Privacy Preserving Data Classification using Inner-product Functional Encryption

Damien Ligier[1,2,3], Sergiu Carpov[1] Caroline Fontaine[2,3] and Renaud Sirdey[1]
[1]*CEA LIST, Point Courrier 172, 91191 Gif-sur-Yvette Cedex, France*
[2]*CNRS/Lab-STICC and Telecom Bretagne, Brest, France*
[3]*UBL, Rennes, France*
{*damien.ligier, sergiu.carpov, renaud.sirdey*}*@cea.fr, caroline.fontaine@telecom-bretagne.eu*

Keywords: Functional Encryption, Inner-Product Encryption, Classification, Linear Classification.

Abstract: In the context of data outsourcing more and more concerns raise about the privacy of user's data. Simultaneously, cryptographers are designing schemes enabling computation on ciphertexts (homomorphic encryption, functional encryption, etc.). Their use in real world applications is difficult. In this work we focus on functional encryption schemes enabling computation of inner-product on encrypted vectors and their use in real world scenarios. We propose a protocol combining such type of functional encryption schemes with machine learning algorithms. Indeed, we think that being able to perform classification over encrypted data is useful in many scenarios, in particular when the owners of the data are not ready to share it. After explaining our protocol, we detail the implemented handwritten digit recognition use case, and then, we study its security.

1 INTRODUCTION

With the generalization of data outsourcing, more and more concerns raise about the privacy and the security of outsourced data. In this context, machine learning methods have to be conceived and deployed, but with users privacy concerns addressed.

In a privacy preserving data classification process, one has to be able to extract knowledge (e.g. in the case of a classifier, deduction of the class label of an individual without compromising his private data) by assuring the protection of the sensitive data and, if possible, by hiding data access patterns from which useful properties could be inferred.

In this work we propose a privacy preserving classification algorithm based on functional encryption, in particular the inner-product encryption. A multi-class prediction algorithm with encrypted input data is described. The privacy of input data is kept due to the inner-product encryption scheme. Roughly speaking the data item on which a prediction must be made is encrypted. From the encrypted data, integer inner-products are extracted and are used afterwards to produce the class of the input data item. The performance of the classification algorithm is evaluated on the MNIST database (LeCun et al.,).

The paper is structured as follows. In Sec. 2, we recall machine learning and cryptographic background that we need in this paper. Sec. 3 presents the protocol for performing classsification over encrypted data, that we introduce. Sec. 4 gives the experimental results we obtained with our protocol. Finally, Sec. 5 concludes this paper.

2 PRELIMINARIES AND RELATED WORK

In the literature one can find several works on privacy-preserving classification. In particular, we refer to (Yang et al., 2005) for a privacy-preserving method that allows to compute frequencies of values, (Mangasarian et al., 2008) for a support vector machine classifier for a private data matrix. In this work we present a new approach which uses functional encryption to perform classification on encrypted data.

2.1 Functional Encryption

Traditional public-key cryptography has been generalized in many ways, among which recently raised the concept of functional encryption (Sahai and Waters, 2005; Boneh et al., 2011). In this paradigm the *authority* is holding a *master secret-key MSK* that allows to derive a secret key sk_f associated with a function

Ligier, D., Carpov, S., Fontaine, C. and Sirdey, R.
Privacy Preserving Data Classification using Inner-product Functional Encryption.
DOI: 10.5220/0006206704230430
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 423-430
ISBN: 978-989-758-209-7

f. It is possible to derive more secret keys associated with different functions. Using the *public key PUB* one can encrypt a message x into a ciphertext c. The user who has sk_f "decrypts" c and does not get x but only the information $f(x)$. Hence, this decryption is not a real decryption in the common sense, but rather an evaluation. Nevertheless, it is called "decryption" in the literature, so we keep this terminology in this paper.

The authority of a functional encryption scheme delivers several secret keys (associated to different functions) to the users. Hence, it requires that if a user owns different secret keys $\{sk_{f_i}\}_i$ and an encryption of x, he cannot learn about x more than $\{f_i(x)\}_i$. This property is called *collusion resistance*, and there are two ways to address it: one relies on *indistinguishability-based security* and the other one on *simulation-based security*.

The holy grail of this domain is to design a scheme that enables to derive a secret key sk_f for any polynomial time computable function f. Goldwasser *et al.* proposed a construction based on fully homomorphic encryption (Goldwasser et al., 2013), Garg *et al.* proposed another construction using an indistinguishability obfuscator (Garg et al., 2013). At present, however, these constructions remain mostly of theorical interest.

