Research Repository UCD

Title	Lightweight Privacy-Preserving Data Classification
Authors(s)	Tran, Ngoc, Le-Khac, Nhien-An, Kechadi, Tahar
Publication date	2020-10
Publication information	Tran, Ngoc, Nhien-An Le-Khac, and Tahar Kechadi. "Lightweight Privacy-Preserving Data Classification." Elsevier, October 2020. https://doi.org/10.1016/j.cose.2020.101835.
Publisher	Elsevier
Item record/more information	http://hdl.handle.net/10197/25909
Publisher's statement	This is the author's version of a work that was accepted for publication in Science Direct Computers & Security. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Science Direct Computers & Security (97,(2020)) https://doi.org/10.1016/j.cose.2020.101835Get
Publisher's version (DOI)	10.1016/j.cose.2020.101835

Downloaded 2026-05-01 23:36:51

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/341428754

Lightweight Privacy-Preserving Data Classification

Article *in* Computers & Security · May 2020

DOI: 10.1016/j.cose.2020.101835

CITATIONS
2

READS
39

3 authors, including:

Nhien-An Le-Khac
University College Dublin
244 PUBLICATIONS **1,623** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

WhatsApp interception View project

Adversarial Deep Learning and XAI for Cyber Security View project

All content following this page was uploaded by Nhien-An Le-Khac on 16 March 2021.

The user has requested enhancement of the downloaded file.

Lightweight Privacy-Preserving Data Classification

Ngoc Hong Tran[a,b], Nhien-An Le-Khac[b] and Mohand Tahar Kechadi[a,b]

[a]The Insight Centre for Data Analytics, University College Dublin
[b]School of Computer Science, University College Dublin
Belfield, Dublin, Ireland.

ARTICLE INFO

Keywords:
Privacy Preserving Data Mining, Elliptic Curve Cryptography,
Oblivious Transfer, Garbled Circuit,
Security, Data Protection

ABSTRACT

Internal attacks are of a huge concern, because they are usually delicately masqueraded under harmless-looking activities, which are very difficult to detect. Machine learning techniques have been successfully applied to identify insider threats. However, they may violate user privacy since they can legally access user's sensitive information. To preserve user privacy, encryption algorithms have been lately exploited as a powerful tool, to hide private data in a multiple-party collaboration. A combination of encryption and data mining techniques raises high computational complexity. Hence, in order to improve the system's performance while securing both user's private data and the classifier, we propose a new secure data analysis protocol, namely SmartClass, by adopting the garbled circuit technique to speed-up the system performance. We developed an efficient encryption step that exploits the additive homomorphism and best properties of the binary Elliptic Curve Cryptography (ECC) algorithm, while keeping the protocol highly secure. We implemented the proposed system and study its effectiveness. Experimental results that show the proposed approach is very promising.

1. Introduction

Internal attacks are constantly growing and they leave severe damage, as the attackers have already privileged accesses to the internal system[1]. IBM X-Force Cyber-Security Intelligence Index presented a high-level overview of the major attacks to businesses worldwide in 2015[2]. They reported that 40% of those threats were external and 60% were internal. An insider threat is usually conducted by two types of users: malicious users that deliberately trigger attacks and users that accidentally cause data breaches and leaks. However, the outcome is the same, as these may lead to breach of confidential information, robbery of important information or intellectual property, the outage of computer systems, etc. According to Tripwire[3], 53% of companies estimate remediation costs of $\in 100K$ and more, with 12% estimating a cost of more than million Euros. 74% of companies feel that they are vulnerable to insider threats, with 7% reporting extreme vulnerability.

To combat the internal attackers, each organisation implements multiple layers of security to guarantee a certain level of defence against attacks. One of the key tasks of these layers is to detect suspicious users based on their malicious behaviours. Recently, data mining techniques, mainly classification techniques, have been proven to be very efficient in detecting anomalous behaviours [1, 2] of both internal and external attackers [3, 4]. Usually the classifier is installed on a server and it receives sessions of data from client machines. As the session data is very sensitive and private, it has to be secured during the transfer. In other words, both server and client need to collaborate in a secret way to enforce the classification process. This is part of privacy-preserving data mining [5] [6].

Over the past decade, many privacy-preserving data mining techniques have been proposed. Most of them focus on how to obfuscate personal information directly in the dataset, by adding some random noise into the data [7][8], or pseudonymising the data (i.e., replacing personal data with artificial values [9] [10]), or anonymising the data (i.e., removing the association between the data and the subjects it represents) [11]. Even though these techniques can hide sensitive information from potential adversaries but still leave a high possibility of re-identification or related attacks [12]. Hence cryptography-based methods [13] were developed to guarantee a stronger security level. However, these methods suffer from high computational complexity. To deal with computational complexity and user privacy issues, we propose a lightweight privacy-aware classification model to detect anomalous behaviours, while keeping the data secure and reducing the computational cost and data payload.

The remainder of the paper is organised as follows. Section 2 discusses the related works on privacy-preserving data mining techniques along with their drawbacks and trade offs. The threat description and the privacy requirements, including essential components of the proposed security system and design, are presented in Section 3. Section 4 describes the proposed protection model, and the security approaches. The base operator design of garbled circuits are presented in Section 5. The circuit driven security design is described in Section 6. Section 7 explains in details the proposed security aware SmartClass protocol. Section 8 gives a formal security analysis. Experimental results are reported in Section 9. We conclude and discuss some future directions in Section 10.

ngoc.tran@insight-centre.org (N.H. Tran); an.lekhac@ucd.ie (N. Le-Khac); tahar.kechadi@ucd.ie (M.T. Kechadi)
ORCID(s): 0000-0001-6369-5338 (N.H. Tran); 0000-0002-0176-6281 (M.T. Kechadi)

[1]S. Malik. Network Security Principles and Practices (CCIE Professional Development), Cisco Press, 2002.

[2]The IBM X-Force 2016 Cyber-Security Intelligence Index.

[3]Tripwire, Insider Threats as the Main Security Threat, 2017 https://www.tripwire.com/state-of-security/security-data-protection/insider-threats-main-security-threat-2017/

Table 1
Recent Works with Different Approaches: Drawbacks and Advantages.

Security	Techniques	Related work	Drawbacks	Advantages
Low	Pseudonymisation	[5]	Re-identification attack	Simple, typical
Medium	Randomisation	[14, 15, 7]	Loss of individuals' information	Simple and useful
Average	Anonymisation	[16, 11, 17, 18]	Loss of information Link-ability attack	Attractive
High	Cryptography	[19]: AES, MD5, SHA-1,2 [20]: SHA-1	High computational cost More complex	Strongly secure No information loss

2. Related work

During the data mining process, usually the user has a direct access to data records in the dataset. This may disclose the data privacy in two ways: 1) access to personal data records and expose its content to unauthorised parties, 2) the returned results may violate the privacy by revealing personal identity and/or personal information. Such risks are more likely to happen since the user data are not encrypted or protected in the storage or through the network transfer. To cope with the above privacy issues, two main categories of solutions were proposed: *at-rest methods* which obfuscate sensitive data in the storage system, and *in-flight methods* which hide sensitive data records as they are exchanged among different parties (see Table 1).

