Illuminating the Dark or how to recover what should not be seen in FE-based classifiers

Sergiu Carpov[1], Caroline Fontaine[2], Damien Ligier[3], and Renaud Sirdey[1]

[1] CEA, LIST, Point Courrier 172, 91191 Gif-sur-Yvette Cedex, France
[2] LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France
[3] Wallix, France

Abstract. Classification algorithms and tools become more and more powerful and pervasive. Yet, for some use cases, it is necessary to be able to protect *data privacy* while benefiting from the functionalities they provide. Among the tools that may be used to ensure such privacy, we are focusing in this paper on *functional encryption*. These relatively new cryptographic primitives enable the evaluation of functions over encrypted inputs, outputting cleartext results. Theoretically, this property makes them well-suited to the process of *classification over encrypted data*. Indeed, its design enables one to perform the classification algorithm over encrypted inputs (i.e. without knowing the inputs) while only getting the input classes as a result in the clear.
In this paper, we study the *security and privacy issues* of classifiers using today practical functional encryption schemes. We provide an *analysis of the information leakage* about the input data that are processed in the encrypted domain with state-of-the-art functional encryption schemes. This study, based on experiments ran on two datasets (MNIST and Census Income), shows that *neural networks* are able to partially recover information that should have been kept secret. Hence, great care should be taken when using the currently available functional encryption schemes to build (seemingly) privacy-preserving classification services. It should be emphasized that this work does not attack the cryptographic security of functional encryption schemes, it rather *warns the community* against the fact that they should be used with caution for some use cases and that the current state-of-the-art may lead to some operational weaknesses that could be mitigated in the future once more powerful functional encryption schemes are available.

Keywords: Functional encryption · information leakage · privacy-preserving classification

Introduction

Nowadays, we have at our disposal many powerful machine learning algorithms. It includes, among others, the capability to classify accurately and efficiently all sorts of data. However for some use cases, dealing with medical data for instance, there is a strong need for privacy as well as being able to perform such algorithms.

In this paper, we focus on Functional Encryption (FE), a primitive that we use to ensure data privacy in a classification use case. Functional Encryption is a generalization of traditional public key cryptography which offers the possibility to only evaluate authorized functions on encrypted inputs. More precisely, inputs are encrypted with a public key, while dedicated secret keys allow their owner to evaluate specific functions (each of these functions being related to a specific secret key) over the encrypted inputs, then providing the resulting evaluation output in the clear. Several works from the literature [30, 13] propose to use simple yet practical FE schemes in order to classify encrypted images with the help of Inner-product FE (IPFE) [30] and quadratic FE [13]. In these works, the MNIST dataset of handwritten digits is employed to assess feasibility and practical performance of such classifiers for "reading blindly".

An inherent drawback of FE schemes, for a use in this context, is that some details about the encrypted input data naturally "leak" via the cleartext output of the authorized functions evaluations. We call *information leakage* of a FE scheme the maximal information one can gain about the private input data from the cleartext output(s) of the FE function(s) evaluation(s). It should be emphasized that this kind of leakage is beyond the scope of the cryptographic security properties of the FE scheme as cleartext evaluations outputs are desired in such primitives, whose security goal is to guarantee that nothing more than the cleartext outputs can be learned about the private inputs. Yet, as we demonstrate in this paper, in the context of classification the information we can infer on a private input from a (set of) cleartext evaluation(s) output(s) may jeopardize the operational security of current (cryptographically secure) FE-schemes deployment. This work also aims at contributing to help deciding if a given neural network is indeed acceptable or not in terms of user privacy, when it is implemented within the constraints imposed by present day practical FE schemes.

Contributions

In the present work, we first analyze the possible designs for FE-based classifiers according to the current state-of-the-art of FE schemes. Current practical FE schemes can only be used to evaluate limited functionalities, in particular linear or quadratic functions over encrypted inputs. This limited functionality of practical FE schemes implies that one needs to cut the classification process into two steps: (i) the first one precisely consists in the evaluations of several linear or quadratic functions over the encrypted inputs, provided by a FE scheme, which produce several cleartext evaluations output values; (ii) then, a second step uses these intermediate values to compute the final output of the classification. Our purpose in this paper is to show how these cleartext intermediate values can be used to recover information about the original encrypted inputs.

In previous works addressing this setup [30], the first step was achieved by a linear FE scheme, namely an IPFE scheme, whereas in the present work we focus on the use in the first step of a quadratic FE scheme, which should *a priori*

provide a higher privacy level as more computation is performed over encrypted inputs.

We also introduce the notion of *minimal* and *operational information leakages* for FE-based classifiers. Roughly speaking, the *minimal leakage* is the minimal information leakage intrinsicly related to the use case goal achievement (e.g. perform a classification task), whereas the *operational leakage* is the information leakage resulting from an actual implementation of the use case (resulting from a dichotomy between encrypted then clear domain calculations). The operational information leakage is necessarily larger than the minimal one, as intermediate use case data are available in the clear, whereas in the minimal case the whole classification process is performed in the encrypted domain.

Our study takes place in the context of an honest-but-curious threat model, where the server evaluating the classification process may act maliciously and attempt to gain as much information as possible on the encrypted inputs, while all parties are behaving as expected. Hence, the attacker has complete access to the classification process, even to the clear intermediate values output by the FE scheme. In this context, we measure and compare operational and minimal information leakage estimations in classification use cases. Information leakage is estimated with the help of machine learning tools, in particular neural networks. We perform extensive experimental studies on two datasets: the MNIST dataset and the Census Income dataset. We show that with the MNIST dataset, an attacker is able to reconstruct input digit images which are close (in terms of MSE distance, as well as visually) to original ones. Hence it is easier to impersonate people who originally wrote the digits used as inputs, or to steal their identities. In the case of the Census Income dataset, ou study shows that an attacker is able to gain more insights on highly-private features, characteristics of individuals. For example, he is able to assert with a higher confidence the gender or the ethnic origin of an individual, among other features.

Finally, we show that, using a second classifier, we are able to decrease the number of needed FE evaluations compared with the classification based on a logistic regression model proposed in [13] (which used a large number of FE evaluations, at least larger than needed). This decreases at the same time the estimated information leakage, thus improving the privacy of encrypted inputs.

Related works

Privacy-preserving classification is a promising research direction in machine learning field, which goal is to perform prediction tasks while preserving the privacy of individual data. Privacy issues may concern the data used during the training phase, the data given to the classifier during the inference phase (prediction), or the classifier itself. Several types of attacks and studies address each of these issues. In the present paper, we focus on the issues concerning the users data during the inference phase. Privacy issue concerning the classification algorithm/process/model, or concerning the users data involved during the training phase are out of the scope of this paper. Please, also note that the present paper

only focuses on privacy issues, and attacks aiming to provoke mis-classification are out of the scope of our study.

Several works found in the literature treat different techniques to perform predictions tasks and to keep the privacy of input data [31, 2]. In [40] the authors provide a review of techniques and methods, ranging from heuristic to cryptography-based ones, for privacy-preserving prediction. In [34], an extensive study on attack surface, adversarial goals and capabilities of privacy preserving machine learning systems is discussed.

In [13], the authors use a quadratic functional encryption scheme to classify data into n classes. They define a neural network with quadratic activation functions and use it to generate n quadratic polynomials, each one describing a class. Afterwards, they generate n FE secret keys associated with each of the above quadratic polynomials. A classification model built in this way corresponds to a multinomial logistic regression model with quadratic features. The authors experimented this privacy-preserving classifier with the MNIST dataset of handwritten digits and got 97.54% of classification accuracy with 10 secret keys. Although straightforward, this classification model does not fully exploit the information contained in the n values of quadratic polynomials evaluation. Indeed, in [30] and also in the present paper, the values obtained after FE evaluation/decryption are considered as intermediate values for a second step of the classification process. This means that a more accurate classification model can be built using the same number of FE secret keys or a model reaching the same classification accuracy using a smaller number FE secret keys. In practice a smaller number of FE secret keys implies less computation, i.e. less FE evaluations/decryptions to perform. Another non-negligible effect of FE secret keys decrease is that the information leakage on input encrypted data is potentially smaller. Our work completes the privacy-preserving classification model from [13] with a second-step of classification and additionally describe the ability of an attacker to gain more information on input data than its class only.

