Spring 4-18-2022

Performance Improvements in Inner Product Encryption

Serena Riback
serena.riback@uconn.edu

Performance Improvements in Inner Product Encryption

Serena Riback

April 2022

University of Connecticut Department of Computer Science and Engineering
Undergraduate Honors Thesis
Thesis Supervisor: Benjamin Fuller
Honors Advisor: Jerry Shi

Abstract

Consider a database that contains thousands of entries of the iris biometric. Each entry identifies an individual, so it is especially important that it remains secure. However, searching for entries among an encrypted database proves to be a security problem - how should one search encrypted data without leaking any information to a potential attacker? The proximity searchable encryption scheme, as discussed in the work by Cachet et al., uses the notions of inner product encryption developed by Kim et al.. In this paper, we will focus on the efficiency of these schemes. Specifically, how the symmetry of the bilinear pairing group effects the time required to execute a search.

1 Introduction

Consider a database of sensitive data. It is easy to see why encrypting this data is important - the owner of the data would not want an attacker to be able to steal it. But encrypting the database creates another issue: how to search for a specific entry. Consider a query, q. The easiest, but likely least efficient, way to search for an entry would be to decrypt everything, and compare all with q until a match is found. However, for a database with thousands of entries, it is easy to see why this might not be the best method of searching. The goal is to be able to efficiently search the database while the entries remain encrypted.

That being said, searching encrypted entries implies there needs to be some way of grouping those that are similar and thus, may match the query q. This opens up the possibility for leakage - i.e. based on q, there is leakage if an attacker can learn any information at all about related entries [WLD+17, IKK12, GLMP18, CGPR15]. This may be the entries themselves, or even just components of them that are similar. While it ultimately depends on the user's definition of security, our goal is to minimize leakage.

There are multiple possible solutions for this problem of searchable encryption [FVY+17, CGKO11], but the solution we will be discussing is based off the work done by Kim et al [KLM+18]. on function hiding inner product encryption and Cachet et al. [ACD+22] on its application to proximity searchable encryption. In order to understand this work, one must first understand inner product encryption. In an inner product encryption scheme, the secret key and ciphertext are given in the form of vectors $\vec{x}$ and $\vec{y}$, respectively. Instead of decrypting normally, the decrypt function in this scheme returns the inner product of these two vectors [OT12, KT14, KLM+18, BJK15].

We will be discussing these schemes in the context of a biometric database. Biometric data, since it relates to attributes of individuals, is especially sensitive [DRD+20]. Biometrics, such as fingerprints, facial, vocal, iris, or vein patterns, are used to identify certain individuals [Dau14] [Fou]. These identifiers are unlikely to change drastically as time goes on, and can be compromised easily [AF19].

As is done in Cachet et al. [ACD+22], we will be focusing on the iris biometric. Similar to all biometrics, iris data is especially noisy [SSF19, Dau09, HBF10]. This means that two scans of the same iris may product similar, but slightly different results [ACD+22]. Therefore, when searching for specific iris data points, it is important to be able to perform a proximity search. Given a query q, the search algorithm should return all entries within a certain distance from the query [ACD+22]. The proximity search algorithm as described in Cachet et al. [ACD+22] uses the function hiding inner product encryption scheme as described in Kim et al. [KLM+18] to locate encrypted entries in an iris biometric database. Given a query q, the algorithm calculates the inner product between q and the stored records, and returns the records with the appropriate inner product.

In this paper, we will mainly focus on the efficiency of the proximity search and inner product encryption algorithms. It was discovered that the efficiency of the search algorithm relates directly to the efficiency of the decryption algorithm. That being said, the speed of the decryption algorithm is dependent on the type of elliptic curve that is used for the group pairing operations. In the scheme defined by Kim et al. [KLM+18], the MNT159 curve, a 159-bit asymmetric pairing curve [AGM+13], is used. By switching to SS512, a 512-bit symmetric pairing curve [AGM+13], the efficiency of the search algorithm improved by a factor of 4. While the security of these algorithms has been proved for an asymmetric pairing [ACD+22, KLM+18], the proof must be extended to include new cases in which objects from the same group can be paired. This paper will set up and show that the new cases do not impact the security of the schemes.

1.1 Contributions

This paper expands on the work done by Cachet et al. [ACD+22]. The results of this thesis are integrated into the latest online version [CAD+20]. The Multi Random Projection IPE scheme uses an asymmetric pairing group as described in Kim et al. [KLM+18]. However, this paper proves that the use of a symmetric pairing group is 4 times more efficient, while still maintaining the security of the scheme. The body of this thesis is organized around showing these two points with Section 4 establishing security of the scheme with symmetric pairings and Section 5 showing performance of the revised scheme.

2 Preliminaries

This section discusses various preliminary topics required for the understanding of the rest of this paper.

2.1 Bilinear Mapping Groups [Wei40, Jou04, BF01]

A bilinear mapping describes a function that maps elements from a set of 2 groups to a 3rd group. These mappings are often described as pairing functions because they pair two elements into a target group. Consider a set of 3 groups, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, with a deterministic map e defined as follows:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators for their respective groups. The 3 groups have a prime order of q. Let $a, b \in \mathbb{Z}_q$.

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \in \mathbb{G}_T$$

$$e(g_1, g_2) \neq 1$$

If $\mathbb{G}_1 \neq \mathbb{G}_2$, we say this is an asymmetric pairing. Likewise, if $\mathbb{G}_1 = \mathbb{G}_2$, it is symmetric.

