3I2-OS-20-3

Locality Sensitive Hashingに基づく暗号理論的に安全なk近傍探索法

Secure k-Nearest Neighbor Search based on Locality Sensitive Hashing

原田 弘毅[*1] 青木 良樹[*2] 佐久間 淳[*2*3]
Hiroki Harada Yoshiki Aoki Jun Sakuma

[*1]筑波大学情報学群情報科学類
College of Information Science, Univercity of Tsukuba

[*2]筑波大学システム情報系
Faculty of Engineering, Information and Systems

[*3]科学技術振興機構 さきがけ
Japan Science and Technology Agency

Recently, the privacy preserving information search is getting attention, which searches information without revealing secrets of data. But its cryptographic operation takes long computing time, and for extensive applications, it need to perform without third party. In this paper,we propose 2 secure protocols. one protocol computes Hamming distance, the other get specific data which server has without revealing what data client want to get. And we realize fast k-NN seach between the two parties.

1. はじめに

近年の計算機性能の向上やインターネット上でのサービスの発展に伴い, 企業や個人の秘密データが増大している. これらの秘密データは利用者にとって有益な情報であることが多く, 秘密データに対する類似検索が可能であれば広範なサービスが期待できる. しかし秘密データを類似検索に利用するためには, データベースを持つサーバに対してクエリの発行が必要であり, そのクエリの発行が秘密情報の暴露につながる可能性がある. このような秘密情報を保護しつつ類似検索を行う手法をプライバシ保護類似検索と呼ぶ.

プライバシ保護類似検索の問題点は, 計算時間が大きい点である. プライバシ保護類似検索では暗号演算が用いられる. この暗号演算は計算コストが大きく, プライバシ保護類似検索における検索時間のボトルネックとなる. Shaneck らの手法では, 暗号理論的に安全な k-近傍点探索を実現しているが, 多くの secure multi-party computation[1] を使っているため, 大きな検索時間がかかる [2].

また Wong らの手法では, SCONEDB モデルを用いて k-近傍点探索を実現している [3]. サードパーティを利用したプライバシ保護類似検索では, 通常のクライアント・サーバ構成に加え新たにサードパーティのサーバを維持する必要があり, 高コストである. そのため, プライバシ保護類似検索ではサードパーティを経由することなく, データを保持するクライアントとサーバ間のみで検索が可能であることが望ましい.

本稿では以下の 2 つのプロトコルを提案し, 2 パーティ間での高速な k-近傍点探索の実現を目指す.1 つ目は, Locality Sensitive Hashing (LSH)[4] を利用した, クライアントが持つクエリとサーバが持つ各データ間の高速な秘密距離計算プロトコル. 2 つ目は, クライアントがサーバから得たいデータを, クエリと得たいデータをサーバに公開することなく取得する秘密データ取得プロトコルである.

また, 提案するプロトコルの高速化手法の処理時間削減量を実験によって測定し, 評価を行う.

連絡先: 原田 弘毅, 筑波大学情報学群情報科学類, 〒 305-8573 茨城県つくば市天王台 1-1-1,h.hiroki@mdl.cs.tsukuba.ac.jp

2. 暗号学的ツール

本節では, 以降で提案するプロトコルで用いる暗号学的ツールを導入する.

2.1 Paillier 暗号

Paillier 暗号とは, 加法準同型性を持つ準同型性公開鍵暗号である. 加法準同型性とは, 共通の公開鍵を用いて暗号化した平文同士の加算を, 暗号文同士の演算で求めることができる性質である. また, その性質を持つ公開鍵暗号を準同型性公開鍵暗号と呼ぶ.

Paillier 暗号では次のように暗号文を生成する.

$$\mathrm{Enc}_{\mathrm{pk}}(m) = g^m r^n \pmod{n^2}$$

g, n は公開鍵, m は平文, r は乱数を示す. 以降では, 秘密鍵を sk, 公開鍵を pk と表記する. また, 暗号処理の記法として, $c = \mathrm{Enc}_{\mathrm{pk}}(t; r)$ を暗号化, $t = \mathrm{Dec}_{\mathrm{sk}}(c)$ を復号化の処理とする.

