Abstract—User-generated data is crucial to predictive modeling in many applications. With a web/mobile/wearable interface, an online service provider (SP) can continuously record user-generated data and depend on various predictive models learned from the data to improve their services and revenue. SPs owning a large collection of user-generated data has raised privacy concerns. We present a privacy-preserving framework, SecureBoost, which allows users to submit encrypted or randomly masked data to SPs that want to learn predictive models but offload the responsibility of protecting sensitive data. Our framework utilizes random linear classifiers (RLCs) as the base classifiers in the boosting framework to simplify the design of the proposed privacy-preserving boosting protocols. A Cryptographic Service Provider (CSP) is used to assist an SP's processing, reducing the complexity of the protocol constructions while the leakage of information to the CSP is limited. We present two constructions of SecureBoost: HE+GC and SecSh+GC, using combinations of homomorphic encryption, garbled circuits, and random masking to achieve both security and efficiency. For a boosted model, the SP learns only the base models (i.e., RLCs) and the CSP learns only the weights of the base models. This separated parameter holding avoids any of the two parties from abusing the final model or conducting model-based attacks. We have conducted extensive experiments to understand the quality of the RLC-based boosting and the cost distribution of the constructions. Our results show that SecureBoost can efficiently learn high-quality boosting models from protected user-generated data.

I. INTRODUCTION

Many applications such as search engine, movie recommendation, healthcare, and social networking depend on learning from user-generated data such as clickthroughs, tweets, reviews and ratings, group interactions, and sensor readings of daily activities [1]–[3]. It is a common scenario in which a service provider (SP) delivers services to its subscribing users (henceforth referred as users) via web/mobile/wearable applications. By tracking users’ activities via the applications, SP can collect a large amount of user-related data, which are used to improve the quality of services and increase revenues from more effective user-targeting.

Though highly beneficial to users and SPs alike, the massive collection of user data along with powerful big data analytics raise great privacy concerns. First, SP’s infrastructures, if poorly secured, can be compromised by external hackers which damages SP’s reputation and users’ privacy. Recent data breach incidents involved Target, Ashley Madison, and Equifax [4], [5]. Second, potential threat of unauthorized retrieval, sharing, or misuse of users’ personal information by insiders [6], [7] are difficult to detect and prevent. SPs have great responsibility for protecting the privacy of massive sensitive data collection. Thus, secure and privacy-preserving data mining frameworks are highly attractive to both SPs and users.

To protect from both internal and external privacy attacks and offload an SP’s burden, one approach is to learn the models from protected data. As SP works with protected data without knowing the decryption keys, any attack on SP side does not breach data privacy as long as the protocol is provably secure. Naive adaptation of the well-known cryptographic/privacy primitives such as the fully homomorphic encryption (FHE) scheme [8], garbled circuits (GC) [9], and secret sharing [10] in building privacy frameworks prove too expensive [11], [12]. A few recent studies [12]–[15] started blending multiple cryptographic primitives and adapted to certain privacy architectures to work around the performance bottlenecks. These “hybrid” constructions mix different cryptographic primitives to implement the key algorithmic components of a protocol with reasonable overheads. In addition, to relieve users from active participation in the protocols, a third party called cryptographic service provider (CSP) is introduced in the frameworks to manage keys, build circuits, and assist SP in the protocols [12].

While adapting a hybrid approach looks promising, there are several challenges. First, it is complex to optimally select and assemble primitives to implement functionalities of existing machine learning algorithms. For example, Nikolaenko et al. [12], [13] develop matrix factorization and linear ridge regression protocols using FastGC [16] to build garbled circuits and with additive homomorphic encryption (AHE). They map the core algorithm, Cholesky decomposition, to a GC implementation, which is quite expensive in communication. Factoring a small $100 \times 100$ matrix with $4096$ non-zero elements takes about 40GB traffic between SP and CSP [12]. This clearly is expensive and indicates the difficulty of balancing cost and utility even with the hybrid approaches.

Moreover, it appears even trickier to select a right machine learning algorithm or modify one to significantly reduce the
complexity of its privacy-preserving version. Fundamentally, there may be tradeoffs among the easiness of developing a privacy-preserving protocol for the learning algorithm, the overhead of the protocol, and the learned model’s prediction power. On one side, it is easier to implement some protocols for less powerful algorithms such as linear classifiers and linear regressions [13], [15], [17]. On the other side, more powerful machine learning algorithms, such as SVM and deep neural networks require complex operations, of which secure implementations do not scale well to large datasets [15].

Finally, existing approaches [13], [15], [17] do not consider model-based inference attacks possibly conducted by a curious SP. Recent studies [18]–[20] have shown that attackers can use a learned predictive model and its output to estimate the sensitive attributes in users’ data. The key step of these attacks is that attackers can freely apply the model to any input data. To protect against these attack, one reasonable approach is to prevent a curious SP from seeing the complete model in the learning process. This change of setting may also bring additional difficulties in algorithm design.

A. Scope of Work and Contributions

Recent studies in learning on protected data have generated some preliminary results for top-quality prediction models, such as logistic regression and neural networks [15]. Surprisingly, no work has been reported on the powerful boosting framework [21]. While deep learning methods [22] have dominated image and sequence-based learning tasks, boosting is among the most powerful methods such as SVM and Random Forest [23] for other prediction tasks. For example, it has also been a popular method (e.g., XGBoost [24]) in learning to rank [25] and a top choice of most Kaggle competition winners. Our framework [21]. While deep learning methods [22] have dominated image and sequence-based learning tasks, boosting is among the most powerful methods such as SVM and Random Forest [23] for other prediction tasks. For example, it has also been a popular method (e.g., XGBoost [24]) in learning to rank [25] and a top choice of most Kaggle competition winners. Our research serves as the first step exploring this learning method on protected data.

The core idea of our SecureBoost approach is to fully utilize the powerful boosting theory [21] that requires only weak classifiers (e.g., each classifier’s accuracy is only slightly exceeding 50% for two-class problems) to derive a powerful prediction model. This flexibility allows us to develop efficient yet powerful hybrid privacy-preserving boosting protocols.

The choice of a weak classifier is thus critical to implement efficient privacy-preserving boosting. In most literature, decision stumps (DS) have been used as the weak classifiers for their simplicity and fast convergence of boosting. Although the training algorithm for a decision stump is quite simple, it appears expensive to implement its privacy-preserving version [26]. Our first principal design is to use random linear classifiers (RLCs) as the weak classifiers. For a linear classifier $f(x) = w^T x$, where $x$ is the feature vector and $w$ is the parameter vector to learn, an RLC $w$ to be random using a specific generation method independent of training data. This random generation of the classifier dramatically simplifies the training step and the only requirement is to determine whether the random classifier is a valid weak classifier (e.g., accuracy $> 50\%$). In experiments, we found that our random RLC generation method works satisfactorily - for every 1-2 random tries we can find a valid weak classifier. The resulting boosting models are comparable to those generated by using decision stumps as base classifiers. The use of RLC also allows us to conveniently protect feature vectors and labels and to greatly reduce the costs of other related steps.

We have designed two privacy-preserving constructions to implement the RLC-based boosting framework to understand the effect of different cryptographic primitives on the associated complexities and expenses. The constructions are based on the non-colluding honest-but-curious SP-CSP setting that has been used by recent related work [12], [13], [15]. CSP is a cryptographic service provider that will be responsible to manage encryption keys and assist SP with the intermediate steps of the boosting framework. SP takes over the major computation and storage burden but is not interested in protecting user privacy. Both of our protocols result in models with distributed parameters between the SP and the CSP: the SP holding the RLCs’ parameters and the CSP holding the base classifier’s weights of the boosted models. An alternate setting (i.e., our SecSh setting) is that two servers take an equal share of computation and storage. For simplicity, we unify the two settings to SP-CSP.

The first construction of SecureBoost, “HE+GC”, uses the additive [27] or somewhat homomorphic encryption schemes (e.g., ring-LWE scheme [28]) to protect user-generated training data. HE enables homomorphic additions and multiplications of encrypted data by SP. A garbled-circuit (GC) based component enables CSP to securely learn the correctness of each base classifier’s prediction on each training example.

The second construction, “SecSh+GC”, uses the secret-sharing scheme to protect user data. Instead of using encryption, the user splits a training example into two random shares (the sum of which recovers the original training example) and distribute them to SP and CSP. SP and CSP use the recently developed protocols [14], [15] to collaboratively compute additions and multiplications with preserved privacy. It saves users’ submission and SP’s storage costs, however, increases the computational burden on both SP and CSP.