Definition 1 ([ACD+22] Asymmetric Bilinear Group)**.** *Suppose $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are three groups (respectively) of prime order q with generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $g_T \in \mathbb{G}_T$ respectively. We denote a value x encoded in $\mathbb{G}_1$ with either g_1^x or $[x]_1$, we denote values encoded in $\mathbb{G}_2$ and $\mathbb{G}_T$ similarly. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate (i.e. $e(g_1, g_2) \neq 1$) bilinear pairing operation such that for all $x, y \in \mathbb{Z}_q$, $e([x]_1, [y]_2) = e(g_1, g_2)^{xy}$. We assume the group operations in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing operation e are efficiently computable, then $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ defines an asymmetric bilinear group.*

Definition 2 (Symmetric Bilinear Group)**.** *Suppose $\mathbb{G}_1$ and $\mathbb{G}_T$ are two groups (respectively) of prime order q with generators $g_1 \in \mathbb{G}_1$ and $g_T \in \mathbb{G}_T$ respectively. We denote a value x encoded in $\mathbb{G}_1$ with either g_1^x or $[x]_1$, we denote values encoded in $\mathbb{G}_T$ similarly. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a non-degenerate (i.e. $e(g_1, g_1) \neq 1$) bilinear pairing operation such that for all $x, y \in \mathbb{Z}_q$, $e([x]_1, [y]_1) = e(g_1, g_1)^{xy}$. We assume the group operations in $\mathbb{G}_1$ and $\mathbb{G}_T$ and the pairing operation e are efficiently computable, then $(\mathbb{G}_1, \mathbb{G}_T, q, e)$ defines an symmetric bilinear group.*

2.2 Generic Group Model [Sho97]

The Generic Group Model is a cryptographic model in which an attacker is able to view handles of group elements, instead of the group elements themselves. A handle is a random string determined by a truly random function, σ. For example, for each $g_1, g_2, g_3 \in \mathbb{G}$ for a group $\mathbb{G}$, the attacker can only see $\sigma(g_1), \sigma(g_2), \sigma(g_3)$. Additionally the attacker has access to a group oracle. Given the handles $\sigma(g_1)$ and $\sigma(g_2)$, they are allowed to query the oracle for $\sigma(g_1 + g_2)$.

The Generic Group Model is often used for assessing the hardness of various assumptions and protocols. That being said, we will now walk through a proof as an example of how certain assumptions are proved to be secure in the Generic Group Model. We will use the following description of Decisional Diffie-Hellman.

2.2.1 Decisional Diffie-Hellman (DDH) [KL14]

Consider a cyclic group $\mathbb{G}$ with order n and generator $g \in \mathbb{G}$. Let $x, y \in \mathbb{Z}_n$ be independently chosen values, and consider g^x and g^y.

Computational hardness in DDH states that an attacker cannot distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, g^z) for independently chosen $z \in \mathbb{Z}_n$. In other words, g^{xy} appears to be a uniformly random element in $\mathbb{G}$.

Theorem 1. *Decisional Diffie-Hellman is secure in the Generic Group Model.*

Proof. Let $\mathbb{G}$ be a cyclic group with prime order q and generator $g \in \mathbb{G}$. Let $x, y, z \in \mathbb{Z}_q$ be independently chosen values.

Consider an efficient attacker. The attacker receives $\sigma(e), \sigma(g), \sigma(g^a), \sigma(g^b), \sigma(g^{ab}), \sigma(g^z)$ and constructs a table of known handles. The attacker can only ask the oracle for handles of elements they have already been given. Note however, that the attacker can query the oracle for $\sigma(g^{a+b})$ and other linear combinations of g^a and g^b. If these resulting handles are not already in their known table, they get added. Additionally, the attacker may test the string equality of different handles - i.e. if $\sigma(g^a) = \sigma(g^b)$.

The scheme is considered secure if the attacker cannot distinguish between $\sigma(g^{ab})$ and $\sigma(g^z)$. We must prove each of the following requirements for security:

1. The distributions of $\sigma(g^{ab})$ and $\sigma(g^z)$ must be identical.
2. The range of σ must be significantly larger than the domain.
3. The attacker cannot calculate $\sigma(g^{ab})$ by adding $\sigma(g^a)$ b-times (or vice-versa).

Claim 1. *The distributions of $\sigma(g^{ab})$ and $\sigma(g^z)$ are identical.*

We know that a, b, z are selected uniformly at random from $\mathbb{Z}_q$. Therefore, a, b, ab and z have the same distributions. This implies g^{ab} and g^z will also have the same distributions.

Claim 2. *The range of σ is significantly larger than the domain.*

The domain of σ is simply the order of $|\mathbb{G}| = q$. Although q is a significantly large prime, the range of σ is approximated to be q^3. Therefore we can say there is a high probability of no repeated handles.

Claim 3. *The probability that the attacker calculates $\sigma(g^{ab})$ by adding $\sigma(g^a)$ b-times (or vice versa) is negligible.*

Given $\sigma(g^a)$ and $\sigma(g^b)$, we say the probability of guessing a or b from these handles is at most $\frac{2}{q}$.

$$\forall a \neq 0, \Pr[A = a | \sigma(g^a), \sigma(g^b)] = \frac{1}{q-1}$$

The attacker must be able to guess a or b in order to compute $\sigma(g^{ab})$ and be able to distinguish from $\sigma(g^z)$. For each query, the probability of the attacker accidentally guessing $\sigma(g^a), \sigma(g^b)$ or $\sigma(g^{ab})$ increases by at most $\frac{3}{q}$. Therefore, for n queries, the probability the attacker will guess one of the handles necessary for distinguishing is $\frac{2+3n}{q}$. If $n = \texttt{poly}(\lambda)$ and $q = \omega(\texttt{poly}(\lambda))$, then this probability is negligible.

□

2.3 Function Hiding Inner Product Encryption (FHIPE) [KLM+18, Lew16]

We now discuss the Function Hiding Inner Product Encryption scheme as it is described in Kim et al..

