

Privacy-Preserving Credit Scoring via Functional Encryption

Lorenzo Andolfo[1], Luigi Coppolino[1], Salvatore D'Antonio[1], Giovanni Mazzeo[1(✉)], Luigi Romano[1], Matthew Ficke[2], Arne Hollum[2], and Darshan Vaydia[2]

[1] Department of Engineering, University of Naples 'Parthenope', Naples, Italy
{lorenzo.andolfo,luigi.coppolino,salvatore.dantonio,giovanni.mazzeo,luigi.romano}@uniparthenope.it

[2] X-Margin Inc., 141 Nantucket Cove, San Rafael, CA 94901, USA
{matthew,arne,darshan}@xmargin.io

Abstract. The majority of financial organizations managing confidential data are aware of security threats and leverage widely accepted solutions (e.g., storage encryption, transport-level encryption, intrusion detection systems) to prevent or detect attacks. Yet these hardening measures do little to face even worse threats posed on *data-in-use*. Solutions such as Homomorphic Encryption (HE) and hardware-assisted Trusted Execution Environment (TEE) are nowadays among the preferred approaches for mitigating this type of threats. However, given the high-performance overhead of HE, financial institutions —whose processing rate requirements are stringent— are more oriented towards TEE-based solutions. The *X-Margin Inc.* company, for example, offers secure financial computations by combining the Intel SGX TEE technology and HE-based *Zero-Knowledge Proofs*, which shield customers' *data-in-use* even against *malicious insiders*, i.e., users having privileged access to the system. Despite such a solution offers strong security guarantees, it is constrained by having to trust Intel and by the SGX hardware extension availability. In this paper, we evaluate a new frontier for *X-Margin*, i.e., performing privacy-preserving credit risk scoring via an emerging cryptographic scheme: *Functional Encryption (FE)*, which allows a user to only learn a function of the encrypted data. We describe how the *X-Margin* application can benefit from this innovative approach and —most importantly— evaluate its performance impact.

Keywords: Credit scoring · Data privacy · Functional encryption · Machine learning

1 Introduction

Data confidentiality has significant relevance in the financial field. The disclosure of sensitive information such as credit scoring or exchange apikeys, can lead to, e.g., learn about users' credit risk —and therefore decide whether to lend money— or ultimately money theft. A data breach inevitably leads to serious consequences

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12956, pp. 31–43, 2021.
https://doi.org/10.1007/978-3-030-87010-2_3

affecting the reputation of the financial institution. The importance of cybersecurity is even more crucial for the cryptocurrency industry, whose current valuation —1.8$ trillion in 2021— attracts a rapidly increasing number of hackers. Only in the last two years, four large hacks occurred in the digital currency space[1].

Companies are therefore pushing on equipping their platforms with state-of-the-art security mechanisms. There are widely accepted solutions such as storage encryption, transport-level encryption, intrusion detection systems, firewalls, IP white-listing, which are used and help in preventing common cyber-attacks. However, there are still open issues —especially in terms of *data-in-use* protection against privileged attackers (e.g., a *malicious insider*)— that requires more advanced approaches. Companies aim for methods where the customer does not need to trust them. There are technologies and cryptography schemes that can be leveraged for this purpose. In this regard, the adoption of hardware-assisted Trusted Execution Environments (TEE) (e.g., Intel SGX) [15] and Homomorphic Encryption (HE) [12] is making its way into the industrial community to ensure this kind of protection. HE is extremely powerful since it enables arbitrary computations on ciphered data and allows the user to trust no one. A TEE such as SGX provides also strong security guarantees enabling the execution of sensitive code in an isolated and measurable area of CPUs, with the only limitation that the user needs to trust the hardware manufacturers such as Intel and that the application's state is sealed in the hardware [6]. Unfortunately, the exclusive use of HE is not doable since it is affected by a non-negligible execution time overhead, and by a large cipher text expansion. Moreover, a major problem of HE is the so-called unverifiable conditional issue, which forces a program running on a third-party host, and processing homomorphically encrypted data, to request that a client decrypts intermediate functional results to proceed further in the execution [7]. This introduces additional synchronization points and performance overhead, and increases the risk of information disclosure (e.g. via side channel analysis of the program workflow). For this reason, the trend for the majority of financial companies is to prefer TEEs.