Both constructions of SecureBoost expose a leakage function to CSP - the correctness of RLC’s prediction on training examples. We carefully analyze the security of the constructions, based on the universally composable (UC) security paradigm [29], [30], and show that no additional information is leaked except for CSP knowing the leakage function. We also analyze the leaked information of the function and show that it is safe to use under our security assumption.

We summarize the unique contributions as follows.

- We propose to use random linear classifiers as the base weak classifiers to effectively simplify the protocol design while still preserving the quality of boosting model.
- We have developed two hybrid constructions: HE+GC and SecSh+GC, with the combination of security primitives such as GC, SHE, Secret Sharing, AHE, and random masking, with several optimization methods such as the use of minimized GC circuits.
- Our framework provably preserves the privacy of users’ submitted data, including both feature vectors and their associated labels, from both SP and CSP, with a leakage function only known by CSP. In addition, our protocols generate model parameters distributed to SP and CSP, which effectively prevents model-based attacks by a curious SP or CSP.
• We have conducted an extensive experimental evaluation of the two SecureBoost constructions with both synthetic and real datasets to fully understand the costs and associated tradeoffs.

II. PRELIMINARY

We use lowercase letters for vectors or scalars; capital letters for matrices, and large integers; single indexed lower case letters for vectors. Let \( \mathbb{R} \) be all real numbers and \( \mathbb{Z} \) be all integers.

A. Boosting

Boosting is an ensemble strategy \cite{31} that generates a high-quality classifier with a linear combination of \( \tau \) weak base classifiers (whose prediction power is slightly better than random guessing). Specifically, given training examples \( \{(x_i, y_i), i = 1 \ldots n\} \), where \( x_i \) are feature vectors and \( y_i \) are labels, it learns a model \( H(x) = \sum_{t=1}^{\tau} \alpha_t h_t(x) \), where \( h_t \) is a weak classifier that outputs the prediction \( \hat{y} \) for the actual label \( y \) and \( \alpha_t \) is the learned weight for \( h_t \). Algorithm 1 outlines the boosting algorithm for the two-class problem. The most popular weak classifier has been the decision stump \cite{21}, which is merely based on conditions like if \( X_j < v_j \), output \( 1 \); otherwise, \(-1\), where \( X_j \) is a certain feature and \( X_j < v_j \) is some optimal split that gives the best prediction accuracy among all possible single-feature splits for the training dataset.

Algorithm 1 Boosting(\( T, \tau \))

1: input: training data samples \( T = \{(x_i, y_i), i = 1 \ldots n\} \), \( x_i \in \mathbb{R} \) and \( y_i \in \{1, -1\} \), number of base classifiers: \( \tau \)
2: Initialize the sample weights \( \delta_{i1} \leftarrow 1/n \) for \( i = 1 \ldots n \);
3: for \( t \leftarrow 1 \) to \( \tau \) do
4: learn a weak classifier \( h_t(x) \) with sample weights \( \delta_{i,t} = 1 \ldots n \);
5: for \( i \leftarrow 1 \) to \( n \) do
6: \( e_{t,i} = 1 \) if \( h_t(x_i) == y_i \) else 0;
7: end for
8: \( \text{error} = \sum_{i=1}^{n} e_{t,i} / \text{error} \);
9: \( \alpha_t = \ln((1-\text{error}) / \text{error}) \);
10: \( \delta_{i,t+1} = \delta_{i,t} \exp(\alpha_t e_{t,i}) \) for \( i = 1 \ldots n \);
11: \( \delta_{i,t+1} = \delta_{i,t+1} / |\delta_{i,t+1}| \);
12: end for
13: Output: \( H(x) = \sum_{t=1}^{\tau} \alpha_t h_t(x) \)

B. Additive Homomorphic Encryption

For any two integers \( \alpha \) and \( \beta \), an AHE scheme allows the additive homomorphic operation: \( E(\alpha + \beta) = f(E(\alpha), E(\beta)) \) where the function \( f \) works on encrypted values without decryption. For example, Paillier encryption \cite{27} is one of the most efficient AHE implementations. Conceptually, with one operand, either \( \alpha \) or \( \beta \), unencrypted, we can derive the pseudo-homomorphic multiplication, e.g., \( E(\alpha \beta) = E(\sum_{i=1}^{\beta} \alpha) \). Similarly, we can derive pseudo-homomorphic vector dot-product, matrix-vector multiplication, and matrix-matrix multiplication, as long as one of the operands is in plaintext. The unencrypted operand either needs to be non-sensitive or must be protected with some masking and de-masking mechanism.

C. RLWE Homomorphic Encryption

The RLWE scheme is based on the intractability of the learning-with-error (LWE) problem in certain polynomial rings \cite{32}. It allows both homomorphic addition and multiplication.

1 Paillier encryption allows more efficient multiplication.
cannot scale with public clouds. It mainly assists SP in intermediate steps, e.g., encrypting/decrypting intermediate results and constructing garbled circuits. CSP is allowed to learn some leakage function but remains oblivious to users’ data. The concept of CSP has been used and justified by other approaches \cite{12, 13} as a practical setting.

A. SecureBoost Learning Protocol

In this section, we describe the rationale and benefits of using RLCs as the base classifiers, the major components of the SecureBoost protocol, and the security goals.

1) RLCs as Base Classifiers.: The original boosting framework uses decision stumps as the base classifiers. However, it is expensive to implement their privacy-preserving versions as shown in previous studies \cite{26}, as each decision stump involves scanning through all the possible splits for all the features in the training data. Instead of using decision stumps, we choose randomly generated linear classifiers (RLC) as the base classifiers in SecureBoost. An RLC generates a classification plane in the form of $w^T x + b$ with randomly selected $w$ and $b$. Apparently, most random settings of $w$ and $b$ do not generate meaningful classifiers. As Figure 2 shows, the generated plane needs to shatter the training data space into two partitions of significant sizes. For this purpose, we require the submitted data be normalized so that the training vectors are distributed near-2, 2, i.e., each dimension $X_i$ is normalized with $(X_i - \mu_i) / \sigma_i$, where $\mu_i$ is the mean and $\sigma_i$ is the standard deviation of the dimension $X_i$, most dimensional values should be in the range $[-2, 2]$. Thus, we can choose $b$, the intercept, in the range $[-2, 2]$ , while each element of $w$ is chosen uniformly from $[-1, 1]$. With this setting, we find in our experiments that a valid random linear classifier can be found in about 1-2 trials. Note that $\mu_i$ and $\sigma_i$ can be roughly estimated with simple sampling and aggregation protocols from users’ submissions. For clarity, we ignore the details of these simple protocols.

![Fig. 2. Effective Random Linear Classifier Generation](image)

The use of RLCs has several unique benefits. First, it enables learning with both the feature vectors and labels that are protected. We can transform the training data as $x \leftarrow (x, 1)$ and $w \leftarrow (w, b)$, with which the hypothesis function simply changes to $h(x) = w^T x$. For a two-class problem with labels $y \in \{-1, 1\}$, if the result $h(x)$ gives a correct prediction, i.e., the same sign as the label $y$, we always get $h(x)y = w^T xy > 0$; otherwise $w^T xy \leq 0$. Note that $xy$ stays together in the evaluation, and thus users can submit the encrypted version of $xy$, protecting both feature vectors and labels. Second, it simplifies the learning of base classifiers. As $w$ is randomly generated, there is no need for SP to consider sample weights during learning. Meanwhile, the learning of the $\alpha_i$ weights can be individually done by CSP. Finally, this process allows only the CSP to learn the weights of base models, and SP to learn the base classifiers, preventing either party from freely using the final model.

We verified with our experiments that boosting with RLCs can generate high-quality models comparable to those based on boosting with decision stumps. Moreover, boosting with RLCs seems to outperform boosting of other more optimized linear classifiers such as perceptrons. The possible reason is that these optimized classifiers tend to be more stable with weighted learning examples than random classifiers and decision stumps. As a result, the boosted model becomes a linear combination of highly similar base classifiers, hence not giving expected good performance.

