A}$ in attacking $\mathcal{E}$ as the quantity $\mathsf{QU\ Adv}_{\mathcal{A}} = |\Pr[\beta' = \beta] - 1/2|$.

Definition 1. *We say that a Φ-searchable system $\mathcal{E}$ is* **secure** *if for all polynomial time adversaries $\mathcal{A}$ attacking $\mathcal{E}$ the function $\mathsf{QU\ Adv}_{\mathcal{A}}$ is a negligible function of λ.*

Another example – equality queries. Let Σ be some finite set. For $\sigma \in \Sigma$ let $P_\sigma(x)$ be an equality predicate, namely

$$P_\sigma(x) = \begin{cases} 1 & \text{if } x = \sigma, \\ 0 & \text{otherwise} \end{cases}$$

Let $\Phi_{\text{eq}} = \{P_\sigma \text{ for all } \sigma \in \Sigma\}$. Then a Φ_{eq}-searchable encryption supports equality queries on ciphertexts. It is easy to see that a secure Φ_{eq}-searchable encryption is also an anonymous IBE system [8, 1, 13] — an Identity Based Encryption system where a ciphertext reveals no useful information about the identity that was used to create it. This should not be too surprising since it was previously shown [8, 1] that anonymous IBE is sufficient for equality searches. A Φ_{eq}-searchable encryption system (*Setup*, *Encrypt*, *GenToken*, *Query*) gives an anonymous IBE as follows:

- $Setup_{\mathrm{IBE}}(\lambda)$ runs $Setup(\lambda)$ and outputs IBE parameters PK and master key SK.
- $Encrypt_{\mathrm{IBE}}(\mathrm{PK}, \mathcal{I}, M)$ where $\mathcal{I} \in \Sigma$ outputs $Encrypt(\mathrm{PK}, \mathcal{I}, M)$.
- $Extract_{\mathrm{IBE}}(\mathrm{SK}, \mathcal{I})$ where $\mathcal{I} \in \Sigma$ outputs $\mathrm{TK}_{\mathcal{I}} \leftarrow GenToken(\mathrm{SK}, \langle P_{\mathcal{I}} \rangle)$.
- $Decrypt_{\mathrm{IBE}}(\mathrm{TK}_{\mathcal{I}},\ C)$ outputs $Query(\mathrm{TK}_{\mathcal{I}},\ C)$.

The correctness property ensures that if C is the result of $Encrypt(\mathrm{PK}, \mathcal{I}, M)$ then $Query(\mathrm{TK}_{\mathcal{I}}, C)$ will output M since $P_{\mathcal{I}}(\mathcal{I}) = 1$. It is not difficult to see that the Φ_{eq}-security game ensures semantic security for both the message and the identity. Hence, the resulting system is an anonymous IBE.

By considering larger classes of predicates Φ we obtain more general searching capabilities. The challenge is then to build secure encryption schemes that are Φ-searchable for the most general Φ possible.

Chosen ciphertext security. Definition 1 easily extends to address chosen ciphertext attacks (CCA), but we do not pursue that here.

2.3 Selective security

We will also need a slightly weaker security definition in which the adversary commits to the search strings I_0, I_1 at the beginning of the game. Everything else remains the same. The game proceeds as follows:

- **Setup.** The adversary outputs two strings $I_0, I_1 \in \Sigma$. The challenger runs $Setup(\lambda)$ and gives the adversary PK.
- **Query phase 1.** The adversary adaptively outputs descriptions of predicates $P_1, P_2, \ldots, P_{q_1} \in \Phi$. The only restriction is that
$$P_j(I_0) = P_j(I_1) \text{ for all } j = 1, 2, \ldots, q_1 \tag{1}$$
The challenger responds with the corresponding tokens $\mathrm{TK}_j \leftarrow GenToken($ $\mathrm{SK}, \langle P_j \rangle)$.
- **Challenge.** The adversary outputs two messages $M_0, M_1 \in \mathcal{M}$ subject to the restriction that:
$$\text{if } M_0 \neq M_1 \text{ then } P_j(I_0) = P_j(I_1) = 0 \text{ for all } j = 1, 2, \ldots, q_1 \tag{2}$$
The challenger flips a coin $\beta \in \{0, 1\}$ and gives $C_* \xleftarrow{R} Encrypt(\mathrm{PK}, I_\beta, M_\beta)$ to the adversary.
- **Query phase 2.** The adversary continues to adaptively request query tokens for predicates $P_{q_1+1}, \ldots, P_q \in \Phi$, subject to the two restrictions (1) and (2). The challenger responds with the corresponding tokens $\mathrm{TK}_j \leftarrow$ $GenToken(\mathrm{SK}, \langle P_j \rangle)$.
- **Guess** The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

The advantage of adversary $\mathcal{A}$ in attacking $\mathcal{E}$ is the quantity $\mathsf{sQU\,Adv}_{\mathcal{A}} = |\Pr[\beta' = \beta] - 1/2|$.

Definition 2. *We say that a Φ-searchable system $\mathcal{E}$ is* **selectively secure** *if for all polynomial time adversaries $\mathcal{A}$ attacking $\mathcal{E}$ the function $\mathsf{sQU\,Adv}_{\mathcal{A}}$ is a negligible functions of λ.*

3 The Trivial Construction

Let Σ be a finite set of binary strings. We build a Φ-searchable public key system $\mathcal{E}_{\text{TR}}$, for *any* set of (polynomial time computable) predicates Φ. We refer to this system as the brute force Φ-searchable system.

The brute force system. Let $\mathcal{E} = (\mathit{Setup}', \mathit{Encrypt}', \mathit{Decrypt}')$ be a public-key system. Let $\Phi = \{P_1, P_2, \ldots, P_t\}$ The Φ-searchable system $\mathcal{E}_{\text{TR}}$ is defined as follows:

Setup(λ) Run $\mathit{Setup}'(\lambda)$ t times to obtain

$$\text{PK} \leftarrow (\text{PK}_1, \ldots, \text{PK}_t) \quad \text{and} \quad \text{SK} \leftarrow (\text{SK}_1, \ldots, \text{SK}_t)$$

Output PK and SK.