加法準同型性公開鍵暗号は次の性質を満たす.

$$\mathrm{Enc}_{\mathrm{pk}}(t_1; r_1) \cdot \mathrm{Enc}_{\mathrm{pk}}(t_2; r_2) = \mathrm{Enc}_{\mathrm{pk}}(t_1 + t_2; r_1 + r_2),$$
$$\mathrm{Enc}_{\mathrm{pk}}(t_1; r_1)^{t_2} = \mathrm{Enc}_{\mathrm{pk}}(t_1 t_2; r_1 r_2)$$

t_1, t_2 は平文, r_1, r_2 は乱数を表す. r_1, r_2 は暗号の安全性を保持するために, 暗号化ごとに変更する. 以降では簡単のため, 乱数 r_1, r_2 の表記を省略し $\mathrm{Enc}_{\mathrm{pk}}(\cdot)$ と表す.

2.2 Random share

random share とは, 2 パーティ間での通信の際に, 両パーティともデータの内容を知らずに持ち合う手法である.

Client と Server が, $\mathbf{x}^{AB} \in \mathbb{Z}_N^l$ を $\mathbf{x}^{AB}$ の内容を知らずに持ち合いたい. まず, $\mathbf{x}^{AB}$ を, $\mathbf{x}^A = (x_1^A, ..., x_l^A)$ と $\mathbf{x}^B = (x_1^B, ..., x_d^B)$ に分割し, Client と Server は $\mathbf{x}^A$ と $\mathbf{x}^B$ をそれぞれ持つ. このとき, $\mathbf{x}^A$ と $\mathbf{x}^B$ が

$$x_i^{AB} = x_i^A + x_i^B \pmod{N} \qquad x_i^A, x_i^B \in_r \mathbb{Z}_N$$

を満たすならば, Client と Server は $\mathbf{x}^{AB}$ を random share している.($\in_r$ は集合からランダムに要素を選ぶ演算子である)

Client と Server が $\mathbf{x}^{AB}$ を random share しているならば, Client と Server は $\mathbf{x}^{AB}$ の内容を知ることなく $\mathbf{x}^{AB}$ を持ち合うことができる.

2.3 Semi-honest モデル

semi-honest モデルとは, プロトコルを利用するユーザの振る舞いのモデルのひとつである. ユーザの振る舞いが semi-honest モデルであるとき, ユーザはプロトコルの正しい利用方法に順ずるが, 通信の中間で発生したデータを保持する.

semi-honest モデルにおいて想定するプロトコルのセキュリティへの攻撃は, プロトコル利用中あるいは利用後に, ユーザが保持しているデータを利用して何らかの知識を得ようとするというものである. これはプロトコルを利用する多くのユーザが実行可能な攻撃であるため, ユーザの多くはプロトコルへの攻撃を考える場合, semi-honest モデルとして振る舞う. 本稿では, ユーザの振る舞いは semi-honest モデルであることを想定してプロトコルの安全性を証明する.

3. 問題の定義

本稿で扱う問題を定義する. この問題では 2 つのエンティティが存在する. 1 つはデータベースを持つサーバである. サーバは, d 次元の特徴ベクトルを n 点持つデータベース $D = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$ $(\mathbf{z}_i \in \mathbb{Z}_N^d)$ を保持している. もう 1 つはクエリ $\mathbf{q} \in \mathbb{Z}_N^d$ を保持するクライアントである. プライバシ保護類似検索における k 近傍点探索では, クライアントはクエリと受け取るデータをサーバに見せずに, データベース D 中の近傍点をクエリと近い点から k 個取得することが目的である. 本稿では以下の 2 つの Statement を満たすプロトコルを提案することで, この問題を解決する.

3.1 距離の評価

まず, クライアントはクエリデータとサーバが持つ各データとの距離を取得する. ここで, サーバが保持するデータおよびクライアントが保持するクエリは LSH などを用いてバイナリコードにコード化されていることを前提とする.(4.1 参照) このとき, クエリとサーバが持つ各データ間の秘密ハミング距離計算は **Statement 1** を満たすプロトコルを繰り返し実行することで達成される.

Statement 1 クライアントとサーバが $\mathbf{x} = (x_1, ..., x_d)$, $\mathbf{y} = (y_1, ..., y_d)$ $(\mathbf{x}, \mathbf{y} \in \{0,1\}^l)$ を持ち, ハミング距離計算プロトコル実行後, クライアントとサーバは $s_A + s_B = H(\mathbf{x}, \mathbf{y}) \pmod N$ を満たす値 s_A, s_B を持ち, 他には何も学習しない.