2) SecureBoost Protocol: The SecureBoost learning protocol is defined with a 4-tuple: $\text{SB-Learning} = \{\text{Setup}, \text{BaseApply}, \text{ResultEval}, \text{Update}\}$. Algorithm 2 shows the use of these components in the boosting framework. For a boosted model $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, SP learns the base models $\{h_t(x) = w_t^T x, t = 1..T\}$, and CSP learns the model weights $\{\alpha_t, t = 1..T\}$.

![Algorithm 2 SecureBoost Framework](image)

(K, E(Z), \{w_i, i = 1..p\}, \delta_t) \leftarrow \text{Setup}(1^k, \tau, p); (1) The key $K$ is generated by a certain party or parties (CSP, SP, or both) as required, with the desired security level $1^k$ and all public keys are published. (2) CSP initializes $\delta_1$ with $1/n$. (3) The training data $Z$ of $n$ instances contains row vectors $z_i = x_i y_i$, which is protected with either a public-key encryption scheme or random masking (e.g., in the secret-sharing construction) to generate $E(Z)$. (4) SP sets the desired number of classifiers, $\tau$, and generates a pool of prospective RLCs with parameters $w_t$ for $t = 1..p$, where $p$ is the pool size proportionally larger than $\tau$, e.g., $p = 1.5\tau$.

$\{E(h_t(x_i)), i = 1..n\} \leftarrow \text{BaseApply}(K, E(Z), w_t)$: With the encrypted training data $E(Z)$ and a model parameter $w_t$, the procedure will output the model $h_t$’s encrypted prediction results on all training instances.

$I_t \leftarrow \text{ResultEval}(K, \{E(h_t(x_i)), i = 1..n\})$: With the encrypted prediction results, ResultEval allows CSP (not SP) to learn the indicator vector $I_t$ of length $n$, indicating the correctness of $h_t$’s prediction for each training instance.
(\delta_{t+1}, \alpha_t, e_t) \leftarrow \textbf{Update}(\delta_t, I_t):$ CSP takes $I_t$, $\delta_t$ to compute the weighted error rate $e_t = I_t^T \delta_t$ and if $h_t$ is a valid base classifier i.e., accuracy $> 50\%$, updates its weight $\alpha_t = 0.5 h_t(1 - e_t)/e_t$ and computes $\delta_{t+1}$ for the next iteration with sample weight updating formula.

In the end, SP learns $\{w_t, t = 1..p\}$ and CSP learns $\{e_t, t = 1..p\}$. A two-party function evaluation protocol can be easily developed for SP to apply the model for classification, which however, is not the focus of this paper.

B. Security Model

We make some relevant security assumptions here: (1) Both SP and CSP are honest-but-curious parties, i.e., they follow the protocols exactly and provide services as expected. However, they are interested in users’ data. (2) SP and CSP do not collude. (3) All infrastructures and communication channels are secure. With these assumptions, we consider the curious SP and CSP the major adversaries in our framework. While the integrity of data and computation is equally important, we consider it orthogonal to our study. We are mainly concerned with the privacy of the following assets.

1. Privacy of training data. User-generated training data may include personal sensitive information. We consider both feature values and the labels sensitive. For example, a user’s fitness activity dataset may contain sensitive features such as heart rate and locations, while the labels, i.e., the type of activity, may imply their activity patterns and health conditions.

2. Privacy of prediction models. Model parameters are split and distributed between SP and CSP. No single party can learn the complete model. The split model allows CSP to restrict SP from misusing/exploiting the framework to carry model-based attacks.

We adopt the universally composable (UC) security [29, 30] to formally define the protocol security. We consider an ideal protocol $\pi$ implementing the ideal functionality $F$ corresponding to a SecureBoost protocol, involving SP and CSP. In the Real world, an honest-but-curious adversary $\mathcal{A}$ can corrupt any of the parties and gain access to all the inputs and outputs of that party. We say that $\pi$ securely realizes $F$ (or $\pi$ is UC-secure) if for any $\mathcal{A}$ in real world there exists an ideal-process simulator $\mathcal{S}$ in ideal world running probabilistic algorithms in polynomial time (i.e., PPT), such that for any environment $\mathcal{E}$ and inputs $m = (m_Z, m_{\mathcal{A}/S}, m_{SP/CSP})$,

$$|Pr(Real_{\pi, A, E}(k, z, m) = 1) − Pr(Ideal_{F, S, E}(k, z, m) = 1)| = negl(k),$$

where $negl(k)$ is a negligible function [37]. In Section VII we propose two theorems that can be proved to show that SecureBoost protocols are UC-secure.

IV. CONSTRUCTION WITH HE AND GC

In this section, we present the homomorphic encryption (HE) and GC based construction of SecureBoost. The use of RLCs greatly simplifies the constructing crypto operations. With the HE encrypted data, the $BaseApply$ procedure is essentially the homomorphic operation $E(Z)w_t$ that is allowed by both Paillier [27] and RLWE [32] cryptosystems. We use a garbled-circuit based protocol to allow only CSP to learn the indicator vector $I_t$, without leaking any other information to the parties. In the following, we first describe the construction of the protocol components and then discuss several key technical details.

Setup. CSP generates the HE public and private key and distributes the public key to the users and SP. Users encrypt their submission. SP generates the pool of $p$ prospective weak classifier vectors, $\{w_t, t = 1..p\}$.

$BaseApply$. With the matrix-vector homomorphic operations enabled by HE, SP computes $E(u_t) = E(Z)w_t$, $t = 1..p$. As this step can be done locally by SP, SP can conduct this work offline before the protocol interactions start.

$ResultEval$. The problem setting is that SP holds $E(u_t)$ and CSP wants to know the sign of each element of $u_t$, i.e., $Zw_t > 0$ implying correct prediction by the RLC, which sets the corresponding element of $I_t$ to 1; otherwise to 0. The sign of element is related to the specific integer encoding, which we will elaborate more. With our encoding scheme, we only need to check a specific bit to determine whether $Zw_t > 0$ is true. To satisfy all the security goals, we decide to use a GC protocol for this step that will be discussed in more detail.

As the last step $Update$ does not involve crypto operations, we can skip its discussion. Algorithm [3] summarizes the operations in this construction and Figure 3 depicts all the associated interactions between SP and CSP in this construction. We call this construction “HE+GC” and depending on the HE scheme used we may call it “Paillier+GC” or “RLWE+GC” for convenience.

Algorithm 3 HE+GC based SecureBoost

1: $\textbf{Setup} (1^l, \tau, p)$: CSP uses an HE scheme to generate a private and public key pair. CSP initializes $\delta_1$. Users use the public key to encrypt data, and SP generates pool of $w_t$;
2: for $t \leftarrow 1$ to $p$ do
3: \hspace{1em} $\textbf{BaseApply}(K_t, E(Z), w_t)$: SP computes $E(u_t) = E(Z)w_t$, which can be done offline in batch;
4: \hspace{1em} $\textbf{ResultEval}(K_t, E(u_t))$:
5: \hspace{2em} SP perturbs $E(u_t)$ with random noise $\lambda_t$, as $E(u_{t,1}) = E(u_t) + E(\lambda_t)$ and sends $E(u_{t,1})$ to CSP; CSP decrypts it;
6: \hspace{2em} CSP constructs and SP evaluates the garbled circuit that de-noises $u_{t,1}$ as $u_t = u_{t,1} - \lambda$, and returns the indicator vector $I_t$ to CSP;
5: \hspace{1em} $\textbf{Update}(K_t, \delta_t, I_t)$;
6: if $\tau$ effective base models have been found then
7: \hspace{1em} stop the iteration;
8: end if
9: end for

Fig. 3. SP-CSP interactions in HE+GC construction. SP-CSP interactions in the SecSh+GC construction. $E_1$ represents HE encryptions whereas $E_2$ represents GC labels for the GC outputs.

A. Technical Detail

In this section we discuss several key problems mentioned in the sketch of the construction above.
Choice of HE Schemes. We consider two choices of HE: Paillier \cite{27} and RLWE \cite{32} in our evaluations. Paillier scheme provides a large bit space allowing to preserve more precisions in floating-integer conversion. Our evaluation shows that with message packing, all RLWE operations including encryption, decryption, addition and one-level multiplication are much faster than Paillier, although the ciphertext size might be larger than that of Paillier.