Encrypt(PK, I, M) For $j = 1, \ldots, t$ define:

$$C_j \overset{R}{\leftarrow} \begin{cases} \mathit{Encrypt}'(\text{PK}_j,\ M) & \text{if } P_j(I) = 1, \\ \mathit{Encrypt}'(\text{PK}_j,\ \perp) & \text{otherwise.} \end{cases}$$

Output $C \leftarrow (C_1, \ldots, C_t)$. Note that the length of C is linear in n.

GenToken$(\text{SK}, \langle P \rangle)$ Here $\langle P \rangle$ (the description of a predicate P) is the index j of P in Φ. Output $\text{TK} \leftarrow (j, \text{SK}_j)$.

Query(TK, C) Let $C = (C_1, \ldots, C_t)$ and $\text{TK} = (j, \text{SK}_j)$.
Output $\mathit{Decrypt}'(\text{SK}_j, C_j)$.

The following lemma proves security of this construction. The proof is a straightforward hybrid argument and is given in Appendix A.

Lemma 1. *The system $\mathcal{E}_{\text{TR}}$ above is a secure Φ-searchable encryption system assuming $\mathcal{E}$ is a semantically secure public key system against chosen plaintext attacks.*

3.1 A third example — conjunctive comparison predicates

Suppose $\Sigma = \{1, \ldots, n\}^w$ for some n, w. Let $\Phi_{n,w}$ be the set of n^w predicates

$$P_{a_1 \ldots a_w}(x_1, \ldots, x_w) = \begin{cases} 1 & \text{if } x_j \geq a_j \text{ for all } j = 1, \ldots, w, \\ 0 & \text{otherwise} \end{cases}$$

for all $\bar{a} = (a_1 \ldots a_w) \in \{1, \ldots, n\}^w$. Then $|\Phi_{n,w}| = n^w$.

The trivial system in this case produces ciphertexts of length $O(n^w)$. Essentially, the system uses a unary encoding of the w columns and assigns a private key to each cell in this n by w matrix. We will construct a much better system in Section 6.

4 Background on pairings and complexity assumptions

Our goal is to construct Φ-searchable systems for a large class of predicates Φ that is much better than the trivial construction. To do so we will make use of bilinear maps.

4.1 Bilinear groups of composite order

We review some general notions about bilinear maps and groups, with an emphasis on groups of *composite order*. We follow [9] in which composite order bilinear groups were first introduced.

Let $\mathcal{G}$ be a an algorithm called a *group generator* that takes as input a security parameter $\lambda \in \mathbb{Z}^{>0}$ and outputs a tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$ where p, q are two distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $n = pq$, and e is a function $e : \mathbb{G}^2 \to \mathbb{G}_T$ satisfying the following properties:

– (Bilinear) $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
– (Non-degenerate) $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order n in $\mathbb{G}_T$.

We assume that the group action in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all computable in polynomial time in λ. Furthermore, we assume that the description of $\mathbb{G}$ and $\mathbb{G}_T$ includes generators of $\mathbb{G}$ and $\mathbb{G}_T$ respectively.

To summarize, $\mathcal{G}$ outputs the description of a group $\mathbb{G}$ of order $n = pq$ with an efficiently computable bilinear map. We will use the notation $\mathbb{G}_p, \mathbb{G}_q$ to denote the respective subgroups of order p and order q of $\mathbb{G}$ and we will use the notation $\mathbb{G}_{T,p}, \mathbb{G}_{T,q}$ to denote the respective subgroups of order p and order q of $\mathbb{G}_T$.

4.2 The bilinear Diffie-Hellman assumption

First we review the standard Bilinear Diffie-Hellman assumption, but in groups of composite order. For a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$
\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{R} \mathbb{G}_p, \quad g_q \xleftarrow{R} \mathbb{G}_q \\
&a, b, c \xleftarrow{R} \mathbb{Z}_n \\
&\bar{Z} \leftarrow \big((n, \mathbb{G}, \mathbb{G}_T, e), \;\; g_q, \; g_p, \;\; g_p^a, \;\; g_p^b, \;\; g_p^c\big) \\
&T \leftarrow e(g_p, \; g_p)^{abc} \\
&\text{Output } \; (\bar{Z}, \; T)
\end{aligned}
$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the composite bilinear Diffie-Hellman problem for $\mathcal{G}$ as:

$$\mathsf{cBDH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) := \Big| \Pr[\mathcal{A}(\bar{Z}, T) = 1] - \Pr[\mathcal{A}(\bar{Z}, R) = 1] \Big|$$

where $(\bar{Z}, T) \xleftarrow{R} P(\lambda)$ and $R \xleftarrow{R} \mathbb{G}_{T,p}$.

Definition 3. *We say that $\mathcal{G}$ satisfies the composite bilinear Diffie-Hellman assumption (cBDH) if for any polynomial time algorithm $\mathcal{A}$ we have that the function* $\mathsf{cBDH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ *is a negligible function of λ.*

4.3 The composite 3-party Diffie-Hellman assumption

Our construction makes use of an additional assumption in composite bilinear groups. For a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$
\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{R} \mathbb{G}_p, \quad g_q \xleftarrow{R} \mathbb{G}_q \\
&R_1, R_2, R_3 \xleftarrow{R} \mathbb{G}_q \\
&a, b, c \xleftarrow{R} \mathbb{Z}_n \\
&\bar{Z} \leftarrow \big((n, \mathbb{G}, \mathbb{G}_T, e), \quad g_q,\ g_p, \quad g_p^a, \quad g_p^b, \quad g_p^{ab} \cdot R_1, \quad g_p^{abc} \cdot R_2\big) \\
&T \leftarrow g_p^c \cdot R_3 \\
&\text{Output} \quad (\bar{Z},\ T)
\end{aligned}
$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the composite 3-party Diffie-Hellman problem for $\mathcal{G}$ as:

$$\mathsf{C3DH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) := \Big| \Pr[\mathcal{A}(\bar{Z}, T) = 1] - \Pr[\mathcal{A}(\bar{Z}, R) = 1] \Big|$$

where $(\bar{Z}, T) \xleftarrow{R} P(\lambda)$ and $R \xleftarrow{R} \mathbb{G}$.