The *X-Margin Inc.*[2] company, instead, uses both security mechanisms. It offers secure credit risk scoring by shielding computations in Intel SGX enclaves and performs HE-based *Zero-Knowledge Proofs* to offer cryptographic Proof of Security, Computation and Encrypted Input. In this paper, we explore an innovative approach for *X-Margin* which would allow to replace the use of SGX, thus removing Intel from the chain-of-trust and being hardware agnostic: *Functional Encryption* (FE) [5]. This is a cryptographic scheme similar to HE that does both evaluation and decryption of the *result of a function* at the same time, without leaking the private key for data decryption and without leaking information about the plaintexts. This means that —unlike HE— there is no need anymore of doing synchronizations between the client and the computation entity for intermediate results. In a FE model, there are special "evaluation keys" that only allow the functional evaluation of ciphered data. An additional advantage given

[1] https://www.investopedia.com/news/largest-cryptocurrency-hacks-so-far-year/.
[2] https://xmargin.io/.

by the FE scheme is that it enables *Attribute-based encryption* (ABE) schemes, where the encrypted data is linked with a set of attributes and secret keys along with certain policies that allow to control which ciphertexts can be decrypted depending on the possessed attributes. Unlike the classical encryption systems, where "*access to the encrypted data is all or nothing, one can either decrypt and read the entire plain or one learns nothing at all about the plain other than its length*", FE tries to change this paradigm allowing a more fine-grained access to encrypted data and improving its flexibility. We describe how FE can be leveraged for *X-Margin* financial computations and —most importantly— evaluate the performance impact of this innovative scheme. We overview a solution that computes credit risk from an encrypted borrower's data with the FE, thus preserving data privacy in a cloud environment even from the company itself. As a first step, we implemented an ad-hoc neural network for the purpose of credit risk scoring, which acts as a binary classifier saying whether a loan defaults. The training is conducted on a plain dataset. The resulting weight configuration downstream the training step is saved for the second step. We built the functional encryption scheme using the weights configuration obtained during the first step. We used a symmetric private key quadratic multivariate polynomial scheme called SGP [8]. As from the definition of functional encryption, the result of decryption is not the record itself, but the evaluation of a function of it, that is represented by the credit risk associated to the borrower. At the end, we computed the credit risk via two FE functions: one that computes the score to say whether the borrower defaults, and the other one that computes the score whether the borrower doesn't default. Then, the final result is normalized among the scores in order to provide the probability of default. Results from experimental evaluation show that the FE entails a non-negligible overhead on the classification time. Using 20 attributes, the application took 17.3 s to compute the score with 200 borrowers in the system. It is important to notice that the scoring classification job is highly parallelizable, thus performance can be improved using multiple nodes.

The rest of this paper is organized as follows: In Sect. 2, we provide a background on credit scoring and on the FE scheme. Afterwards, in Sect. 3, the X-Margin case study is presented. In Sect. 4, the architecture of the proposed solution is discussed. Section 5 presents results from the experimental evaluation. Finally, Sect. 6 concludes the document.

2 Background

In this section, we provide a background on the two main elements covered in this work, i.e., credit scoring and functional encryption.

2.1 Principles of Credit Scoring

Credit scoring is a method that is historically used by banks to estimate the risk of lending money to an individual. It determines the ability of a person to

repay the debts. The higher is the credit score, the higher is the probability of obtaining a loan with low interest rates. On the other hand, people with a lower credit score must pay higher interest rates on their loans. The process of giving a loan is influenced by many factors: characteristics of borrowers in terms of who they are, their economic situation, the amount of the requested loan, its purpose, and the collateral type. The risk is estimated using elements of quantitative and qualitative analysis. The quantitative analysis takes into account an assessment of the financial standing of customers. It could also consider cash flow analysis of customers' accounts together with their credit history. While the qualitative assessment includes other information such as the education, employment, industry in which they operate, or the way of keeping accounts.