Integer Conversion. The HE schemes work on integers only. For a floating-point value \( x, x \in \mathbb{R} \), to preserve \( m \)-digit precision after the decimal point upon conversion and recovery, we have: \( v = \lfloor 10^m x \rfloor \mod q \), where \( q \) is a large integer such that \( 10^m x \in (-q/2, q/2) \). Let the modulo operation map the values to \([0, q)\), in such a way that the negative values are mapped to the upper range \((q/2, q)\). It is easy to check that \( x \) is recoverable: if \( v > q/2, x \approx (v - q)/10^m \); otherwise, \( x \approx v/10^m \). The modulo additions and multiplications preserve the signs and are thus recoverable. Furthermore, this encoding simplifies the evaluation of the RLC base classifiers, which involves checking the sign of \( h_t(x) \). Let \( b \) be the total number of bits to represent the values in \([0, q)\). It is trivial to learn that if the \( b \)-th bit of a value in the range \([0, q)\) is 1, then the value is in the range \((q/2, q)\), which is negative; otherwise the value is positive. With large enough \( q \) we can accommodate the desired multiplication and addition results without overflow. An \( n \)-bit plaintext space that allows one multiplication followed by \( \alpha \) additions, as used in our protocol, spares \((n - \alpha)/2\) bits to encode the original value. For easier processing, we normalize the original real values in the same dimension of training data before converting them to \( b \) bit integers.

Secure Matrix-Vector Multiplication. The core operation \( E(Zw_t) \) involves encrypted \( E(Z) \) and SP randomly generated plaintext \( w_t \). Thus, both AHE and SHE schemes can be applied.

Securely Checking Signs of \( E(u_t) \). CSP needs to check the result of base classifier prediction, \( E(u_t) = E(Zw_t) \) to learn the correctness of prediction on each instance, so that the error rate, the model weight, and the sample weight update can be computed. With the described integer conversion encoding method, the sign checking \( u_{t,i} < 0? \) is determined by a specific bit in the result. Note that letting CSP know \( u_t \) directly may reveal too much information significantly weakening the security. To balance between security and efficiency, we decide to let CSP only learn the signs indicating if the base classifier \( h_t \) correctly classified the training instances, and nothing else is leaked. Lu et al. \cite{11} have proposed a comparison protocol based only on RLWE, however, it is extremely expensive to be adapted to our framework. Therefore, we rely on a noise addition procedure to hide the decrypted \( u_t \) from CSP and a GC-based denoising and bit extraction procedure to let CSP learn the specific bit for sign checking. We give the details of these procedures next.

To hide the plaintext \( u_t \) from CSP, we use a noise addition method that can be easily implemented by SP on the encrypted vector with homomorphic addition: \( E'(u_{t,0}) = E(u_{t,0}) + E(\lambda_t) \), where \( \lambda_t \) is a noise vector generated by the pseudo-random number generator \( G \). Then, CSP can decrypt \( E(u_{t,0}) \) to learn the noisy result. Let \( u_{t,1} = \lambda_t \) held by SP. Now the problem is turned to using a GC to securely compute \( u_t = u_{t,0} - u_{t,1} \) and return the specific bit of each element of \( u_t \).

![Fig. 4. GC-based sign checking protocol.](image)

Figure 4 shows the GC based denoising and bit extraction protocol. CSP’s input to the circuit is the binary form of \( u'_t \) elements whereas SP’s inputs are the binary form of \( \lambda_t \) elements. With associated oblivious transfer (OT) protocol and wire label transfers, the circuit can securely evaluate \( u'_{t} - \lambda_t \) and extract the most significant bit, \( msb(u'_{t,j}), j = 1..n \), of the result without leaking anything else. SP evaluates the circuits and returns the extracted encrypted bits (represented as output labels in GC) to CSP. CSP can then decrypt (re-map) the labels to generate the indicator vector \( I_t \).

V. CONSTRUCTION WITH SECSh AND GC

Alternatively, we design our framework with a mixture of secret sharing and garbled circuit techniques. A somewhat similar approach was taken by \cite{15} in constructing privacy-preserving gradient-descent based learning. It differs from the HE based construction in two aspects: 1) user data protection uses secret sharing, and 2) matrix-vector multiplication happen over secret random splits of training data held by SP and CSP.

Instead of encryption, users randomly split their training data into two shares, one for SP and the other for CSP. The sum of shares recovers the original values. Any intermediate results that need protection are also in the form of random shares distributed between SP and CSP. As a result, multiplication of two values, say, \( a \) and \( b \), each as random shares (e.g., SP holds \( a_0 \) and \( a_1 \) while CSP holds \( b_0 \) and \( b_1 \), where \( a_0 + a_1 = a \) and \( b_0 + b_1 = b \)), needs the help of AHE encryption to compute each party’s random share for \( ab \). As for sign checking, we reuse the GC protocol designed earlier for HE+GC.

Setup. Each user splits their data \( Z \) into a random matrix \( Z_0 \) and \( Z_1 \), where \( Z_1 = Z - Z_0 \), and securely distributes \( Z_0 \) to SP and \( Z_1 \) to CSP. SP also generates a key pair for a chosen AHE scheme and shares the public key with CSP.

BaseApply. With SP holding \( Z_0 \) and \( w_t \), and CSP holding \( Z_1 \), BaseApply will generate random shares of the result \( u_t = Z w_t = u_{t,0} - u_{t,1}; u_{t,0} \) and \( u_{t,1} \) held by SP and CSP, respectively. This is implemented with a special matrix-vector multiplication algorithm, which we will describe later.

ResultEval. With the random shares: \( u_{t,0} \) and \( u_{t,1} \) held by SP and CSP respectively, we can apply the same GC protocol presented in the last section for computing \( u = u_{t,0} - u_{t,1} \) and extracting the specific bits.

Algorithm 4 outlines the construction and Figure 5 summarizes all the interactions between SP and CSP. We call this construction “SecSh + GC” henceforth.
Algorithm 4 SecSh+GC based SecureBoost

1: **Setup:** SP and CSP possess their share of training data $Z_0$ and $Z_1$ respectively; SP generates $p > \tau$ many random vectors $w_i$; SP also generates an AHE key pair, and distributes the public key to CSP; CSP initializes the sample weights $\delta_{i,t} = 1/n$ for $i = 1 \ldots n$
2: for $t \leftarrow 1$ to $p$ do
3: **BaseApply:** apply the random-share based matrix-vector multiplication; in the end, SP holds $u_{t,0}$ and CSP holds $u_{t,1}$
4: **ResultEval:** use the GC protocol to compute the subtraction of $u_{t,0}$ and $u_{t,1}$ and extract the specific bits of $u_t = u_{t0} - u_{t1}$
5: **Update:** $(K, \delta_{t} , I_t)$;
6: if $\tau$ effective base models have been found then
7: stop the iteration;
8: **end if**
9: **end for**

Fig. 5. SP-CSP interactions in the SecSh+GC construction. $E_1$ represents HE encryptions whereas $E_2$ represents GC labels for the GC outputs.

A. Technical Detail

The SecSh+GC construction reuses the integer conversion and the GC-based sign checking components. Here, we focus on the major difference: the protocol for computing matrix-vector multiplication with random shares.

Random-Share-Based Matrix-vector Multiplication. To initiate, SP and CSP respectively hold the two shares $Z_0$ and $Z_1$ of user data in plaintext, and SP also holds $w_t$ in plaintext. The goal is to derive random shares of $Z w_t$ and each party learns only one of the shares.

SP computes the part $Z_0 w_t$ in plaintext by itself. The challenge is to collect the other part $Z_1 w_t$ without CSP knowing $w_t$ and no party knowing the complete result of $Z w_t$. We use the following procedure to achieve this security goal. (1) SP encrypts $w_t$ with an AHE scheme and sends $E(w_t)$ to CSP so that CSP can apply pseudo-homomorphic multiplication to compute $E(Z_1 w_t) = Z_1 E(w_t)$. (2) CSP generates a random vector $\lambda_t$ with the pseudo-random number generator $\mathcal{G}$, encrypts it with the public key provided by SP, and apply homomorphic addition to get $E(Z_1 w_t + \lambda_t)$, which is sent back to SP. (3) SP decrypts it and sums up with the other part $Z_0 w_t$ to get $Z w_t + \lambda_t$. In the end, SP gets $u_{t,0} = Z w_t + \lambda_t$ and CSP gets $u_{t,1} = \lambda_t$. At this point SP and CSP can use the GC protocol for sign checking developed in Section IV.

VI. COST ANALYSIS

Table I summarizes the associated big-O estimation of communication and computation broken down into different operations/components.