As stated above, FE is a cryptographic primitive (alongside others) which can be used to accomplish privacy preserving prediction. Our work focuses on information leakage estimation linked with the use of current state-of-the-art FE-based classifiers. We exploit intermediate values revealed in the clear by the use of linear or quadratic FE schemes to infer private information about the original inputs.

Our approach shares some similarities with *model inversion attacks*, which use the output of a model applied to an unknown input and infer certain features about this input. More precisely, inversion attacks use the knowledge of a model – resulting from the learning phase – to infer private information about the inputs. Fredrikson et al. [15] investigated model inversion attacks applied to a linear regression model in the field of pharmacogenetics. Their study reveals that such a model leaks sensitive information about patient genotype and would pose a danger to genomic privacy. In a subsequent work [14], this model inversion attack is further generalized to different machine learning primitives: decision trees for lifestyle surveys and neural networks for facial recognition. In both

cases, confidence values – which are revealed in order to make predictions – are used to estimate input features or reconstruct recognizable face images. Our approach and inversion attacks are similar as they both use some knowledge about intermediary data used in the classification step – which depend on the inputs – to infer some private information about this input. At the same time, both attacks are different as in our case the attacker should have access to the intermediate values in the clear, and then should have access to the whole classifier, whereas in the inversion model attack the attacker may be external and will first query the classifier to estimate the classification model and coefficients.

In the context of searchable encryption [11] and of order-revealing encryption [18] the type of leakage attacks studied in our work are called *leakage-abuse* attacks. The goal of leakage-abuse attack is to exploit the leakage explicitly allowed by a cryptographic scheme in order to reveal/obtain information about input data, which is similar to our work except that the cryptographic construction is different. In any case, the leakage-abuse attack does not attack the cryptographic primitive itself but only its use in an operational context (use case).

Other types of attacks have been run on machine learning use cases, such as membership inference attacks [38, 6, 37, 19, 32] or model extraction attacks [5, 39]. On one hand, membership inference attacks – also referred as tracing attacks – aim to determine whether or not an input was a part of the dataset used during the training phase of a target model. In this paper, we do not focus on this membership privacy concern. On the other hand, model extraction attacks aim to extract the parameters of a target model trained on a private dataset. The main motivation is to construct a model whose predictive performance is similar to the targeted model. Compared to this attack in our case the model is already available to the attacker.

Organization of the paper

This paper is organized as follows. Section 1 comprehensively details the kind of deployment scenarios and the underlying threat model which we address in this work. We then proceed in Section 2 with a formalization of generic FE schemes and a description of linear and quadratic schemes, which are the only practical schemes available today. Afterwards, in Section 3 we describe in details the information leakage in FE based classification use cases, and how to estimate it with machine learning tools. In Section 4 we describe the datasets we used, and the experiments we performed, and provide an analysis of the results we obtained concerning the information leakage. Finally, we conclude this work and provide some perspectives for future works in Section 5.

1 Classification over encrypted data by means of Functional Encryption: scenario and threat model

This section aims at precising the kind of classification use case that can be meaningfully addressed by means of Functional Encryption (FE) and, then, the

resulting threat model. More technical details about the structure of FE are provided in Section 2. We start by considering the following concrete application scenario. There is a pharmaceutical firm wishing to conduct an epidemiologic study over, say, the population of a given country. In order to do so, they need to evaluate e.g. a specific neural network on some health related data over a large set of patients. The point is that (1) the evaluation of the neural network should be done on their own servers (i.e. at their own cost) without interaction and (2) to conduct their study there is no legitimate need to have access to the inputs of the neural network but rather only to its outputs. Bringing FE into the picture, we consider that a health authority is the (trusted) owner of a FE scheme instance that is a public key which any patient can use to encrypt some data and the associated master secret key which allows to spawn additional secret evaluation keys. In this setup, the firm needs first to submit its study to the health authority which includes full disclosure (to that authority) of its neural network. The authority then decides whether or not the network is acceptable with respect to patient privacy (for example, a malicious neural network outputting its inputs would straightforwardly not be acceptable as its evaluation would grant access to the input data). Deciding whether or not a neural network (or any other algorithm) is acceptable is easier said than done, as the present paper contributes to demonstrate. This decision is under the responsibility of the authority. If acceptable, the authority gives to the firm a specific evaluation secret key, which allows the firm to evaluate a specific function over inputs that have been *encrypted* under the FE scheme public key; in the FE paradigm, the output of this evaluation is provided as a *cleartext* result, which is then readable by nature. Patients (or perhaps rather the doctors to which patients give their consent) can then encrypt the relevant data under the FE public key (as provided by the health authority) and send them to the pharmaceutical company to contribute to the study. This setup leads to the following properties:

- The pharmaceutical firm (evaluation server) has no access to its neural network inputs (x) but only to its output ($f(x)$) and cannot compute another function unless the authority provides it with a new secret key tight to that new function.
- The health authority (authority) has access to neither patient data nor neural network output and has no role to play in the operational running of the study (i.e. no server to operate or no cost to incur on a per-patient basis).
- The patients (users) have to trust that the health authority will allow the evaluation only of acceptable (in terms of private information leakage) functions on their data and that it will not collude with the pharmaceutical firm.

In terms of threat model, this setup allows to address *confidentiality threats* on *user inputs* from the *evaluation server*. Our study takes place in the honest-but-curious model, where all parties follow the attended protocol, but can try to get advantage of the information they are allowed to access. In our case, we focus on a malicious classifier (evaluation server), which may try to learn

sensitive information about the users inputs from what it can observe during the classification process.

It should be emphasized that such type of privacy could also be achieved by means of Fully Homomorphic Encryption (FHE) but with different consequences in terms of system architecture. In essence, in a FHE-based setup, the pharmaceutical company could send its network to the authority, and the patient would send their data encrypted under the FHE public key of the company to the authority, which would homomorphically evaluate the network, and get an *encrypted* result, which would be sent back to the company for decryption and further exploitation. This FHE-based setup achieves the same security properties as the previous FE-based one, but this time the health authority becomes an FHE-computation operator which most likely requires to acquire fairly large-scale computing resources involved on a per-patient basis. On the contrary, the FE setup duly puts the main computing burden on the pharmaceutical firm, involving the health authority on per-study rather than per-patient basis. This is why we focus in this paper on the FE-based classification setup, which is more pertinent from a practical point of view in such a privacy-preserving use case.

Unfortunately, yet, the current state-of-art in FE does not credibly allow to practically evaluate a full neural network, so the evaluation of the network has to be split into a first encrypted-domain evaluation phase and a second clear-domain one, meaning that the server has also access to intermediate results rather than just the final network outputs. In essence, this paper investigates the consequences of this partitioning on user input privacy, which is really different to what is going on in other threat models where the attacker infers some parameters of the classifier (in our study, the classifier is supposed to be completely known by the attacker). In fact, this work can be seen as taking the point of view of the (health) authority above in contributing to help deciding on whether or not a given neural network is indeed acceptable in terms of user privacy when implemented within the constraints imposed by present day practical FE schemes.

2 Functional encryption

Functional Encryption (FE) is a quite recent generalization of public-key cryptography, which can be used in cloud computing [9] and verifiable computation [35]. Functional encryption also generalizes attribute-based encryption [42, 28, 41], identity based encryption [8, 36] or predicate encryption [23, 33]. It can also be used to preserve data privacy in some classification use cases, as in [30, 13]. In this section, we will recall definitions and specificities concerning functional encryption, and will summarize the state-of-the-art concerning the current available schemes proposed in the literature.