The Setup function takes in 2 important parameters; the dimension (n), and the group (group_name). Unless otherwise specified, group_name is set to MNT159. This denotes an asymmetric elliptic curve with a 159-bit base field. This function initializes the pairing groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and two random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Matrix B is also initialized from the group $\mathbb{GL}_n(\mathbb{Z}_q)$, along with its determinant $\det(B)$, and B^*. Note that $B^* = det(B)(B^{-1})^T$. B is a matrix with size $n \times n$ and entries from the group $\mathbb{Z}_q$. Setup returns the master secret key, ms_k, and public parameters, pp, which is empty by default. The master secret key is equal to the following:

$$ms_k = (det(B), B, B^*, \mathbb{G}, g_1, g_2)$$

The TokGen function takes in 2 parameters, ms_k and $\hat{x}$, a vector of length n from the group $\mathbb{Z}_q^n$. Random variable α is initialized from $\mathbb{Z}_q$. This function returns the key in the form (k_1, k_2). This key is equal to the following:

$$sk_x = (k_1, k_2) = (g_1^{\alpha det(B)}, g_1^{\alpha \hat{x} B})$$

The Encrypt function takes in a similar 2 parameters as TokGen, ms_k and $\hat{y}$, a vector of length n from the group $\mathbb{Z}_q^n$. Random variable β is initialized group $\mathbb{Z}_q$. This function returns the ciphertext in the form (c_1, c_2). The ciphertext is equal to the following:

$$ct_y = (c_1, c_2) = (g_2^{\beta}, g_2^{\beta \hat{x} B^*})$$

The Decrypt function takes in 4 parameters: pp, the public parameters, sk_x, the secret key, ct_y, the ciphertext, and a value for the max inner product. The output of this function is $\langle x, y \rangle$ as long as it is between 0 and the max inner product. This function computes t_1 and t_2, then uses the Baby-step/Giant-step algorithm [KM17] to solve for $\langle x, y \rangle$ where $t_2^{\langle x,y \rangle} = t_1$.

$$t_1 = \langle g_2^{\beta x B^*}, g_1^{\alpha x B} \rangle$$

$$t_2 = (g_2, g_1)^{\alpha \beta B B^*}$$

2.4 Multi-random Projection IPE [ACD+22]

This implementation of the inner product encryption scheme uses the same structure and makes references to the FHIPE code [KLM+18, Lew16]; however, improves performance by partitioning the matrices and vectors into bases. This saves storage space and computation time.

Performance analysis was done on the code written by Cachet et al. [ACD+21] in order to speed up the execution of the proximity search function. Although the main purpose is to search for nearby queries of certain data, this code calls FHIPE scheme. In order to make Search faster, the following functions must also be made faster: TokGen, Encrypt, Decrypt.

Ultimately, it was found that calling versions of the TokGen and Encrypt functions that leave out the calls to the pairing group, makes a huge difference. This led to the discovery of utilizing a different type of pairing curve. Originally, the curve MNT159 was being used [Lew16]. By switching to SS512, results improved significantly.

3 Definitions

This section provides various definitions that we will need for the main results that have not previously been given.

Definition 3 ([ACD+22] Secret key predicate encryption)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $\mathcal{M}$ be the set of attributes and $\mathcal{F}$ be a set of predicates. We define $PE = (PE.Setup, PE.Encrypt, PE.TokGen, PE.Decrypt)$, a secret-key predicate encryption scheme, as follows: $PE.Setup(1^\lambda) \to (sk, pp)$, $PE.Encrypt(sk, x) \to ct_x$, $PE.TokGen(sk, f) \to tk_f$, and $PE.Decrypt(pp, tk_f, ct_x) \to b$.*
We require the scheme to have the following properties:

1. Challenger draws $b \leftarrow^{\$} \{0,1\}$
2. Challenger computes $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{PE.Setup}(1^\lambda)$ and sends pp to $\mathcal{A}$ and $\mathcal{A}'$.
3. For $1 \leq i \leq s$ $\mathcal{A}$ chooses $X_i \in X$.
4. For $1 \leq j \leq r$ $\mathcal{A}$ chooses $Y_j \in Y$.
5. Define $S = X_1, ..., X_s$ and $R = Y_1, .., Y_r$.
6. $\mathcal{A}$ sends S,R to $\mathsf{PE.enc}$. Let $x_i \leftarrow X_i$ and $y_j \leftarrow Y_j$ respectively.
7. Let $z \leftarrow \mathcal{A}(\{\mathsf{PE.enc}(x_i)\}_{i=1}^r, \{\mathsf{PE.TokGen}(y_i)\}_{j=1}^s)$
8. Let $z' \leftarrow \mathcal{A}'\{\langle x_i, y_j \rangle \stackrel{?}{=} 0\}_{i=1,j=1}^{r,s}$
9. $\mathcal{A}$ outputs some value denoted as z.
10. The advantage of $\mathcal{A}$ is
$$\Pr[z = f(X,Y)] - \Pr[z' = f(X,Y)].$$

Figure 1: Semantic Security Definition of Function-Hiding, Predicate Inner Product Encryption

Correctness: *For any $x \in \mathcal{M}, f \in \mathcal{F}$,*

$$\Pr\left[f(x) = b \;\middle|\; \begin{array}{c} ct_x \leftarrow PE.Encrypt(sk,x) \\ tk_f \leftarrow PE.TokGen(sk,f) \\ b \leftarrow PE.Decrypt(pp, tk_f, ct_x) \end{array}\right] \geq 1 - (\lambda).$$

Security of admissible queries: *Let $r = \mathtt{poly}(\lambda)$ and $s = \mathtt{poly}(\lambda)$. Any PPT adversary $\mathcal{A}$ has only (λ) advantage in the PE_{IND} game defined in Figure 1. Token and encryption queries must meet the following admissibility requirements, $\forall j \in [1,r], \forall i \in [1,s]$,*

$$PE.Decrypt(pp, tk_j^{(0)}, ct_i^{(0)}) = PE.Decrypt(pp, tk_j^{(1)}, ct_i^{(1)}).$$

Remark: When we discuss performance numbers for searchable encryption, algorithms have different names. For searchable encryption we consider four different algorithms $\mathsf{BuildIndex}, \mathsf{Setup}, \mathsf{Trapdoor}, \mathsf{Search}$ which correspond to $\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{TokGen}, \mathsf{Decrypt}$ respectively.

4 Main Result

In this section we show that the construction of Cachet et al. [ACD+22] is secure when using symmetric bilinear groups. We present their scheme adapted to use symmetric bilinear groups in Figure 2. The result is the symmetric bilinear group analogue of [ACD+22, Theorem 1].