HE+GC Costs. For $n$ records with $k$ features, each user performs $nk$ (Paillier) or $[nk/h]$ (RLWE) encryptions where $h$ is the number of messages packed in one ciphertext, in forming encrypted matrix $Z$. The user’s upload cost is exactly $nk$ (Paillier) or $[nk/h]$ (RLWE) ciphertexts. For evaluating $p$ RLCs over encrypted training data $Z_{nxk}$ SP overtakes $O(pnk)$ homomorphic multiplications and additions when using Paillier. If using RLWE, SP performs $pk$ full replication operations and $O(p[n/h]k)$ homomorphic multiplications and additions to perform the matrix vector multiplication.

In passing the masked RLC evaluation results to CSP, SP takes $p[n/h]$ (RLWE) or $pn$ (Paillier) homomorphic additions while adding noise vector $\lambda_t$ to $u_t = Zw_t$. If using RLWE, SP also needs to encrypt $w_t$ first to do the homomorphic additions, an additional cost of $p([n/h][k/h])$ encryptions. However, there is no need to encrypt the weight vectors if Paillier is used. After receiving the data, CSP conducts $p[n/h]$ RLWE or $pn$ Paillier decryptions.

The computational and communication costs on GC are all determined by the size of GC. For $b$-bit addition/subtraction, the size of GC is linear to $b$. Specifically, CSP will need $O(pnb)$ symmetric encryptions to construct the GC and SP takes the same number of decryptions in evaluating the GC. In addition, GC of size $O(pnb)$ should be transferred to SP.

SecSh+GC. This construction relieves the end users from any sort of cryptographic operations. Users’ only cost is uploading $Z_0$ and $Z_1$. In matrix-vector multiplication, SP takes $pk$ encryptions in encrypting $w_t$ and passes the $pk$ ciphertexts to CSP. CSP then takes $pn$ encryptions to generate the noise vectors and conduct the homomorphic operations $Z_1 w_t + \lambda_t$, which include $pnk$ homomorphic multiplications and additions in total. Finally, CSP sends back the $pn$ ciphertexts and SP needs $pn$ decryptions to get the random shares. As this protocol shares exactly the same GC component in HE+GC, we ignore the GC cost analysis.

In summary, we observe that HE+GC constructions demand no CSP storage and CSP only needs to conduct decryptions and GC constructions. In contrast, in SecSh+GC, the workload and storage is almost equally distributed to SP and CSP. However, as user-generated data is not encrypted but split into random shares in SecSh+GC, users’ costs and overall storage costs are much lower.

VII. SECURITY ANALYSIS

According to the security model outlined in Section III-B, we focus on the subcomponents of the protocols that involve both SP and CSP and implement a specific ideal function $\mathcal{F}$. The security is proved by finding a simulator $\mathcal{S}$ in the ideal scenario corresponding to the adversary $\mathcal{A}$ in the real scenario such that the environment $\mathcal{Z}$ cannot distinguish the probabilistic outputs of Ideal and Real.
The major interaction happens in computing the indicator vector \( I_t \) for an iteration \( t \). The corresponding ideal function is defined as \( \mathcal{F}(m_{SP,t}, m_{CSP,t}) \rightarrow I_t \), where \( m_{SP,t}, m_{CSP,t} \) are SP and CSP’s inputs to the function and the function’s output is the indicator vector \( I_t \) as defined by our protocols. We present two theorems next, along with their sketch proofs.

**Theorem 1:** If the random number generator \( \mathcal{G} \) is pseudo-random, and both the HE scheme and GC are CPA-secure, then the HE+GC construction of SecureBoost is secure in computing \( I_t \) with an honest-but-curious adversary.

**Sketch Proof for Theorem 1.**

In the real protocol, the environment machine \( Z \) can observe the inputs \( i_{SP,t} = (E(Z), w_t, GC_t, Label_{CSP,t}) \) and \( i_{CSP,t} = (E(Zw_t + \lambda_t), Label_{output,t}, GC_{key}) \), and outputs: \( o_{SP,t} = (\lambda_t, Label_{output,t}) \) and \( o_{CSP,t} = (GC_t, I_t) \), where \( Label \) refers to the GC wire labels. By the security definition of HE, GC, and pseudo-randomness, the variables except for Label \( Z \) is defined as uniformly random values in polynomial time. Therefore, both worlds compute the same \( I_t \), but \( Z \) cannot distinguish the outputs Ideal and Real computationally from what it has observed. For a compromised CSP, \( S \) does the same operation, i.e. generate a random vector \( \lambda_t \) using \( \mathcal{G} \) and computes \( E(Zw_t + \lambda_t) \) which is sent to SP and simply forwards \( \lambda_t \) to the ideal functionality. This proves that our protocol \( \pi \) UC-realizes the ideal functionality \( \mathcal{F}(m_{SP,t}, m_{CSP,t}) \rightarrow I_t \).

**Theorem 2:** If the random number generator \( \mathcal{G} \) is pseudo-random, both the AHE scheme and GC are CPA-secure, then the SecSH+GC construction is secure in computing \( I_t \) with an honest-but-curious adversary.

**Sketch Proof for Theorem 2.**

In the real protocol, the environment machine \( Z \) can observe the inputs \( i_{SP,t} = (Z_0, w_t, E_1(Zw_t) + \lambda_t, GC_t, Label_{CSP,t}) \) and \( i_{CSP,t} = (Z_1, E_1(w_t), Label_{output,t}, GC_{key}) \), and outputs: \( o_{SP,t} = (Label_{output,t}) \) and \( o_{CSP,t} = (\lambda_t, GC_t, I_t) \), where \( Label \) refers to the GC wire labels. By the security definition of AHE, GC, and pseudo-randomness, the variables except for the plaintext \( w_t \) and \( I_t \) cannot be distinguished from uniformly random values in polynomial time by the environment \( Z \).

Now, let us define an ideal world parallel to this real protocol. We design an ideal functionality \( \mathcal{F} \): with the inputs \( m_{SP,t} = Zw_t + \lambda_t \) and \( m_{CSP,t} = E(\lambda_t) \), it decrypts and removes the noise \( \lambda_t \), and computes the indicator vector \( I_t \). Assuming \( A \) compromises SP in the real world, we can think of a simulator \( S \) in the ideal world as follows. \( S \) generates a random vector \( \lambda_t \) using \( \mathcal{G} \) and computes \( Zw_t + \lambda_t \) which is passed on to the ideal functionality. Again, if \( \mathcal{G} \) is pseudo-random and AHE is CPA-secure, it is impossible for \( Z \) to distinguish \( \lambda_t \) and \( E(Zw_t + \lambda_t) \) from uniformly random values in polynomial time. Therefore, both worlds compute the same \( I_t \), and both worlds compute the same \( I_t \), but \( Z \) cannot distinguish the outputs Ideal and Real computationally from what it has observed. For a compromised CSP, \( S \) does the same operation, i.e. generate a random vector \( \lambda_t \) using \( \mathcal{G} \) and computes \( E(Zw_t + \lambda_t) \) which is sent to SP, and simply forwards \( \lambda_t \) to the ideal functionality. This proves that our protocol \( \pi \) UC-realizes the ideal functionality \( \mathcal{F}(m_{SP,t}, m_{CSP,t}) \rightarrow I_t \).

**A. Implication of Revealing \( I_t \) to CSP.**

CSP learns the indicator function \( I_t, i(h_t(x_i) == y_i) \), for \( i \in 1..n \) in the iteration \( t \) of SecureBoost. It is clear that this leakage does not help CSP learn the complete boosted model \( H(x) \) as long as SP holds \( \{w_t, t = 1..T\} \) as secrets. However, we must understand if such leakage may help CSP learn anything about the training data.

Recall that an element of indicator vector \( I_t, i(h_t(x_i) == y_i) \) represent if the base RLC \( h_t \) classifies the training instance \( x_i \) correctly or incorrectly (1 and 0, respectively). At the end of learning, each record \( x_i \) gets \( p \) prediction results for \( p \) base classifiers \( h_{t,1}, t = 1..p \), respectively, which is denoted as \( c_i = (c_{i,1}, \ldots, c_{i,p}) \), \( c_{i,j} \in \{0,1\} \). Let \( c_i \) be the characterization vector for the record \( x_i \). The intuition tells that two similar records (i.e. relatively small Euclidean distance) with the same label will get an identical characterization vector with high probability. If this is true, this knowledge can be utilized by a malicious CSP, which is, however, excluded by our semi-honest assumption, injecting anchor vectors to the training dataset and estimating those similar to the anchor vectors.