At-Rest Methods: These methods do not allow direct access to the plain data, rather they obfuscate the data in the data store. Some of them encrypt directly the entire dataset. *Pseudonymisation* is a simple technique for protecting the user identity [5]. However, it is not strong enough, as there is a high possibility of re-identifying users [9]. *Randomisation* is a technique that consists of adding noise to various attributes of records that carry sensitive information [14] [15]. There are straightforward and useful techniques for keeping the user data secret. However, they suffer from loss of individuals' information, because they cannot guarantee the recovery of the original data from the distorted data. Furthermore, they are not adequate for the datasets containing several attributes.

Anonymisation is a good approach for protecting sensitive or private data and has attracted much attention from the research community. [11] presented a review on data anonymisation that preserves user privacy in a data mining process. They considered three k-anonymity algorithms: Samarati's algorithm [17] looks for the possible k-anonymity solutions by jumping at different levels in Domain Generalisation Hierarchy (DGH). Sweeney's algorithm [18], considered to be the best, generalises variables with the most distinct values (unique items). The Incognito algorithm [16] produces all possible k-anonymous full-domain generalisations of a relation, with an optional tuple suppression threshold. However, this technique has significant information loss and fails to correctly link attacks.

Encryption techniques attempt to protect the whole dataset by encrypting it. Some modern database management systems are already using cryptographic techniques to protect the data at-rest. However, they do not protect the data during the transfer. In this study, we focus on data protection and security during transfer (In-Flight method).

In-Flight Methods: These techniques attempt to protect the data in transit. In this case, cryptographic approaches can be used for data exchange between multiple parties [13]; the data is protected and they provide better privacy. However, they do not scale well for a large number of involved parties [21]. In [22], the authors leveraged the homomorphic encryption algorithms to preserve the user privacy in a collaborative environment among different peers of a decentralised system. Some similar studies, but in centralised systems (a server and many clients), enforce a web service composition based on specific web service executive languages [23]. These studies did not focus on optimising the computational cost since their dataset is small, (e.g., XML-based file or a local text-file). However, they do not provide a secure solution between two parties during the execution of the classification process on big datasets.

The concept of *garbled circuit* was proposed for preserving data privacy between two-party communications [20]. they optimised the arithmetic operator design using SHA-1 [20]. However, the SHA-1 algorithm has a shorter key length and hashed value, so it is not secure enough. Others extended the garbled circuit approach to the three- party communications using AES, MD5, and SHA-1, 2 [19], however, the computation cost is high. Mohasse et al. [24] applied the garbled circuit on some machine learning techniques; linear regression and neuron networks. These techniques, which combine encryption algorithms and gabled circuit, do not sufficiently focus on preserving the privacy of data mining techniques.

In this paper we propose a secure protocol for two-party communications to protect the users against the honest-but-curious and malicious adversaries. To reduce computation cost, we exploit the Elliptic Curve Cryptography (ECC) [25] in the transmission process. The whole system is evaluated on the Naive Bayes [26] and SVM [27] classifiers.

3. System Requirements

Consider a distributed computing system that consists of N processing nodes, which can be used by M users. The proposed security model divides the system into *clients* and *servers*. Its main goal is to protect the users from honest-but-

curious attackers [28]. The servers run a security controllers and a security guard. The security controller identifies suspicious attackers and the security guard makes decisions based on the security controller reports. The clients handle users' processes and activities. The key module in this system is the security controller that implements a data collector (DC) from various clients and a data server classifier(SC) for detecting suspicious attacks (see Figure 1).

3.1. Privacy Requirements

To classify a new data, SC and DC, as shown in Figure 1, need to collaborate, which may cause privacy violation (e.g., user privacy), since DC and SC may be honest, honest-but-curious or malicious attackers. SC can get users' private information, while DC can infer the steps and the criteria of the SC's analysis model. Let us consider the following simple example.

Example 1. *Assume that DC transfers the feature vector X to SC, which contains personal activities of the user. Thus, SC knows that the user is working at time x_1, and is using the folder, namely* `HR-2020`*, at time x_1 based on the coordinates in x_2. Meanwhile, DC can get the details of SC's analysis model w, so, it can infer its thresholds to avoid anomalous detection or to create a fraudulent forensic data of other users to deceit SC.*

To preserve the user privacy and to secure the data, we consider three security requirements, which are; *(R_1) Secure DC's data*: SC is not allowed to get access to DC's data; *(R_2) Secure SC's data*: DC is not allowed to access SC's dataset and its model w; *(R_3) Secure computed data and model*: the results of the classification phase must be secured against DC and third party. Hence, SC and DC need to collaborate in a secret way to obtain the results without leaking any of their own information to each other.

3.2. Data Collection

DC is in charge of collecting the unclassified forensic data regarding a given user access to system resources, such as computer mouse, keyboard, file system, tools, web browser, etc. This data collection process is periodically enforced. A client sends its data as a feature vector. For example, a feature vector X having four components; $X = (x_0, x_1, x_2, x_3)$, which represent mouse motion information such as cursor position $x_0 = (y_0, y_1)$ on the screen, click time x_1, left/right mouse button x_2, and single/double click x_3. Moreover, there may be more than one DC in the system, depending on the number of active clients. Each DC connects to a cluster of clients to collect their data, then collaborates with the classifier to find out whether the owner of that data is likely to be a malicious user or not.

3.3. Data Classification

The classifier has two main components: the training dataset, which has to be validated and annotated by experts, and the classification model that implements appropriate machine learning techniques for detecting anomalous behaviours

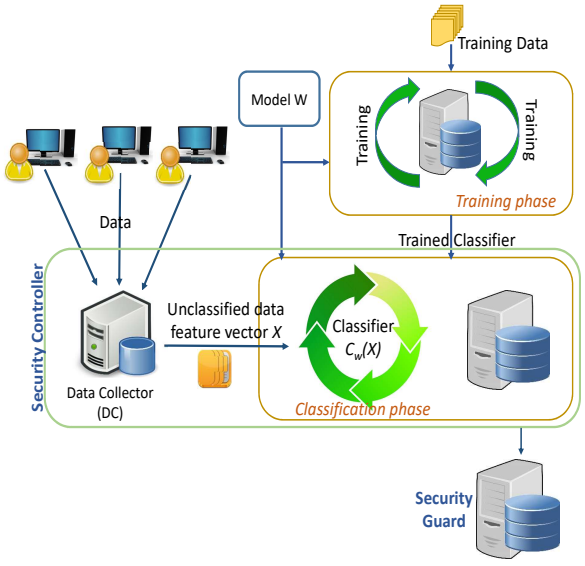

Figure 1: System Architecture.

in the system. The classification model is also kept secret by encrypting its outputs. The data consists of user interactions with the system, which include key stroke, mouse motion, opening folder, saving file, used path, printing/scanning activities, etc. The analysis of this data can conclude on the user's habit/behaviour or frequent operations on a given computer, and can also be used for tracing incidents internal cyber crimes. This data X is represented as a vector of features; $X = \{x_0, x_1, \cdots, x_{n-1}\}^l$ where n is the number of features in X, x_i is the value of the i^{th} feature, and l is the number of bits in x_i. Consider the mouse motion of a user u. At time t, u hovers the mouse at the point (c_0, c_1) where c_0 is the horizontal coordinate and c_1 is the vertical coordinate. The mouse status (i.e., the clicked button, the button clicking type, and the dragging status) can be represented as follows: $X = \{x_0 = t, x_1 = c_0, x_2 = c_1, null, null, null, null\}$.