Theorem 2. *In the Generic Group Model for symmetric bilinear groups the construction in Figure 2 is a secure $\mathsf{IPE}_{fh,sk,pred}$ scheme according to Definition 3 for the family of predicates $\mathcal{F} = \{f_y | y \in \mathbb{Z}_q^n\}$ such that for all vectors $x \in \mathbb{Z}_q^n$, $f_y(x) = (\langle x, y \rangle \stackrel{?}{=} 0)$.*

Proof. In this proof we do not provide a full overview of the simulator's behavior which remains unchanged in our setting. We must argue however, that the simulator continues to be correct with the additional flexibility provided to the adversary by the ability to pair multiple elements in $\mathbb{G}$. Concretely, this

$\underline{\mathsf{Setup}(1^\lambda, n, \sigma)}$:

1. Sample $(G_1, G_T, q, e) \leftarrow \mathcal{G}_{abg}$ and randomly sample generators $g_1 \in G_1$.
2. For $1 \leq \ell \leq \sigma$, randomly samples an invertible square matrix $\mathbb{B}_\ell \in \mathbb{Z}_q^{N \times N}$ and sets $\mathbb{B}_\ell^* = (\mathbb{B}_\ell^{-1})^T$, with $N = \lceil n/\sigma \rceil + 1$.
3. Outputs $\mathsf{pp} = (G_1, G_T, q, e, n, \sigma)$ as public parameters and $\mathsf{sk} = (g_1, \{\mathbb{B}_\ell, \mathbb{B}_\ell^*\}_{\ell=1}^\sigma)$.

$\underline{\mathsf{TokGen}(\mathsf{pp}, \mathsf{sk}, y)}$:

1. Sample $\alpha \xleftarrow{\$} \mathbb{Z}_q$.
2. Splits y into σ subvectors y_ℓ of size $\lceil n/\sigma \rceil$ and pads with zeroes if needed.
3. For $1 \leq \ell \leq \sigma$, defines $y'_\ell = 1 \mid\mid y_\ell$ and sets $\mathsf{tk}_\ell = [\ \alpha \cdot (y'_\ell)^T \cdot \mathbb{B}_\ell\ ]_1$, a vector in G_1.
4. Outputs $\mathsf{tk} = (\mathsf{tk}_1, \cdots, \mathsf{tk}_\sigma)$.

$\underline{\mathsf{Encrypt}(\mathsf{pp}, \mathsf{sk}, x)}$:

1. Samples $\beta \xleftarrow{\$} \mathbb{Z}_q$.
2. Splits x into σ subvectors x_ℓ of size $\lceil n/\sigma \rceil$, and pads with zeroes if needed.
3. For $1 \leq \ell \leq \sigma - 1$, samples $\zeta_\ell \xleftarrow{\$} \mathbb{Z}_q$ then sets $\zeta_\sigma = -\sum_{\ell=1}^{\sigma-1} \zeta_\ell$.
4. For $1 \leq \ell \leq \sigma$ defines $x'_\ell = \zeta_\ell \mid\mid x_\ell$ and sets $\mathsf{ct}_\ell = [\ \beta \cdot (x'_\ell)^T \cdot \mathbb{B}_\ell^*\ ]_2$, a vector in G_1.
5. Outputs $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_\sigma)$.

$\underline{\mathsf{Decrypt}(\mathsf{pp}, \mathsf{tk}, \mathsf{ct})}$:

Computes $\left(\Pi_{\ell=1}^\sigma \Pi_{i=1}^N e(\mathsf{tk}_\ell[i], \mathsf{ct}_\ell[i])\right)$ and returns $\top$ if the results is equal to $1 \in \mathbb{G}_T$, $\perp$ otherwise.

Figure 2: Construction of MRProjSym. The only difference between this and MRProj [ACD+22] is that MRProjSym construction uses a symmetric bilinear group while MRProj uses an assymetric bilinear group.

means that the adversary has the ability to ask to pair elements of tk with other elements of tk and elements of ct with elements of ct which was not possible before. In the asymmetric group setting, the adversary was limiting to pairing elements in ct with elements in tk. We refer the reader to [ACD+22, Equation 3] which shows how the simulator splits each query into two parts $p_{i,j}$ which consists of valid decryptions (scaled by some values) and $f_{i,j}$ which consist of some other elements. As a quick review of notation:

- Let Q be the maximum number of queries made by an adversary.
- Let σ and N be as in Figure 2.
- For all $i \in [Q]$, $\ell \in [\sigma]$ and $k \in [N]$,
  - Let $\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}, \hat{x}_{\ell,k}^{(i)}, \hat{y}_{\ell,k}^{(i)}$ represent the hidden variables $\alpha^{(i)}, \beta^{(i)}, x_{\ell,k}^{(i)}, y_{\ell,k}^{(i)}$,
  - Let $\hat{b}_{\ell,k,m}$ and $\hat{b}_{\ell,k,m}^*$ represent the entry in position (k, m) of the $\mathbb{B}_\ell$ and $\mathbb{B}_\ell^*$ matrices respectively,
  - Let $\hat{\zeta}_\ell^{(i)}$ be the formal variables for $\zeta_\ell^{(i)}$ where the simulator tracks the constraints that for each $i \in [Q]$, $\sum_{\ell=1}^\sigma \hat{\zeta}_\ell^{(i)} = 0$ and let $\hat{s}_{\ell,m}^{(i)}$ and $\hat{t}_{\ell,m}^{(i)}$ represent formal polynomials:

$$\hat{s}_{\ell,m}^{(i)} = \sum_{k=1}^N \hat{y}'^{(i)}_{\ell,k} \cdot \hat{b}_{\ell,k,m} = \hat{b}_{\ell,1,m} + \sum_{k=2}^N \hat{y}_{\ell,k\text{-}1}^{(i)} \cdot \hat{b}_{\ell,k,m} \tag{1}$$

$$\hat{t}_{\ell,m}^{(i)} = \sum_{k=1}^N \hat{x}'^{(i)}_{\ell,k} \cdot \hat{b}^*_{\ell,k,m} = \hat{\zeta}_\ell^{(i)} \cdot \hat{b}^*_{\ell,1,m} + \sum_{k=2}^N \hat{x}_{\ell,k\text{-}1}^{(i)} \cdot \hat{b}^*_{\ell,k,m} \tag{2}$$