ここで, $H(\mathbf{x}, \mathbf{y})$ は 2 つのバイナリデータ間のハミング距離を表す. また, クライアントとサーバが得た s_A, s_B は random share で生成された値である.

3.2 近傍点の取得

続いて, 秘密ハミング距離計算によって得られたクエリと各データの距離を用いて, クライアントはサーバが持つ近傍点 top k のサーバ内でのインデックスを得る. 次に, クライアントはサーバ上のこれら k 個のデータを, 要求するインデックスをサーバに知られることなく取得する. クライアントの秘密データ取得は **Statement 2** を満たすプロトコルを繰り返し実行することで達成される.

Statement 2 クライアントがサーバに要求するインデックス u を持ち, サーバが $D = \{\mathbf{z}_1, .., \mathbf{z}_u, .., \mathbf{z}_n\} (\mathbf{z}_i \in \mathbb{Z}_N^d)$ を持つ. このとき, 秘密データ取得プロトコル実行後には, サーバは何も学習しない. また, クライアントは $\mathbf{z}_u$ を持ち, 他には何も学習しない.

結果, クライアントはクエリと受け取るデータをサーバに見せずに,k 近傍探索を実現する. 以降ではこれらの問題を解決するプロトコルを提案する.

4. 秘密ハミング距離計算プロトコル

本節では, クライアントのクエリとサーバが持つ各データ間のハミング距離を安全に計算するプロトコルを提案する.

4.1 LSH によるバイナリコード化

秘密ハミング距離計算プロトコルでは, 高次元データに対しても高速な計算を行うため, Locality Sensitive Hashing を利用する.

LSH は, データの特徴量を局所鋭敏性を持つハッシュ関数を用いてバイナリコードにコード化する手法である. 局所鋭敏性とは, 2 つのデータをそれぞれバイナリにコード化したとき, データ同士が特徴空間において隣接している場合ハミング距離において近いバイナリコードに, データ同士が離れている場合ハミング距離において遠いバイナリコードにハッシュされる性質である. LSH によってハッシュされたバイナリコードは, 特徴空間での距離関係をバイナリコード間のハミング距離で近似的に表現する.

LSH はデータを低次元バイナリ空間にコード化するため, 高次元のデータに対しても高速で類似検索を行うことができる. しかし LSH は高速な計算時間の代償として, ハミング距離を用いて検索される近傍点に false positive を含む. そのため, LSH の予測精度と検索精度にはトレードオフの関係がある.

4.2 プロトコル設計

2 つのバイナリコードのハミング距離計算の手法を次に示す. 2 つのバイナリコード $\mathbf{x} = (x_1, ..., x_l), \mathbf{y} = (y_1, ..., y_l)$ $(x, y \in \{0,1\}^l)$ について, このハミング距離 $H(\mathbf{x}, \mathbf{y})$ は次のように計算される [5].

$$\begin{aligned} H(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{l} (x_i - y_i)^2 \\ &= \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{y} \end{aligned} \quad (1)$$

バイナリコード間のハミング距離は, 式 (1) のように変形される. 式 (1) の第一項と第二項は, ローカルのデータのみで計算を行うので各パーティで計算可能である. 一方, 第三項の $-2\mathbf{x}^T\mathbf{y}$ は両パーティのデータを用いて評価する必要がある. そこで, 第三項を準同型性暗号を用いて秘密計算することで, 秘密ハミング距離計算を達成する. 提案する秘密ハミング距離計算プロトコルを Algorithm 1 に示す.

4.3 暗号理論的なプロトコルの安全性証明

このプロトコルの安全性について, 次の補助定理が導かれる.

Lemma1 Client と Server が semi-honest モデルで振る舞うとき, 秘密ハミング距離計算プロトコルは **Statement 1** の意味において安全である.

提案するプロトコルについて, **Lemma1** を view を用いた証明法によって証明する.