4. Security & Protection Model

The proposed security system, called *SmartClass*, implements a secure data exchange between two parties. The system uses a garbled circuit technique in cooperation with Elliptic Curve Cryptography (ECC) algorithm [29] and oblivious transfer mechanism (see Figure 2).

The main components of the system are: SC, DC, and Security Guard (SG). The user processes send to DC blocks of data about the user's interaction with the system (Step 1). For each unlabelled data X, DC collaborates with SC to infer the class of the corresponding user data X; *normal* or *anomalous*. SC generates a set of garbled circuits (GCs) (Step 2). GCs are encrypted, with all its rows randomly shuffled, and sends them to DC along with its public key, required encrypted feature values of the relevant random numbers, etc. (Steps 3, 4). DC encrypts all of its bits with the received public key from SC (Steps 5,6). To exe-

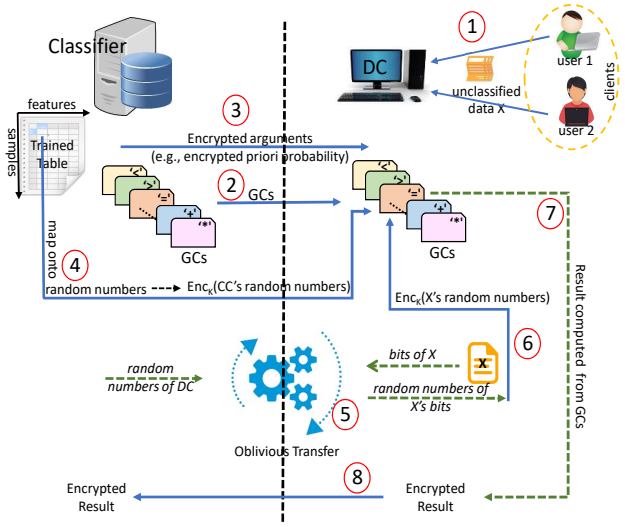

Figure 2: SmartClass Secure Architecture

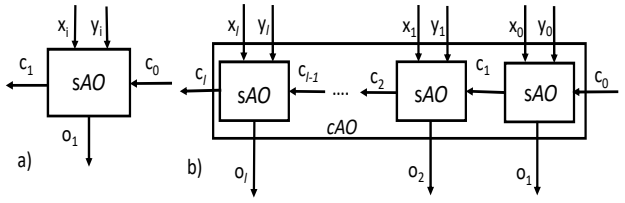

Figure 3: Arithmetic Operator (AO) a) Single AO; b) Complex AO

cute GCs, DC inputs the required encrypted random numbers from SC and its encryptions of random numbers. Then, DC receives the encrypted result and returns it back to SC (Steps 7, 8). Finally, SC sends it to the Security Guard. The Security Guard makes the decision about what to do with the given client based on the result from SC.

5. SmartClass Operators

We have two types of circuits: arithmetic circuits for $+$, $-$, $*$, $/$, and relational circuits for $<$, $\leq$, $=$, $>$, and $\geq$. All operands of these operators are of l-bit size. The complex garbled circuits involving a number of arithmetic and relational operators. Both arithmetic and relational circuits of l-bit numbers are built using single 1-bit circuits (see Figures 3 and 4).

Most secure relational operations are usually enforced by cryptographic protocols. However, in this study, we design logical circuits of relational operators to exploit the garbled circuit technique. We optimised the implementation of these logical circuits in terms of number of transistors. Table 2 shows a comparison between SmartClass logical circuit design and the circuits that are given in [30]. The number of transistors are calculated based on the definition of the number of transistors per type of logical circuit. We can see that the proposed circuits use less transistors then similar state-of-the-art techniques.

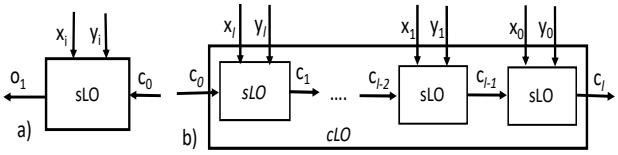

Figure 4: Relational Operator (LO): a) Single LO; b) Complex LO

6. Garbled Circuit-based Encryption

We use the binary Elliptic Curve Cryptography (ECC) technique to generate a pair of keys (one public key and one private key). The public key is used to encrypt the circuits. The generated keys need to satisfy the NIST conditions[4], with regard to the key length. ECC uses different random numbers to encrypt a plain text. The second phase of the model is to use a permutation technique to shuffle the elements of the encrypted circuit to produce a garbled circuit. This ensures the security of the messages exchanged, and the random permutation makes even stronger.

Our approach uses additive homomorphism and ECC, which is very quick (linear complexity). So the computations are significantly reduced while certifying several keys. The inputs and outputs are hidden behind the random numbers. Hence, one cannot recognise which bit value connects to which random number. The random numbers must be positive and unique. The system generates six random numbers; four random numbers for two inputs and two random numbers for one output. Whereas, for each single arithmetic circuit there are eight random numbers; four random numbers for two inputs, two random numbers for one input carry bit, and two random numbers for one output.

We encrypt the circuit to secure it during the exchange between SC and DC. The details of the attack resistance are further described in Section 8. In our model, SC generates the circuits for all operators that are used by the classifier.

There are 2^{2l+1} sets of inputs and output values for a complex arithmetic or relational circuit, which is equal to the number of rows in their truth table. However, for a complex relational circuit, the number of columns is $(2l + 2)$, while for a complex arithmetic circuit, the number of columns is $(3l + 1)$.

Example 2. *Consider a 2-bit relational circuit for performing the comparison $x < y$. The classifier generates six random numbers for two input bits and one carry bit. We have $\mathcal{R}_{C0} = \{r_{x=0}, r_{y=0}, r_{c=0}, r_{c=0}\}$ and $\mathcal{R}_{C1} = \{r_{x=1}, r_{y=1}, r_{c=1}, r_{c=1}\}$. The truth table of the 4 bits x, y, c_0, c_1 is $\{\{0,0,0,0\}, \{0,0,1,1\}, \{0,1,0,1\}, \{0,1,1,1\}, \{1,0,0,0\}, \{1,0,1,1\}, \{1,1,0,0\}, \{1,1,1,1\}\}$. At the same time, the classifier generates a public key K to encrypt the circuit. The encryptions of the complex circuit vector are $CEV = \{CE_0, CE_1, CE_2, CE_3, CE_4, CE_5, CE_6, CE_7\}$.*

To strengthen the security of CEV, its elements are randomly permuted. We apply the random permutation algo-

[4]https://www.keylength.com/en/4/

Table 2
Comparison between SmartClass Relational Circuits and Kolesnikov et al. [30] in terms of the Number of Used Transistors.

Operator	Kolesnikov et. al.[30]		SmartClass	
	2-bit Logical Gate	Number of Transistors	2-bit Logical Gate	Number of Transistors
$<$	3 XOR, 1 AND	30/36	1 AND, 1 OR, 1 NEG	14
$>$	3 XOR, 1 AND	30/36	1 AND, 1 OR, 1 NEG	14
$\leq$	3 XOR, 1 AND	30/36	1 AND, 1 OR, 1 NEG	14
$\geq$	3 XOR, 1 AND	30/36	1 AND, 1 OR, 1 NEG	14
$=$	3 XOR, 1 AND	30/36	1 XOR, 1 AND, 1 NEG	16/18

rithm; *Fisher-Yates* [31] for its optimal and effective properties compared to the other random permutation algorithms, with the complexity $O(n)$.