2.2 Fundamentals of Functional Encryption

Functional encryption (FE) is relatively a new encryption technique, which was introduced by Amit Sahai and Brent Water in 2008 [16]. FE enables selective access control of sensitive data d basing on specific functions $f(d)$. In an FE scheme, a decryption key $sfKey(f)$ is associated with the function f. Therefore, the decryption of an encrypted data d through $sfKey(f)$, provides the function evaluation $f(d)$, and nothing more about d. Concerning differences between FE and FHE, from one side they result similar, because once the message has been encrypted, both evaluate a function of the message. However, FHE suffers from to the additional step of decrypting the evaluation of the function, since the result of its computation is still an encrypted result. With FE instead, the result of the decryption already represents the evaluation of the function. FE has been successfully applied to cloud environments and to obfuscation mechanisms [9], that can be useful for example to protect intellectual property. Unfortunately, up to now, industries seldom adopt such advanced cryptography technologies: the majority of cryptography technologies used by industries were developed in early 2000s [14]. Only recently, thanks to the ambitious project FENTEC (Functional ENcryption TEChnologies)[3], there has been a tentative to propose FE solutions for privacy preserving in a wide range of sectors from clinical data[4], to public transportation[5].

There are several types of schemes that can be assimilated to the FE, these are: attribute-based encryption (ABE) [3,10,13], identity-based encryption (IBE) [4], the ones that implement inner product functions [1,2] and nonlinear (at most quadratic) polynomials. In this work, the focus is on a symmetric private key FE scheme called SGP, that implements a two dimensional quadratic polynomial function.

$$f(x,y) : \sum_{i=1,j=1}^{m,n} f_{i,j} x_i y_j = \underset{1\times m}{X^T} \cdot \underset{m\times n}{\mathrm{F}} \cdot \underset{n\times 1}{Y} \tag{1}$$

[3] https://fentec.eu/.
[4] https://github.com/fentec-project/Selective-Access-to-Clinical-Data.
[5] https://github.com/fentec-project/FE-anonymous-heatmap.

Here, X and Y represents the vectors we want to encrypt, while F is a custom matrix. The result of decryption in this case will be the evaluation of the function above.

Regarding security, it provides an adaptive security under chosen-plaintext attacks (IND-CPA security) which is based on bilinear pairings, with the following primitives:

1. $msk \leftarrow GenerateMasterKey()$: generates a secret key for the SGP scheme.
2. $c \leftarrow Encrypt(X, Y, msk)$: encrypts input vectors X and Y with the secret key msk and returns the appropriate ciphertext c.
3. $feKey \leftarrow DeriveKey(msk, F)$: given in input a custom matrix F and the secret key msk, derives the FE key $feKey$.
4. $X^T \cdot \mathrm{F} \cdot Y \leftarrow Decrypt(c, feKey, F)$: given in input the ciphertext c the FE key $feKey$ and the custom matrix F, gets the evaluation of $f(x, y)$ that is $X^T \cdot \mathrm{F} \cdot Y$.

3 The X-Margin Case Study

X-Margin Inc. provides privacy-preserving credit scoring in the crypto-currency credit market. With over $100B of crypto-collateral being used to generate over $1.25B of interest on a quarterly basis, credit is one of the most rapidly growing sectors of the emerging crypto-currency finance ecosystem. *X-Margin* allows borrowers to supply lenders with real-time portfolio risk metrics, while preserving the privacy of trades, positions, and other sensitive information. Borrowers benefit from improved lending terms, as they can display their risk in real-time and assure lenders they are trading responsibly. Lenders benefit from increased visibility and real-time information. *X-Margin* calculates a variety of Risk Metrics on each user's portfolio, including Equity, Balance, Margin Usage, Maximum Loss (SPAN or VaR calculations), Aggregate absolute Delta, and the company uses Zero Knowledge technology to ensure that the credit scoring system is functioning in an unbiased and privacy preserving way, and as the users expect.

X-Margin leverages TEE technologies, along with HE-based cryptographic Proof of Security, Computation and Encrypted Input, to ensure the privacy of users' sensitive data and guarantee risk analysis. Intel Software Guard Extensions (SGX), is the enabling TEE technology that is currently used to protect and attest the sensitive computations. SGX is a recent extension of modern CPUs, which ensures confidentiality even against super-privileged users running as *root*. *X-Margin* uses HE-based schemes for Proof of Computation including a Proof of Encrypted Input. Together, the proofs demonstrate that a given function being executed within the SGX enclave corresponds exactly to a symbolic representation stored in the server, outside the enclave —the untrusted 'host'— and that the data on which the function operates is received encrypted from an external source. *X-Margin* proves that the functions the system is instructed to run are run in the enclave, without revealing the functions themselves. The Proof of Computation & Encrypt ed Input uses the Enhanced Skyline protocol, which is a

proprietary protocol developed by *X-Margin*'s team of cryptographers. Figure 1 shows the current architecture of the *X-Margin* solution. There is a SGX-based Key Management System (KMS) that holds users' apikeys, and a CScore unit where the actual private computation takes place, protected by the SGX secure enclave, and proven by cryptographic proofs. It queries spot/derivative exchanges and provide risk score metrics to a Time Series Database (TSDB).