The FE paradigm adds a new party, an *authority*, to the two traditional asymmetric parties. The authority generates a *master secret key* and a *public key*. The master secret key is known only by the authority. This particular key is necessary to derive what are called *secret keys*. The secret keys are associated

with functions; for instance, we denote by sk_f the secret key associated with function f. The public key, as expected, is used to encrypt messages. Let ct_x be an encryption of a message x. A user owning sk_f and ct_x can run the evaluation algorithm and get $f(x)$ as plaintext output. Hence, there is no traditional way to decrypt, but a kind of evaluation of some functions f (one function for each secret key) over the encrypted messages, with unencrypted evaluation results at the end. Boneh *et al.* provide in [9] the following standard definitions for functional encryption using the notion of *functionality*.

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 tuple composed 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 $sk \leftarrow \texttt{keyGen}(MSK, k)$ and $c \leftarrow \texttt{encrypt}(PUB, x)$ we have $F(K, X) = \texttt{decrypt}(PUB, sk, c)$ with an overwhelming probability. Figure 1 illustrates FE actors and their roles as described above.

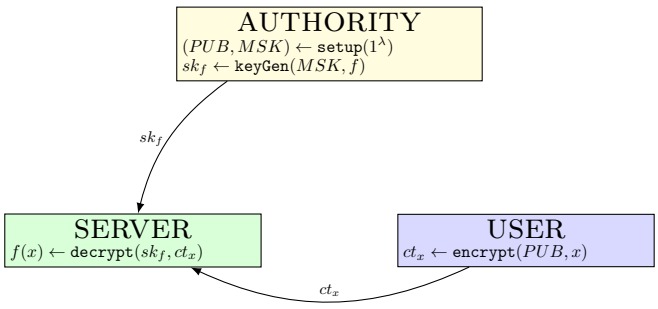


Fig. 1: The three actors of a functional encryption system, the algorithm they use and their communications. In this figure, the public key, the master secret key and the secret key associated with function f are respectively called PUB, MSK and sk_f.

The cryptographic community is currently looking for public-key functional encryption schemes enabling to evaluate any polynomial time computable function. Goldwasser *et al.* proposed a construction based on fully homomorphic encryption [17], Garg *et al.* proposed another construction using an indistinguishability obfuscator [16]. At present, however, these constructions remain mostly of theoretical interest. Nevertheless, more recent schemes for simpler functionalities have been proposed, for example FE schemes supporting linear [1, 3] (also called inner-product functional encryption or IPFE) or quadratic [7] polynomial evaluation. The advantage of these schemes is their decent performance and applicability in real applications.

Linear FE or inner-product FE We call *linear functional encryption* a scheme which enables the evaluation of degree-one polynomials. In the literature these schemes are also called *functional encryption for the inner-product functionality* or *inner-product functional encryption (IPFE)*. Let $\boldsymbol{v}$ be a vector, $ct_{\boldsymbol{v}}$ an encryption of $\boldsymbol{v}$, $\boldsymbol{w}$ be a vector of coefficients, and $sk_{\boldsymbol{w}}$ the secret key associated with $\boldsymbol{w}$. The decryption of ciphertext $ct_{\boldsymbol{v}}$ with secret key $sk_{\boldsymbol{w}}$ returns $\boldsymbol{v}^T \cdot \boldsymbol{w} = \sum_i w_i \cdot v_i$, thus a linear polynomial evaluated at $\boldsymbol{v}$.

Abdalla *et al.* [1] proposed constructions for the inner-product encryption schemes satisfying standard security definitions, under well-understood assumptions like the *Decisional Diffie-Hellman* and *Learning With Errors*. However they only proved their schemes to be secure against *selective adversaries*. Agrawal *et al.* [3] upgraded those schemes to provide them a *full security* (security against *adaptive attacks*).

Quadratic FE We call *quadratic functional encryption* a functional encryption system which can evaluate degree-two polynomials. A first construction of this type has been recently provided in [7]. Let $\boldsymbol{v}$ be a vector, $ct_{\boldsymbol{v}}$ an encryption of $\boldsymbol{v}$, $\boldsymbol{W}$ a square matrix of coefficients and $sk_{\boldsymbol{W}}$ the secret key associated with $\boldsymbol{W}$. The decryption of ciphertext $ct_{\boldsymbol{v}}$ with secret key $sk_{\boldsymbol{W}}$ returns $\boldsymbol{v} \cdot \boldsymbol{W} \cdot \boldsymbol{v}^T = \sum_{i,j} W_{i,j} \cdot v_i \cdot v_j$, thus a quadratic polynomial evaluated at $\boldsymbol{v}$.

3 Information leakage in typical FE deployment scenarios

3.1 Classification use case

A natural use case for functional encryption schemes is the classification over encrypted data. Several classification algorithms from the machine learning field use polynomial projection of input data features, e.g. Linear Classification, Gaussian Mixture Model, etc.

Unfortunately, as mentioned in Section 2, today no functional encryption scheme can be found in the literature that would allow to perform practical arbitrary computations over encrypted data. Available schemes which are practical from a computation point of view only allow to evaluate linear and quadratic

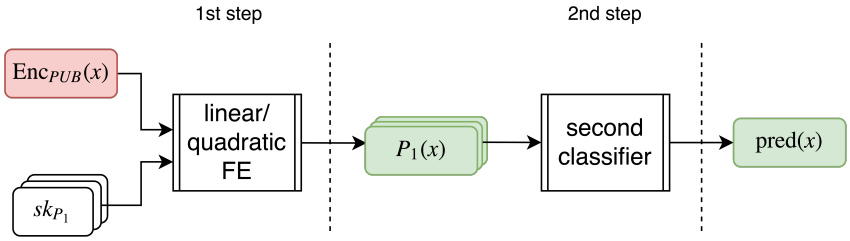


Fig. 2: Algorithm decomposition in a FE use case. Red and green boxes denote respectively encrypted and clear data.

polynomials. These schemes cannot be used as is in classification use cases because most of them require more complex processing. A solution to bypass this limitation is to adapt the use case algorithm and decompose it in two steps: (i) linear/quadratic polynomial evaluations over encrypted data, (ii) any computation performed on clear data resulting from the first step. Figure 2 illustrates this decomposition. Here, in the first step, k polynomials $P_1(.), \ldots, P_k(.)$ are evaluated over an encrypted input sample x using the functional encryption scheme, with the help of the corresponding FE secret keys $sk_{P_1}, \ldots, sk_{P_k}$. As an inherent property of functional encryption, the resulting values $P_1(x), \ldots, P_k(x)$ are provided in the clear. They are used as inputs for the second step, which performs the additional computations related to the use case. This second step does no more involve encrypted data and only performs in the clear.

Classification over encrypted data can be built upon this decomposition. The first step of the use case algorithm is either a linear or a quadratic polynomial evaluation over encrypted data. After this step, any computation can be performed on the first step output values, because they are available in the clear form. For the second step, we have several possibilities (non-exhaustive list):

- "sign" function for binary classification,
- "argmax" function for one-vs-all multi-class classification,
- a full-fledged classification algorithm.

The last possibility needs some clarification. Any classification algorithm taking as input the polynomial evaluations from the first step can be used as a "final" classification algorithm, and the intermediate polynomial evaluations obtained with the help of the FE scheme can be seen as linear or quadratic projections of the input dataset.

3.2 Information leakage

Each output of a FE scheme consists in the cleartext data representing the evaluation of the functionality embedded in the corresponding FE secret key. An attacker could use this clear text data in order to infer more information about the input data than it is supposed to know. This is particularly relevant when

several functions can be evaluated by the same entity over the same sensitive encrypted input. We call *information leakage* this downside of FE schemes. A more precise (yet still informal) definition is given below:

Definition 3 (Information leakage). *Let sk_f be a functional encryption secret key, embedding functionality f. The* `decrypt` *operation (described in previous section) allows to obtain a cleartext value of $f(x)$ from an encrypted x. The* information leakage *is defined as the maximal information that can be inferred about x from the knowledge of both $f(x)$ and function f specification.*

Knowing the specification of f we are able to evaluate f and consequently to obtain the cleartext value $f(x)$ for any x. This information leakage definition is straightforwardly generalized to FE scheme instantiations where several secret keys $sk_{f_1}, \ldots, sk_{f_k}$ are available.