7. Security-aware Protocol

In this section, we describe how the classification phase is enforced in a secret way in order to guarantee the requirements R_1, R_2, and R_3 (c.f. Section 3.1). The SmartClass protocol is depicted in Table 3. SC uses a training table having different columns (i.e., features) and values (i.e., frequencies) depending on the classification techniques. SC first generates a pair of public and private keys (K, K_S). Then, SC creates the garbled circuits representing the supported operators.

The number of garbled circuits is equal to the number of operators in the system. For example, if the system supports nine operators, then there are nine garbled circuits. Generating garbled circuits is costly, but this can be done as part of the pre-processing phase. Moreover, in order to strengthen the security policy, the public key used for making encryptions can be changed periodically. As a consequence, all the garbled circuits are regenerated again.

Algorithm 1 *generateGarbledCircuits()*

Input: $K, \mathcal{T}_{\mathcal{R}}$
Output: GCs
1: $GCs = \phi$; $\mathcal{T}^{\mathcal{C}}_{\mathcal{V}} = \phi$;
2: **for** $(\forall \mathcal{R}_{Ci} \in \mathcal{T}_{\mathcal{R}} : \overline{i = 0..l})$ **do**
3: $\quad \mathcal{T}^{\mathcal{C}}_{\mathcal{V}} = genRandTruthTable(\mathcal{R}_i)$;
4: $\quad GC = \phi$;
5: $\quad GC = initOneGarbledCircuit(K, \mathcal{T}^{\mathcal{C}}_{\mathcal{V}})$;
6: $\quad GCs.Add(GC)$;
7: **end for**
8: *return* GCs;

SC generates the tables of random numbers (step 1.1 - Table 3). Each garbled circuit uses two tables of random numbers. Let $\mathcal{T}_{\mathcal{R}} = \{\mathcal{R}_{Ci}\}_{i=0}^{l}$ be a set of tables of random numbers, where l is the number of garbled circuits. Algorithm 1 describes the steps for generating the garbled circuits GCs. For each $\mathcal{R}_{Ci} \in \mathcal{T}_{\mathcal{R}}$, a random-driven truth table $\mathcal{T}^{\mathcal{C}}_{\mathcal{V}}$ is computed (c.f. lines 2, 3 - Algorithm 1, step 1.2 - Table 3). For each $\mathcal{T}^{\mathcal{C}}_{\mathcal{V}}$, its garbled circuit GC is computed by the function *initOneGarbledCircuit()* (see Algorithm 1) as presented in Algorithm 2. GC is then returned and added into the list of GC (line 6 - Algorithm 1). A complex encrypted circuit,

Table 3
SmartClass Protocol

1.	SC**:**
1.1.	- Generates the random numbers $\mathcal{R}$ for complex circuits
1.2.	- Creates truth tables $T^{\mathcal{C}}_{\mathcal{V}}$ based on $\mathcal{R}$ for complex circuits
1.3.	- Computes the complex circuit encryption vector CEVs for complex circuits
1.4.	- Permutes elements of CEVs to output garbled circuits GCs.
2.	$SC \rightarrow DC$**:**
2.1.	- Sends SC's public key K.
2.2.	- Sends GCs.
2.3.	- Sends $Enc_K(CCRand)$.
3.	SC & DC enforce oblivious transfer process.
3.1.	- DC inputs bits.
3.2.	- SC inputs encrypted random numbers generated for possible DC's input bits.
3.3.	- DC receives encrypted random numbers $Enc_K(DCRand)$.
4.	Execute one of the below classifiers.
4.1.	- Naive Bayes Classifier based protocol in Table 4.
4.2.	- SVM Classifier based protocol in Table 5.
5.	SC**:**
5.1.	- Decrypts $Enc_K(C_i)$ and concludes on user class of X.

CEV, is computed based on each $\mathcal{T}^{\mathcal{C}}_{\mathcal{V}}$ (step 1.3 - Table 3). Each element $CE_i \in CEV$ is calculated by adding the encryptions of all elements $\mathcal{T}^{\mathcal{C}}_{\mathcal{V}}[i, j]$ in its random-driven truth table (c.f. lines 3, 4, 5, 7 - Algorithm 2). After that, the circuit encryption CEV is randomly permuted (i.e., garbled) using shuffle function $GC = f_{FY}(CE)$ (line 9 - Algorithm 2, step 1.4 - Table 3).

7.1. Enforcing Secure Classification

After generating the garbled circuits (GCs), SC shares GCs and its valid public key K with DC (steps 2.1, 2.2 - Table 3). DC cannot know which operator each GC represents, since all GCs are lists of encryptions that are randomly permuted.

For each GC, DC inputs the two sets of encrypted random numbers. For that purpose, SC sends DC a corresponding set of encrypted random numbers of its value's input bits, (step 2.3 - Table 3). DC inputs its bits to the oblivious transfer process and SC inputs the encrypted random numbers

Algorithm 2 *initOneGarbledCircuit()*

Input: K, T_V^C
Output: GC
1: $GC = \phi$;
2: $CEV = \phi$;
3: **for** $(\forall CE_i \in CEV, i = \overline{0..2^{2l+1}-1})$ **do**
4: **for** $(\forall \mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j]), j = \overline{0..m})$ **do**
5: $CE_i += Enc_K(\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j]))$;
6: **end for**
7: $CEV.Add(CE_i)$;
8: **end for**
9: $GC = f_{FY}(CEV)$;
10: *return* GC;

of DC (steps 3.1, 3.2 - Table 3). For each specific value of DC's input bit, DC gets a respective encrypted random number and adds it into $Enc_{(DCRand,K)}$ (step 3.3 - Table 3). Note that DC cannot infer any random number of SC from its plain bits, since all random numbers SC generated for every bits are different.

More particularly, let us consider a relational $GC = \{ GC_0, GC_1, \cdots \}$. After retrieving $DCRand$, DC computes the GC's result (c.f. Algorithm 3). In particular, we have two input data; $Enc_{(CCRand,K)} = \{ Enc_K(rc_{x_0}), \cdots, Enc_K(rc_{x_l}), Enc_K(rc_{c_0}) \}$, and $Enc_{(DCRand,K)} = \{ Enc_K(rd_{y_0}), \cdots, Enc_K(rd_{y_l}) \}$, where rc, rd are random numbers of $CCRand$ and $DCRand$, respectively. Let ADD be the resulted set containing additions of each GC_i with $Enc_{(CCRand,K)}$ and $Enc_{(DCRand,K)}$. DC computes ADD and sends it to SC (c.f. line 2). The encrypted result can be used as an input to the following relational or arithmetic GCs.

Algorithm 3 *computeResult()*

Input: $Enc_{(CCRand,K)} = \{Enc_K(rc_{(x_0^1)}, \cdots, Enc_K(rc_{(x_l)}), Enc_K(rc_{(c_0)})\}$,
$Enc_{(DCRand,K)} = \{Enc_K(rd_{(y_0)}, \cdots, Enc_K(rd_{(y_l)}\}$
Output: ADD
ADD = ϕ;
1: **for** $(\forall GC_i \in GC)$ **do**
2: $ADD_i = GC_i + Enc_{(CCRand,K)} + Enc_{(DCRand,K)}$;
3: **end for**
4: *return* ADD;

In the case of a few GCs, in order to avoid DC to predict the used GC's operator, SC duplicates the requested GCs then shuffles the orders of elements of those GCs. We describe the enforcement of two secure classifiers Naive Bayes (step 4.1 - Table 3) and SVM (step 4.2 - Table 3) in the following.