The goal of the company is to further improve their solution by raising the concept of *trust* to a higher level. In the current situation, in fact, its customers must trust the hardware manufacturer, i.e., *Intel*. It would be ideal for the company to remove also Intel from the *chain-of-trust*. Furthermore, in the current status, the credit scoring application is strictly bounded to a particular hardware where the SGX ISA extension —and the related motherboard support— is available. This also entails migration constraints because the application's state is sealed in the hardware. *X-Margin*'s R&D is therefore exploring new innovative security techniques to cope with the mentioned limitations.

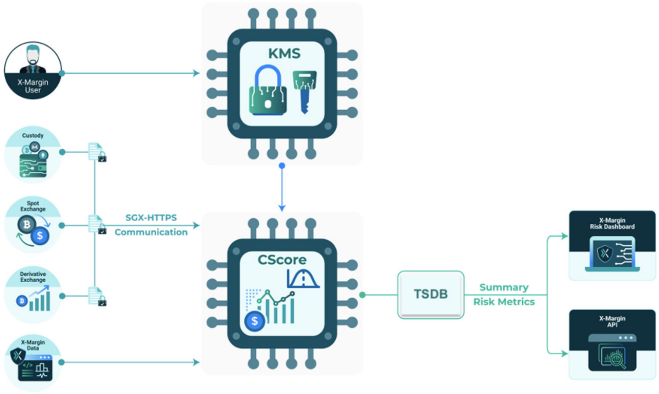

Fig. 1. The *X-Margin* architecture

4 The FE-Based Credit Scoring Solution

The functional encryption paradigm represents an opportunity to overcome limitations posed by the SGX-based credit scoring solution. It ensures security properties similar to HE but with the advantage that it allows to learn specific functions of the encrypted data. Overall, the FE-based credit scoring works as follows. In a first phase, the neural network is trained leveraging plain datasets in *X-Margin*'s premises. This is acceptable since the training process does not use any customers' sensitive data but it is conducted on anonymous data. At the end of this learning phase, coefficients needed to compute the credit risk of a given borrower are obtained. These are then used to generate keys. In this regard, according to the FE scheme principles, a trusted *key authority* receives the coefficients and generates two types of keys, i.e.: public/private keys for the encryption/decryption of data,

and operational keys needed to perform the functional computation on encrypted data, which can only be used to decrypt the result of the computation but not the data itself. At the end of this phase, the *X-Margin* credit scoring application is finally configured. From now on, the customer will encrypt the data along with the public keys received by the authority. The ciphered information is then sent to the untrusted *X-Margin* premises running on cloud. Here, the credit risk associated to the borrower is computed on ciphered data. It is important to notice that the computation of the credit risk associated to the borrower does not include any confidential borrower's data. Only the result of this computation (i.e., the scoring evaluation) will be accessible to *X-Margin*.

4.1 Architecture

Figure 2 shows the overall architecture of the FE-based credit scoring solution, together with the steps performed during normal functioning.