For some use cases one has to hide information inside the function f associated with the decryption keys sk_f. This functionality is not covered by public-key functional encryption. A solution to this problem is to use *functional encryption in a private-key setting* (Brakerski and Segev, 2015; Bishop et al., 2015) (*private-key functional encryption*). There is no more master public key and the encryption algorithm takes as input the master secret key; consequently, an attacker is not able to encrypt whatever he wants. We do not focus on the private-key setting in this paper.

2.1.1 Definition

Boneh *et al.*. (Boneh et al., 2011) gave the following standard definitions for functional encryption. In this definition, the previous function f is represented with the function $F(K,\cdot)$.

Definition 1. *A* functionality F *defined with* (K,X) *is a function* $F : K \times X \to \Sigma \cup \{\perp\}$. *The set* K *is the key space, the set* X *is the plaintext space, and the set* Σ *is the output space and does not contain the special symbol* $\perp$.

Definition 2. *A* functional encryption *scheme for a functionality* F *is a tuple* $\mathcal{FE} = (\texttt{Setup},\texttt{Keygen},\texttt{Encrypt},\texttt{Decrypt})$ *of four algorithms with the following properties.*

- *The* `Setup` *algorithm takes as input the security parameter* 1^λ *and outputs a pair of a public key and a master secret key* (PUB,MSK).
- *The* `KeyGen` *algorithm takes as inputs the master secret key* MSK *and* $k \in K$ *which is a key of the functionality* F. *It outputs a secret key* sk *for* k.
- *The* `Encrypt` *algorithm takes as inputs the public key* PUB *and a plaintext* $x \in X$. *This randomized algorithm outputs a ciphertext* c_x *for* x.
- *The* `Decrypt` *algorithm takes as inputs the public key* PUB, *a secret key and a ciphertext. It outputs* $y \in \Sigma \cup \{\perp\}$.

It is required that for all $(PUB,MSK) \leftarrow \texttt{Setup}(1^\lambda)$, *all keys* $k \in K$ *and all plaintexts* $x \in X$, *if* $sk \leftarrow \texttt{KeyGer}(MSK,k)$ *and* $c \leftarrow \texttt{Encrypt}(PUB,x)$ *we have* $F(K,X) = \texttt{Decrypt}(PUB,sk,c)$ *except with a negligible probability.*

2.1.2 Inner-Product Functional Encryption (IPFE)

Functional encryption schemes that enable the evaluation of inner products are called *functional encryption for the inner-product functionality*, *inner-product functional encryption*, or *inner-product encryption*. In those schemes the plaintext space X and the functionality key space K are vector spaces $\mathbb{K}^n$, and F is the inner-product function. We now consider a plaintext $W \in \mathbb{K}^n$, and a secret key sk_V which is associated with a vector $V \in \mathbb{K}^n$. If c_W is an encryption of W, when one decrypts c_W with sk_V he only gets $\langle v,w \rangle$, thus the inner product of v and w. The owner of sk_V has to know V in order to decrypt with it, i.e. V cannot be hidden to the decryptor.

Recently, Abdalla *et al.* (Abdalla et al., 2015) proposed constructions for the inner product encryption schemes satisfying standard security definitions, under well-understood assumptions like the *Decisional Diffie-Hellman* (DDH) and *Learning With Errors*. However they only proved their schemes to be secure against *selective adversaries*. Agrawal *et al.* (Agrawal et al., 2015) upgraded those schemes to provide them a *full security* (security against *adaptive attacks*). In this work we focus on these inner product schemes, thus on the fully secure functional encryption for the inner product functionality under the DDH assumption (Agrawal et al., 2015).

We now recall the functional encryption for inner-product scheme of the Agrawal *et al.* (Agrawal et al., 2015) that provides full security under the DDH assumption. The notation $\alpha \xleftarrow{R} K$ means that α is randomly choosen from the set K.

Theorem 1. *(Boneh, 1998) In a cyclic group* $\mathbb{G}$ *of prime order q, the* Decisional Diffie-Hellman (DDH) *problem is to distinguish the distribution* $D_0 = \{(g, g^a, g^b, g^{ab}) | g \xleftarrow{R} \mathbb{G},\ a, b \xleftarrow{R} \mathbb{Z}_q\}$ *and* $D_1 = \{(g, g^a, g^b, g^c) | g \xleftarrow{R} \mathbb{G},\ a, b, c \xleftarrow{R} \mathbb{Z}_q\}$.

We recall the four algorithms of the scheme.

Algorithm 1: `Setup`(1^λ,1^m).