7.2. Secure Naive Bayes Classifier

We developed a Naive Bayes algorithm following our approach (see Table 4). The classifier calculates prior probabilities of all user classes $Pr(C_i)$ (step 1.1). Then, for each class C_i, the classifier needs to get the posterior probabilities $Pr(C_i|X)$ of the normal or anomalous user types based on data X.

Example 3. *Given C = { 'Normal User', 'Anomalous User' }, and X = {23:04:23, 10, 12, CLICKED, LEFT, SINGLE, DRAGGING }. In order to compute Pr('Anomalous User' | X), SmartClass needs to compute Pr('Anomalous User'), Pr(X | 'Anomalous User'), Pr(X). We need to compute the partial probabilities including Pr('23:04:23' | 'Anomalous User'), Pr('23:04:23' | 'Normal User'), ···, Pr(DRAGGING | 'Anomalous User'), Pr(DRAGGING | 'Normal User'). To compute the partial probability Pr(DRAGGING | 'Anomalous User'), one needs to calculate the three following frequencies, that is, the frequency of samples having values equal to "Anomalous user" denoted as q, the frequency of samples having values equal to "DRAGGING" and "Anomalous User" denoted as p. So that we can have Pr(DRAGGING | 'Anomalous User') = p/q.*

Based on the generic Naive Bayes algorithm, we generated the necessary steps of the encrypted version of the algorithm based on our approach, as given in Table 4. All it is required is to make sure that the algorithm's steps are encrypted using a number of GCs, and then follow strictly the exchange secure protocol to avoid breaches from both sides (SC and DC). For instance, after obtaining all needed frequencies (step 2.5), DC can compute the corresponding partial probabilities using arithmetic GCs. DC can compute the probability $Pr(C_i|X)$ for each user class C_i. The process is iterated for the other user classes. When a probability $Pr(C_i|X)$ is computed (step 2.6), DC compares it to the previously computed one using the relational GC of the operator $>$, to find the greater probability (step 2.7). SC and DC continue to execute the protocol until the last posterior probability.

7.3. Secure SVM Classifier

The SVM Classifier is simpler than Naive-Bayes classifier. SVM requires two arithmetic operators $+$ and $*$, and two operators $>$ and $<$. SC needs to duplicate GCs, as its has few operators, permute the elements of each duplicated GC, and then shuffle the GCs. In order to classify an unlabelled data $X = \{x_0, x_1, \cdots, x_n\}$, we need to compute $f(X)$ to find out which side of the hyperplane X is located. The secure version of SVM is summarised in Table 5.

In our protocol, SC sends a set of GCs (lines 1.1, 1.2) including duplicates. Next, SC needs to send its random numbers representing its input bits (line 1.3). DC iteratively inputs the random numbers of SC into GCs (line 2.1) until there is no GC. The final result is compared with the encrypted value, which is previously sent from SC (line 2.3).

Classification Result Decision: The decision is made by SC. After receiving the final result from DC, SC decrypts it. Based on the result outcome, SC sends user ID to the Security Guard. The Security Guard makes a decision on X's behaviour.

8. Security Analysis

In this section, we analyse the security properties of the SmartClass secure protocol against the honest-but-curious and malicious attacks. We show how SmartClass can protect private data on one side and classifier on other side.

Table 4
Naive Bayes Classifier based SmartClass Protocol

1.	SC:
1.1.	- Computes prior probabilities of all user classes $Pr(C_i)$.
1.2.	- Traverses each user class $C_i \in C = \{Normal\ User, Anomalous\ User\}$. If $C = \phi$, do step 2.9.
1.3.	- Sends DC $Enc_K(Pr(C_i))$, $Enc_K(C_i)$ where K is SC's public key.
1.4.	- Traverses each feature column of the training data table to compute encrypted $Pr(x_j\|C_i)$.
1.5.	- Sends DC the needed GCs' ordinaries to be used for the indicated feature column.
1.6.	- Traverses each feature value, sends DC its encrypted random numbers and the related user class' bits for comparing.
2.	DC:
2.1.	- Use relational GCs to compare X's feature values and the received from SC.
2.2.	- Inputs the result from 2.1. into GC of operator "+" to count the feature frequency.
2.3.	- Back to step 2.1. when the feature column is not ended.
2.4.	- Do step 1.4. when all feature columns excluding user class feature is not traversed.
2.5.	- All frequencies are ready, compute the encrypted partial probability $Pr(x_j\|C_i)$.
2.6.	- Computes the encrypted $Pr(C_i\|X)$ using GC of operator "$*$".
2.7.	- Compares $Pr(C_i\|X)$s to get the greater probability. Saves $Enc_K(C_i)$ of the greater probability.
2.8.	- Do step 1.2.
2.9.	- Send $Enc_K(C_i)$ of the maximum posterior probability to SC.

Table 5
SVM Classifier based SmartClass Protocol.

1.	$SC \rightarrow DC$:
1.1.	- Sends a set of GCs.
1.2.	- Sends the needed GCs's ordinaries
1.3.	- Sends the set of encrypted random numbers.
2.	DC:
2.1.	- For each GC, inputs the random number of SC and its random number.
2.2.	- Transfers the encrypted result of the previous calculation into the following GC (i.e., +).
2.3.	- Compares the final computed result with the cipher text of 0 by relational GC (i.e., >).
2.4.	- Returns the relational encrypted result to SC.

The classifier training dataset should not be altered by any insider or outsider attacks. It is only used during the training phase and it is not part of the collaborative process between SC and DC. The training dataset is encrypted and it is only accessed by the classifier. In the following, we look at some possible scenarios of data security enforcement.

8.1. Encryption Approach Efficiency

Let us investigate the possibility that DC can decrypt SC's data. In order to decrypt any ECC encryption, DC needs a private key of SC. To do so, the attacker can attempt of the following methods.

Trial-and-Error Method: ECC private key k is an n-bit number that is randomly generated, where n is the key size. To get the right k, the adversary can try 2^n keys to decrypt a cipher text, where n is a large number, which is very time consuming. For instance, for $n = 163$, there are 2^{163} possible keys.

Plain-Text Analysis: The adversary can attempt to input the selected numbers (e.g., x) and encrypt them with the public key $k.B$, and then compare the result, $Enc_{k.B}(x)$, to $Enc_K(rc_{(x_0)})$. Even if they are equal, the adversary still cannot infer the value of bits as it does have the right random numbers, which is the second step of the proposed encryption approach. Since we do not generate the same random number for the same input value, this makes it very difficult to associate the right random number to a given input. In more details

- The adversary tries to guess $\Sigma_{j=0}^{j=m}(\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j])$, encrypts them with the public key $k.B$, compares them with the ones received from SC, then chooses possible values giving the correct answer.
- The adversary needs to try with different l to compute different m, i.e., $m = 2.l + 2$ or $m = 3.l + 1$.
- For each $\Sigma_{j=0}^{j=m}(\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j])$, the adversary needs to factorise it to get m random numbers $\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j]$. The random number is an integer r of 64 bits, which means there are 2^{64} numbers for one $\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j]$. At this step, the adversary cannot know which random number is correct, as well as, cannot map the random number to the real value of bits because the random numbers for the same values of different bits are also different.