Proof. Client の view について, Client は Server から最終的な出力以外のメッセージを受け取っていないため, Server は

Algorithm 1 秘密ハミング距離計算プロトコル

-Client's input: $\mathbf{x} \in \{0,1\}^l$
-Server's input: $\mathbf{y} \in \{0,1\}^l$
-Client's output: $s_A \in \mathbb{Z}_N$
-Server's output: $s_B \in \mathbb{Z}_N$
(ただし $s_A + s_B = H(\mathbf{x},\mathbf{y}) \pmod N$)

1. Client は鍵 (pk, sk) を生成する.
2. Client は $i = 1,...,l$ に関して $c_i = \mathrm{Enc}_{\mathrm{pk}}(x_i)$ を計算し, $\mathbf{c} = (c_1,...,c_l)$ と pk を Server に送る.
3. Server は乱数 $r_B \in_r \mathbb{Z}_N$ を生成し, $c' = \mathrm{Enc}_{\mathrm{pk}}(-r_B) \cdot \prod_{i=1}^{l} c_i^{-2y_i}$ を計算し, c' を Client に送る.
4. Client は $s_A = \mathrm{Dec}_{\mathrm{sk}}(c') + \sum_{i=1}^{l} x_i^2$, Server は $s_B = r_B + \sum_{i=1}^{l} y_i^2$ をそれぞれ出力する.

安全である. Server の view について, Server は Client から $\mathbf{c} = (c_1,...,c_l)$ を受け取っている. しかし Server は復号化のための秘密鍵を持たず, また c_i は一様分布から生成された乱数を用いて強秘匿性を有するように暗号化されている. そのため, Client と同じ挙動をする Client Simulator が一様分布から乱数を生成し, Client と同時に Server に送信した場合, Server は Client のメッセージと Client Simulator のメッセージを判別することができない. よって, Client は安全である.

以上より, Client と Server の view でそれぞれが安全であることがいえるため, **Lemma1** が証明された. □

4.4 プロトコルの高速化

Algorithm 1 のハミング距離計算プロトコルを高速化する手法を導入したプロトコルを Algorithm 2 に示す.

暗号化回数の削減

Algorithm 1 の step.2 に関して, 暗号化の処理の回数を削減する高速化手法を提案する. プロトコル上で l 次元のバイナリコード $\mathbf{x} \in \{0,1\}^l$ を扱うとき, 各 x_i を暗号化する処理はプロトコル全体の計算コストの多くを占める. そこで事前に l 通りの暗号文を生成しておくことで, この計算回数を削減しプロトコルを高速化する.

この高速化手法では, クエリの暗号化計算コストを $O(n)$ から $O(1)$ に減少させるため, 大幅な高速化が期待できる.

計算処理の並列処理化

Algorithm 1 のプロトコルでは, Server の計算資源のみを用いて暗号演算を行っている. そこで, Client と Server が計算コストを負担し合う, プロトコル上での演算処理を並列化する高速化手法を提案する. この高速化では, l を偶数とする.

両パーティが同等の計算資源を持つとき, この高速化手法は計算時間を半減させることができる.

5. 秘密データ取得プロトコル

5.1 プロトコル設計

クライアントがクエリ及び受け取るデータをサーバに公開することなく, サーバからあるインデックスを持つデータを取得するプロトコルを提案する. インデックス u を持つデータを要求するクエリ $\mathbf{q}' \in \{0,1\}^n$ を, クライアントは次のように

Algorithm 2 高速化手法を用いたプロトコル

-Client's input: $\mathbf{x} \in \{0,1\}^l$
-Server's input: $\mathbf{y} \in \{0,1\}^l$
-Client's output: $s_{AA}, s_{AB} \in \mathbb{Z}_N$
-Server's output: $s_{BB}, s_{BA} \in \mathbb{Z}_N$
(ただし $s_{AA} + s_{AB} + s_{BB} + s_{BA} = H(\mathbf{x},\mathbf{y}) \pmod N$)

0. Client は $\mathbf{x}$ を $\mathbf{x}^{\mathrm{pre}} = \{x_1,....,x_t\}(t = l/2)$ と $\mathbf{x}^{\mathrm{suf}} = \{x_{t+1},....,x_l\}$ に分割し, Server は $\mathbf{y}$ を $\mathbf{y}^{\mathrm{pre}} = \{y_1,....,y_t\}$ と $\mathbf{y}^{\mathrm{suf}} = \{y_{t+1},....,y_l\}$ に分割する.

1a. Client の鍵 $(\mathrm{pk_A}, \mathrm{sk_A})$ を生成し, $v^A = \mathrm{Enc}_{\mathrm{pk_A}}(0)$ を t 通り持つ集合 $U_{A0} = \{v_1^A,...,v_t^A\}$ と $w^A = \mathrm{Enc}_{\mathrm{pk_A}}(1)$ を格納しておく.