1: choose a cyclic group $\mathbb{G}$ of prime order $q > 2^\lambda$ with generators $g, h \in \mathbb{G}$
2: **for all** $1 \leq i \leq m$ **do**
3: $\quad s_i, t_i \xleftarrow{R} \mathbb{Z}_q$
4: $\quad h_i \leftarrow g^{s_i} \cdot h^{t_i}$
5: $PUB \leftarrow (\mathbb{G}, g, h, \{h_i\}_{1 \leq i \leq m})$
6: $MSK \leftarrow (\{s_i\}_{1 \leq i \leq m}, \{t_i\}_{1 \leq i \leq m})$
7: **return** (PUB, MSK)

Let $\boldsymbol{v} = (v_1, \cdots, v_m) \in \mathbb{Z}_q^m$ be the vector we want to associate a key.

Algorithm 2: `Keygen`(MSK,$\boldsymbol{v}$).

1: $s_{\boldsymbol{v}} \leftarrow \sum_{i=1}^m s_i \cdot v_i$
2: $t_{\boldsymbol{v}} \leftarrow \sum_{i=1}^m t_i \cdot v_i$
3: **return** $sk \leftarrow (s_{\boldsymbol{v}}, t_{\boldsymbol{v}})$

Let $\boldsymbol{w} = (w_1, \cdots, w_m) \in \mathbb{Z}_q^m$ be a plaintext we want to encrypt.

Algorithm 3: `Encrypt`(PUB,$\boldsymbol{w}$).

1: $r \xleftarrow{R} \mathbb{Z}_q$
2: $C \leftarrow g^r$, $D \leftarrow h^r$
3: **for all** $1 \leq i \leq m$ **do**
4: $\quad E_i = g^{w_i} \cdot h_i^r$
5: **return** $c \leftarrow (C, D, \{E_i\}_{1 \leq i \leq m})$

Decryption algorithm uses a discrete logarithm computation in a large size group (which in general is hard to compute). Coefficients of the plaintext vector $\boldsymbol{w}$ and the key vector $\boldsymbol{v}$ belong to $\{-\beta, ..., 0, ..., \beta\}$ where β is a small integer, so the possible interval of $\langle \boldsymbol{v}^i, \boldsymbol{w} \rangle$ is small as well. When the output interval of the discrete logarithm is small and known we can use Shank's baby step giant step algorithm (Shanks, 1971) to compute it efficiently or simply use a lookup table.

Algorithm 4: `Decrypt`(PUB,sk,c).

1: $E \leftarrow \prod_{i=1}^m E_i^{v_i} / (C^{s_v} \cdot D^{t_v})$
2: $r \leftarrow log_g(E)$
3: **return** r

2.2 Classification

Machine learning (ML) is a sub-field of computer science. It studies and builds algorithms for learning, predicting, classifying, and more generally extracting knowledge from data. In *supervised ML*, the algorithms are trained with example input data and desired output results, before being used as classifiers. One can distinguish two phases in the application of supervised ML algorithms: *learning* and *prediction* (classification). During the first phase, ML algorithms analyze example input data and learns how to make proper predictions on such kind of data. During the prediction phase, ML algorithms predict how new data has to be classified, as a function of the first learning step.

Depending on the type of prediction results, ML techniques can be of two different kinds: (i) classification – when the result is a discrete value (e.g. predict class membership) or (ii) regression – when the result is a continuous value (e.g. predict temperature). In what follows we focus on the first kind, and describe two classification algorithms used in this work.

2.2.1 Linear Classifier

A linear classification algorithm makes a decision on the membership of an input data object, based on a linear combination of its features (characteristics). For example, in an image classification algorithm the input object is an image and the features can be image pixels. In a binary classification, the decision is made as a function of a threshold overrun by the dot product between object features and linear classifier coefficients. For multi-class classification, one can distinguish two possibilities:

- *One-vs.-rest*, in which a binary classifier is built for each class in order to distinguish between this class and all the others. The decision is made as a function of the resulting dot product amplitude.
- *One-vs.-all*, in which a binary classifier is built for any pair of classes. The decision is made as a function of the number of positive votes received by each class.

More details about linear classification can be found in (McCullagh and Nelder, 1989). In this work we focus on one-vs.-rest multi-class classification. Considering $n \in \mathbb{N}$ classes, $n \geq 2$, we want to classify vectors (objects features) in a specific subset of $\mathbb{Z}^m$, $m \in \mathbb{N}$. Let $\boldsymbol{w}^T = (w_1, w_2, \cdots, w_m) \in \mathbb{Z}^m$ be one of the objects we want to classify. The set $\{\boldsymbol{v}^i\}_{1 \leq i \leq n}$ contains n vectors of $\mathbb{Z}^m$, each of them being associated with one class. The vector $\boldsymbol{v}^i = (v_1^i, v_2^i, \cdots, v_m^i) \in \mathbb{Z}^m$

represents binary classifier coefficients used to distinguish between class i and the other classes. For all $1 \leq i \leq n$ we compute $ip^i = \langle \boldsymbol{v}^i, \boldsymbol{w} \rangle$. The class of input object $\boldsymbol{w}^T$ is given by $\arg\max_{1 \leq i \leq n} r^i$.