Definition 4. *We say that $\mathcal{G}$ satisfies the composite 3-party Diffie-Hellman assumption (C3DH) if for any polynomial time algorithm $\mathcal{A}$ we have that the function* $\mathsf{C3DH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ *is a negligible function of λ.*

The assumption is formed around the intuition that it is hard to test for Diffie-Hellman tuples in the order p subgroup if the elements to be tested have a random order q subgroup component.

5 Hidden Vector Encryption

We construct a Φ-searchable encryption system for a general class of equality predicates. We call such systems Hidden Vector Systems or HVEs for short. We then show in Section 6 that our HVE system leads to comparison and subset queries far more efficient than the trivial system.

5.1 HVE Definition

Let Σ be a finite set and let $*$ be a special symbol not in Σ. Define $\Sigma_* = \Sigma \cup \{*\}$. The star $*$ plays the role of a wildcard or "don't care" value. In our subset and

range query applications we typically set $\Sigma = \{0, 1\}$. Note that here we use the symbol Σ differently than how it was used in Section 2.1.
For $\sigma = (\sigma_1, \ldots, \sigma_\ell) \in \Sigma_*^\ell$ define a predicate P_σ^{HVE} over Σ^ℓ as follows. For $x = (x_1, \ldots, x_\ell) \in \Sigma^\ell$ set:

$$P_\sigma^{\text{HVE}}(x) = \begin{cases} 1 & \text{if for all } i = 1, \ldots, \ell: \ (\sigma_i = x_i \text{ or } \sigma_i = *), \\ 0 & \text{otherwise} \end{cases}$$

In other words, the vector x matches σ in all the coordinates where σ is not $*$. Let $\Phi_{\text{HVE}} = \{P_\sigma^{\text{HVE}} \text{ for all } \sigma \in \Sigma_*^\ell\}$. We refer to ℓ as the **width** of the HVE.

Definition 5. *A Hidden Vector System (HVE) over Σ^ℓ is a selectively secure Φ_{HVE}-searchable encryption system.*

The case $\ell = 1$ degenerates to the example discussed in Section 2.2 where we showed equivalence to anonymous IBE [8, 1, 13]. For larger ℓ we obtain a more general concept that is much harder to build. In particular, the wildcard character '$*$' — which is essential for the applications we have in mind — makes it challenging to construct a Φ_{HVE}-searchable system. We construct an HVE with the following parameters:

$$\text{CT-size} = O(\ell) \quad \text{and} \quad \text{TK-size} = O(\ \text{weight}(\sigma)\ )$$

where $\text{weight}\big(\sigma = (\sigma_1, \ldots, \sigma_\ell)\big)$ is the number of coordinates where $\sigma_i \neq *$.

5.2 Construction

For our particular HVE construction we will let $\Sigma = \mathbb{Z}_m$ for some integer m. We set $\Sigma_* = \mathbb{Z}_m \cup \{*\}$. We describe an HVE where the payload M is in a small subset $\mathcal{M}$ of $\mathbb{G}_T$, namely $|\mathcal{M}| < |\mathbb{G}_T|^{1/4}$. This is not a serious restriction since the payload M is typically a short symmetric message key. Our HVE system works as follows:

Setup(λ) The setup algorithm first chooses random primes $p, q > m$ and creates a bilinear group $\mathbb{G}$ of composite order $n = pq$, as specified in Section 4.1. Next, it picks random elements

$$(u_1, h_1, w_1), \ldots, (u_\ell, h_\ell, w_\ell) \in \mathbb{G}_p^3\ , \qquad g, v \in \mathbb{G}_p\ , \qquad g_q \in \mathbb{G}_q.$$

and an exponent $\alpha \in \mathbb{Z}_p$. It keeps all these as the secret key SK.
It then chooses $3\ell + 1$ random blinding factors in $\mathbb{G}_q$:

$$(R_{u,1}, R_{h,1}, R_{w,1}), \ldots, (R_{u,\ell}, R_{h,\ell}, R_{w,\ell}) \in \mathbb{G}_q \text{ and } R_v \in \mathbb{G}_q.$$

For the public key, PK, it publishes the description of the group $\mathbb{G}$ and the values

$$g_q, \quad V = vR_v, \quad A = e(g, v)^\alpha, \quad \begin{pmatrix} U_1 = u_1 R_{u,1}, & H_1 = h_1 R_{h,1}, & W_1 = w_1 R_{w,1} \\ & \vdots & \\ U_\ell = u_\ell R_{u,\ell}, & H_\ell = h_\ell R_{h,\ell}, & W_\ell = w_\ell R_{w,\ell} \end{pmatrix}$$

The message space $\mathcal{M}$ is set to be a subset of $\mathbb{G}_T$ of size less than $n^{1/4}$.