However, our experiments show that the above intuition deviates from the reality dramatically. In particular, for the record pairs having identical characterization vectors, their average distances and standard deviations are not much different from other random pairs (Figure 17 in Section VIII). Note that CSP knowing the leakage function is a trade-off made between efficiency and security. We will study the candidate methods for further disguising \( I_t \) and evaluate the additional costs as future work.

**VIII. Experiments.**

We design our experiment set on both real and synthetic datasets with three goals: (1) show random linear classifiers are effective weak classifiers for boosting; (2) evaluate associated computation, communication, and storage costs, and their distributions amongst the users, SP, and CSP for both the constructions; and (3) understand the trade-off between costs and model quality, including a comparison with another state-of-the-art privacy-preserving classification learning framework.

**Implementation.** We adopt the HELib library [38] for the RLWE encryption scheme, implement the Paillier cryptosystem [27] for the AHE encryption scheme, and use the ObliVM (oblimv.com) library for the garbled circuits. ObliVM has included the state-of-the-art GC optimization techniques such as half AND gates, free XOR gates, and OT extension.
The core algorithms for data encoding, encryption, matrix-vector multiplications, and additive perturbation are implemented with C++ using the GMP library. Users’ submissions are encoded with the 7-bit floating-integer conversion method (Section IV-A). We use the scikit-learn toolkit (scikit-learn.org) to evaluate the model quality for existing classifier learning methods selected for comparison purpose.

**Parameter selection.** We pick cryptographic parameters corresponding to 112-bit security. The RLWE parameters allow 32-bit message-space overall, 1 full vector replication, and at least 2 levels of multiplication. The degree of the corresponding cyclotomic polynomial is set to $\phi(m) = 12,000$ and $e = 7$ modulus switching matrices, which gives us $h = 600$ slots for message packing. The Paillier cryptosystem uses 2048-bit key-size to achieve approximately 112-bit security. Our GC-based sign checking protocol accommodates $(2b + \log_2(k))$-bit inputs, where $b$ is the bit-precision (i.e., $b=7$ in experiments) and $k$ is the dimension of the training data. Note that HELib uses a text format to store the ciphertext which we zip to minimize the costs.

**Datasets.** We test SecureBoost with both the synthetic and real datasets. Table II summarizes the dataset properties. Datasets are selected to cover a disparate range of dimensions and number of instances. All selected datasets contain only two classes to simplify the evaluations. The real datasets come from the UCI Machine Learning Repository [39]. The synthetic dataset is deliberately designed to generate non-linearly separable classes. It is used to conveniently explore and understand the behaviors of RLC-based boosting and the quality of non-linear classification modeling methods.

### Table II. Dataset Statistics.

| Dataset   | Instances | Attributes | Adaboost Accuracy | Number of decision stumps |
|-----------|-----------|------------|-------------------|--------------------------|
| ionosphere| 351       | 34         | 92.02% +/- 4.26%  | 50                       |
| credit    | 1,000     | 24         | 74.80% +/- 3.50%  | 100                      |
| spambase  | 4,601     | 57         | 92.31% +/- 4.40%  | 75                       |
| epileptic | 11,500    | 179        | 86.95% +/- 3.40%  | 200                      |
| synthetic | 150,000   | 10         | 89.51% +/- 2.10%  | 75                       |

**A. Effectiveness of RLC Boosting**

The performance of boosting is characterized by the convergence rate and the final accuracy. The speed of convergence is directly related to the overall cost of the SecureBoost protocols. We look at the number of base classifiers ($\tau$) needed to attain a certain level of accuracy. As a randomly generated RLC may fail (i.e., RLCs having $\approx 50\%$ accuracy for the two-class datasets) and be discarded in some of the rounds, we also assess the actual number ($p$) of RLCs that are tried to generate the final model. All the accuracy results are for 10-fold cross-validation. The following results can be reproduced and verified with the scripts we have uploaded to https://sites.google.com/site/testsboost/.

Figure 6 analyzes the convergence of RLC-based boosting. The left subfigure identifies for each dataset the number of iterations (or base classifiers) that lead to a stable model accuracy level. We observe that overall only about 200 base classifiers are sufficient for the considered datasets. The right subfigure compares boosting with different base classifiers: RLC, decision stumps (DS), and linear means classifiers (LMC) with one another when learning on the synthetic dataset. Clearly, DS has the advantage of converging faster in about 75-80 rounds. On the other hand, boosting with LMC does not reach the desired accuracy, because the centers of class (i.e., the “means”) that are used to define the classification plane stay stable even with changed sample weights. The result is a bunch of highly similar base classifiers in the final boosting model, which does not take advantage of the boosting framework.

![Figure 6](image)

**Figure 6.** Convergence of boosting with RLCs (left). Convergence of boosting with RLCs, LMCs, and DSs for the synthetic dataset (right)

The left subfigure in Figure 7 shows the final model quality produced by RLC boosting and the DS boosting (i.e., the default boosting method). We use 200 RLCs and varying number of DSs as shown in Table III as the base classifiers for the datasets. In every case, both methods generate models with almost identical accuracy. All of the above results suggest that RLC boosting is a robust boosting method to get high-quality classification models.

**Encoding Bits.** The number of bits for encoding affects the cost of GC-related components and the precision in floating-integer conversion, which in turn affects the final model quality. The right subfigure in Figure 7 shows the effect of preserved bits on model accuracy. It seems preserving 7 bits is sufficient to get optimal quality models.

![Figure 7](image)

**Figure 7.** Model quality: boosting with random linear classifiers vs. boosting with decision stumps (left). Bit precision vs. model accuracy (right)

Table III reports the cost comparison of RLC and DS-based boosting just for the GC comparison portion of the protocol with the synthetic data encoded with 7 bits precision. The DS learning algorithm is based on Lindell et al. [26]. DS-based boosting incurs both computation and communication costs larger than that of the RLC-based by magnitudes. Note that we use 75 DS base classifiers to reach the optimal model quality as we mentioned. To be conservative in cost estimation, only 10 bins per feature is used for finding the optimal splits in learning decision stumps.
TABLE III. GC COMPARISON COST. BOOSTING WITH RLCs VS. DECISION STUMPS (DS). p- NUMBER OF BASE CLASSIFIERS

| Dataset  | RLC | DS     |
|----------|-----|--------|
| ionosphere | 244 | 75  |
| credit     | 8.51| 261.64|
| spambase    | 5,009.6| 1320.15|
| epileptic   | 87,549,500| 371.2|
| synthetic   | 8,840,000| 342|

B. Cost Distribution

We now inspect the associated costs for each involved party in the two constructions.

Table [IV] shows the parameter settings for different datasets that led to the desired model quality. \( \tau \) is the number of base classifiers in the final boosting model. \( p \) represents the total number of RLCs that are tried in the modeling process, which determines the actual protocol costs. Overall, in about 1-2 tries on average, we can find a valid RLC (with accuracy > 50%).

TABLE IV. SecureBoost parameter setting for cost evaluation. \( \tau \) and \( p \)- NUMBER OF DESIRED AND TRIED RLCs

| Dataset  | \( \tau \) | \( p \) | Accuracy |
|----------|----------|--------|----------|
| ionosphere | 200 | 226 | 91.5% +/- 3.1% |
| credit     | 200 | 342 | 73.4 % +/- 2.4 % |
| spambase    | 200 | 229 | 87.4 % +/- 4.8 % |
| epileptic   | 200 | 331 | 84.4% +/- 2.9 % |
| synthetic   | 200 | 244 | 87.91% +/- 3.2 % |

User’s Costs. A user’s costs depend on the size of training data, i.e. the number of training records \( n \) and the number of dimensions \( k \) per record. The Paillier+GC construction requires each user to encrypt their submission element-wise in a streaming or batched manner. The RLWE+GC construction requires each user to batch her submissions and encrypt them as a column-wise matrix \( E(Z) \) with message padding (see Section II-C). For the SecSh+GC construction, users simply apply the one-time padding method to generate the masks and distribute the splits to SP and CSP, respectively.