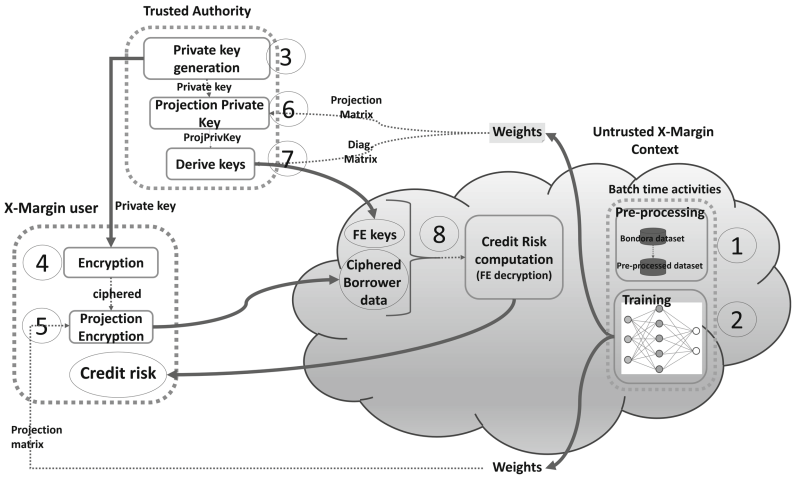

Fig. 2. The credit risk score architecture

1. **Data pre-processing**: At startup, *X-Margin* performs a pre-processing activity on the data-set that will be used during the training phase. Modifications to the dataset are needed to adapt the subsequent computation to the FE requirements, and also to improve the accuracy of the neural network model, such as:
   (a) Removes columns and rows containing null values, discard columns with a high percentage of null values, as well as rows with null values inside.
   (b) Removes outliers.
   (c) Adapt timestamps
   (d) Scales and normalizes columns.

(e) Replaces object columns with dummy variables and one-hot encodes the output.
(f) Transform data in integer form in order to ensure compatibility with the FE scheme properties,

2. **Neural Network Training:** In this phase, the training phase takes place using clear data. This is acceptable given that the leveraged dataset contains anonymous entries. An ad-hoc neural network was defined. We leveraged and adapted the *Adam optimization algorithm*, with a learning rate of 10^{-4} and a batch size of 32. Training epochs have been set to 50. The neural network has a single hidden layer and its activation layer is represented by a *square element-wise*, instead of *sigmoid*. This choice will be better explained in the rest of this section.
   Let $\underset{1\times n}{X}$ be our input vector, that is the borrower's record, $\underset{n\times d}{Pr}$ be the weight matrix of the first layer and $\underset{d\times l}{D}$ be the weight matrix of the hidden layer. In our case, $n = 130$, $d = 20$ and $l = 2$. The output of the neural network is:
   $$prediction : \text{squareElemWise}(\underset{1\times n}{X} \cdot \underset{n\times d}{Pr}) \cdot \underset{d\times l}{D} = \underset{1\times l}{DefaultScores}.$$
   The result of the classification is represented by a pair of scores, since $l = 2$, which are mutually exclusive and that are not normalized. In order to get the final result in terms of probability, a softmax function has been applied in the untrusted context side.
   At the end of the training step, Pr and D are saved and used afterwards for the encryption task and for deriving FE keys in the trusted context.
3. **Secret Key Generation.** The trusted authority generates a secret key for the symmetric SGP scheme according to the FE scheme, that is:
   $$msk \leftarrow GenerateMasterKey() \tag{2}$$
   It then sends msk to the *X-Margin* user.
4. **Data Encryption.** The X-margin user uses msk to encrypt the borrower's data X and returns the appropriate ciphertext that is:
   $$c \leftarrow Encrypt(X, X, msk) \tag{3}$$
   Here, respect to the classical SGP scheme, we have replaced the parameter Y by X.
5. **Encryption Projection.** The *X-Margin* user takes the encryption c and weight matrix Pr obtained at step 2 to produce a projection of c, that is:
   $$ProjC \leftarrow projectEncryption(c, Pr) \tag{4}$$
   $ProjC$ represents the encryption of $(Pr \cdot X)$, that is sent to the untrusted context.
6. **Secret Key Projection.** The trusted authority takes the secret key msk obtained at step 3 and the weight matrix Pr obtained at step 2 to get a projection of the secret key:
   $$ProjSecKey \leftarrow projectSecKey(msk, Pr) \tag{5}$$
   $ProjSecKey$ represents the secret key for encryption of $(Pr \cdot X)$.

7. **FE Key Generation.** The trusted authority provides also the FE keys to the untrusted party to compute the credit risk of the borrower. There are two FE keys: one used to decrypt the score of default and one used to decrypt the score of not default of the borrower. Therefore, each FE key is used to decrypt a different output i of the neural network:

$$feKey_i \leftarrow DeriveKey(ProjSecKey, Diag_i) \forall i \in l \tag{6}$$

   $DeriveKey$ accepts as parameter the secret key $ProjSecKey$, obtained at the previous step, and the diagonal matrix $Diag_i$, that is the diagonalized version of D_i, that represents the $i-th$ row of the matrix D. It should be noted that D_i is the vector containing the weights connected to a single label. Then, the trusted authority sends the obtained FE keys to the untrusted context.
8. **Credit Score Computation.** The untrusted context, computes the credit risk scores of the borrower through the FE key provided by the trusted authority at the previous step. This means that for each label i it evaluates the function of $(Pr \cdot X)$, that is $(Pr \cdot X)^T \cdot Diag_i \cdot (Pr \cdot X)$. In order to do that, it exploits the following decryption primitive of the SGP scheme:

$$\begin{aligned} defaultScore_i &= (Pr \cdot X)^T \cdot Diag_i \cdot (Pr \cdot X) \\ &\leftarrow Decrypt(ProjC, feKey_i, Diag_i) \forall i \in l \end{aligned} \tag{7}$$