Encrypt(PK, $\mathcal{I} \in \mathbb{Z}_m^\ell$, $M \in \mathcal{M} \subseteq \mathbb{G}_T$) Let $\mathcal{I} = (\mathcal{I}_1, \dots, \mathcal{I}_\ell) \in \mathbb{Z}_m^\ell$. The encryption algorithm works as follows:

- choose a random $s \in \mathbb{Z}_n$ and random Z, $(Z_{1,1}, Z_{1,2}), \dots, (Z_{\ell,1}, Z_{\ell,2}) \in \mathbb{G}_q$. (The algorithm picks random elements in $\mathbb{G}_q$ by raising g_q to random exponents from $\mathbb{Z}_n$.)
- Output the ciphertext:

$$C = \left(C' = MA^s, \; C_0 = V^s Z, \; \begin{pmatrix} C_{1,1} = (U_1^{\mathcal{I}_1} H_1)^s Z_{1,1}, & C_{1,2} = W_1^s Z_{1,2} \\ \vdots & \\ C_{\ell,1} = (U_\ell^{\mathcal{I}_\ell} H_\ell)^s Z_{\ell,1}, & C_{\ell,2} = W_\ell^s Z_{\ell,2} \end{pmatrix} \right)$$

GenToken(SK, $\mathcal{I}_* \in \Sigma_*^\ell$) The key generation algorithm will take as input the secret key and an ℓ-tuple $\mathcal{I}_* = (\mathcal{I}_1, \dots, \mathcal{I}_\ell) \in \{\mathbb{Z}_m \cup \{*\}\}^\ell$. Let S be the set of all indexes i such that $\mathcal{I}_i \neq *$. To generate a token for the predicate $P_{\mathcal{I}_*}^{\text{HVE}}$ choose random $(r_{i,1}, r_{i,2}) \in \mathbb{Z}_p^2$ for all $i \in S$ and output:

$$\text{TK} = \left(\mathcal{I}_*, \quad K_0 = g^\alpha \prod_{i \in S} (u_i^{\mathcal{I}_i} h_i)^{r_{i,1}} w_i^{r_{i,2}}, \quad \forall i \in S: \quad K_{i,1} = v^{r_{i,1}}, \quad K_{i,2} = v^{r_{i,2}} \right)$$

Query(TK, C) Using the notation in the description of *Encrypt* and *GenToken* do:

- First, compute

$$M \leftarrow C' \;/\; \left(e(C_0, K_0) \;/\; \prod_{i \in S} e(C_{i,1}, K_{i,1}) \; e(C_{i,2}, K_{i,2}) \right) \tag{3}$$

- If $M \notin \mathcal{M}$ output $\perp$. Otherwise, output M.

Correctness Before proving security we first show that the system satisfies the correctness property defined in Section 2.1. Let $(\mathcal{I}, M)$ be a pair in $\Sigma^\ell \times \mathcal{M}$ and let $B_* \in \Sigma_*^\ell$. This B_* defines a predicate P_{B_*} in Φ_{HVE}.

Let $(\text{PK}, \text{SK}) \xleftarrow{R} \textit{Setup}(\lambda)$, $C \xleftarrow{R} \textit{Encrypt}(\text{PK}, \mathcal{I}, M)$,
and $\text{TK} \xleftarrow{R} \textit{GenToken}(\text{ SK}, B_*)$.

- If $P_{B_*}(\mathcal{I}) = 1$ then a simple calculation shows that $\textit{Query}(\text{TK}, C) = M$. This uses in a crucial way the fact that $e(h_p, h_q) = 1$ for all $h_p \in \mathbb{G}_p$ and $h_q \in \mathbb{G}_q$.
- If $P_{B_*}(\mathcal{I}) = 0$ the following lemma shows that when the message space $\mathcal{M}$ satisfies $|\mathcal{M}| < n^{1/4}$ then $\Pr[\textit{Query}(\text{TK}, C) \neq \perp]$ is negligible.
  Here the probability is over the random bits used to create the ciphertext.

Lemma 2. *With the notation as above, and assuming $|\mathcal{M}| < n^{1/4}$, whenever $P_{B_*}(\mathcal{I}) = 0$ the quantity $\Pr[\textit{Query}(\text{TK}, C) \neq \perp]$ is negligible.*
The probability is over the random bits used to create the ciphertext.

Proof. Let $\mathcal{I} = (\mathcal{I}_1, \ldots, \mathcal{I}_\ell) \in \Sigma$ and let $B_* = (B_1, \ldots, B_\ell) \in \Sigma_*^\ell$. Let S be the set of all indexes i such that B_i is not a wildcard $*$ at index i. Since $P_{B_*}(\mathcal{I}) = 0$ we know that there is some $i \in S$ such that $B_i \neq \mathcal{I}_i$. Then the decryption equation (3) contains a factor

$$e(C_0, K_0) \ / \ e(C_{i,1}, K_{i,1}) \ e(C_{i,2}, K_{i,2}) = e(v, u_i)^{(B_i - \mathcal{I}_i) \cdot s r_{i,1}}$$

which is a uniformly distributed value in $\mathbb{G}_{T,p}$ and is independent of the rest of the equation. Since the message space is of size $n^{1/4}$ and the size of $\mathbb{G}_{T,p}$ is approximately $n^{1/2}$, the false positive probability is at most $1/n^{1/4}$, which is negligible in the security parameter as required. □

We note that in practice there is no need to use a small message space $\mathcal{M} \subseteq \mathbb{G}_T$ to determine if decryption succeeded. We only use $\mathcal{M}$ to simplify the description of the system. In practice, one could do the following. The encryptor first picks a random $k \in \mathbb{G}_T$ and derives two uniform and independent b-bit symmetric keys (k_0, k_1) from k. It encrypts the payload M using a symmetric encryption system under key k_0 to obtain C_1. Next, it runs our *Encrypt*(PK, $\mathcal{I}$, k) to obtain C. The final ciphertext is the tuple (C, C_1, k_1). Now, our *Query* algorithm works as follows. It first recovers a k' from C using the given token TK. Next, it derives (k'_0, k'_1) from k' and outputs $\perp$ if $k'_1 \neq k_1$. Otherwise, it outputs the decryption of C_1 under k'_0 using a symmetric system. Lemma 2 shows that the false error probability is now $1/2^b$. Alternatively, if the symmetric encryption system provides authenticated encryption, then one could decide if *Query* produced the right value based on whether symmetric decryption succeeded.