2.2.2 Extremely Randomized Trees Classifier

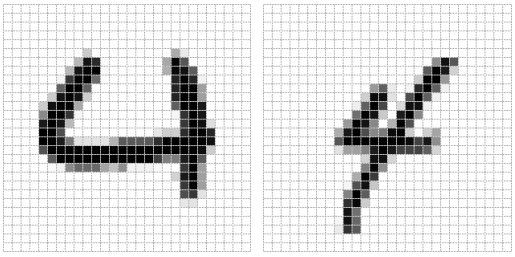
In ensemble learning methods, the predictions of several (usually small) base classifiers are combined in order to make an aggregated classifier which is more powerful and more robust than separate ones (Dietterich, 2000). One of the possibilities to build an ensemble method is to average the decisions of many base classifiers. The combined classifier is stronger than any of the base classifiers.

A decision tree classifier (Safavian and Landgrebe, 1991) represents a tree-like structure where an internal node is a test on a single data feature, node output edges are the outcomes of this test and the tree leafs are decision classes. A decision tree classifier prediction is built by following a tree path from the root node to a leaf node. At each step a decision is made as a function of node condition.

Extremely randomized trees (ERT) classifier is an ensemble learning method in which base classifiers are decision trees. Roughly speaking, an ERT classifier builds many decision trees on different sub-sets of input data features. Prediction is performed by averaging the classes resulting from each decision tree. For more details, please refer to (Geurts et al., 2006).

2.2.3 MNIST Dataset

The performance of the privacy preserving classification methods proposed in this work, is tested using the MNIST dataset. The MNIST database is a collection of handwritten digit images (LeCun et al.,). Fig. 1 presents sample images from it. Dataset images have size $784 = 28 \times 28$ and each pixel has 256 levels of grey. The handwritten digits are normalized in size and are centered. There are 10 output classes in the dataset (digits from 0 to 9). The MNIST dataset has been extensively used by the ML community for classifier validation. For a review of ML methods applied to MNIST, please refer to (Bottou et al., 1994; LeCun et al., 1998).



(a) (b)

Figure 1: Sample digit images from the MNIST database: (a) is the 60th image and (b) is the 61st.

3 CONTRIBUTION

In this work we propose a privacy preserving data classification method. Input data is encrypted using an inner product encryption scheme. In the context of ML algorithms, the inner product encryption can be seen as a linear binary classifier. In order to perform a multi-class linear classification, we need to compute several inner products on the same input data. Usually, linear classifiers provide worse results when compared to other more elaborate classification methods. Nevertheless, only a linear classifier is able to provide a prediction for data encrypted using the inner product encryption (if we don't send more information as encryption of some pieces of precalculus for example).

3.1 Privacy Preserving Classification Protocol

In our protocol, there is an entity called *server* that has performed a training step of a linear classifier. The server has a set of linear classification coefficients $\{\boldsymbol{v}^i\}$ and he wants to keep them secret, but he wants to classify data with it, and not delegate its computation. There are many *users* which have information that they want to keep secret but at the same time, they also want to release classification results to the server (for example in order to obtain a service). We introduce a third party that both the server and the users can trust. We call it *authority* and it is not meant to perform computation. Its goal is in a first time to check that the server's $\{\boldsymbol{v}^i\}$ are not dishonest (the server is not trying to cheat) and in a second time, to generate an instance of an inner-product encryption scheme. We now describe in detail this protocol, illustrated by Fig. 2. The initialization phase has two steps:

First, the authority generates the public key and the master secret key with the `Setup` algorithm of the IPFE, and sends the public key to the users.

The following steps are repeated each time a new server wants to join the system:

(*A*) The server uses the training algorithm on his training set.

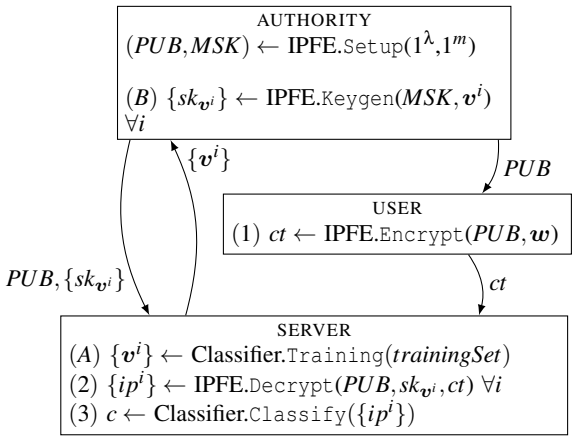


Figure 2: Privacy preserving classification protocol.

