Poster: Privacy-Preserving Boosting with Random Linear Classifiers

Sagar Sharma, Keke Chen
Data Intensive Analysis and Computing (DIAC) Lab, Kno.e.sis Center, Wright State University
{sharma.74,keke.chen}@wright.edu

ABSTRACT

We propose SecureBoost, a privacy-preserving predictive modeling framework, that allows service providers (SPs) to build powerful boosting models over encrypted or randomly masked user submitted data. SecureBoost uses random linear classifiers (RLCs) as the base classifiers. A Cryptographic Service Provider (CSP) manages keys and assists the SP's processing to reduce the complexity of the protocol constructions. The SP learns only the base models (i.e., RLCs) and the CSP learns only the weights of the base models and a limited leakage function. This separated parameter holding avoids any party from abusing the final model or conducting model-based attacks. We evaluate two constructions of SecureBoost: HE+GC and SecSh+GC using combinations of primitives - homomorphic encryption, garbled circuits, and random masking. We show that SecureBoost efficiently learns high-quality boosting models from protected user-generated data with practical costs.

ACM Reference Format:
Sagar Sharma, Keke Chen. 2018. Poster: Privacy-Preserving Boosting with Random Linear Classifiers. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3243734.3278520

1 INTRODUCTION

Web and mobile service providers (SPs) are collecting user-generated data, e.g., clicks, reviews, friend connections at an unprecedent scale. Such collection of user data together with powerful big data analytics help SP improve the quality of their services and increase revenue, however, at the same time raise privacy concerns. SP's infrastructures, if poorly secured, can be compromised by external hackers, leaking sensitive user data [8]. Furthermore, unauthorized retrieval, sharing, or misuse of users' personal information by insiders [1] are difficult to track and prevent. Even worse, users may not be aware of any misuse or leakage of their data.

An interesting solution is SP storing protected data and only the users who generate the data possessing the data recovery key. This has two unique advantages: users reclaim the ownership of their data and SP eliminates the responsibility of protecting privacy. The challenge here is to preserve the utility of the protected data. There should be some mechanisms for SP to mine (e.g., learn predictive models) the protected data. Recent advances in cryptography have provided several tools such as homomorphic encryption (HE), e.g., somewhat HE (SHE) and additive HE (AHE), and garbled circuits (GC) for achieving this goal. However, it is difficult to optimally balance cost with utility when assorting these privacy primitives to build the privacy-preserving machine learning algorithms [6].

It is even trickier to select and modify a machine learning algorithm to significantly reduce the complexity of its privacy-preserving version. It might be easier to implement privacy protocols for less powerful algorithms such as linear classifiers and linear regressions [3, 5, 6], however, extremely complex and impractical to build more powerful algorithms such as SVM and deep neural networks [5].

Finally, existing approaches [3, 5, 6] disregard model-based attacks [2, 7]. A reasonable approach to prevent such attacks is to prevent the curious SP from holding the *complete* model.

Scope of work and contributions. SecureBoost addresses the fundamental conflict between the model prediction power and the complexity of building privacy-preserving learning algorithms for protected data. We utilize the powerful boosting framework and simplify it by use of random linear classifiers (RLC) as the base weak classifiers. We adopt the setting of Crypto Service Provider (CSP) [6] or two non-colluding servers [5]. We develop two constructions: HE+GC and SecSh+GC, with novel combinations of security primitives such as GC, SHE, Secret Sharing, AHE, and random masking. We preserve the privacy of users' submitted data, including both the feature vectors and their labels, with a minimal leakage function known only to CSP. SP and CSP learn only a part of the model parameters, effectively preventing model-based attacks by a curious SP or CSP. We conduct an extensive experimental evaluations of SecureBoost to explore the associated costs and tradeoffs.

2 FRAMEWORK

Figure 1 shows the SecureBoost framework and the involved parties: the Service Provider (SP), the users who contribute personal data for model training, and the Cryptographic Service Provider (CSP). The learning protocols consist of multiple rounds of SP-CSP interactions to build a boosted model on the global pool of user training data. Ultimately, SP learns the parameters of each base classifier whereas CSP learns the weights of the base classifiers. SP undertakes the major storage and computation cost of the privacy-preserving protocol. CSP assists SP in intermediate steps, e.g. encrypting and decrypting intermediate results and constructing garbled circuits. CSP is allowed to learn a leakage function, however, remains oblivious to users' data or the learnt models. Note: The dotted line in Figure 1 shows the random share submission by users to SP and CSP.