In the introduction, we defined *minimal* and *operational information leakages*, which are both beyond the scope of the underlying FE scheme security properties which guarantee that given n decryption keys $sk_{f_1}, ..., sk_{f_k}$ and an input x an attacker learns nothing more than $f_1(x), \ldots, f_k(x)$. We define the *minimal leakage* as the minimal information leakage intrinsicly related to the use case goal achievement, e.g. here perform a classification task. Hence ideally in our use case only the final classification result should be revealed about the input data x. We also define the *operational leakage* as the information leakage resulting from an actual implementation of the use case.

Hence, in our use case minimal and operational information leakages may be equal if the whole classification could be processed over encrypted data with a unique authorized function evaluation, revealing in the clear only the final classification result. But, according to the current necessary dichotomy between encrypted then clear domain calculations, we need in practice to evaluate k polynomials over the input x. Hence, in this case the attacker will have access to the k evaluations results, which may reveal more information about x than the final classification result. In this case, operational information leakage may then be much larger than the minimal one.

However we keep the above definition informal and provide more intuition below. Indeed measuring the absolute information leakage of a function (for a given input or a given set of inputs) is not easy (not computable), as it is linked to the Kolmogorov complexity (as a generalization of entropy) of that input. In essence, the output of a function $f : \{0,1\}^m \longrightarrow \{0,1\}^n$ with $m >> n$ leaks *at most* n bits about the corresponding input (at most because some functions, e.g. a constant function, lead to no leak at all). Intuitively, feeding random data, or equivalently an input with large Kolmogorov complexity, into any function appears benign because the leak is indeed limited to n bits. Difficulties start to crop up when low Kolmogorov complexity data are fed into a given function because in that case n may not be much smaller than the minimum number of bits required to describe that input. Therefore, the induced leak could (in principle, yet intrinsically) allow to retrieve the input in question with enough precision. When considering machine learning algorithms, we are typically in a setup where the function purpose is to extract a few highly discriminant bits

from highly correlated partially redundant data i.e. data of relatively low Kolmogorov complexity. So, we are bending towards the dark side. Yet, due to the non-computability of Kolmogorov complexity, it is difficult to quantitatively apprehend the above intuitions. Still, the neural network-based leakage estimation technique presented subsequently provides an (efficient) approximation of an oracle able to retrieve low Kolmogorov complexity inputs from the corresponding outputs. Full (formal) details on the deep connections between Kolmogorov complexity and machine learning underlying the above intuitive arguments may be found in [20].

3.3 Leakage estimation

In this section, we introduce a method for estimating the information leakage resulting from the use of functional encryption schemes in use cases implying the classification of encrypted data. In this context, which has been described in Section 3.1, we propose to estimate the leakage with the help of the information discovery capabilities of a Neural Network (NN).

Leakage estimation protocol Let f be a functionality encoded in an FE scheme and let X be a representative dataset. By "representative dataset" we understand a dataset that follows the same distribution as the dataset used by the classification algorithm, i.e. X contains typical classification inputs. We suppose that such a dataset X is available in the clear and that we can straightforwardly obtain $f(x)$ for any $x \in X$ because functionality f is not hidden in public-key FE schemes. Our goal is to accurately predict/reconstruct dataset samples x (whole x or a part of it) from $f(x)$. The accuracy is measured by using a suitable metric (more details are given below). To accomplish this goal, a neural network is employed.

Achieving maximum accuracy is not possible as the problem of obtaining an universal predictor [21, 22, 27] is intractable in the general case. Hence, the estimated information leakage (measured by the accuracy metric) will be a lower bound to the operational information leakage. Nevertheless, such an estimation proves to be useful for comparing information leakages of different FE instantiations. We assume that the NN model has the same information extraction power, independently of FE instantiation used upon. This is (supposedly) the case when comparing estimated minimal and operational information leakages of classification use cases.

We place ourselves in the context of the FE-based classification use case described in Section 3.1. A Neural Network is used to reconstruct an input x from use case data available in the clear. Figure 3 illustrates this methodology. We have two possible leakage estimation NNs, depending of the data which is available in the clear form

- second/final classifier prediction $\mathrm{pred}(x)$ only,
- or FE outputs $P_1(x), \ldots, P_k(x)$ (linear/quadratic polynomial evaluations).

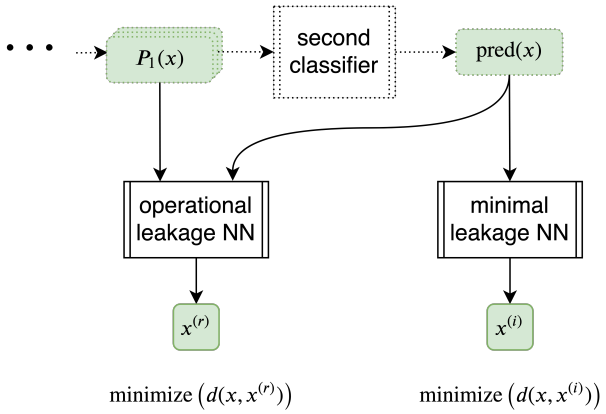


Fig. 3: Operational and minimal information leakage estimation using Neural Network. Top part of this illustration is the FE use case from Figure 2.

An estimation of the minimal information leakage is obtained in the first case, whereas operational information leakage in the second one. By comparing the prediction accuracies of these two NNs we should be able to better understand and to gain insights about classification use case information leakage.

4 Experiments

We start this section by providing more details about the employed datasets and the neural networks used for classification and for information leakage estimation. Afterwards, we provide an aggregation of the results we obtained. We shall note that no FE schemes evaluation were performed in this work since, as already emphasized, we are not attacking the cryptographic security of FE schemes but exploiting the fact that they by intent provide cleartext output (as well as the fact that existing practical FE schemes are limited in terms of expressiveness). Typical execution times for linear and quadratic FE schemes decryption (corresponding to polynomial evaluation) is small (few seconds). Please refer to [30, 13] for more details about FE performance.

4.1 Datasets

Two datasets are used in our experimentation. The first one is the well known *MNIST* dataset [26]. The MNIST database is a collection of handwritten digit images. Dataset images have size 28×28 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 [10, 25].

The second one is the *Census Income* dataset introduced in [24], and which can be found in [4]. This dataset contains highly-sensitive personal data (age, education, race, etc.) of approximately 50 thousands individuals given by 14 features. 6 features are continuous (e.g. age) and other 8 are categorical (e.g. race). Two redundant features (fnlwgt, Education) have been removed. The missing values in categorical features were replaced by the most frequent value. Continuous features have been scaled to zero mean and unit variance. Categorical features have been one-hot-encoded, i.e. transformed to binary features, denoting whether an individual is in the corresponding category. The transformed dataset has 82 features, of which 5 continuous and 77 are binary. The prediction task is to determine whether a person earns over 50K$ a year.

These datasets are split into training, validation and test subsets. The training subset is used to train both the prediction classifier and the information leakage estimation neural network. The validation subset is used to choose the final neural network (prediction and information leakages). The test subset is used for asserting both the prediction accuracy (i.e. for the nominal use of the system) and the estimated information leakage of the chosen neural network (i.e. during the attack).

4.2 Neural networks structure

We describe in more details the neural networks used for prediction and for information leakage estimation.

Prediction NN The neural network used for prediction (i.e. for the nominal use of the system) has the following structure:

- linear or quadratic first layer,
- one hidden layer (optional),
- one output layer.

First layer The first layer corresponds to the functionality of a linear, respectively quadratic, FE scheme. In this way, the neural network model automatically learns the coefficients of FE scheme secret keys. This layer has k outputs, corresponding to k FE secret keys/evaluations. In our experiments k belongs to $\{1, \ldots, 10\}$. No activation function is used on this layer, so that the hidden and output layers can be directly evaluated from the outputs of a FE scheme.

This linear layer simply performs inner products or, equivalently, a projection to a k-dimensional space. For the quadratic layer, we used the same approximation of quadratic polynomials as in [13]. In particular, input data is firstly projected to a d-dimensional space for $d \in \{50, 100\}$. Each component of the d-dimensional space is then individually squared. Finally, these components are projected to a k-dimensional space.