Extensions In our description above we limited the index space Σ to be $\mathbb{Z}_m$. We can expand this space to all of $\{0,1\}^*$ by taking a large enough m to contain the range of a collision-resistant hash function. Then *Encrypt*(PK, $\mathcal{I} \in (\{0,1\}^*)^\ell$, $M \in \mathbb{G}_T$) first hashes all the coordinates of $\mathcal{I}$ into $\mathbb{Z}_m$ using the collision resistant hash and then applies the *Encrypt* algorithm described above.

5.3 Proof of Security

We prove our scheme selectively secure (as defined in Section 2.3) under the composite 3-party Diffie-Hellman assumption and the bilinear Diffie-Hellman assumption. We give the high-level arguments of the proof in this section and defer the proofs of some lemmas to the full version of our paper [11].

Suppose the adversary commits to vectors $L_0, L_1 \in \Sigma^\ell$ at the beginning of the game. Let X be the set of indexes i such that $L_{0,i} = L_{1,i}$ and $\overline{X}$ be the set of indexes i such that $L_{0,i} \neq L_{1,i}$.

The proof uses a sequence of $2\ell+2$ games to argue that the adversary cannot win the original security game of Section 2.3 which we denote by G. We begin by slightly modifying the game G into a game G'. Games G and G' are identical except for how the challenge ciphertext is generated. In G' if $M_0 \neq M_1$ then the adversary multiplies the challenge ciphertext component C' by a random element of $\mathbb{G}_{T,p}$. The rest of the ciphertext is generated as usual. Additionally, if $M_0 = M_1$ then the challenge ciphertext is generated correctly.

Lemma 3. *Assume that the Bilinear Diffie-Hellman assumption holds. Then for any polynomial time adversary $\mathcal{A}$ the difference of advantage of $\mathcal{A}$ in game G and game G' is negligible.*

The proof is in the full version of our paper [11].

Next, we define a game $\tilde{G}$. In this game the adversary will give two challenge messages, M_0, M_1. If $M_0 \neq M_1$ then the challenger outputs a random element of $\mathbb{G}_T$ as the C' component of the challenge ciphertext. The rest of ciphertext is constructed as normal. If $M_0 = M_1$ the challenger outputs the challenge ciphertext as normal.

Lemma 4. *Assume that the Composite 3-party Diffie-Hellman assumption holds. Then for any polynomial time adversary $\mathcal{A}$ the difference of advantage of $\mathcal{A}$ in game G' and game $\tilde{G}$ is negligible.*

The proof is in the full version of our paper [11].

Finally, we define two sequences of hybrid games G_j and G'_j for $j = 1, \ldots, |\overline{X}|$. We define the game G_j as follows. Let $\tilde{X}$ be a set containing the first j indexes in $\overline{X}$. The challenger creates the challenge ciphertext components C_0 and $C_{i,1}, C_{i,2}$ as normal for all $i \notin \tilde{X}$. However, for all $i \in \tilde{X}$ the challenger creates $C_{i,1}, C_{i,2}$ as completely random group elements in $\mathbb{G}$. Additionally, if $M_0 \neq M_1$ then C' is replaced by a completely random element from $\mathbb{G}_T$ (otherwise it is created as normal).

We define a game G'_j as follows. Let $\tilde{X}$ be a set containing the first j indexes in $\overline{X}$ and let δ be the $(j+1)$-th index in $\overline{X}$. In the challenge ciphertext the challenger creates C_0 and $C_{i,1}, C_{i,2}$ as normal for all $i \notin \tilde{X}$ and $i \neq \delta$. For all $i \in \tilde{X}$ the challenger creates $C_{i,1}, C_{i,2}$ as completely random group elements in $\mathbb{G}$. Finally, the challenger chooses a random s' and creates

$$C_{\delta,1} = (u_p^{\mathcal{I}_\delta} h_p)^{s'} g_q^{z_{\delta,1}}, \quad C_{\delta,2} = g_p^{s'} g_q^{z_{\delta,2}}.$$

Additionally, if $M_0 \neq M_1$ then C' is replaced by a completely random element from $\mathbb{G}_T$ (otherwise it is created as normal).

Observe that for all i in $\tilde{X}$ the challenge ciphertext contains no information about $L_{\beta,i}$. Therefore the adversary's advantage in game $G_{|\overline{X}|}$ is 0. Additionally, game G_0 is equivalent to $\tilde{G}$. We state the following two lemmas whose proofs are given in the full version of our paper [11].

Lemma 5. *Assume the Composite 3-party Diffie-Hellman assumption holds. Then for all j and any polynomial time adversary $\mathcal{A}$ the difference of advantage of $\mathcal{A}$ in game G_j and game G'_j is negligible.*

Lemma 6. *Assume the Composite 3-party Diffie-Hellman assumption holds. Then for all j and any polynomial time adversary $\mathcal{A}$ the difference of advantage of $\mathcal{A}$ in game G'_j and game G_{j+1} is negligible.*

It now follows that if the Composite 3-party Diffie-Hellman and Bilinear Diffie-Hellman assumptions hold then no polynomial-time adversary can break our scheme with non-negligible advantage. This follows from the sequence of hybrid games starting with the original game G:

$$G,\ \tilde{G},\ G'_0,\ G_1,\ G_{1'},\ G_2,\ G_{2'},\ \ldots,\ G_{|\overline{X}|}.$$

The adversary's advantage in the game $G_{|\overline{X}|}$ is 0 and the difference in adversary's advantage between any two consecutive hybrid games is negligible by the lemmas above. Hence, no polynomial adversary can win game G with non-negligible advantage.

6 Applications of HVE

We show how HVE leads to efficient systems for subset queries and conjunctive comparison queries. Throughout the section we let $\Sigma_{01} = \{0,1\}$ and $\Sigma_{01*} = \{0,1,*\}$.