2.1 SecureBoost Learning Protocol

Random Linear Classifiers. It is expensive to implement the privacy-preserving versions of boosting with the commonly used decision stumps (DS) as shown in previous studies [4]. Each DS involves scanning through all possible splits for all the features in the training data. Instead, we choose randomly generated linear classifiers (RLC) as the base classifiers. RLC is defined by a boundary

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3278520

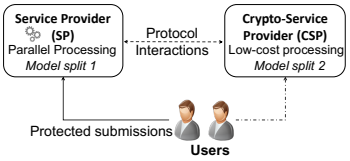

Figure 1: Secure-Boost Framework. SP and CSP collaboratively learn predictive models over encrypted user data. The learned model is split between SP and CSP.

plane in the form of $w^T x + b$ with randomly selected w and b. However, not all arbitrary RLCs would work as shown in Figure 2. With standardization, training data values are mostly in range $[-2, 2]$. Thus, we can simply sample b from the range $[-2, 2]$ and each element of w uniformly from [-1, 1]. With this setting, we were able to find *effective* random linear classifier in about 1-3 tries.

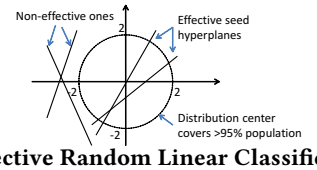

Figure 2: Effective Random Linear Classifier Generation

SecureBoost Protocol. The SecureBoost learning protocol as shown in Algorithm 1 is defined with a 4-tuple: SB-Learning = (**Setup, BaseApply, ResultEval, Update**). For a boosted model $H(x) = \sum_{t=1}^{\tau} \alpha_t h_t(x)$, SP learns the base models $\{h_t(x) = w_t^T x, t = 1..\tau\}$ whereas CSP learns the model weights $\{\alpha_t, t = 1..\tau\}$.

Algorithm 1 SecureBoost Framework

1: $(K, E(Z), \{w_i, i = 1..p\}, \delta_1) \leftarrow$**Setup**$(1^k, \tau, p)$;
2: **for** $t \leftarrow 1$ **to** p **do**
3: $\{E(h_t(x_i)), i = 1..n\} \leftarrow$ **BaseApply**$(K, E(Z), w_t)$;
4: $I_t \leftarrow$ **ResultEval**$(K, \{E(h_t(x_i), i = 1..n\})$;
5: $(\delta_{t+1}, \alpha_t, e_t) \leftarrow$ **Update**(K, δ_t, I_t); // by CSP only
6: **if** τ effective base models have been found **then**
7: stop the iteration;
8: **end if**
9: **end for**

$(K, E(Z), \{w_i, i = 1..p\}, \delta_1) \leftarrow$**Setup**$(1^k, \tau, p)$**:** (1) The key K with security level 1^k is generated by CSP (or SP if required); all public keys are published. (2) CSP initializes δ_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, τ, and generates a pool of prospective RLCs with parameters w_t for $t = 1 \ldots p$, where p is the pool size proportional to τ, e.g., $p = 1.5\tau$.

$\{E(h_t(x_i)), i = 1..n\} \leftarrow$ **BaseApply**$(K, E(Z), w_t)$: With the encrypted training data $E(Z)$ and the model parameter w_t, the procedure outputs the model h_t's encrypted prediction results for all the training instances.

$I_t \leftarrow$ **ResultEval**$(K, \{E(h_t(x_i)), i = 1..n\})$**:** 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 (not the actual predictions).

$(\delta_{t+1}, \alpha_t, e_t) \leftarrow$ **Update**(δ_t, I_t): CSP takes I_t, δ_t to compute the weighted error rate $e_t = I_t^T \delta_t$ and then the weight $\alpha_t = 0.5ln((1 - e_t)/e_t)$ for the base classifier h_t.

Security Model. We make some security assumptions: (1) Both SP and CSP are honest-but-curious parties. They follow exactly the protocols and provide services as expected but interested in users' data, hence are the major adversaries in the framework. (2) SP and CSP do not collude as it would break the security of the protocols. (3) All infrastructures and communication channels are secure.

SecureBoost protocol does not leak any information enabling SP or CSP to breach data privacy, except for CSP learning a leakage function, which is the indicator function $I_{t,i}(h_t(x_i) == y_i)$, for $i = 1..n$ in iteration t. Model parameters are split and distributed between SP and CSP, hence no single party can learn the complete model or launch model-based attacks.

2.2 Technical Detail

In this section we discuss some key details in the framework.