Hidden layer The hidden layer is optional. When there is no intermediary layer the neural network corresponds to a logistic regression model (or multinomial logistic regression in case of MNIST). In this way we can test simple linear or quadratic prediction models, and compare our prediction results with those from [29, 30, 13]. A ReLU (Rectified Linear Unit) activation function is used here. This layer has 256 nodes. Empirical results have shown that there is no need for more than one hidden layer in our experimental setup.

Output layer The output layer has a single node for the Census Income and 10 nodes for the MNIST dataset. The activation functions are sigmoid for the first one, and softmax for the second one. The sign and argmax functions (used to transform continuous neural network output values into labels) are not included in the neural network, otherwise we will not be able to train the model. The employed validation metric is the prediction accuracy of the obtained neural network classifier. The prediction accuracy is the ratio of correct predictions to the total number of predictions made and belongs to $[0, 1]$ range.

Attack NN The neural networks used to estimate the information leakage (i.e. NN used during the attack) follow the same structure as above: (i) an input layer to which the information available to attacker is fed, (ii) several hidden layers, (iii) and an output layer.

Information leakage – MNIST use case The input layer for the minimal information leakage estimation neural network has 11 input nodes. One input node is the digit label (integer value from 0 to 9) and 10 other nodes represent the one-hot-encoding of this label [4]. We have chosen to provide different encodings of digit label in order to ease the neural network training. Usually, initial weights of NN layers are randomly chosen so that gradient descent like training algorithms avoid local minima. We have selected the initial weights in such a way that the NN outputs an average of images in the train dataset for a digit when the one-hot-encoding input corresponding to this digit is activated. Actually, a small random noise is added to the average in order to avoid local minima. In practice training algorithm convergence is accelerated by this initialization process instead of randomly selecting network weights.

In the case of the neural network for operational information leakage the input layer has k additional nodes which correspond to the outputs of the FE scheme. Two hidden layers with 256 nodes each are used. The output layer has 784 nodes corresponding to each pixel of the image to reconstruct. All the layers use the ReLU activation function. The validation metric is the MSE (mean squared error) score.

Information leakage – Census Income use case The minimal information leakage estimation network has a single input node. To this input the binary output

[4] The one-hot-encoding of a digit m is the vector v where $v_m = 1$ and $v_i = 0$ for all other $i \neq m$.

(whether a person earns over 50K$ a year) of the prediction network is fed. In the case of operational information leakage estimation the neural network has additionally k input nodes corresponding to FE scheme decryption outputs. Two hidden layers with 256 and 32 nodes are used. The output layer has a single node. Hidden layers use ReLU activation function and output layer the sigmoid activation.

The information leakage for the Census Income dataset is measured as the ability for the leakage estimation model to make good predictions on a feature of the input dataset. In our experiments the information estimate leakage is estimated for binary input dataset features, in particular: Sex_Male, Race_White, Race_Black, Race_Asian-Pac-Islander and Race_Other. The validation metric is the ROC AUC score (Area Under the Receiver Operating Characteristic Curve). To summarize, the prediction is performed on the full Census Income Dataset and the goal of the attack network is to predict one input dataset feature from the output of the prediction network.

4.3 Results

In this subsection, we present the experimental results we have obtained for the prediction accuracy and the information leakage estimation.

We have used 2 datasets, 3 types of FEs (1 linear and 2 quadratic), 11 different number of FE evaluations (10 for the NN models and 1 for the logistic regression model). In total, we have modeled 66 neural networks for the prediction task (2 datasets $\times$ 3 FE types $\times$ 11 FE evaluations). A minimal and a operational leakage analysis NN was built for each MNIST model, a total of 66 neural networks (33 prediction NN $\times$ 2 leakage analysis types) are obtained. For the Census Income dataset the operational and the minimal information leakage estimation is performed on 5 different input features, a total of 330 neural networks (33 prediction NN $\times$ 5 features $\times$ 2 leakage analysis types).

Keras framework [12] is used to implement these neural networks (note that the linear logistic regression model is simply a 1-layer network and the quadratic one is a 2-layer network). A batched training (batch size 32) over 100 epochs is performed. sgd (stochastic gradient descent) optimizer is used for training NNs except for the Census Income information leakage NNs where adam (a variant of stochastic gradient-based optimization method) optimizer is employed. Binary (Census Income) and categorical (MNIST) cross-entropy are used as loss functions during the training phase. Other optimization parameters are the default ones.

The best network (after each epoch) in terms of optimization metric value on the validation set is chosen as the neural network to keep. The training process is executed 5 times with different random seeds. The average of metric value over the test dataset is used in illustrations. The best prediction model, in terms of metric value over the validation dataset, is used for information leakage estimation. Prediction models are denoted as:

- linear NN model (fe1): linear FE and a neural network as second classifier,

– linear logistic model (fe1_logit): linear FE and a logistic regression model as second classifier,
– quadratic NN model (fe2_d): quadratic FE d-space projection and a neural network as second classifier.
– quadratic logistic model (fe2_d_logit): quadratic FE with d-space projection and a logistic regression model as second classifier.

Neural networks structure (number of layers, layer sizes, etc.) and training hyper-parameters have been manually chosen for maximizing prediction accuracy. We have tried to increase hidden layer count, increase/decrease hidden layer sizes, change activation functions, use different optimizer and number of training epochs. The obtained NN accuracies were practically the same. Nevertheless, we are not able to formally ensure that the NN structure and the training hyper-parameters we have chosen are the best possible ones.

MNIST dataset

MNIST – accuracy Figure 4 shows how the final prediction accuracy (i.e. the ratio between the number of good predictions vs. the total number of samples) evolves, according to the number of FE evaluations used in the first step (i.e. according ot the number of intermediate values provided in the clear form by the FE scheme). We can observe that there is no significant difference in terms of accuracy between quadratic models with different projection sizes (50 or 100) in our experimental settings. As expected, the quadratic models always give better accuracies than the linear ones. Even if prediction accuracies using a single FE evaluation are not overwhelming (0.5 for linear and > 0.7 for quadratic polynomials), it is much better than one would expect using random guessing (≈ 0.1).

The quadratic logistic model (fe2_d_logit) attains the same accuracy as the quadratic NN model (fe2_d). Although, comparable accuracies are obtained by the quadratic NN model starting with only 5 FE evaluations. Using a NN model is more interesting as the number of FE decryptions to perform will be smaller in this case.

The accuracy of the linear logistic model (fe1_logit) corresponds to the accuracy of a linear NN model with $5 - 6$ FE evaluations.

MNIST – information leakage Figure 5 illustrates the information leakage in terms of MSE (the lower MSE, the greater information leakage). Obviously, MSE corresponding to the minimal information leakage is always larger than the one corresponding to the operational information leakage.

At first glance it may seem counter-intuitive that the minimal leakage is not the same for different numbers of FE evaluations (secret keys). As the minimal leakage estimation NN uses predicted digits labels by the classification NN and not real ones, the prediction error translates to smaller minimal leakage (larger MSE values). For a given digit, the reconstructed image obtained by the minimal leakage NN should correspond to the average of images of this digit in the train

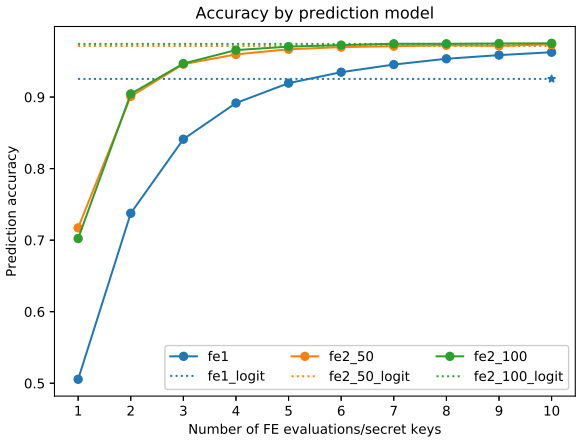


Fig. 4: MNIST dataset – prediction accuracy.

dataset. Actually, it corresponds to the average of the images which are classified as this digit by the classification NN.