Conjunctive comparison queries. In Section 3.1 we defined conjunctive comparison queries and the predicate family $\Phi_{n,w}$. We use HVE to build a $\Phi_{n,w}$-searchable encryption system with ciphertext size $O(nw)$ and token size $O(w)$.

Let ($Setup_{HVE}$, $Encrypt_{HVE}$, $GenToken_{HVE}$, $Query_{HVE}$) be a secure HVE over Σ_{01}^{nw}. Thus, the width of this HVE is $\ell = nw$. We construct a $\Phi_{n,w}$-searchable system as follows:

- *Setup*(λ) is the same as $Setup_{HVE}(\lambda)$.
- *Encrypt*(PK, I, M) where $I = (x_1, \ldots, x_w) \in \{1, \ldots, n\}^w$. Build a vector $\sigma(I) = (\sigma_{i,j}) \in \Sigma_{01}^{nw}$ as follows:

$$\sigma_{i,j} = \begin{cases} 1 & \text{if } j \geq x_i, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

  Then output $Encrypt_{HVE}$(PK, $\sigma(I)$, M) which gives a ciphertext of size $O(nw)$. For example, for $w = 2$ and $I = (x_1, x_2)$ the vector $\sigma(I)$ looks like:

$$\sigma(S) = \begin{array}{|c|c|c|c|c|c|c||c|c|c|c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{x_1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{n} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{x_2} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{n} \\ \hline 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ \hline \end{array} \in \{0,1\}^{2n}$$

- *GenToken*(SK, $\langle P_{\bar{a}} \rangle$) where $\bar{a} = (a_1, \ldots, a_w) \in \{1, \ldots, n\}^w$. Define $\sigma_*(\bar{a}) = (\sigma_{i,j}) \in \Sigma_{01*}^{nw}$ as follows:

$$\sigma_{i,j} = \begin{cases} 1 & \text{if } x_i = j, \\ * & \text{otherwise} \end{cases} \tag{5}$$

  Output $\mathrm{TK}_{\bar{a}} \xleftarrow{R} GenToken_{HVE}(\mathrm{SK}, \sigma_*(\bar{a}))$ which gives a token of size $O(w)$. For example, for $w = 2$ and $\bar{a} = (x_1, x_2)$ the vector $\sigma_*(\bar{a})$ looks like:

$$\sigma_*(\bar{a}) = \begin{array}{|c|c|c|c|c|c|c||c|c|c|c|c|c|c|}
\multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{x_1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{n} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{x_2} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{n} \\
\hline
* & \cdots & * & 1 & * & \cdots & * & * & \cdots & * & 1 & * & \cdots & * \\
\hline
\end{array} \in \{0,1,*\}^{2n}$$

– *Query*$(\mathrm{TK}_{\bar{a}}, C)$ output *Query*$_{HVE}(\mathrm{TK}_{\bar{a}}, C)$

To argue correctness and security, observe that for a predicate $P_{\bar{a}} \in \Phi_{n,w}$ and an index $I \in \{1, \dots, n\}^w$ we have that: $P_{\bar{a}}(I) = 1$ if and only if $P^{\mathrm{HVE}}_{\sigma_*(\bar{a})}(\sigma(I)) = 1$. Therefore, correctness and security follow from the properties of the HVE. We thus obtain the following immediate theorem.

Theorem 1. *(Setup, Encrypt, GenToken, Query) is a selectively secure $\Phi_{n,w}$- searchable system assuming ($Setup_{HVE}$, $Encrypt_{HVE}$, $GenToken_{HVE}$, $Query_{HVE}$) is an HVE over Σ_{01}^{nw}.*

Conjunctive range queries. We note that a system that supports comparison queries can also support range queries. To search for plaintexts where $x \in [a, b]$ the encryptor encrypts the pair (x, x). The predicate then tests $x \geq a \ \wedge \ x \leq b$.

6.1 Subset queries

Next, we show how to search for general subset predicates. Let T be a set of size n. For a subset $A \subseteq T$ we define a subset predicate as follows:

$$P_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

We wish to support searches for any subset predicate. More generally, we wish to support searches for conjunctive subset predicates over T^w. That is, let $\sigma = (A_1, \dots, A_w)$ be a w-tuple where $A_i \in T$ for all $i = 1, \dots, w$. Then σ is an elements of $(2^T)^w$. Define the predicate $P_\sigma : T^w \to \{0, 1\}$ as follows:

$$P_\sigma\big((x_1, \dots, x_w)\big) = \begin{cases} 1 & \text{if } x_i \in A_i \text{ for all } i = 1, \dots, w, \\ 0 & \text{otherwise} \end{cases}$$

Let $\Phi = \{\ P_\sigma$ for all $\sigma \in (2^T)^w\}$. Note that Φ is huge — its size is 2^{nw}.
The Φ-searchable system is as follows:

– *Encrypt*(PK, I, M) where $I = (x_1, \dots, x_w) \in T^w$. Build a vector $\sigma(S) = (\sigma_{i,j}) \in \Sigma_{01}^{nw}$ as:
$$\sigma_{i,j} = \begin{cases} 1 & \text{if } x_i = j, \\ 0 & \text{otherwise} \end{cases} \tag{6}$$
Then output *Encrypt*$_{HVE}(\mathrm{PK}, \sigma(I), M)$. The ciphertext size is $O(nw)$ as was the case for comparison queries.

– *GenToken*(SK, $\langle P_\alpha \rangle$) where $\alpha = (A_1, \ldots, A_w)$. Define $\sigma_*(\alpha) = (\sigma_{i,j}) \in \Sigma_{01*}^{nw}$ as follows:

$$\sigma_{i,j} = \begin{cases} 0 & \text{if } j \notin A_i, \\ * & \text{otherwise} \end{cases} \tag{7}$$

Output $\mathrm{TK}_\alpha \stackrel{R}{\leftarrow} \mathit{GenToken}_{HVE}(\mathrm{SK}, \sigma_*(\alpha))$. The token size is $O(nw)$, which is bigger than tokens for comparison queries.