   For each label i, $Decrypt$ accepts $ProjC$, that as we saw represents the encryption of $(Pr \cdot X)$ sent by the *X-Margin* user, the derived FE key $feKey_i$ and the diagonal matrix $Diag_i$ sent by the trusted authority.
   Since $l = 2$, there are two functions: one that computes the score that the borrower defaults and one that he/she doesn't default. Such scores can be seen as mutually exclusive, therefore we can apply the soft max function on them in order to normalize the results and obtain the related probabilities.

$$\underset{1 \times l}{DefaultProbabilities} = \text{softMax}(\underset{1 \times l}{Scores}) \tag{8}$$

   Finally, the untrusted context sends back to the *X-Margin* user the score obtained at 8.

4.2 Details on the FE Scheme for Credit Risk Scoring

The aim of this section is to draw connections between the neural network output function provided at 2 and the FE function at 1, so as to understand what is the information needed to build the FE scheme.

Given the FE function at 1, we can transform it from a multivariate to a single variable function.

$$f(k) : \underset{1 \times d}{K^T} \cdot \underset{d \times d}{\mathrm{F}} \cdot \underset{d \times 1}{K} \tag{9}$$

From the other side, we can see the neural network function at 2 as the result obtained for each single label:

$$defaultScore_i : \text{squareElemWise}(\underset{1\times n}{X} \cdot \underset{n\times d}{Pr}) \cdot \underset{d\times 1}{D_i} \forall i \in l. \tag{10}$$

where D_i is a vector containing the weights connected to a single label i, X is the borrower's vector and Pr is the weight matrix associated to the first layer. Applying the following manipulation to each function i obtained in 10, it is possible to get the FE function at 9:

$$\begin{aligned} defaultScore_i : \text{squareElemWise}(X \cdot Pr) \cdot D_i \\ = ((Pr)^T \cdot (X)^T)^T \cdot Diag_i \cdot ((Pr)^T \cdot (X)^T) \\ = K^T \cdot Diag_i \cdot K \\ \forall i \in l. \end{aligned} \tag{11}$$

Here, we have transformed the first matrix product, switching the operands and applying the transpose on them. Indeed, we have removed the square element-wise operation on it and, in order to keep the equality we have replaced D_i with its diagonalized version $Diag_i$ and we have multiplied by the product $((Pr)^T \cdot (X)^T)$. Finally, we have replaced the products with the obtained vector K in order to highlight the equivalence with the 9. The custom matrix F is the diagonal matrix $Diag_i$ that is different for each label, while the other parts of the function don't change.

We have shown that such neural network architecture computes the FE function 9, for each label l. Each function has, as variable, the matrix product $((Pr)^T \cdot (X)^T)$ named K, and produces as output respectively the score that a borrower defaults and not. From now on $((Pr)^T \cdot (X)^T)$ will be indicated without the transpose operations inside, in order to not burden the discussion.

5 Experimental Performance Evaluation

We conducted an experimental evaluation of the privacy-preserving scoring solution. The goal was to estimate the performance overhead given by the adoption of the FE-based processing. More precisely, our focus was on the execution time needed to encrypt borrowers' data, and to compute the credit risk score, i.e., the evaluation of the functional decryption occurring on *X-Margin* premises. These tests did not take into account the classification algorithm accuracy since our main goal is to focus on the sole performance.

We have evaluated our model on a Windows 10 machine equipped with an AMD Ryzen 3600X with 6-Core Processor at 3.80 GHz and 16 GB of RAM. Execution times (in ms) are the average of 5 repeated tests.

As a first experiment, we evaluated the encryption phase (i.e., steps 4 & 5) by increasing the number of users in the system up to 1000 borrowers, and also varying the number of borrowers' attributes (in the range [5,25]).