TABLE V. User’s Cost for a Batch of 600 Records

| Dataset   | HE+GC (RLWE / Pailler) | SecSh+GC |
|-----------|------------------------|----------|
| ionosphere| 1.34 / 235.83          | 0.04     |
| credit    | 1.09 / 168.45          | 0.03     |
| spambase  | 2.54 / 390.80          | 0.07     |
| epileptic | 7.91 / 1,212.84        | 0.09     |
| synthetic | 0.48 / 74.12           | 0.05     |

Table [V] depicts the user’s costs in encrypting and submitting one batch of records with the batch size \( h = 600 \). The HE+GC constructions are more expensive than SecSh+GC in all aspects, but still quite acceptable in most cases. RLWE+GC results in larger ciphertext but far less computations than Paillier+GC.

Data and CSP cost distribution. As SP’s and CSP’s costs are highly inter-related in the SecureBoost constructions we discuss them together. First, we analyze the cost growth of the construction for with increasing number of records and dimensions. Then, we analyze the shared GC components and the overall cost of the constructions with the selected real and synthetic datasets. Note: We use the Paillier cryptosystem in SecSh+GC as the required AHE scheme.

We try to understand the relationship between the size of training data and associated costs using synthetic datasets of several sizes and dimensions. First, we fix the number of dimensions \( k = 20 \) and see how number of records \( n \) affects the costs. Figure [S] (top) shows that both SP’s and CSP’s costs in RLWE+GC grow much slower than the other two’s. CSP’s costs are linear to the number of records \( n \) as they involve the same number of decryption operations. Figure [S] (bottom) depicts the effect of increasing the dimensions while fixing the number of records to \( n = 10,000 \). We observe that RLWE+GC cost for SP grows much faster for the larger dimensions. This is due to the associated dimension-wise RLWE replication cost in the matrix-vector multiplication. On the other hand, CSP’s cost when using RLWE+GC is much lower than with the other two constructions, as the RLWE decryptions are much cheaper than that of Paillier. Both SP and CSP’s costs when using Paillier+GC and SecSh+GC stay almost flat as only \( n \) dominates the overall cost.

Now we present the associated costs for the selected real and synthetic datasets listed in Table [II] as all the constructions share the same GC component for sign checking, we list the GC costs separately in Table [VI]. The number of AND gates represents the size of GC. The computational and communication costs include the total of SP’s and CSP’s. GC’s associated costs are linear to \( n \) and bit precision \( b \).

TABLE VI. Costs of the GC Component: Computation (Comp.) and Communication (Comm.)

| Dataset   | AND Gates | Comp. (M) | Comm. (MB) |
|-----------|-----------|-----------|------------|
| ionosphere| 2,016,846 | 5.1       | 43.1       |
| credit    | 8,840,000 | 20.3      | 371.2      |
| spambase  | 37,268,100| 47.2      | 1,202.6    |
| epileptic | 87,549,500| 101.3     | 5,009.6    |
| synthetic | 695,400,000| 927.4    | 39791.1    |

Finally, Table [VII] sums up the costs for all the components. For the smaller datasets, the RLWE+GC construction does not show much advantage over the other two. For datasets with the larger number of records such as the synthetic dataset, both SP and CSP take less computational time with RLWE+GC construction in comparison with the other two. For datasets with larger dimensions such as the epileptic dataset, RLWE+GC is more onerous to the SP whereas beneficial to the CSP in terms of computation cost. As for storage and communication costs, Paillier+GC and SecSh+GC are favorable across the board. By comparing Table [VII] and [VI], it is clear that the GC-component dominates the communication cost.
C. Comparing with Other Methods

In this section, we compare SecureBoost with the recently developed SecureML method [15]. It implements the stochastic gradient-descent (SGD) learning based on secret sharing [14], which is then used for logistic regression (LR) and neural network (NN) [31]. We tried different shapes of inner hidden layers and found the minimum-cost setting for satisfactorily handle the non-linearly separable synthetic dataset. SGD is conducted with a mini-batch size of 128 records in training. Both algorithms are run enough iterations until convergence.

Next, we compare the costs of learning focusing on the same synthetic dataset. Figure 10 shows that all the SecureBoost constructions are much more efficient than the neural network (NN) [31]. We tried different shapes of inner hidden layers and found the minimum-cost setting for satisfactorily handle the non-linearly separable synthetic dataset. SGD is conducted with a mini-batch size of 128 records in training. Both algorithms are run enough iterations until convergence.

D. Effect of Releasing $I_t$

We want to verify if similar characterization vectors infer similar training records to understand the leaked information by $I_t$. Figure 11 measures the average euclidean distances between the training record pairs corresponding to the characteristic vectors differing by $k$ bits. It is evident that the similarity of characterization vectors does not infer similarity of training records.

### Table VII. Overall SP and CSP Costs: Storage, Comp. (Computation cost), Comm. (Communication cost)

| Dataset | Storage (MB) | Comp. (minutes) | Comm. (MB) | St. (MB) | Comp. (minutes) | Comm. (MB) |
|---------|--------------|-----------------|------------|----------|-----------------|------------|
| ionosphere | 38.3 / 6.0 | 13.5 / 21.1 | 119.2 / 81.0 | 2.6 | 2.6 | 17.8 |
| credit | 55.0 / 12.2 | 28.0 / 83.2 | 1,119.2 / 537.2 | 8.1 | 8.1 | 72.1 |
| spambase | 41.4 / 5.0 | 129.5 / 586.5 | 384.6 / 1,876.6 | 76.4 | 76.4 | 271.8 |
| epileptic | 3,960.0 / 1,010.7 | 932.2 / 1,453.0 | 12,291.6 / 6,868.3 | 653.4 | 653.4 | 788.1 |
| synthetic | 3,025.0 / 805.7 | 1,414.7 / 874.7 | 106,891.1 / 57,662.2 | 383.9 | 383.9 | 1,842.5 |

![Figure 9. Comparison of model accuracy: Secure-Boost vs. SecureML - Logistic Regression and Neural Network](image)

![Figure 10. Overall cost comparison: SecureBoost constructions vs. SecureML neural network and SecureML logistic regression for the synthetic dataset.](image)

![Figure 11. Average distance between training record pairs generating characterization vectors differing by $k$-bits.](image)

**IX. Related Work**

The current implementations of FHE are still too expensive to apply on complex functions. ML Confidential [17] shows that simple linear models can be learned by a semi-honest SP from FHE-encrypted data with acceptable costs. However, these simple models are unable to handle non-linearly separable datasets. Their algorithms require the labels of training data to be revealed to SP. Lu et al. [11] show that PCA and linear regression can be implemented on FHE encrypted data with reasonable costs for a strictly small number of iterations in the algorithms. Moreover, the comparison operation based on FHE is very expensive [11], [40], which hinders the FHE’s application in many algorithms.

Despite new optimization of GC with techniques such as free XOR gates [35], half AND gates [36], and OT Extension [34], its adaptation in privacy frameworks is still costly. Nikolaenko et al. [12], [13] use FastGC [16] and AHE to implement matrix factorization and linear ridge regression solutions. Use of GCs in the core operations led these protocols to suffer from unbearable communication costs between CSP and SP. In our designs, we carefully craft the primitive operations to minimize the performance impact of the GC-related operations.

Demmler et al. [14] have shown that basic matrix operations can be implemented on random shares held by different parties when using secret sharing secure multi-party computations. SecureML [15] utilized these operations and GC to implement the gradient-descent learning method with a two-server model. However, we note that these models are more expensive than ours to achieve the same level of model quality.

Locally differential privacy [41], [42] was developed for users submitting their perturbed private data to a service provider, without being identified or distinguished from other users - a different approach to preserve users’ privacy. These studies differ from our approach in two aspects. First, perturbation involves a fundamental trade-off between data quality and privacy protection, while ours does not. Second, SP learns the model which facilitates model-based attacks.
Recent studies on model-based attacks [18–20] show that adversaries may explore exposed predictive models to breach user data privacy. The success of these attacks depends on the adversary’s unrestrained access to the models. Thus, a reasonable approach is to prevent curious parties seeing the entire model in the secure learning procedure and freely applying the learned models later. We have considered this attack in our approach.