Therefore, in this Scenario, the adversary cannot gain access to SC's private data.

8.2. Process of non-Reversibility

The adversary, (e.g., DC), can put an effort on tracing step-by-step the process that SC uses to encrypt the list of GCs, which are:

Step 1: The classifier operators are implemented as logical circuits. Each logical circuit is represented by its binary truth table.

Step 2: The truth table is randomised into $\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}$.

Step 3: $\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}$ is encrypted to $CEV = \bigcup_{i=0}^{2^{2l+1}-1}\{CE_i\} = \bigcup_{i=0}^{2^{2l+1}-1}\{Enc_K(\Sigma_{j=0}^{j=m}(\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j])\}$.

Step 4: CEV is permuted to $GC = f_{FY}(CEV)$.

By reverse engineering, DC should attempts the following:

Step 4: SmartClass uses the shuffle algorithm Fisher-Yates [31] that randomises elements of CEV. However, to look for an order of the encryption elements CE_i in

CEV, one needs to test a very huge number of combinations in CEV, which is equal to 2^{4l+1} combinations. In addition, the adversary needs to execute also the steps 3, 2, and 1 to test which combination is the right answer.

Step 3: The adversary tries to seek the plain text of encryption as in the previous scenario with both methods of trial-and-error method and plain-text analysis to obtain the array of $\{(\mathcal{T}_{\mathcal{V}}^{\mathcal{C}}[i,j])\}$, where $i = 0, \cdots, 2^{2.l+1} - 1$ and $j = 0, \cdots, m$, which is very complex. Step 3 is the most challenging for the adversary to obtain the plain text data, as the computing cost is much higher.

Step 2: The used random numbers are generated for each bit following Hypothesis 1. The difficulty in predicting the random number at Step 2 is similar to Plain-text Analysis scenario discussed above.

To strengthen even more the security of the proposed model, we added the following constraints:

Constraint 1. *The random numbers for different bits are distinguished and re-generated periodically.*

Constraint 2. *The oblivious transfer protocol is completely secure.*

Constraint 1 guarantees that although the bits have the same value, they are represented by different numbers. Therefore, the adversary cannot analyse the random numbers easily, and use it for the other bits having the same value. Whereas, Constraint 2 ensures that DC's random numbers cannot be mapped to the real bits of DC to avoid SC accessing DC's data.

8.3. DC's Data Security

There are two cases where SC is likely to infer the bits of DC. We show how our model deals with DC's data in each case.

1. **Random Number Mapping.** Each bit of X is pseudonymised by a different random number. SC is responsible for generating random numbers for DC on request. Therefore, SC can take this chance to map random numbers and DC's data. However, to get random numbers, both SC and DC execute a secure oblivious transfer protocol. Constraint 2 makes the transmission of these random numbers secure to DC. Hence, SC cannot access DC's data. Additionally, SC cannot map DC's bits to their random numbers as each bit is represented by a different random number according Constraint 1.
2. **Classification Results.** After getting the classification results, DC returns them to SC. SC decrypts the results, however, it still cannot trace backwards the original input data of DC. If DC's input data has l bits, there is a minimum $\lfloor \frac{2^{2l+1}}{2} \rfloor$ same output bits, which increases the complexity for SC to analyse. Hence, SC cannot exactly identify DC's input data.

Table 6
Combination Key Sizes of Cryptographic Algorithms ECC, SHA, and Paillier.

Combination Ord.	ECC Key Size (bit)	SHA Key Size (bit)	Paillier Key Size (bit)
1	233	224	1024
2	283	256	2048
3	409	384	3072
4	571	512	4096

Finally, the proposed analysis of private data is strong, and each step of the technique requires enormous amount of computations to break the security policy used at that step. The data of the user is kept private, SC cannot access it. The classifier is also secure, as each of its processing steps are encrypted and kept secret. The results of the analysis are also secure, as they follow the same constraints and security policies implemented at each step of the whole execution. In the following section, we perform some experiments to validate the model.

9. Experimentation

To evaluate our approach effectiveness and efficiency, as well as its lightweight property, we performed several experiments with various cryptographic algorithms and their key sizes, different garbled circuit types and sizes, and different sizes of the training table. These experiments are run on a computer with 8 GB of RAM and CPU i7 1.8 GHz. Each experiment was run 20 times and we take the average execution time.

We use SHA-2, Paillier cryptographic algorithms, and Koblitz elliptic curves (or anomalous binary curves), as the other algorithms, either they are not suitable, such as AES, or they are weak, such as SHA-1 and MD5. Koblitz elliptic curves is one of the recommended ECC algorithms. For simplicity, it is denoted ECC in the rest of the paper.

The key sizes used for ECC, SHA-2, and Paillier are $\{233, 283, 409, 571\}$ bits, $\{224, 256, 384, 512\}$ bits, and $\{$ 1024, 2048, 3072, 4096$\}$ bits, respectively. Table 6 summarises the sizes used which satisfy the NIST's conditions against the analysis attacks[5]. In this study, we focus only on numerical data type. Other data types will be studied in future work.

9.1. Data Size vs Execution Time

Our solution is built around the data size used either in the operands or encryption keys. The objective here is to study the SmartClass performance by evaluating the garbled circuits with different data sizes. The performance evaluation is crucial as the size of the garbled circuits depends directly on the size of its inputs. The sizes of the training table of Naive Bayes classifier are 50, 100, 150. The number of tested GCs inputs are also 50, 100, 150. Each value contains l bits, with $l \in \{8, 16, 32\}$.

[5] https:// www. keylength. com/ en/4/

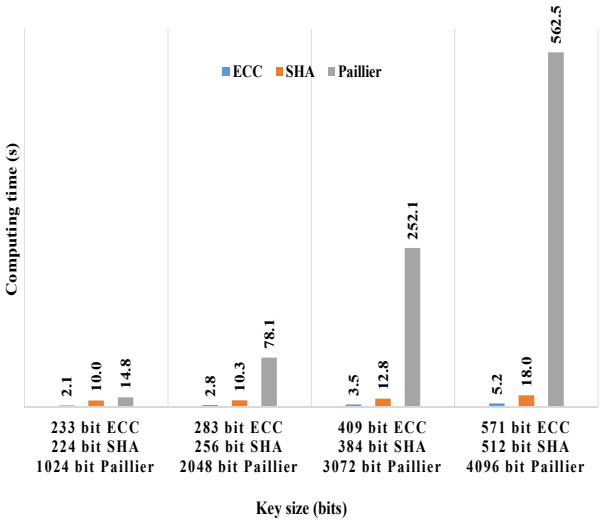

Figure 5: Computing time on 50 garble circuits evaluation with 8-bit numeric data and different key sizes of ECC, SHA, Paillier algorithms.

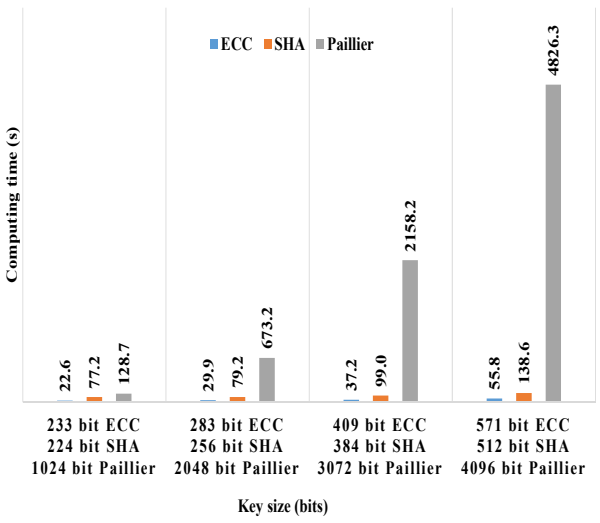

Figure 6: Computing time on 150 garble circuits evaluation with 32-bit numeric data and different key sizes of ECC, SHA, Paillier algorithms.