- The set of formal variables is $\mathcal{R} \cup \mathcal{T}$, where

$$\mathcal{R} = \left\{\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}\right\}_{i \in [Q]} \cup \left\{\hat{s}_{\ell,m}^{(i)}\ ,\ \hat{t}_{\ell,m}^{(i)}\right\}_{i \in [Q],\ \ell \in [\sigma],\ m \in [N]}$$

and

$$\mathcal{T} = \left\{\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}\right\}_{i \in [Q]} \cup \left\{\hat{x}'^{(i)}_{\ell,k},\ \hat{y}'^{(i)}_{\ell,k},\ \hat{\zeta}_\ell^{(i)}\right\}_{i \in [Q],\ \ell \in [\sigma],\ k \in [N]} \cup \left\{\hat{b}_{\ell,k,m}\ ,\ \hat{b}^*_{\ell,k,m}\right\}_{\ell \in [\sigma],\ m,k \in [N]}$$

For completeness we reproduce the simulator behavior on zero-test queries below:

1. It "canonicalizes" the polynomial p by expressing it as a sum of products of formal variables in $\mathcal{T}$ with $\mathtt{poly}(\lambda)$ terms.
2. If $\tau = 1$ and p is the zero polynomial, $\mathcal{S}$ outputs "zero". Otherwise it outputs "non-zero".
3. If $\tau = T$ the simulator decomposes p into the form

$$p = \sum_{i,j=1}^{Q} \hat{\alpha}^{(i)} \hat{\beta}^{(j)} \cdot \left(p_{i,j}\left(\{\hat{s}^{(i)}_{\ell,m},\ \hat{t}^{(j)}_{\ell,m}\}_{\ell\in[\sigma],m\in[N]} \right) + f_{i,j}\left(\{\hat{s}^{(i)}_{\ell,m},\ \hat{t}^{(j)}_{\ell,m}\}_{\ell\in[\sigma],m\in[N]} \right) \right) \tag{3}$$

   where for $1 \le i, j \le Q$, $p_{i,j}$ is defined as

$$p_{i,j} = c_{i,j} \cdot \left(\sum_{\ell,m=1}^{\sigma,N} \hat{s}^{(i)}_{\ell,m}\ \hat{t}^{(j)}_{\ell,m} \right)$$

   where $c_{i,j} \in \mathbb{Z}_q$ is the coefficient of the term $\hat{s}^{(i)}_{1,1}\hat{t}^{(j)}_{1,1}$, and $f_{i,j}$ consists of the remaining terms.
4. If for all $1 \le i, j \le Q$, $(i, j) = 0$ in $\mathcal{C}_{\mathsf{ip}}$ (corresponding to a zero inner product) and $f_{i,j}$ does not contain any non-zero term, $\mathcal{S}$ outputs "zero". Otherwise it outputs "non-zero".

The goal of the proof is to show for the polynomial $f_{i,j}$ we have 2 things we want to be sure of:

1. The terms of $f_{i,j}$ are low degree polynomials of the hidden variables $\mathbb{B}_\ell$ and the values α, β, ζ. The polynomial $f_{i,j}$ is either 0 for all values of x, y encrypted by the adversary or non-zero across all values of x, y.
2. That the polynomials are low-degree enough that we can use the Schwartz-Zippel to show that the nonzero polynomial $f_{i,j}$ evaluates to 0 with low probability. The probability space is the hidden randomness of the scheme, specifically the choice of $\mathbb{B}_\ell$ and the values α, β, ζ.

We note that the canonicalization into p as described in Equation 3 is efficient as discussed in [KLM+18, ACD+22].

Claim 4. *For $\tau = 1$ the simulator's behavior is correct with overwhelming probability.*

Proof of Claim 4. Note that the only monomials that the adversary obtains are in response to key generation and ciphertext queries. The canonical polynomial is of the form

$$p = \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m} \cdot \hat{s}^{(i)}_{\ell,m} \right) + \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)\prime}_{\ell,m} \cdot \hat{t}^{(i)}_{\ell,m} \right) \tag{4}$$

$$= \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m} \sum_{k=1}^{N} \hat{y}'^{(i)}_{\ell,k} \cdot \hat{b}_{\ell,k,m} \right) + \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)\prime}_{\ell,m} \sum_{k=1}^{N} \hat{x}'^{(i)}_{\ell,k} \cdot \hat{b}^{*}_{\ell,k,m} \right)$$

$$= \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m} \left(\hat{b}_{\ell,1,m} + \sum_{k=2}^{N} \hat{y}^{(i)}_{\ell,k} \cdot \hat{b}_{\ell,k,m} \right) \right) + \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)\prime}_{\ell,m} \left(\zeta^{(i)}_{\ell} \cdot \hat{b}^{*}_{\ell,1,m} + \sum_{k=2}^{N} \hat{x}^{(i)}_{\ell,k} \cdot \hat{b}^{*}_{\ell,k,m} \right) \right) \tag{5}$$

where the variables

$$\left\{ c^{(i)}_{\ell,m}, c^{(i)\prime}_{\ell,m} \right\}_{1\le\ell\le\sigma}^{1\le m\le N} \in \mathbb{Z}_q.$$