(*B*) The authority receives the $\{\boldsymbol{v}^i\}_{1\leq i\leq m}$ from the server and generates the $\{sk_{\boldsymbol{v}^i}\}_{1\leq i\leq m}$ using the `Keygen` algorithm of the IPFE, and afterwards sends them to the server.

The following steps are repeated each time a user wants to send its data to a server:

(1) A user encrypts a private data vector w with the `Encrypt` algorithm of the IPFE, and sends it to the server.

(2) The server decrypts it with all of his secret keys $\{sk_{\boldsymbol{v}^i}\}_{1\leq i\leq m}$ using the `Decrypt` algorithm of the IPFE, and gets $\{ip^i\}_{1\leq i\leq m}$ such that $ip^i = \langle \boldsymbol{v}^i, \boldsymbol{w}\rangle, \forall 1 \leq i \leq m$.

(3) The server uses a classification algorithm in order to predict the class of w using the inner products $\{ip^i\}_{1\leq i\leq m}$.

To illustrate the advantage of our construction, we now describe a realistic scenario. We assume that there is a pharmaceutical company which has constructed a classifier that takes as input medical and private pieces of information. The company does not want to divulge its secret classifier. However, it wants to conduct a study on real human beings (for example the patients of a hospital). The law about medical data can be very restrictive depending on the country. For example in France, a hospital cannot easily share the data of its patients. Our construction provides a possible solution. We simply need a third party that can be trusted by the company and the hospital. A governmental agency should be able to do it. So the agency generates the public key and the secret keys associated with the company classifier. The hospital encrypts the data of its patients and sends it to the company. The company decrypts them with its secret keys. After decryption, the company only gets the result of a computation which involves the patients data. It is important to notice that the critical data remained encrypted during the whole process. Nevertheless, the company can use the computation result to perform the classification. Proceeding this way, it can perform its study on real medical data without endangering the patients data privacy. Moreover, our solution can involve several hospitals and companies if needed (the agency has to generate new secret keys for each new classifier).

3.2 Combined Classifier

The simplest way to classify is to use a linear classifier. The only thing to perform is the $argMax(\{\langle \boldsymbol{v}^i, \boldsymbol{w}\rangle\}_{1\leq i\leq m})$ which give the class of the vector w.

In order to increase the results of linear classification, we propose a combined classification method, in which linear classification is applied to encrypted data, then followed by a more complex classification algorithm (for example an ensemble method in our case but not limited to). For each piece of input encrypted data, several inner products are computed. These products are then used as input features for a second, more elaborate, classifier. In this way, we are able to perform classification of encrypted data with increased performance in terms of an evaluation metric (e.g. error rate).

A linear one-vs.-rest classifier is applied on the dataset. On the obtained output values of this classifier (i.e. dot products), we train an ERT classifier, i.e. a pipeline of classifiers is used. The ERT classifier succeeds in extracting information from the n (number of classes) inner products values. In order to increase the success rate of the combined classifier, we can further split each input class into several subclasses. This corresponds to assigning new "artificial" labels to input data set (simply relabelling the target values). The previously described combined classifier is applied to the input data set with relabelled outcomes except that for the second step "real" labels are used instead of "artificial" ones.

3.3 Classification Security

In this section, we look beyond the cryptographic security of the IPFE scheme. Indeed, the result of the computation provided by a secure IPFE scheme may leak some information about the plaintext. If this information leakage can be used to recover enough information about the plaintext to compromise it, the privacy preserving classification protocol may not be considered as secure even if the underlying IPFE scheme is proved to be secure.

Let R be a ring. We consider a plaintext vector $\boldsymbol{w}^T = (w_1, \cdots, w_m) \in R^m$ and $c_{\boldsymbol{w}}$ one of its encryptions. Let $\{sk_{\boldsymbol{v}^i}\}_{1 \leq i \leq n}$ be a set of n secret keys. For all $1 \leq i \leq n$, $sk_{\boldsymbol{v}^i}$ denotes the secret key associated with the vector $\boldsymbol{v}^i = (v_1^i, \cdots, v_m^i) \in R^m$. So the question is: if we have $c_{\boldsymbol{w}}$ and $\{sk_{\boldsymbol{v}^i}\}_{1 \leq i \leq n}$, what do we exactly know about $\boldsymbol{w}$? Using the decryption algorithm we get $\{ip^i\}_{1 \leq i \leq n}$. So we have the following system with m unknowns: $w'_1, \cdots, w'_m$.

$$\begin{cases} ip^1 &= \sum_{j=1}^m w'_j \cdot v_j^1 \\ ip^2 &= \sum_{j=1}^m w'_j \cdot v_j^2 \\ \vdots & \vdots \quad\quad \vdots \\ ip^n &= \sum_{j=1}^m w'_j \cdot v_j^n \end{cases} \tag{1}$$