In the first experiment (see Figure 5), the execution times are measured on evaluated 50 circuits and 8-bit input numbers. We can see that ECC returned the shortest time, while Paillier returned the longest for all key sizes.

Furthermore, in the second experiment (c.f. Figure 6), we investigate the execution time when the number of garbled circuits was increased. We have the same observation as in the previous experiment; ECC is the shortest and Paillier is the longest. It is interesting to note that the ECC time went from 5 seconds for 50 GCs to 55 seconds for 150 GCs using its largest key 571, while the inputs numbers varied from 16 bits to 32 bits. This can be significantly reduced if we use multicore feature of the current CPUs. 55 seconds still very quick for 150 GCs, compared to over 2 minutes for SHA-2 and more than 80 hours for Paillier.

Naive Bayes vs SVM: Performance Comparison. It is expected that SVM is much quicker but we would like to know how they behave in the context of our privacy-preserving analysis model. We consider all the parameters affecting the vulnerability of the model. Particularly, we choose different l input bits for each GC, that is, $l \in \{8, 16, 32\}$ bits (it is denoted IB in the Figures). Moreover, we use variable combinations of key sizes of ECC, SHA and Paillier. For each unlabelled data X, it is set to contain 10 features. SVM based model needs 22 GCs in total for checking the class of X, while the Naive Bayes based model needs a higher number of GCs, 550 GCs with a training table of 10 features and 50 samples.

In the first experiment (c.f. Figure 7), the key sizes of SHA, ECC and Paillier are respectively 224, 233 and 1024 bits. The input data size IB took values 8, 16, 32 bits. We can see that, in all cases the Naive Bayes execution time is always much higher than SVM based model. More specifically, we consider the key size combination includes 224-bit

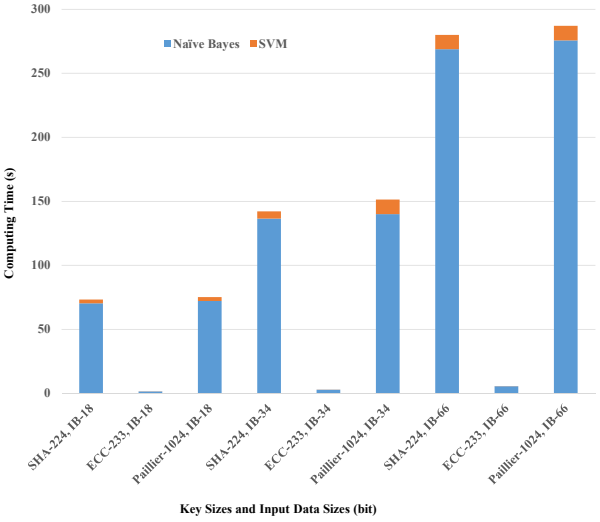

Figure 7: Computing time (s) of Naive Bayes based model vs SVM based model on: 10 features of unlabelled data X (i.e., 550 GCs for Naive Bayes and 22 GCs for SVM); different numeric data sizes (IB = {8, 16, 32} bits); different key sizes (233-bit ECC, 224-bit SHA, 1024-bit Paillier algorithms).

SHA, 233-bit ECC and 1024-bit Paillier. We can also notice that the combination of SVM and ECC has extremely short execution time, which is 0.06s, while Naive Bayes is much more slower, which ever the encryption algorithm used, and does not exploit the best features of the ECC. The Naive Bayes performance becomes worse with the increasing sizes of the encryption keys; see Figure 8.

It is worth noting the execution time is one measure. In the privacy-preserving data analysis one should consider the accuracy of the results produced by the analyser, in this case the classifier. In many situations, we can find Naive Bayes classifier is much more accurate and robust, despite its poor

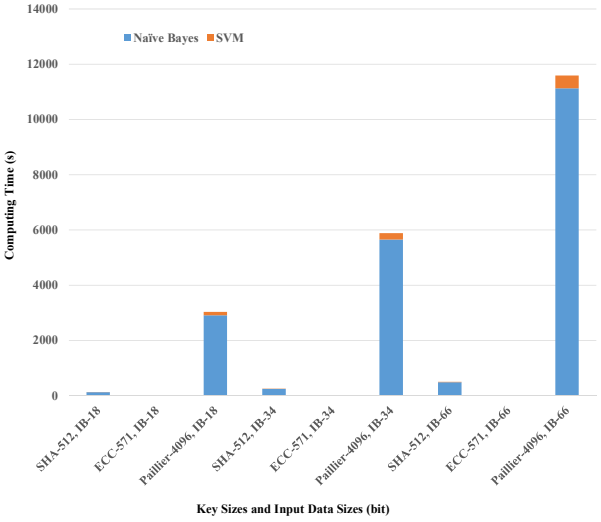

Figure 8: Computing time (s) of Naive Bayes based model vs SVM based model on: 10 features of unlabelled data X (i.e., 550 GCs for Naive Bayes and 22 GCs for SVM); different numeric data sizes (IB = {8, 16, 32} bits); different key sizes (571-bit ECC, 512-bit SHA, 4096-bit Paillier algorithms).

speed-up. Moreover, in some applications with small number of GCs, SVM is not desirable, as it can compromise its integrity and security. In these experiments, we wanted to give a good idea about the Naive Bayes complexity and its association with the analysis of private datasets. By using distributed SCs, one can improve significantly the performance of Naive Bayes. This will be subject of our future work.

9.2. Garbled Circuit Size

The size of GCs transferred between SC and DC are quantified in order to evaluate the bandwidth consumption. We measure the sizes of relational and arithmetic GCs that SmartClass generates using the ECC cryptographic algorithm. As the data size for each attribute in the training set varies within $\{1, 3, 11, 17\}$ bits, the input data size for the GC is $2l + 1$; $\{3, 7, 23, 35\}$ bits.

The experimental results are shown in Table 7. For numbers of sizes 3 and 7 bits, the GC sizes are 0.91KB and 14.56KB, respectively, which are relatively small. For numbers of sizes 23 and 35 bits, the largest GC size is about 9 GB. Moreover, the GC size depends also on the key size used in the encryption algorithms. This is not really a problem, as we do not exchange the GC between SC and DC completely. Only part of the circuit is exchanged at any one time. Therefore, the size of the circuit is not an issue for the transmission. The local execution of the circuit is also not an issue, as there is enough memory locally and it is very fast to execute. Moreover, the GC is only updated when the encryption keys are renewed.

Table 7
Garbled Circuit Sizes (KB) vs ECC Key Sizes (Bits) vs Data Unit Size (Bits).