Note that the sums

$$\hat{b}_{\ell,1,m} + \sum_{k=2}^{N} \hat{y}^{(i)}_{\ell,k} \cdot \hat{b}_{\ell,k,m}$$

can not be the identically zero polynomial over the formal variables $\{\hat{b}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$. The sums

$$\zeta^{(i)}_{\ell} \cdot \hat{b}^{*}_{\ell,1,m} + \sum_{k=2}^{N} \hat{x}^{(i)}_{\ell,k} \cdot \hat{b}^{*}_{\ell,k,m}$$

can only be the identically zero polynomial over the formal variables $\{\hat{b}^{*}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$ if $\zeta^{(i)}_{\ell} = 0$ which happens with negligible probability. Both of these facts are true regardless of the actual values of the adversary's queries. Recall $\{\hat{\alpha}^{(i)}\}_{i\in[Q]}$, $\{\hat{\beta}^{(i)}\}_{i\in[Q]}$, and $\{\hat{b}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$ are sampled uniformly and independently in the real game. Furthermore, the values $\{\hat{b}^{*}_{\ell,k,m}\}_{\ell\in[\sigma],\ k,m\in[N]}$ in the real game are products formed by the inverse computation which are the sum of monomials of degree N. Thus, under the assumption that the above sums are nonzero, the entire value of p can be expressed as a nonzero polynomial of degree at most $N+1 = \mathtt{poly}(\lambda)$ in $\alpha, \beta, \hat{b}$. By the Schwartz-Zippel lemma [KLM+18, Lemma 2.9], p evaluates to non-zero with overwhelming probability for random $\alpha, \beta, \hat{b}$. This implies that the simulator is correct with overwhelming probability. This completes the proof of Claim 4. □

Claim 5. *For $\tau = T$ the simulator's behavior is correct with overwhelming probability.*

Proof of Claim 5. From [KLM+18, Lemma 3.3] the polynomial $f_{i,j}$ must contain

Case 1 A "cross-term" of the form $c * s^{(i)}_{l_i} t^{(j)}_{l_j}$ where $c \in \mathbb{Z}_q$ is non-zero and $l_1 \neq l_2$.

Case 2 Some partial inner product of the form $c * s^{(i)}_{l,k} t^{(j)}_{l,k}$ where $c \in \mathbb{Z}_q$ and $l \in [\sigma], k \in [n]$.

Case 3 Neither a cross-term or a partial inner product and must be a polynomial of only $s^{(i)}_{l}$ or $t^{(j)}_{l}$.

The above 3 cases have already been proved for an asymmetric pairing group. However, there are additional cases when the pairing group is made symmetric. Therefore, we must show that in the 3 cases, $f_{i,j}$ cannot be identically zero. We refer to the previous proofs in [KLM+18, ACD+22] for proofs that if the polynomials are nonzero they have bounded degree and are unlikely to evaluate to 0.
Note that in all settings we need to consider the fact that the $f_{i,j}$ can contain terms of the form $c * s^{(i)}_{\ell_i} s^{(j)}_{\ell_j}$ or $c * t^{(i)}_{\ell_i} t^{(j)}_{\ell_j}$ where $c \in \mathbb{Z}_q$ is non-zero. Note in the above that ℓ_i, ℓ_j may be the same or different. In all 3 cases the goal is to show that $f_{i,j}$ cannot be identically zero in each of the above cases regardless of the adversary's choice of $\{x^{(i)}, y^{(j)}\}$.

Case 1: In this case there is some cross term $c \cdot s^{(i)}_{l_i,m_i} \cdot t^{(j)}_{l_j,m_j}$ where $c \in \mathbb{Z}_q$ is non-zero and $(l_i, m_i) \neq (l_j, m_j)$. The value $c * s^{(i)}_{\ell_i,m_i} t^{(j)}_{\ell_j,m_j}$ cross-terms were constructed by strictly multiplying elements of $\mathbb{B}_\ell$ with $\mathbb{B}^{-1}_\ell$. Specifically, one can rewrite the above as

$$c \cdot s^{(i)}_{\ell_i,m_i} \cdot t^{(j)}_{\ell_j,m_j} = c\beta^{(i)}\alpha^{(j)} \left(\hat{b}_{\ell_i,1,m} + \sum_{k=2}^{N} \hat{y}^{(i)}_{\ell_i,k} \cdot \hat{b}_{\ell_i,k,m} \right) \left(\hat{\zeta}^{(j)}_{\ell_j} \cdot \hat{b}^{*}_{\ell_j,1,m_j} + \sum_{k=2}^{N} \hat{x}^{(j)}_{\ell_j,k\text{-}1} \cdot \hat{b}^{*}_{\ell_j,k,m_j} \right). \quad (6)$$

We now recall the form of $\hat{b}^*$ terms. For a value $k \in [N]$, let $S_{\neg k}$ denote $[N] \setminus k$ and let $\pi_{k,\ell}$ denote the set of bijections from $S_{\neg k} \to S_{\neg \ell}$, and $\mathtt{sgn}$ represent a sign function that maps inputs to $\{-1, 1\}$. The variable

$$\hat{b}^{*}_{\ell_i,k,m_i} = \frac{\sum_{\pi_{k,m_i}} \mathtt{sgn}(\pi, k, m_i) \prod_{r\in S_{\neg k}} b_{\ell_i,r,\pi(r)}}{\mathtt{det}(\mathbb{B})}. \quad (7)$$

Consider the expansion of Equation 6 into monomials of $\hat{b}$. Each monomial in the expansion contains the product of exactly two variables in column ℓ_i of $\mathbf{B}$ but no variables in column ℓ_j. However, we now evaluate whether $f_{i,j}$ can be identically zero should these cross-terms include polynomials multiples elements from $\mathbb{B}_\ell$ together and/or elements of $\mathbb{B}_\ell^{-1}$ together. First, we must show that these newly formed cross-terms do not contain terms that would cancel with terms from previously constructed cross-terms. This can be shown by evaluating the degrees of the polynomials. All of the monomials in numerator of the definition of Equation 6 have degree exactly N, the new polynomials available to the adversary do not. (We refer the reader to [KLM+18] for why different cross terms cannot cancel one another. Here we focus on why the newly available terms cannot cancel any cross term.) Specifically, all terms of the form $c * s^{(i)}_{\ell_i} s^{(j)}_{\ell_j}$ consist of monomials of degree exactly 2. All terms of the form $c * t^{(i)}_{\ell_i} t^{(j)}_{\ell_j}$ consist of monomials of total degree exactly $2(n-1)$ (see Equation 7). Therefore, if you have a combination of all these types of cross terms, the resulting polynomial $f_{i,j}$ could not be identically zero.