– *Setup* and *Query* are the same algorithms from the HVE system, as for comparison queries.

It is easiest to see how this works in the one dimensional setting, namely $w = 1$. We encrypt a value $x \in T$ using an HVE vector

$$\sigma(x) = \begin{array}{c} \begin{array}{ccccccc} 1 & & & x & & & n \end{array} \\ \boxed{0 \mid \cdots \mid 0 \mid 1 \mid 0 \mid \cdots \mid 0} \end{array} \quad \in \{0,1\}^n$$

Consider a predicate P_A where, for example, $A = \{2, 3, n\} \subseteq T$. We generate a token for P_A by calling $\mathit{GenToken}_{HVE}(\mathrm{SK}, \sigma_*(A))$ using the HVE vector

$$\sigma_*(A) = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & & & n \end{array} \\ \boxed{0 \mid * \mid * \mid 0 \mid 0 \mid \cdots \mid 0 \mid *} \end{array} \quad \in \{*,1\}^n$$

The main point is that $x \in A$ if and only if $P^{\mathrm{HVE}}_{\sigma_*(A)}(\sigma(x)) = 1$. Therefore, correctness and security follow from the properties of the HVE. We obtain a secure system for subset queries for *arbitrary subsets*.

Theorem 2. *(Setup, Encrypt, GenToken, Query) is a selectively secure Φ- searchable system assuming ($Setup_{HVE}$, $Encrypt_{HVE}$, $GenToken_{HVE}$, $Query_{HVE}$) is an HVE over Σ_{01}^{nw}.*

Note that the trivial system of Section 3 for subset queries produces ciphertexts of size $O(2^n)$. The construction above generates ciphertexts of size $O(n)$.

Subset queries on large domains using Bloom filters. So far we considered subset queries over a domain of size n. In Section 1 we presented examples where one wishes to test a subset relation over a large domain. For example, we discussed email filtering queries of type ($\mathsf{sender} \in S$) where S is a set of email addresses. To use our construction one would first hash email addresses to a set $\{1, \ldots, n\}$ for some n, using a publicly known hash function, and then use the HVE for small domain.

Unfortunately, by hashing into a small domain there is some chance for false positives, namely *Query* may output M even though ($\mathsf{sender} \notin S$). False positives result from hash collisions. The false positive probability can be reduced by a standard application of Bloom filters [5]. Instead of using one hash function, we use multiple functions $H_1, \ldots, H_d : \{0,1\}^* \to T$. Again, consider the one-dimensional case, namely $w = 1$. To encrypt a word $W \in \{0,1\}^*$ the encryptor creates a vector $\sigma(W) \in \{0,1\}^n$ that contains a '1' at positions

$H_1(W), \dots, H_d(W)$ and '0' everywhere else. The encryptor then runs *Encrypt*(PK, $\sigma(W), M$).

To generate a token for a set $A = \{W_1, \dots, W_s\}$ the *GenToken* algorithm builds a vector $\sigma_*(A) \in \{0, *\}^n$ that contains $*$ at positions $H_i(W_j)$, for all $i = 1, \dots, d$ and $j = 1, \dots, s$, and contains '0' everywhere else. By choosing n and d appropriately, the false positive probability can be made arbitrarily small.

Another subset query application. In our subset query application we identified a ciphertext with an element x and a user's token with a set A. This allowed us to test whether $x \in A$. We observe that we can easily apply HVE to achieve the opposite semantics where a user's key is associated with an element x and the ciphertext with a set A. This could be used by a gateway to test if a particular user was one of the (possibly) many receivers of an email. We expect there to be several other applications that one can build with HVE.

7 Extensions

Privacy for search queries. In some cases one may want the token TK_P not to identify which predicate P is being queried. For example, in the anti-spam example from the introduction, the user may not want to reveal his anti-spam predicate to the server. A similar problem was studied by Ostrovsky and Skeith [18] and is related to Private Information Retrieval [16]. For public-key systems supporting comparison queries this is clearly not possible since, given TK_P the server can identify the threshold in P with a simple binary search. It is an open problem to convert our system to a symmetric-key system where TK_P does not expose P. One approach is to simply keep the public key secret from the server; however, this is not sufficient in our system.

Validating ciphertexts. Throughout the paper we assumed that the encryptor is honestly creating ciphertexts as specified by the encryption system. For some applications discussed in the introduction (e.g. spam filtering) this may not be the case. By creating malformed ciphertexts an attacker may generate false-positive or false-negatives for the server using the tokens.

Fortunately, in some settings including a payment gateway or spam filter, this is easily avoidable. Briefly, one technique is as follows. The recipient who has SK will also publish a regular public-key PK_1 and ask the encryptor to encrypt the plaintext (I, M) with both the searchable system and with PK_1. The resulting ciphertext is the pair $C = \big(\mathit{Encrypt}(\mathrm{PK}, I, M),\ \mathit{Encrypt}_{PKE}(\mathrm{PK}_1, (I, M))\big)$. When the recipient receives a ciphertext $C = (C_0, C_1)$ it recovers (I, M) from C_1 and uses SK to test that C_0 is a valid encryption of (I, M). If not then the ciphertext is immediately rejected. In doing so, the recipient automatically drops invalid ciphertexts. More precisely, a Φ-searchable system could provide an algorithm $\mathit{Test}(C, I, M, \mathrm{SK})$ that outputs `true` when C is a valid encryption of (I, M) and `false` otherwise. Our HVE system supports this type of test.

Alternatively, one could require the encryptor to prove that his ciphertext is well formed, for example to prove that C_0 is consistent with C_1. This can be done using non-interactive proof techniques [6, 7].

8 Conclusion

In public key systems supporting queries on encrypted data a secret key can produce tokens for testing any supported query predicate. The token lets anyone test the predicate on a given ciphertext without learning any other information about the plaintext. We presented a general framework for analyzing security of searching on encrypted data systems. We then constructed systems for comparisons and subset queries as well as conjunctive versions of these predicates.