Solving this system is equivalent to finding all the vectors $\boldsymbol{w'} \in R^m$ that satisfy the equation:

$$\boldsymbol{ip} = A \cdot \boldsymbol{w'} \tag{2}$$

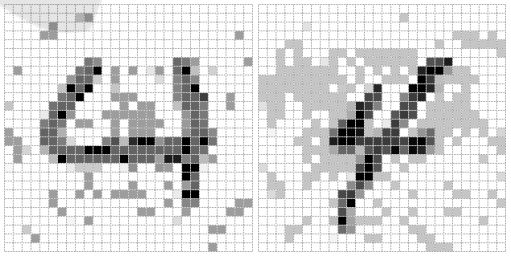
with $\boldsymbol{ip}^T = (ip^1, \cdots, ip^n)$, $\boldsymbol{w'}^T = (w'_1, \cdots, w'_m)$ and the matrix $A = (v_j^i)_{1 \leq i \leq n, 1 \leq j \leq m}$ with n lines and m columns. The plaintext $\boldsymbol{w}$ is one of the solutions of the system and the difficulty to find it depends on m, n, R, $\{\boldsymbol{v}^i\}_{1 \leq i \leq n}$ and on the intrinsic properties of the used messages (images with properties, data with properties, random, ...). To illustrate this we will consider the following example.

Let $R = \mathbb{Z}$, $m = 16$ and $n = 4$. The plaintexts, are in $\{0,1\}^{16} \subset \mathbb{Z}^{16}$. Let $\boldsymbol{w}^T = (0,1,1,0,1,0,1,0,0,0,1,0,0,1,1,1)$. Vectors $\boldsymbol{v}^1 = (2,81,80,82,3,78,90,14,66,29,52,36,11,40,83,31)$, $\boldsymbol{v}^2 = (70,64,65,46,74,10,2,85,23,54,2,41,95,83,38,6)$, $\boldsymbol{v}^3 = (54,43,98,0,93,78,23,91,52,39,43,62,19,57,95,50)$ and $\boldsymbol{v}^4 = (87,49,3,33,28,47,96,18,17,8,92,69,89,38,84,10)$ are randomly chosen in $\{0,\cdots,99\}^{16}$. So, we have $\boldsymbol{ip}^T = (460,334,502,400)$.

There are an infinite number of solutions in $\mathbb{Z}^m$ but only one in the subset our plaintexts come from: $\boldsymbol{w} \in \{0,1\}^{16}$. With a brute force attack we find it in a few seconds even if the inner product encryption scheme is secure!

A small space is clearly insecure. That is why on the one hand the parameter m and the size of the "realistic" plaintext space has to be large enough, and on the other hand the number of inner products n has to be limited. Within our use case, a brute force attack is unthinkable because of the size of our plaintext space: $2^{8 \cdot 784}$. Nevertheless it does not mean that it is not possible to get more information than the inner-product (e.g. about the handwriting we want to hide).

Now, with an explicit example we will discuss an ideal property that would ensure an ideal security level for our handwriting hiding scenario. Let us suppose that $\boldsymbol{x}_{\text{fat}}$ is the vector of the "fat" four (Fig. 1a), $\boldsymbol{x}_{\text{thin}}$ is the vector of the "thin" four (Fig. 1b), and that $A \cdot \boldsymbol{x}_{\text{fat}} = A \cdot \boldsymbol{x}_{\text{thin}}$ (which is not true in general). Let us denote by $\boldsymbol{ip}$ this value ($\boldsymbol{ip} = A \cdot \boldsymbol{x}_{\text{fat}} = A \cdot \boldsymbol{x}_{\text{thin}}$). Now, let us suppose that someone encrypts one image among $\boldsymbol{x}_{\text{fat}}$ and $\boldsymbol{x}_{\text{thin}}$ and sends it to us. After its decryption we get $\boldsymbol{ip}$. Using the classifier we get the class of the image: 4. Of course, it is impossible to know which of $\boldsymbol{x}_{\text{fat}}$ or $\boldsymbol{x}_{\text{thin}}$ was the original image. However $\boldsymbol{x}_{\text{fat}}$ and $\boldsymbol{x}_{\text{thin}}$ do not have the same character shape at all, so we can say that the handwriting is kept secret in this case. Indeed, we know that an image of a four has been encrypted but not if it is the thin four or the fat four or even an other four. In the real life $A \cdot \boldsymbol{x}_{\text{fat}} \neq A \cdot \boldsymbol{x}_{\text{thin}}$. Let us denote $\boldsymbol{ip}_{\text{fat}} = A \cdot \boldsymbol{x}_{\text{fat}}$ and $\boldsymbol{ip}_{\text{thin}} = A \cdot \boldsymbol{x}_{\text{thin}}$. With the previous idea we searched two things. First, x'_{fat} the closest[1] image to $\boldsymbol{x}_{\text{fat}}$ such that $A \cdot x'_{\text{fat}} = \boldsymbol{ip}_{\text{thin}} = A \cdot \boldsymbol{x}_{\text{thin}}$ and we got Fig. 3a. Similarly, in a second time, we searched for x'_{thin} the closest image to $\boldsymbol{x}_{\text{thin}}$ such that $A \cdot x'_{\text{thin}} = \boldsymbol{ip}_{\text{fat}} = A \cdot \boldsymbol{x}_{\text{fat}}$ and we got Fig. 3b. Those two images are acceptable in the sense that they are almost like an original image. In both cases, we used CPLEX (ILOG, 2006) for the search. More generally, if for any image $\boldsymbol{x}$ of a digit there exists an other image $\boldsymbol{y}$ of the same digit but with a different shape such that $A \cdot \boldsymbol{x} = A \cdot \boldsymbol{y}$, then the handwriting is kept secret. This property depends on the nature of the objects handled by the protocol and the matrix A. In practice, this ideal property may not be satisfied, but we can get close, as finding an image really close to the original one when we only know its $\boldsymbol{ip}$ vector value seems to be hard. Such a study will be addressed in future work to better understand how close we are from the ideal security model.