Case 2: In this case there is some partial inner product $c * s^{(i)}_{l,k} t^{(j)}_{l,k}$ where $c \in \mathbb{Z}_q$ and $l \in [\sigma], k \in [n]$. Kim et al. [KLM+18] showed that no form of this term $s^{(i)}_{l,k} t^{(j)}_{l,k}$ consists of monomials of degree exactly N. As before, these terms will not cancel with the new terms available to the adversary which consist of monomials of degree exactly 2 and $2(n-1)$. Therefore, if you have a combination of partial inner products, the resulting polynomial $f_{i,j}$ could not be identically zero.

Case 3: Lastly, we consider the case where $f_{i,j}$ consists of nonzero terms that have no products between s and t. Kim et al. [KLM+18] showed non-zero terms of the form $s^{(i)}_{l,m}$ or $t^{(j)}_{l,m}$ will not cancel out with each other in $f_{i,j}$. However, we now need to consider the case in which we have squared terms. Consider the canonical polynomial in Claim 4 as described in Equation 4 where there are additionally square terms. That is,

$$p = \sum_{i=1}^{Q} \hat{\alpha}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m,1} \cdot \hat{s}^{(i)}_{\ell,m} \right) + (\hat{\alpha}^{(i)})^2 \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m,3} \cdot {\hat{s}^{(i)}_{\ell,m}}^2 \right) \tag{8}$$

$$+ \hat{\beta}^{(i)} \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m,3} \cdot \hat{t}^{(i)}_{\ell,m} \right) + (\hat{\beta}^{(i)})^2 \left(\sum_{\ell,m=1}^{\sigma,N} c^{(i)}_{\ell,m,4} \cdot \hat{t}^{(i)}_{\ell,m} \right) \tag{9}$$

Recall, that the expansion of $\hat{t}$ terms are monomials of degree exactly $n-1$ as described in Equation 7. Thus, note that terms of the type $\hat{s}$ have degree exactly 1, terms of the type $\hat{s}^2$ have degree exactly 2, terms of the type $\hat{s}$ have degree exactly $n-1$ and terms of the type $\hat{s}^2$ have degree exactly $2(n-1)$. Thus, expanding Equation 9 in terms of $\hat{b}$ yields a polynomial that is not identically zero and whose monomials are linearly independent. So since there is at least one nonzero coefficient the resulting polynomial $f_{i,j}$ could not be identically zero.

Since we have shown that the polynomial $f_{i,j}$ can not be identically in the 3 new cases, then the simulator will correctly output "non-zero" with overwhelming probability. This completes the proof of Claim 5. □

This completes the proof of Theorem 2. □

5 Data

Table 1 compares the timings for the MRProj and MRProjSym proximity search schemes. The MRProj scheme is asymmetric and uses the MNT159 elliptic curve as described in Kim et al. [KLM+18]. The MRProjSym scheme is symmetric and uses the SS512 elliptic curve. Overall, the search algorithm proved to be more efficient in the symmetric scheme, improving the overall speed by approximately a factor of 4. Additionally, the efficiency of the Setup and BuildIndex operations improved as well. The only operation that is not more efficient in the symmetric setting is the Trapdoor algorithm. This is likely due to the way that the SS512 curve and TokGen algorithm are constructed. However, notice that the effect is overall smaller than the improvement in search time.

			Time							
			MRProj				MRProjSym			
n	b	t	Setup	BuildIndex	Trapdoor	Search	Setup	BuildIndex	Trapdoor	Search
128	3	38	75	1.5	.36	234	34	.7	.8	58
192	5	57	47	2.2	.8	495	38	.9	1.8	130
256	7	76	57	2.9	1.4	850	43	1.2	3.2	228
384	10	115	94	4.4	3.1	1870	73	1.7	7.4	514
512	13	153	153	5.7	5.7	3282	106	2.3	13	907
768	19	230	269	8.6	13.4	7210	169	3.4	28	2030
1024	25	307	268	10.8	22.4	12600	225	4.3	52	3580

Table 1: Operations timing (in seconds) of the proximity search algorithm for different vector sizes. n is the vector length, b the number of bases used, and $t = .30$ the distance tolerance.

		Time											
		MNT159						SS512					
		TokGen		Encrypt		Decrypt		TokGen		Encrypt		Decrypt	
n	b	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
20	1	.0012	9.3×10^{-13}	0.0038	1.5×10^{-13}	0.0359	3.0×10^{-9}	0.0031	9.5×10^{-12}	0.0016	6.1×10^{-13}	0.0080	3.3×10^{-11}
30	1	0.0018	2.1×10^{-12}	0.0054	2.3×10^{-11}	0.0434	1.1×10^{-6}	0.0046	7.0×10^{-12}	0.0025	1.4×10^{-11}	0.0108	2.3×10^{-10}
40	1	0.0026	2.0×10^{-13}	0.0075	2.7×10^{-12}	0.0546	9.0×10^{-8}	0.0061	1.8×10^{-11}	0.0032	4.1×10^{-12}	0.0131	1.6×10^{-10}
50	1	0.0030	3.7×10^{-11}	0.0179	2.0×10^{-11}	0.1147	1.1×10^{-7}	0.0078	4.8×10^{-11}	0.0076	1.8×10^{-11}	0.0303	1.9×10^{-10}
60	1	0.0036	3.6×10^{-12}	0.0210	7.9×10^{-12}	0.1310	5.0×10^{-7}	0.0091	1.6×10^{-11}	0.0089	9.2×10^{-12}	0.0355	1.2×10^{-9}
70	1	0.0044	1.8×10^{-11}	0.0252	6.7×10^{-10}	0.1543	1.7×10^{-7}	0.0107	8.1×10^{-11}	0.0104	4.2×10^{-13}	0.0404	2.1×10^{-12}
80	2	0.0051	2.1×10^{-11}	0.0282	1.3×10^{-11}	0.1708	8.6×10^{-9}	0.0128	3.1×10^{-11}	0.0123	9.6×10^{-12}	0.0471	1.9×10^{-10}
90	2	0.0056	2.8×10^{-11}	0.0330	1.1×10^{-11}	0.1984	8.5×10^{-8}	0.0138	1.2×10^{-10}	0.0135	7.3×10^{-11}	0.0518	7.1×10^{-11}
100	2	0.0063	8.9×10^{-12}	0.0369	5.8×10^{-10}	0.2189	3.9×10^{-7}	0.0159	2.4×10^{-10}	0.0156	1.0×10^{-10}	0.0589	6.6×10^{-10}

Table 2: Operations timing (in seconds) of the predicate encryption algorithm for different vector sizes. n is the vector length, and b the number of bases used.