Figure 3 shows the outcomes of the evaluation. It can be noticed that the encryption is not significantly time consuming. The encryption of one borrower in what we defined the worst situation —i.e., characterized by 50 attributes— takes on average 52.3 ms, while in the best situation —i.e., 5 attributes— it takes 14.3 ms. The execution time increases linearly with the number of borrowers.

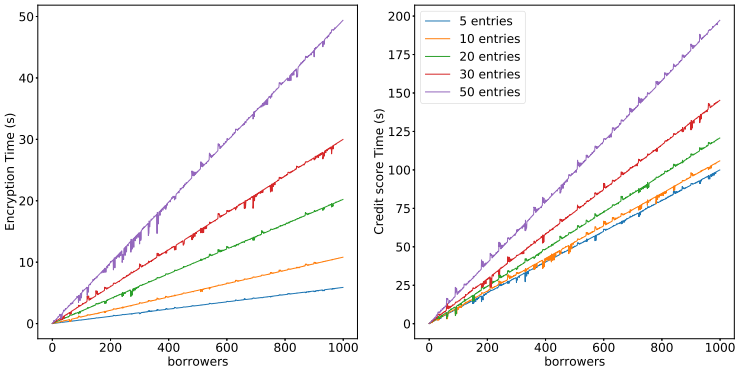

Fig. 3. Encryption and credit scoring performance

The second evaluation we made refers to the FE-based scoring computation time, which is critical for the actual usability of the proposed solution. It is important to notice that this operation includes both the actual functional computation and the final result decryption. Figure 3 shows that the overhead is significant, reaching in the worst condition —i.e., 50 attributes and 1000 users— a computation time of 170 s. It must be noticed that in real situations, the number of attributes is lower, in the order of 20. Nevertheless, performance are still not acceptable for the adoption in real contexts such as the *X-Margin* one. However, the type of job we implemented with FE is highly parallelizable, and the deployment on multiple nodes could make the adoption of the FE-based credit scoring solution acceptable.

6 Conclusion

In this paper, we presented an innovative solution for privacy-preserving credit risk scoring, which leverages the emerging *Functional Encryption* cryptography that allows to learn the result of a specific function using only encrypted data. We built a credit risk model using the quadratic polynomial FE symmetric scheme SGP for the X-Margin Inc. company, whose case study was used to validate the effectiveness of the proposed approach. Results from the experimental campaign show that the solution performs well under certain conditions, but the overhead needs to be properly managed in case of a real production deployment. We discovered that with a limited number of borrowers' attributes (e.g. < 20) the

encryption time and the classification time are acceptable. Future developments of the current solution will be focused on improving performance by evaluating new public key quadratic schemes (e.g., [11]), and setting up a multi-node parallel architecture.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 597–627. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_20
3. Agrawal, S., Chase, M.: Fame: fast attribute-based message encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp. 665–682. Association for Computing Machinery (2017). https://doi.org/10.1145/3133956.3134014
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_14
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
6. Campanile, F., et al.: Cloudifying critical applications: a use case from the power grid domain. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 363–370 (2017). https://doi.org/10.1109/PDP.2017.50
7. Coppolino, L., D'Antonio, S., Formicola, V., Mazzeo, G., Romano, L.: VISE: combining intel SGX and homomorphic encryption for cloud industrial control systems. IEEE Trans. Comput. **70**(5), 711–724 (2020). https://doi.org/10.1109/TC.2020.2995638
8. 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
9. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016)
10. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
11. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 95–120. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_4

12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC 2009, New York, NY, USA, pp. 169–178. ACM (2009). https://doi.org/10.1145/1536414.1536440
13. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, New York, NY, USA, pp. 89–98. Association for Computing Machinery (2006). https://doi.org/10.1145/1180405.1180418
14. Green, M.: A few thoughts on cryptographic engineering. https://blog.cryptographyengineering.com/2017/07/02/beyond-public-key-encryption/ (2017)
15. McKeen, F., et al.: Innovative instructions and software model for isolated execution. In: Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy, New York, NY, USA, HASP 2013, pp. 10:1–10:1. ACM (2013). https://doi.org/10.1145/2487726.2488368
16. Sahai, A., Waters, B.: Slides on functional encryption. http://www.cs.utexas.edu/~bwaters/presentations/files/function.ppt (2008)