1b. Server の鍵 $(\mathrm{pk_B}, \mathrm{sk_B})$ を生成し, $v^B = \mathrm{Enc}_{\mathrm{pk_B}}(0)$ を t 通り持つ集合 $U_{B0} = \{v_1^B,...,v_t^B\}$ と $w^B = \mathrm{Enc}_{\mathrm{pk_B}}(1)$ を格納しておく.

2a. Client は $i = 1,...,t$ に関して,
$$c_i = \begin{cases} v_r^A & (x_i = 0) \\ w^A \cdot v_r^A & (x_i = 1) \end{cases}$$
を計算し ($v_r^A \in_r U_{A0}$), $\mathbf{c}_A = (c_1,...,c_t)$ と $\mathrm{pk_A}$ を Server に送る.

2b. Server は $j = t+1,...,l$ に関して,
$$c_j = \begin{cases} v_r^B & (y_j = 0) \\ w^B \cdot v_r^B & (y_j = 1) \end{cases}$$
を計算し ($v_r^B \in_r U_{B0}$), $\mathbf{c}_B = (c_{t+1},...,c_l)$ と $\mathrm{pk_B}$ を Client に送る.

3a. Bob は乱数 $r_B \in_r \mathbb{Z}_N$ を生成し、$c'_B = \mathrm{Enc}_{\mathrm{pk_A}}(-r_B) \cdot \prod_{i=1}^{t} c_i^{-2y_i}$ を計算し、c'_A を Alice に送る.

3b. Alice は乱数 $r_A \in_r \mathbb{Z}_N$ を生成し、$c'_A = \mathrm{Enc}_{\mathrm{pk_B}}(-r_A) \cdot \prod_{j=t+1}^{l} c_j^{-2x_j}$ を計算し、c'_B を Alice に送る.

4a. Client は $s_{AA} = \mathrm{Dec}_{\mathrm{sk_A}}(c'_B) + \sum_{i=1}^{t} (x_i^{\mathrm{pre}})^2$, Server は $s_{AB} = r_B + \sum_{i=1}^{t} (y_i^{\mathrm{pre}})^2$ をそれぞれ出力する.

4b. Server は $s_{BB} = \mathrm{Dec}_{\mathrm{sk_B}}(c'_A) + \sum_{i=t+1}^{l} (y_i^{\mathrm{suf}})^2$, Client は $s_{BA} = r_A + \sum_{i=t+1}^{l} (x_i^{\mathrm{suf}})^2$ をそれぞれ出力する.

生成する.

$$q'_i = \begin{cases} 1 & (i = u) \\ 0 & (i \neq u) \end{cases}$$

クライアントは $\mathbf{q}'$ の各要素を暗号化したベクトルを $\mathbf{c}$ とし, $\mathbf{c}$ をサーバに送信する. ここで, サーバが持つデータセットを $D \in \mathbb{Z}_N^{d \times n}$ とする. サーバは $j = 1,...,d$ に関して, 暗号の準同型性を利用して行列演算

$$c'_j = \prod_{i=1}^{n} c_i^{D_{ji}}$$

を計算し, クライアントにデータを送信する. 秘密データ取得プロトコルを Algorithm 3 に示す.

クライアントは Algorithm 3 のプロトコルを用いて任意のインデックスを持つデータをサーバから取得することができ

Algorithm 3 秘密データ取得プロトコル

-Client's input: $\mathbf{q}' \in \{0,1\}^n$
-Server's input:$D = \{\mathbf{z}_i | i = 1, .., u, .., n\}(\mathbf{z}_i \in \mathbb{Z}_N^d)$
-Client's output: $\mathbf{z}_u \in \mathbb{Z}_N^d$

1. Client の鍵 (pk, sk) を生成する.
2. Client は $i = 1, ..., n$ に関して $c_i = \mathrm{Enc}_{\mathrm{pk}}(q'_i)$ を計算し, $\mathbf{c} = (c_1, ..., c_n)$ を Server に送る.
3. Server は $j = 1, ..., d$ に対して, $c'_j = \prod_{i=1}^{n} c_i^{D_{ji}}$ を計算し, $\mathbf{c}'$ を Client に送る.
4. Client は $j = 1, ..., d$ に関して $z_{ju} = \mathrm{Dec}_{\mathrm{sk}}(c'_j)$ を計算し, $\mathbf{z}_u$ を出力する.