Data Unit Size (bit)	ECC Key Size (bit)			
	233	283	409	571
3	0.91 KB	1.11 KB	1.60 KB	1.12 MB
7	14.56 KB	17.69 KB	25.56 KB	4.46 MB
23	932 MB	1132 MB	1636 MB	2284 MB
35	3728 MB	4528 MB	6544 MB	9136 MB

10. Conclusion and Future Work

We proposed a secure two-party model for privacy preserving data mining, called *SmartClass* that does not only protect user data but also protects the mining model for data analysis. Specifically, we use the arithmetic garbled circuits as part of the model, as well as design a set of garbled circuits for the relational operators. Additionally, we leverage the additive property of the binary ECC, and apply it as part of the protocol for a private communication between SC and DC. This has reduced significantly the system complexity and therefore its response time. The proposed model is very efficient as shown theoretically and experimentally. Furthermore, the SmartClass protocol is more suitable for very big data analytics, running on cloud infrastructure. Our system can take advantage of Spark[6].

As future works, we extend this work to support more data mining techniques, such as Decision Tree, Deep learning, etc. We will also modify SmartClass to process various data types such as categorical data, etc. We will also improve authentication of the GCs delivery transactions and improve the performance of some robust and very popular data mining algorithms. We also look at the use of ontology-based approach [32] to apply the proposed approach in analysing the digital incidents.

Acknowledgement

This work is supported by Science Foundation Ireland under grant number SFI/12/RC/2289. We would like to thank Dr. Martin Ochoa, SUTD, Singapore for helpful discussions.

References

[1] S. Budhaditya, D. Pham, M. Lazarescu, and S. Venkatesh. Effective anomaly detection in sensor networks data streams. In *2009 Ninth IEEE International Conference on Data Mining*, pages 722–727, 2009. doi: 10.1109/ICDM.2009.110.

[2] R. Fujimaki. Anomaly detection support vector machine and its application to fault diagnosis. In *2008 Eighth IEEE International Conference on Data Mining*, pages 797–802, 2008. doi: 10.1109/ICDM.2008.69.

[3] A.L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016. ISSN 1553-877X. doi: 10.1109/COMST.2015.2494502.

[6] https://spark.apache.org/docs/latest/

[4] P. Parveen, Z.R. Weger, B.M. Thuraisingham, K.W. Hamlen, and L. Khan. Supervised learning for insider threat detection using stream mining. In *23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1032–1039. IEEE Computer Society, 2011.

[5] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Third IEEE International Conference on Data Mining*, pages 99–106, 2003. doi: 10.1109/ICDM.2003.1250908.

[6] M. Kantarcioglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally partitioned data. In *IEEE ICDM Workshop on Privacy Preserving Data Mining*, pages 3–9, 2003.

[7] C.C. Aggarwal and P.S. Yu. *A General Survey of Privacy-Preserving Data Mining Models and Algorithms*. Springer US, Boston, MA, 2008. ISBN 978-0-387-70992-5. doi: 10.1007/978-0-387-70992-5_2. URL https://doi.org/10.1007/978-0-387-70992-5_2.

[8] D. Agrawal and C.C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 247–255, New York, NY, USA, 2001. ACM. ISBN 1-58113-361-8. doi: 10.1145/375551.375602. URL http://doi.acm.org/10.1145/375551.375602.

[9] S.L. Garfinkel. Nistir 8053: De-identification of personal information. NIST: National Institute of Standards and Technology, http://dx.doi.org/10.6028/NIST.IR.8053, 2015.

[10] C. Bauer and et al. White paper on pseudonymization drafted by the data protection focus group for the safety, protection, and trust platform for society and businesses in connection with the 2017 digital summit. https://www.eprivacy.eu/fileadmin/Redakteur/News/2017_Data_Protection_Focus_Group-White_Paper_Pseudonymization.pdf, 2017.

[11] S. Vinita and K. Kinjal Parmar. A review on data anonymization in privacy preserving data mining. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(2):75–79, 2016. doi: 10.17148/IJARCCE.2016.5216.

[12] K. El Emam, E. Jonker, L. Arbuckle, and B. Malin. A systematic review of re-identification attacks on health data. In *Scherer RW, ed. PLoS ONE. 6(12):e28071*, pages 439–450, 2011. doi: 10.1371/journal.pone.0028071.

[13] S. Bhanumathi and P. Sakthivel. Preservation of private information using secure multi-party computation. *Indian Journal of Science and Technology*, 9(14):1–6, 2016. ISSN 0974-6846. doi: 10.17485/ijst/2016/v9i14/74588.

[14] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 439–450, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4. doi: 10.1145/342009.335438. URL http://doi.acm.org/10.1145/342009.335438.

[15] L. L. Liu, M. M. Kantarcioglu, and B. B. Thuraisingham. Privacy preserving decision tree mining from perturbed data. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–10, 2009. doi: 10.1109/HICSS.2009.353.

[16] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 49–60, New York, NY, USA, 2005. ACM. ISBN 1-59593-060-4. doi: 10.1145/1066157.1066164. URL http://doi.acm.org/10.1145/1066157.1066164.

[17] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001. ISSN 1041-4347. doi: 10.1109/69.971193.

[18] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002. ISSN 0218-4885. doi: 10.1142/S021848850200165X.

[19] P. Mohassel, M. Rosulek, and Y. Zhang. Fast and secure three-party computation: The garbled circuit approach. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 591–602, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813705.

[20] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 35–35, Berkeley, CA, USA, 2011. USENIX Association. URL http://dl.acm.org/citation.cfm?id=2028067.2028102.

[21] G.J. Taric and E. Poovammal. A survey on privacy preserving data mining techniques. *Indian Journal of Science and Technology*, 10(5), 2017. doi: 10.17485/ijst/2017/v10i5/111138.

[22] B. Carminati, E. Ferrari, and N.H. Tran. Enforcing trust preferences in mobile person-to-person payments. In *2013 International Conference on Social Computing*, pages 429–434, Sep. 2013. doi: 10.1109/SocialCom.2013.67.

[23] B. Carminati, E. Ferrari, and N.H. Tran. Secure web service composition with untrusted broker. In *2014 IEEE International Conference on Web Services*, pages 137–144, June 2014. doi: 10.1109/ICWS.2014.31.

[24] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, May 2017. doi: 10.1109/SP.2017.12.

[25] V. Katiyar, K. Dutta, and S. Gupta. A survey on elliptic curve cryptography for pervasive computing environment. *International Journal of Computer Applications*, 11(10):41–46, 2010. ISSN 0975 âĂŞ 8887.

[26] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, pages 41–46. IBM New York, 2001. doi: 10.1109/HICSS.2009.353.

[27] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. ISBN ISBN 0-521-78019-5.

[28] Q. Do, M. Ben, and K.KR. Choo. The role of the adversary model in applied security research. *Computers & Security*, 81:156–181, 2019.

[29] D. Hankerson and A. Menezes. Chapter nist elliptic curves, encyclopedia of cryptography and security (2nd ed.), 2011.

[30] V. Kolesnikov, A-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, pages 1–20, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10432-9. doi: 10.1007/978-3-642-10433-6_1.

[31] R.A. Fisher and F. Yates. *Statistical tables for biological, agricultural and medical research (3rd ed.)*. Oliver and Boyd, Edinburgh, 1949. ISBN 0-02-844720-4.

[32] Chabot Y., Bertaux A., Nicolle C., and Kechadi M-T. An ontology-based approach for the reconstruction and analysis of digital incidents timelines. *Digital Investigation*, 15:83–100, 2015. doi: 10.1016/j.diin.2015.07.005.

View publication stats