The underlying tool behind these new constructions is a primitive we call HVE. The one-dimensional version of HVE (namely $\ell = 1$) is essentially an Anonymous IBE system. For large ℓ we obtain a new concept that is extremely useful for a large variety of searching predicates. We note that by setting $\ell = 1$ in our HVE construction we obtain a new simple anonymous IBE system secure without random oracles.

This work posses many challenging open problems. For example, the best non-conjunctive (i.e. $w = 1$) comparison system we currently have requires ciphertexts of size $O(\sqrt{n})$ where n is the domain size. In principal it should be possible to improve this to $O(\log n)$, but this is currently a wide open problem that will require new ideas. Similarly, for non-conjunctive subset queries the best we have requires ciphertexts of size $O(n)$. Again, can this be improved to $O(\log n)$? Our results mostly focus on conjunction. Are there similar results for disjunctive queries? More generally, what other classes of predicates can we search on?

Acknowledgments

We thank Amit Sahai and Alice Silverberg for helpful comments about this work.

References

[1] 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. In *CRYPTO*, pages 205–222, 2005.

[2] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Efficiently-searchable and deterministic asymmetric encryption. `http://eprint.iacr.org/2006/186`, 2006.

[3] J. Bethencourt, H. Chan, A. Perrig, E. Shi, and D. Song. Anonymous multi-attribute encryption with range query and conditional decryption. Technical report, C.M.U, 2006. CMU-CS-06-135.

[4] John Bethencourt, Dawn Song, and Brent Waters. New constructions and practical applications for private stream searching. In *Proceeding of 2006 IEEE Symposium on Security and Privacy*, 2006.

[5] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.

[6] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.

[7] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafial Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of Eurocrypt '04*, 2004.

[9] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Eurocrypt '06*, 2006.

[11] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. Cryptology ePrint Archive, Report 2006/287, 2006. `http://eprint.iacr.org/`.

[12] Dan Boneh and Brent Waters. A fully collusion resistant broadcast trace and revoke system with public traceability. In *ACM Conference on Computer and Communication Security (CCS)*, 2006.

[13] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Crypto '06*, 2006.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation by oblivious rams. *JACM*, 1996.

[15] Philippe Golle, Jessica Staddon, and Brent R. Waters. Secure conjunctive keyword search over encrypted data. In *ACNS*, pages 31–45, 2004.

[16] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[17] Rafail Ostrovsky. *Software protection and simulation on oblivious RAMs.* PhD thesis, M.I.T, 1992. Preliminary version in STOC 1990.

[18] Rafail Ostrovsky and William Skeith. Private searching on streaming data. In *Proceedings of Crypto 2005*, LNCS. Springer, 2005.

[19] Dawn Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE symposium on Security and Privacy (S&P 2000)*, 2000.

[20] Brent Waters, Dirk Balfanz, Glenn Durfee, and Dianna Smetters. Building an encrypted and searchabe audit log. In *Proceedings of NDSS '04*, 2004.

A Proof of Lemma 1

We prove that the trivial system presented in Section 3 is secure.

Proof. Showing that $\mathsf{QU\ Adv}_{\mathcal{A}}$ is negligible is a straight forward hybrid argument. Let $\mathcal{A}$ be an adversary playing the query security game. For $i = 1, \ldots, n+1$ we define experiment number i as follows:

– The challenger runs $\mathit{Setup}(\lambda)$ to obtain

$$\mathrm{PK} \leftarrow (\mathrm{PK}_1, \ldots, \mathrm{PK}_n) \quad \text{and} \quad \mathrm{SK} \leftarrow (\mathrm{SK}_1, \ldots, \mathrm{SK}_n)$$

It gives PK to $\mathcal{A}$. Next, $\mathcal{A}$ is given the tokens for any predicates of its choice.

– Then $\mathcal{A}$ outputs two pairs (I_0, M_0) and (I_1, M_1) subject to the restrictions of the query security game challenge phase. For $j = 1, \ldots, n$ the challenger constructs the following ciphertexts:

$$C_j \xleftarrow{R} \begin{cases} \mathit{Encrypt}'(\mathrm{PK}_j,\ M_0) & \text{if } P_j(I_0) = 1 \text{ and } j \geq i, \\ \mathit{Encrypt}'(\mathrm{PK}_j,\ M_1) & \text{if } P_j(I_1) = 1 \text{ and } j < i, \\ \mathit{Encrypt}'(\mathrm{PK}_j,\ \perp) & \text{otherwise} \end{cases}$$

The challenger gives $C \leftarrow (C_1, \ldots, C_n)$ to $\mathcal{A}$.

– The adversary continues to adaptively request query tokens subject to the restrictions of the query security game. Finally, $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. We let $\mathsf{EXP}^{(i)}_{\mathrm{QU}}[\mathcal{A}]$ denote the probability that β' equals 1.

This completes the description of experiment i. A standard argument shows that

$$2 \cdot \mathsf{QU\,Adv}_{\mathcal{A}} = \left|\mathsf{EXP}^{(1)}_{\mathrm{QU}}[\mathcal{A}] - \mathsf{EXP}^{(n+1)}_{\mathrm{QU}}[\mathcal{A}]\right| \leq \sum_{i=1}^{n} \left|\mathsf{EXP}^{(i)}_{\mathrm{QU}}[\mathcal{A}] - \mathsf{EXP}^{(i+1)}_{\mathrm{QU}}[\mathcal{A}]\right|$$

But $\left|\mathsf{EXP}^{(i)}_{\mathrm{QU}}[\mathcal{A}] - \mathsf{EXP}^{(i+1)}_{\mathrm{QU}}[\mathcal{A}]\right|$ is clearly negligible assuming $\mathcal{E}$ is semantically secure against chosen plaintext attacks.