る. そこで, top k の各インデックスに対してクエリを生成し, 秘密データ取得プロトコルを用いることで, k-近傍探索を実現することができる.

5.2 暗号理論的なプロトコルの安全性証明

このプロトコルの安全性について, 次の補助定理が導かれる. 秘密データ取得プロトコルについて, **Lemma2** を view を用いた証明法によって証明する.

Lemma2 Client と Server が semi-honest モデルで振る舞うとき, 秘密データ取得プロトコルは **Statement 2** の意味において安全である.

Proof. Client の view について, Client は Server から最終的な出力以外のメッセージを受け取っていないため, Server は安全である. Server の view について, Server は Client から $\mathbf{c} = (c_1, ..., c_n)$ を受け取っている. しかし Server は復号化のための秘密鍵を持たず, また c_i は一様分布から生成された乱数を用いて強秘匿性を有するように暗号化されている. そのため, Client と同じ挙動をする Client Simulator が一様分布から乱数を生成し, Client と同時に Server に送信した場合, Server は Client のメッセージと Client Simulator のメッセージを判別することができない. よって, Client は安全である.

以上より, Client と Server の view でそれぞれが安全であることがいえるため, **Lemma 2** が証明された. □

6. 秘密ハミング距離計算の高速化手法による時間削減量の実験的評価

高速化手法を適用した秘密ハミング距離計算の計算時間を, 既存手法との比較によって評価する.

計算機は, Intel Core i7 3.07GHz(CPU), 6GB(RAM) の Windows 機を用いた. 実験に用いるデータセットは The MIT-CBCL face recognition database [6] の各データを 2500 次元に圧縮したデータセットと, 32256 次元のデータを持つ The Extended Yale Face Database B [7] データセットである. なお, データセットの各要素は LSH で用いるために中央化されており, 各データセットから 400 点のデータを選び, これを検索対象とした.

また, LSH を用いた類似検索では, バイナリコード長を変化させることで false positive の発生率を調整できる. 今回はバイナリコード長 7500 を用いる. この長さでは, 今回用いるデータセットにおいて false positive の発生率は 5% 以下に抑えられる. なお, 分散計算を用いた高速化は, 今回は予測値を算出し実験結果とした.

表 1: 秘密ハミング距離計算プロトコルの高速化効果の検証 (20 回試行平均秒数)

手法	高速化手法	MIT	Yale
提案手法	Algorithm 1	26.9	27.1
	Algorithm 1 + pre-encryption	12.1	12.7
	Algorithm 2	6.05	6.35
秘匿内積計算 [8]		45.2	547

既存の秘密距離計算手法として, 秘匿内積計算プロトコルを用いる. この手法は, コサイン類似度を暗号理論的に安全に計算する手法であり, 計算時間はデータの次元数に依存する.

表 1 より, 高速化手法を用いたプロトコルは秘匿内積計算プロトコルと比較して約 1/5 の計算時間を達成し, 既存手法の計算時間を 1/80 に短縮することが確認された.

7. おわりに

本稿では, プライバシ保護類似検索において 2 パーティ間での高速な k-近傍点探索を実現するプロトコルを提案した. 今後はより高速なデータ取得プロトコルを設計する.

謝辞

本研究は科学研究費 (24680015) および部分的に JST さきがけ「知の創成と情報社会」の助成を受けたものである.

参考文献

[1] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[2] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. In *Proceedings of the Sixth IEEE International Conference on Data Mining-Workshops*, pp. 541–545. IEEE Computer Society, 2006.

[3] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis. Secure $k$nn computation on encrypted databases. In *Proceedings of the 35th SIGMOD International Conference on Management of data*, pp. 139–152. ACM, 2009.

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pp. 459–468. Ieee, 2006.

[5] Jun Sakuma and Rebecca N. Wright. Privacy-preserving evaluation of generalization error and its application to model and attribute selection. In *ACML*, pp. 338–353, 2009.

[6] Jennifer Huang, Bernd Heisele, and Volker Blanz. Component-based face recognition with 3d morphable models. In *AVBPA*, pp. 27–34, 2003.

[7] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 23, No. 6, pp. 643–660, 2001.

[8] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. *Information Security and Cryptology–ICISC 2004*, pp. 23–25, 2005.