(a) (b)

Figure 3: (a) is a solution $\boldsymbol{w'}$ of Equation 2 that is the closest to Fig. 1a where $\boldsymbol{w}$ is Fig. 1b. In the same way, (b) is a solution $\boldsymbol{w'}$ of Equation 2 that is the closest to Fig. 1b where $\boldsymbol{w}$ is Fig. 1a.

[1]By closest we mean that we bound the maximum difference for each pixel.

4 EXPERIMENTATION & RESULTS

We give the results and the performance of our implementation when we use it to perform handwritten digit recognition.

4.1 MNIST Digit Classification

In Sec. 3.2 we have introduced a combined classification algorithm which can be used with an inner product encryption scheme. In our implementation we have employed the scikit-learn[2] library. The scikit-learn is an easy way to use machine learning library with an extensive set of available ML algorithms. Linear and extremely randomized trees classifiers from this library were used. These methods were called with default parameters. Four classifiers were implemented: linear classifier, combined classifier with 10, 20 and 30 intermediate classes.

As said earlier for the combined classifier with 10 intermediate classes we have simply used the inner products from the linear classifier for a new learning process. In the case of the combined classifier with 20 and 30 classes, each digit class was clustered [3] into 2 and respectively 3 "artificial" sub-classes. Each classifier was trained on the MNIST training set and the error rate on the test set was measured. Linear classifier training method in scikit-learn returns floating-point coefficients v^i (recall Sec. 2.2.1). In order to be able to use them in the inner product encryption scheme we scale and round these coefficients to signed 8-bit integers[4]. The loss in classification precision is minor in this case and can be neglected.

We implemented the fully secured functional encryption for inner product functionality (Agrawal et al., 2015) within a prime field. Our group $\mathbb{G}$ is $\mathbb{F}_p^*$ such that p is a safe prime of approximatively 2048-bits. So, the group where DDH is assumed to be difficult is the subgroup of $\mathbb{F}_p^*$ which has the prime order size $(p-1)/2$. The IPFE algorithms were implemented in C++ using the FLINT library (Hart et al., 2013) for field $\mathbb{F}_p^*$ computations. The experiments were performed on a regular laptop computer with an Intel Core i7-4650U CPU and 8GB of RAM. The sizes of the elements we manipulate in the IPFE instantiation are given in Table 1. The sizes are given in the context of a classification algorithm applied to the MNIST database: the plaintext is an image, the ciphertext is an encrypted image.

Table 1: Size of the implementation. The secret key also counts the coefficients of the vector that the secret key is associated with.

plaintext	ciphertext	secret key
784 B	196 kB	1296 B

Table 2: Execution times (in seconds) for the inner product encryption part of the classifier. The decryption column contains the time needed for all the IPFE.`Decrypt` (not parallelized) and the computaion of the classification.

Classifier	Keys gen	Encryption	Decryption
Linear or Combined 10	0.0037	0.15	23
Combined 20	0.0079		46
Combined 30	0.012		69

Table 3: Execution results of the proposed classifiers.