The minimal leakage is roughly the same for both quadratic models. We observe that the minimal leakage is approximately equal to the operational leakage of linear/quadratic NN models when one FE evaluation is used. This means that the operational leakage estimation network has no significant advantage over the minimal one, i.e. approximately the same information about input images is leaked in both cases. As said earlier, even one FE evaluation allows to increase the prediction accuracy to a much better value than random guessing. So in these conditions, using a FE-based classifier leaks the same amount of information as a cleartext one.

There is no significant difference between linear and quadratic NN models in terms of operational information leakage. Although, for a higher number of FE evaluations the linear NN model (f1) leaks a little more information than the quadratic counterpart (fe2_d).

The leakage of logistic models is equivalent to the leakage of linear/quadratic NN models with 10 FE evaluations. We may conclude that there is a strong correlation between the number of FE evaluations and the estimated information leakage. At least, stronger than the correlation between evaluated polynomials themselves (i.e. theirs coefficients) and the information leakage. There is a negative correlation between prediction accuracy and operational information leakage (MSE) (correlation coefficient ≈ -0.8) for quadratic models and stronger for linear ones (correlation coefficient ≈ -0.87). As expected, better classification accuracy induces larger information leakage.

Estimated information leakages presented above are average measures over all digits. Figure 6 illustrates estimated information leakage per digit for models using 5 FE evaluations. Minimal leakage depends mainly on the average image

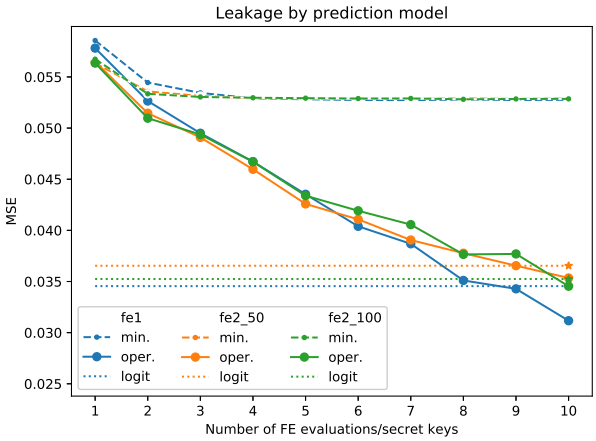


Fig. 5: MNIST – information leakage estimation.

of a digit in the train dataset. There is no significant difference between minimal leakages for different attack models (fe1, fe1_50 and fe2_100) as expected. Minimal leakage values in the figure are equal to the average of variance of image pixels for a given digit.

The operational information leakage is larger (smaller MSE) than the minimal one for every digit. The ratio between minimal and operational leakages (averaged over attack models) is shown below each bar plot. These ratios differ from one digit to another and vary from 1.11 to 1.5. As FE secret keys are the same (i.e. same linear/quadratic polynomial coefficients) we conclude that the operational information leakages depends on the input data distribution also. That is to say some digits (e.g. digit 1) are easier to reconstruct/recover from FE evaluations than others (e.g. digit 8).

Figure 7 illustrates the impact of the number of FE evaluations on the resemblance of reconstructed images with the input ones. In this figure, each image contains 4 lines of digits from 0 to 9. First and third lines are samples of input images. Second and fourth lines are the reconstructed images with, respectively, the minimal and maximal MSE score. Thus, second line images are the best reconstructed digits and the fourth line images the worst reconstructed ones.

We observe that for the minimal leakage model there is no difference between the reconstructed images (lines 2 and 4 are the same). These images are very close (MSE difference is under 10^{-5}) to the average of images of the same digit from the train dataset. On the other side, when 10 FE evaluations are available the reconstruction quality is quite good. We can easily see the handwritten traits and style of writing digits in the reconstructed images. Reconstructed images for other operational information leakage estimation models follow the same resemblance pattern.

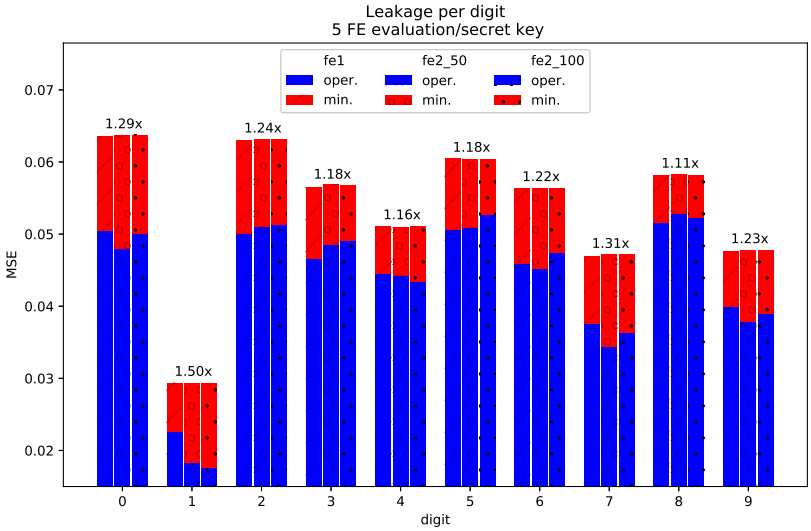
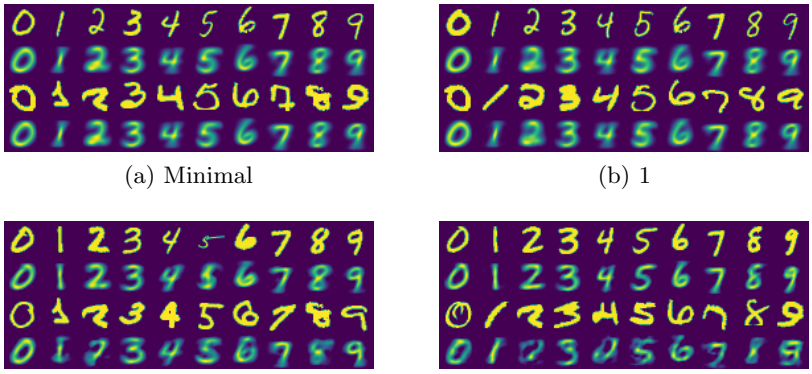


Fig. 6: MNIST – per digit information leakage estimation.



(a) Minimal (b) 1

(c) 5 (d) 10

Fig. 7: Samples of reconstructed images by the quadratic NN model fe2_100 with different numbers of FE evaluations/secret keys (b-d) and the minimal leakage estimation model (a).

Census Income dataset

Census Income dataset – accuracy The prediction accuracy obtained for the Census Income dataset is given in Figure 8. The NN models have better performance than the logistic regression ones with the same number of FE evaluations.

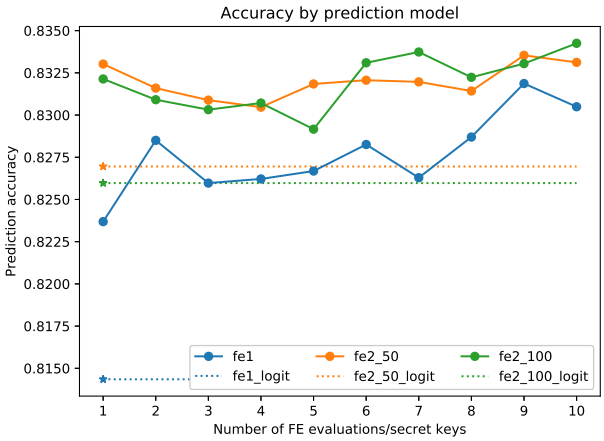


Fig. 8: Census Income – prediction accuracy.

The number of FE evaluations does not impact a lot the classification performance, although a positive trend is observed when number of FE evaluations increases. The fluctuations in the prediction accuracy we observe are due to the fact that these NN models are hard to learn. Partially because, there is no strong dependence between the loss function (binary cross-entropy) and the validation metric (prediction accuracy). For this particular dataset, using a NN model does not significantly modify the prediction accuracy ($< 2\%$) of a basic logistic model. Although, when quadratic features are used better results are obtained.