Table 2 compares the timings for the predicate encryption algorithm when it uses the MNT159 or SS512 curves for its pairing group. For each vector length n, each algorithm (TokGen, Encrypt, Decrypt) was run 100 times. The timings for each trial were averaged and the variance calculated. Overall, these results are consistent with the results we see in 1. Again, we notice that Encrypt and Decrypt run faster in the symmetric scheme (SS512), but TokGen is marginally slower. While TokGen slows down by a factor of 3, Encrypt and Decrypt speed up by factors of 3 and 4, respectively.

6 Conclusion

In this paper we discussed the work done by Cachet et al. [ACD+22] on proximity searchable encryption. We reviewed the inner product encryption scheme as described by Kim et al. [KLM+18] and analyzed the efficiency. It was discovered that using a symmetric pairing group greatly improves the efficiency of the decryption algorithm and thus, we were able to improve the efficiency of search by a factor of 4.

6.1 Acknowledgements

I'd like to take a moment to thank my advisor throughout this whole process, Benjamin Fuller. When I first approached him in the spring of 2021, I was overwhelmed and unsure where to start with the honors thesis project. He not only helped me find a topic that aligned with my interests, but he was patient and kind as I slowly worked my way through the material. I sincerely appreciate the amount of time Professor Fuller has dedicated each week of this past year towards helping me succeed with this thesis, and I could not have asked for a better mentor.

References

[ACD+21] Sohaib Ahmad, Chloe Cachet, Luke Demarest, Benjamin Fuller, and Ariel Hamlin. An implementation of proximity searchable encryption (PSE). `https://github.com/chloecachet/pse`, 2021.

[ACD+22] Sohaib Ahmad, Chloe Cachet, Luke Demarest, Benjamin Fuller, and Ariel Hamlin. Proximity searchable encryption for the iris biometric. In *Asia CCS*, 2022. `https://ia.cr/2020/1174`.

[AF19] Sohaib Ahmad and Benjamin Fuller. Thirdeye: Triplet-based iris recognition without normalization. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2019.

[AGM+13] Joseph Akinyele, Christina Garman, Ian Miers, Matthew Pagano, Michael Rushanan, Matthew Green, and Aviel Rubin. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3, 06 2013.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 213–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *Advances in Cryptology – ASIACRYPT 2015*, pages 470–491, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[CAD+20] Chloe Cachet, Sohaib Ahmad, Luke Demarest, Serena Riback, Ariel Hamlin, and Benjamin Fuller. Multi random projection inner product encryption, applications to proximity searchable encryption for the iris biometric. Cryptology ePrint Archive, Report 2020/1174, 2020. `https://ia.cr/2020/1174`.

[CGKO11] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19:895–934, 01 2011.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 668–679, 2015.

[Dau09] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.

[Dau14] John Daugman. 600 million citizens of India are now enrolled with biometric id,". *SPIE newsroom*, 7, 2014.

[DRD+20] Pawel Drozdowski, Christian Rathgeb, Antitza Dantcheva, Naser Damer, and Christoph Busch. Demographic bias in biometrics: A survey on an emerging challenge. *IEEE Transactions on Technology and Society*, 1(2):89–103, 2020.

[Fou] Electronic Frontier Foundation. Mandatory national ids and biometric databases.

[FVY+17] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. SoK: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 172–191. IEEE, 2017.

[GLMP18] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 315–331, 2018.

[HBF10] Karen Hollingsworth, Kevin W Bowyer, and Patrick J Flynn. Similarity of iris texture between identical twins. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 22–29. IEEE, 2010.

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *NDSS*, volume 20, page 12. Citeseer, 2012.

[Jou04] Antoine Joux. A one round protocol for tripartite diffie–hellman. *Journal of cryptology*, 17(4):263–276, 2004.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman Hall/CRC, 2nd edition, 2014.

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J Wu. Function-hiding inner product encryption is practical. In *International Conference on Security and Cryptography for Networks*, pages 544–562. Springer, 2018.

[KM17] Prabhat Kushwaha and Ayan Mahalanobis. A probabilistic baby-step giant-step algorithm. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2017.

[KT14] Yutaka Kawai and Katsuyuki Takashima. Predicate- and attribute-hiding inner product encryption in a public key setting. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography – Pairing 2013*, pages 113–130, Cham, 2014. Springer International Publishing.

[Lew16] Kevin Lewi. FHIPE github. `https://github.com/kevinlewi/fhipe`, 2016.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT 2012*, pages 591–608, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[SSF19] Sailesh Simhadri, James Steel, and Benjamin Fuller. Cryptographic authentication from the iris. In *International Conference on Information Security*, pages 465–485. Springer, 2019.

[Wei40] André Weil. Sur les fonctions algébriquesa corps de constantes fini. *CR Acad. Sci. Paris*, 210(1940):592–594, 1940.

[WLD+17] Guofeng Wang, Chuanyi Liu, Yingfei Dong, Hezhong Pan, Peiyi Han, and Binxing Fang. Query recovery attacks on searchable encryption based on partial knowledge. In *International Conference on Security and Privacy in Communication Systems*, pages 530–549. Springer, 2017.