Classifier	Learning	Prediction	Error rate
Linear	6 sec.	< 0.1 sec.	13.93 %
Combined 10	30 sec.		7.32 %
Combined 20	77 sec.		4.86 %
Combined 30	94 sec.		4.36 %

The decryption of the IPFE scheme is performed by computing a discrete logarithm. In our case, the inner products obtained by the application of the learned linear coefficients on the MNIST database belong to a small interval. We have pre-computed a lookup table with all the (α, g^α) for α in $\{-933197 \dots 424769\}$. This pre-computation took 6.2 seconds and the size of the obtained lookup table is about 337MB. The size of the lookup can be reduced to 10.4MB if only the first 32-bits of all the g^α are kept, which can be seen as a hashing of g^α (less than 0.1% of collisions are obtained in this case). Solving the discrete logarithm using the lookup table takes under 0.18 seconds. We measured the execution time of the algorithms: 0.00037 seconds for `Keygen`, 0.15 seconds for `Encrypt` and 2.3 seconds for `Decrypt` (without the discrete logarithm part because we used the lookup table).

The execution times of the IPFE part are presented in Tab. 2. The keys generation is fast. In the "encryption" column is provided the execution time for encrypting an image and the last column is the execution time for computing the inner product values (several IPFE decryptions). We shall note that this step can be easily parallelized and then, the computation time can be divided by 10.

In Tab. 3 we present execution results for the 4 proposed classifiers. The classification algorithms were executed on the same laptop as previously. Learning is executed only once for estimating clas-

[2] http://scikit-learn.org

[3] The k-means clustering was used.

[4] We shall note that the range of coefficients is not limited to $[-127 \dots 127]$ as regular signed 8-bit integers. The inner product encryption scheme allows to encrypt any integer modulo the prime order of the cyclic group (see Alg. 1).

sifier models. Classifier error rate is the percentage of miss-predictions reported to the total number of predictions. As we can observe the introduction of a second step after the linear classifier allows to decrease the error rate by at least 6 percentage points. Using 20 intermediate inner products allows furthermore to decrease the percentage of miss-predictions by $\approx 2.5\%$. In contrast using 30 intermediate inner products instead of 20 increase the performance by less than 0.5%. We suppose that using different number of "artificial" sub-classes for each digit will allow to obtain better results.

5 CONCLUSION AND FUTURE WORK

In this work we have used an instantiation of an inner-product functional encryption scheme in order to perform classification over encrypted data. The learning process is kept secret and only linear classifiers coefficients are shared with the authority. In the protocol we introduce, we have a trusted authority, some servers computing classifications and the users who encrypt their data. Obtained execution times are reasonably small (a prediction is made in approximatively 69 seconds without any parallelization) just like the size of the ciphertexts. We have studied a method for ensuring that we cannot find the original image from the inner product values. In perspective, we consider to study more deeply the information leakage of inner product encryption schemes used in classification and to propose methods to lower it. We also consider to improve our implementation (with elliptic curve for example) in order to have smaller sizes.

REFERENCES

Abdalla, M., Bourse, F., De Caro, A., and Pointcheval, D. (2015). Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer.

Agrawal, S., Libert, B., and Stehlé, D. (2015). Fully secure functional encryption for inner products, from standard assumptions.

Bishop, A., Jain, A., and Kowalczyk, L. (2015). Function-hiding inner product encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 470–491. Springer.

Boneh, D. (1998). The decision diffie-hellman problem. In *International Algorithmic Number Theory Symposium*, pages 48–63. Springer.

Boneh, D., Sahai, A., and Waters, B. (2011). Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *International conference on pattern recognition*, pages 77–77. IEEE Computer Society Press.

Brakerski, Z. and Segev, G. (2015). Function-private functional encryption in the private-key setting. In *Theory of Cryptography Conference*, pages 306–324. Springer.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., and Waters, B. (2013). Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. volume 63, pages 3–42. Springer.

Goldwasser, S., Kalai, Y., Popa, R. A., Vaikuntanathan, V., and Zeldovich, N. (2013). Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM.

Hart, W., Johansson, F., and Pancratz, S. (2013). FLINT: Fast Library for Number Theory. Version 2.4.0, http://flintlib.org.

ILOG, I. (2006). ILOG CPLEX: High-performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex/.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. volume 86, pages 2278–2324. IEEE.

LeCun, Y., Cortes, C., and Burges, C. J. The MNIST Database. http://yann.lecun.com/exdb/mnist/.

Mangasarian, O. L., Wild, E. W., and Fung, G. M. (2008). Privacy-preserving classification of vertically partitioned data via random kernels. *ACM Trans. Knowl. Discov. Data*, 2(3):12:1–12:16.

McCullagh, P. and Nelder, J. A. (1989). *Generalized linear models*, volume 37. CRC press.

Safavian, S. R. and Landgrebe, D. A. (1991). A survey of decision tree classifier methodology. volume 21, pages 660–674.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer.

Shanks, D. (1971). Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math*, volume 20, pages 415–440.

Yang, Z., Zhong, S., and Wright, R. N. (2005). Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 5th SIAM International Conference on Data Mining*.