Census Income dataset – information leakage The bar plot in Figure 9 illustrates the ROC AUC score for input dataset feature leakage we study for each prediction model with one FE evaluation. We note that a predictor giving a ROC AUC score of 0.5 is equivalent to random guessing. In our context, this means that the information leakage estimated by the NN model is equal to zero. The minimal leakage ROC AUC score is never 0.5 because of dataset skewness allowing the NN to perform better than random guessing. Supposedly, the NN model uses the correlation between attacked feature and prediction model output (whether person earns over 50K$ a year) in order to better estimate the information leakage.

Minimal (blue) and operational (red) leakages are plotted on the same bar to ease the comparison. As expected, the operational information leakage is always larger than the minimal one. It means, for example, that an attacker will be able to increase its confidence in the fact that a given individual is a male.

In our experiments we have observed that information leakage estimation depends a lot on the frequency of a studied feature in the input dataset. Information leakages of features which are balanced in the input dataset are better estimated. It is a well know fact that machine learning algorithms learn bet-

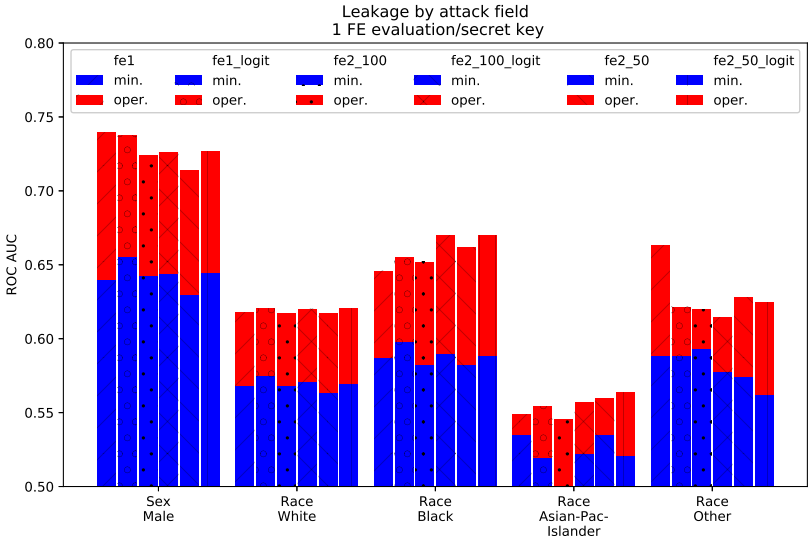


Fig. 9: Census Income – information leakage estimation.

ter on un-skewed datasets. The skewness of Census Income dataset features is given in Table 1. We note that information leakage attack results for dataset features which are heavily under-represented or over-represented) (in particular Race_Asian-Pac-Islander and Race_Other) should be interpreted with care.

Dataset feature	Frequency
Sex_Male	66.92%
Race_White	86.39%
Race_Black	9.59%
Race_Asian-Pac-Islander	3.19%
Race_Other	0.83%

Table 1: The frequencies of Census Income dataset features (number of individuals possessing this feature).

The operational information leakage for the MNIST dataset increases with the number of FE evaluations. In order to see if this pattern is also verified for the Census Income dataset we have estimated the leakage of the Sex_Male feature as a function of the number of secret keys. operational and minimal information leakages for this feature are plotted in Figure 10.

As expected, the operational information leakage increases when the number of FE evaluations increases. The leakage of NN and logistic models is very close for one FE evaluation. When the number of FE evaluations belongs to range

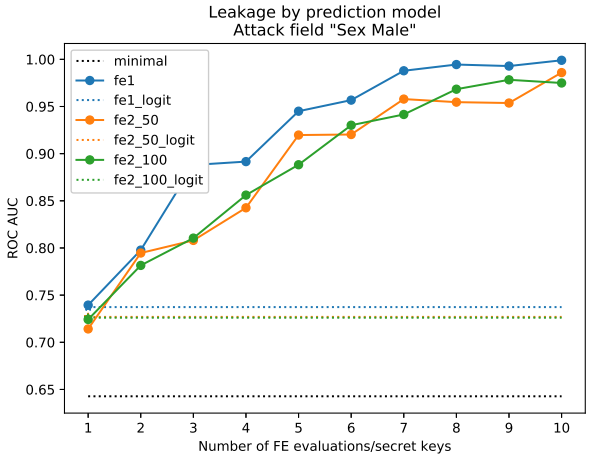


Fig. 10: Census Income – Sex_Male feature information leakage estimation.

$6 - 10$, an attacker will be able to state with very high confidence that a given individual is a male. Deploying a prediction model using 6 to 10 FE evaluations will clearly represent a real risk to individual privacy.

Information leakage on other features of the Census Income dataset are presented in the Appendix. The leakage on these fields follow same trend as the leakage on Sex_Male feature.

5 Conclusion and future works

Functional encryption schemes offer the ability to evaluate functions over encrypted inputs while granting access to the evaluation result in clear form i.e., meaning that an owner of a secret key sk_f is able to get no more than $f(m)$ from an encryption of m. According to this specific property, this cryptographic primitive seems perfectly suited for privacy-preserving classification, as it is able to keep inputs private while also allowing to perform computations over ciphertexts. However, we show in the present paper that due to the limitations on the classes of functions supported in today *practical* FE schemes, input data privacy cannot be ensured. Indeed, the current state-of-art of functional encryption only provides schemes for linear or quadratic functions evaluation, which forces us to finalize the classification process from clear-domain intermediate values. With that respect, the goal of this article was to study how much information these intermediate values reveal about the private input of a neural network-based classifier. To do so, we proposed a methodology to estimate and compare the information leakage of classification use cases and provide extensive experimental results on classification and information leakage estimations on two well-known datasets: the MNIST dataset of handwritten digits and the

Census Income dataset, showing in realistic conditions how much information an attacker can infer about the inputs that were supposed to be kept private. More precisely, concerning the MNIST dataset, we showed that an attacker is able to reconstruct input digit images which are close (in terms of MSE distance, as well as visually) to the original ones. Hence it is easier to recover the handwriting of the people who originally wrote the digits used as inputs. In the case of the Census Income dataset, our study showed that an attacker is able to gain more insights on highly private characteristics of individuals. For example, it is possible to assert with a higher confidence the gender or the ethnic origin of an individual, among other features. Finally, we showed that, using a second classifier, we are able to decrease the number of needed function encryption evaluations, when compared with the classification based on a logistic regression model proposed in [13] (which used a large number of functional encryption evaluations, at least larger than needed). This decreases at the same time the information leakage, and thus improves the privacy of the input data.

It is important to notice that beyond the privacy-preserving classification use-case discussed in the present paper, this study is relevant for other use-cases implying functional encryption. Also, this methodology of attack is not limited to linear or quadratic functional encryption schemes, neither to the public key setting. Indeed, as long as some part of the classification process is performed over clear intermediate values, our attack line can be used. Of course, the higher the degrees of the polynomials that are handled by the functional encryption scheme, the higher privacy can be reached as long as this results in smaller sets of intermediate cleartext values.