X. Conclusion

We develop the SecureBoost protocol for SPs to learn high-quality boosted classification models from encrypted or randomly partitioned users’ data. The key idea is to use random linear classifiers as the base classifiers to simplify the protocol design. Two constructions: HE+GC and SecSh+GC have been developed, using a novel combination of homomorphic encryption, garbled circuits, and randomized secret sharing to protect privacy and achieve efficiency. We formally analyze the security of protocol and show that SecureBoost constructions satisfy the universally composable security for multiparty computation. Our experimental evaluation examines the intrinsic relationships among primitive selection, cost distribution, and model quality. Our results show that the SecureBoost approach is very practical in learning high-quality classification models. Our constructions are the first batch of boosting protocols with practical costs, compared to the expenses of the start-of-the-art implementation of other major predictive modeling methods (e.g., Neural Networks by SecureML). We will extend the study to explore the effect of sub-sampling the training data and differentially private release of the leakage function in future. Similarly, we will extend the work to multi-class classification problem and other types of boosting.

References

[1] P. Larraaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armaanzas, G. Santafé, A. Prez, and V. Robles, “Machine learning in bioinformatics,” Briefings in Bioinformatics, vol. 7, no. 1, pp. 86–112, 2006. DOI: 10.1093/bib/bbk007.

[2] I. Kononenko, “Machine learning for medical diagnosis: History, state of the art and perspective,” Artificial Intelligence in Medicine, vol. 23, no. 1, pp. 89–109, 2001. ISSN: 0933-3657. DOI: http://dx.doi.org/10.1016/S0933-3657(01)00077-X. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S093336570100077X.

[3] P. Galan-Garcia, J. G. De La Puerta, C. L. Gomez, I. Santos, and P. G. Bringas, “Supervised machine learning for the detection of troll profiles in Twitter social network: Application to a real case of cyberbullying,” Logic Journal of the IGPL, vol. 24, no. 1, pp. 42 –53, 2016. ISSN: 13670751.

[4] S. Mansfield-Devine, “The Ashley Madison affair,” Network Security, vol. 2015, no. 9, pp. 8 –16, 2015. ISSN: 13534858. [Online]. Available: http://ezproxy.libraries.wright.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=109503050&site=eds-live.

[5] L. Unger, “Breaches to customer account data.,” Computer and Internet Law, vol. 32, no. 2, pp. 14 –20, 2015. ISSN: 15314944. [Online]. Available: http://ezproxy.libraries.wright.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cph&AN=100337545&site=eds-live.

[6] A. Chen, “GCReep: Google engineer stalked teens, spied on chats.,” Gawker, http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats; 2010.

[7] A. J. Duncan, S. Creece, and M. Goldsmith, “Insider attacks in cloud computing,” in 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012.

[8] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in ANNUAL ACM Symposium on Theory of Computing, Bethesda, MD, USA: ACM, 2009, pp. 169–178.

[9] A. C. Yao, “How to generate and exchange secrets,” in IEEE Symposium on Foundations of Computer Science, 1986, pp. 162–167.

[10] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, Nov. 1979. ISSN: 0001-0782. DOI: 10.1145/359168.359176.

[11] W. Lu, S. Kawasaki, and J. Sakuma, “Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data.,” IACR Cryptology ePrint Archive, vol. 2016, p. 1163, 2016.

[12] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” in Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security, Berlin, Germany: ACM, 2013, pp. 801–812, ISBN: 978-1-4503-2477-9.

[13] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, “Privacy-preserving ridge regression on hundreds of millions of records,” in Proceedings of the 2013 IEEE Symposium on Security and Privacy, Washington, DC, USA: IEEE Computer Society, 2013, pp. 334–348.

[14] D. Demmler, T. Schneider, and M. Zohner, “ABY—A framework for efficient mixed-protocol secure two-party computation,” in 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015, 2015. [Online]. Available: https://www.ndss-symposium.org/ndss2015/aby-framework-efficient-mixed-protocol-secure-two-party-computation.

[15] P. Mohassel and Y. Zhang, “SecureML: A system for scalable privacy-preserving machine learning,” in 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 19–38. DOI: 10.1109/SP.2017.12.

[16] 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, ser. SEC’11, San Francisco, CA: USENIX Association, 2011, pp. 35–35.

[17] T. Graepel, K. Lauter, and M. Naehrig, “ML confidential: Machine learning on encrypted data,” in Proceedings of the 15th International Conference on Information Security and Cryptology, ser. ICISC’12, Seoul, Korea: Springer-Verlag, 2013, pp. 1–21, ISBN: 978-3-642-37681-8.
[18] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in 23rd USENIX Security Symposium USENIX Security 14, San Diego, CA: USENIX Association, 2014, pp. 17–32, ISBN: 978-1-931971-15-7. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson_matthew

[19] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures.” Conference on Computer and Communications Security, p. 1322, 2015. [Online]. Available: http://ezproxy.libraries.wright.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsb&AN=111436607&site=eds-live

[20] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models.” 2016. [Online]. Available: http://ezproxy.libraries.wright.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=edsarx.1610.05820&site=eds-live

[21] Y. Freund and R. E. Schapire, “A short introduction to boosting,” in International Joint Conferences on Artificial Intelligence, Morgan Kaufmann, 1999, pp. 1401–1406.

[22] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning.” Nature, vol. 521, no. 7553, p. 436, 2015, ISSN: 00280836. [Online]. Available: http://ezproxy.libraries.wright.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=f5h&AN=103043672&site=eds-live

[23] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in PROCEEDINGS OF INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), Pittsburgh, Pennsylvania: ACM, 2006, pp. 161–168.

[24] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in SIGKDD Conference on Knowledge Discovery and Data Mining, 2016.

[25] O. Chapelle and Y. Chang, “Yahoo! learning to rank challenge overview.” Journal of Machine Learning Research - Proceedings Track, vol. 14, pp. 1–24, 2011.

[26] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” Journal of Cryptology, vol. 15, no. 3, pp. 177–206, 2000.

[27] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in EUROCRYPT, Springer-Verlag, 1999, pp. 223–238.

[28] Z. Brakerski and V. Vaikuntanathan, “Fully homomorphic encryption from Ring-LWE and security for key dependent messages,” in Proceedings of the 31st Annual Conference on Advances in Cryptology, ser. CRYPTO’11, Santa Barbara, CA: Springer-Verlag, 2011, pp. 505–524, ISBN: 978-3-642-22791-2.

[29] R. Canetti, “Universally composable security: A new paradigm for cryptographic protocols,” in Proceedings 2001 IEEE International Conference on Cluster Computing, 2001, pp. 136–145.

[30] R. Canetti, A. Cohen, and Y. Lindell, A simpler variant of universally composable security for standard multi-party computation, 2015.

[31] T. Hastie, R. Tibshirani, and J. Friedman, The elements of statistical learning. Springer-Verlag, 2001.

[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) fully homomorphic encryption without bootstrapping,” in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ser. ITCS ’12, Cambridge, Massachusetts: ACM, 2012, pp. 309–325, ISBN: 978-1-4503-1115-1.

[33] S. Halevi and V. Shoup, “Algorithms in HELib,” in International Cryptology Conference, Springer, 2014, pp. 554–571.

[34] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, “More efficient oblivious transfer and extensions for faster secure computation,” in 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013, pp. 535–548. DOI: 10.1145/2508859.2516738 [Online]. Available: http://doi.acm.org/10.1145/2508859.2516738

[35] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free XOR gates and applications,” in Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations, 2008, pp. 486–498. DOI: 10.1007/978-3-540-70583-3_40 [Online]. Available: https://doi.org/10.1007/978-3-540-70583-3_40

[36] S. Zahur, M. Rosulek, and D. Evans, “Two halves make a whole,” in Advances in Cryptology - EUROCRYPT 2015, E. Oswald and M. Fischlin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 220–250.

[37] J. Katz and Y. Lindell, Introduction to modern cryptography. Chapman and Hall/CRC, 2007.

[38] S. Halevi and V. Shoup, “Introduction to modern cryptography.” Chapman and Hall/CRC, 2007.

[39] S. Halevi and V. Shoup, “Introduction to modern cryptography.” Chapman and Hall/CRC, 2007.

[40] A. Frank and A. Asuncion, UCI machine learning repository, 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[41] P. Golle, “A private stable matching algorithm,” in International Conference on Financial Cryptography and Data Security, Springer, 2006, pp. 65–80.

[42] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Local privacy and statistical minimax rates,” 2013.

[43] U. Erlingsson, A. Korolova, and V. Pihur, “RAPPOR: randomized aggregatable privacy-preserving ordinal response,” CoRR, vol. abs/1407.6981, 2014. arXiv:1407.6981 [Online]. Available: http://arxiv.org/abs/1407.6981