We hope that this work will be followed by other studies, helping to decide if a given neural network is acceptable or not in terms of user privacy, when it is implemented within the constraints imposed by today practical functional encryption schemes. In this direction, it would be interesting to explore other machine learning models than neural networks for the second step of the prediction and also for the information leakage estimation. Also, further studies should explore other choices for the function used to estimate and measure the leakage. In our work, we measured the information leakage by the ability to predict input data set features. We chose the gain in prediction accuracy of input dataset features for Census Income, and the MSE score between input image and reconstructed images for MNIST. In future work, we envisage to develop more refined methods to measure the information leakage. For instance, in the case of MNIST, an image resemblance metric would be more suitable than the MSE, such as the structural similarity (SSIM) index.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: IACR International Workshop on Public Key Cryptography. pp. 733–751. Springer (2015)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining, vol. 29. ACM (2000)

3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Annual Cryptology Conference. pp. 333–362. Springer (2016)
4. Asuncion, A., Newman, D.: UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (2007)
5. Ateniese, G., Felici, G., Mancini, L.V., Spognardi, A., Villani, A., Vitali, D.: Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. arXiv preprint arXiv:1306.4447 (2013)
6. Backes, M., Berrang, P., Humbert, M., Manoharan, P.: Membership privacy in microrna-based studies. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 319–330. ACM (2016)
7. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Annual International Cryptology Conference. pp. 67–98. Springer (2017)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in Cryptology – CRYPTO 2001. pp. 213–229. Springer (2001)
9. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
10. 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.: Comparison of classifier methods: a case study in handwritten digit recognition. In: International conference on pattern recognition. pp. 77–77. IEEE Computer Society Press (1994)
11. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 668–679. ACM (2015)
12. Chollet, F., et al.: Keras. `https://github.com/fchollet/keras` (2015)
13. Dufour Sans, E., Gay, R., Pointcheval, D.: Reading in the dark: Classifying encrypted digits with functional encryption. Cryptology ePrint Archive, Report 2018/206 (2018), `https://eprint.iacr.org/2018/206`
14. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1322–1333. ACM (2015)
15. Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., Ristenpart, T.: Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In: USENIX Security Symposium. pp. 17–32 (2014)
16. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on. pp. 40–49. IEEE (2013)
17. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 555–564. ACM (2013)
18. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 655–672. IEEE (2017)
19. Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J.V., Stephan, D.A., Nelson, S.F., Craig, D.W.: Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. PLoS genetics **4**(8), e1000167 (2008)

20. Hutter, M.: Universal Artificial Intelligence—Sequential Decisions Based on Algorithmic Probability. Springer (2005)
21. Hutter, M.: Universal artificial intelligence: Sequential decisions based on algorithmic probability. Springer Science & Business Media (2004)
22. Hutter, M.: On the foundations of universal sequence prediction. In: International Conference on Theory and Applications of Models of Computation. pp. 408–420. Springer (2006)
23. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Advances in Cryptology – EUROCRYPT 2008. pp. 146–162. Springer (2008)
24. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 202–207. AAAI Press (1996)
25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. vol. 86, pp. 2278–2324. IEEE (1998)
26. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST Database. http://yann.lecun.com/exdb/mnist/
27. Legg, S.: Is there an elegant universal theory of prediction? In: International Conference on Algorithmic Learning Theory. pp. 274–287. Springer (2006)
28. Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W.: Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE transactions on parallel and distributed systems **24**(1), 131–143 (2013)
29. Ligier, D., Carpov, S., Fontaine, C., Sirdey, R.: Information leakage analysis of inner-product functional encryption based data classification. In: PST'17: 15th International Conference on Privacy, Security and Trust. IEEE (2017)
30. Ligier, D., Carpov, S., Fontaine, C., Sirdey, R.: Privacy preserving data classification using inner-product functional encryption. In: ICISSP. pp. 423–430 (2017)
31. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Annual International Cryptology Conference. pp. 36–54. Springer (2000)
32. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. arXiv preprint arXiv:1812.00910 (2018)
33. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 214–231. Springer (2009)
34. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814 (2016)
35. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Theory of Cryptography Conference. pp. 422–439. Springer (2012)
36. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–473. Springer (2005)
37. Sankararaman, S., Obozinski, G., Jordan, M.I., Halperin, E.: Genomic privacy and limits of individual detection in a pool. Nature genetics **41**(9), 965 (2009)
38. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: Security and Privacy (SP), 2017 IEEE Symposium on. pp. 3–18. IEEE (2017)

39. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: USENIX Security Symposium. pp. 601–618 (2016)
40. Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. ACM Sigmod Record **33**(1), 50–57 (2004)
41. Waters, B.: Efficient identity-based encryption without random oracles. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 114–127. Springer (2005)
42. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: International Workshop on Public Key Cryptography. pp. 53–70. Springer (2011)

A Information leakage on Census Income dataset

In this appendix are illustrated the operational and the minimal information leakage on Race_White, Race_Black, Race_Asian-Pac-Islander and Race_Other fields of the Census Income dataset.

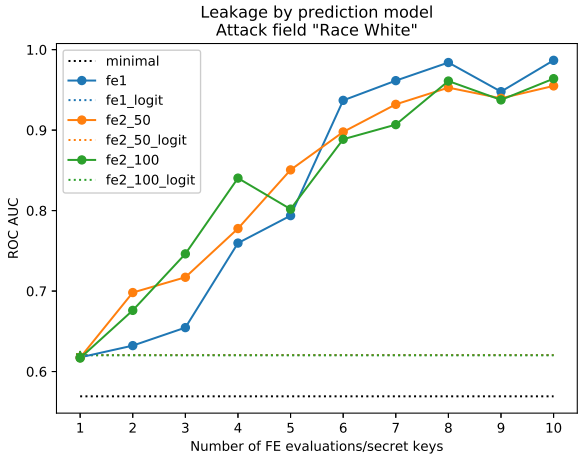


Fig. 11: Census Income – Race_White feature information leakage estimation.

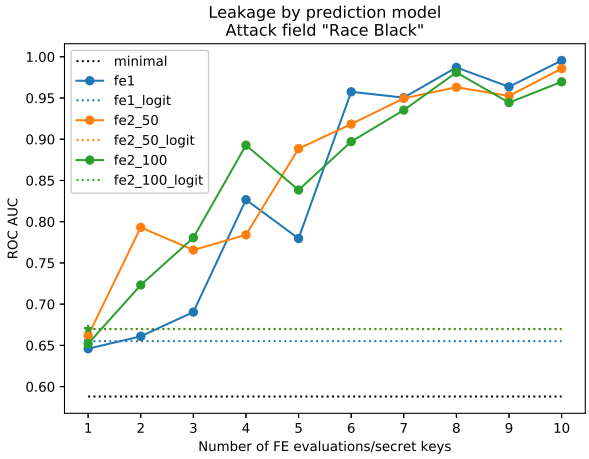


Fig. 12: Census Income – Race_Black feature information leakage estimation.

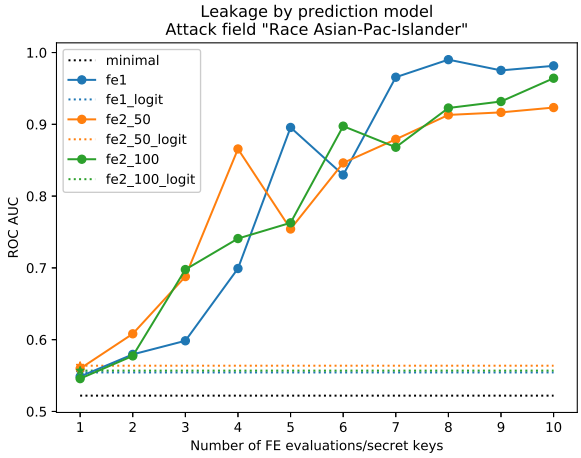


Fig. 13: Census Income – Race_Asian-Pac-Islander feature information leakage estimation.

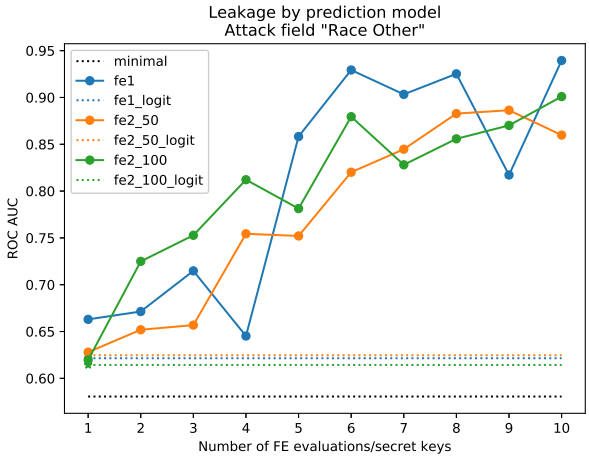


Fig. 14: Census Income – Race_Other feature information leakage estimation.