Integer Conversion. For HE compatibility, we convert floating-point values x, $x \in \mathbf{R}$ to $v = \lfloor 10^m x \rfloor \mod q$, where q is a large integer such that $10^m x \in (-q/2, q/2)$ preserves m-digit precision. The negative values are mapped to the range $(q/2, q)$. With large enough q, x is recoverable and desired operations do not overflow.

Data Submission. For n records $x_i y_i$ of training data, the matrix Z is encoded column-wise before encryption with RLWE message packing. With AHE, users encrypt Z record-wise. With SecSh, instead of encryption, users randomly split Z into Z_0 for SP and Z_1 for CSP, such that $Z = Z_0 + Z_1$. Users batch their submissions with RLWE whereas either batch or stream with AHE and SecSh.

Secure Matrix-Vector Multiplication. With an AHE-encrypted Z and the plaintext w_t, it is straightforward to compute the matrix-vector multiplication $E(Zw_t)$ in **BaseApply** homomorphically. With RLWE, SP encrypts w_t and follows the homomorphic matrix-multiplication scheme of RLWE. With SecSh, SP holds Z_0 and w_0 while CSP holds Z_1 and w_1 ($Z_0 + Z_1 = Z$ and $w_0 + w_1 = w$). SP and CSP compute the random shares of Zw_t using an AHE scheme.

Securely Checking Signs of a protected vector. CSP needs to learn the correctness of RLCs when predicting for the training instances during **ResultEval**, i.e. $sign(E(u_t) = E(Zw_t))$. u_t is split between SP and CSP as $u_{t,0}$ and $u_{t,1}$ in SecSh+GC construction. To hide u_t from CSP in HE+GC, SP performs $E(u_{t,0}) = E(u_t) + E(\lambda_t)$, where λ_t is a noise vector. Denoting λ_t by u_{t_1}, a GC (shown by Figure 3) computes $u_t = u_{t,0} - u_{t,1}$ ($u_t = u_{t,0} + u_{t,1}$ for HE+GC) and returns a specific bit of each element of u_t.

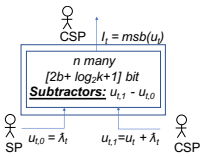

Figure 3: GC sign checking protocol. CSP inputs $u_{t,1}$, SP provides $u_{t,0} = \lambda_t$. The circuit outputs $msb(u_t)$ denoting $sign(u_t)$.

Cost Analysis. Table 1 summarizes the associated Big O estimation for each party broken down into different operations.

Table 1: BigO estimation for SecureBoost constructions

Construction	Party	Encryption	Decryption	Enc. Mult/Add	Enc. Comm.	GC Comm.	Storage
HE+GC	User	$O(nk)$	-	-	$O(nk)$	-	-
	SP	$O(pn)$	-	$O(pnk)$	$O(pn)$	$O(pnb)$	$O(nk)$
	CSP	-	$O(pn)$	-	-	-	-
SecSh+GC	User	-	-	-	-	-	-
	SP	$O(pk)$	$O(pn)$	-	$O(p(n+k))$	$O(pnb)$	$O(nk)$
	CSP	$O(pn)$	-	$O(pnk)$	$O(pn)$	-	$O(nk)$

3 EXPERIMENTS

We pick cryptographic parameters corresponding to 80-bit security. The RLWE parameters allow 32-bit message-space, 1 full vector replication, and at least 1 level of multiplication. Our GC-based sign checking protocol accommodates $(2b + log_2(k))$-bit inputs, where $b = 7$ is the bit-precision and k is the dimension of the training data. We test SecureBoost [1] with the datasets summarized by Table 2. The synthetic dataset consists non-linearly separable classes.

Table 2: 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

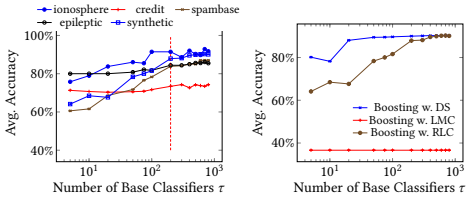

Figure 4: Convergence of boosting with RLCs (Left). Convergence of boosting with RLCs, LMCs, and DSs (Right)

Effectiveness of RLC Boosting. Figure 4 [Left] shows that about 200 RLCs lead to a stable model accuracy for each dataset. Figure 4 [Right] compares amongst boosting with RLCs, DS, and linear means classifiers (LMC) over the synthetic dataset; DS converges the fastest (about 75-80 rounds) while LMC fails to improve at all. Figure 5 [Left] shows that the model quality produced by 200 RLCs vs. 75 DS are almost identical (note 200 RLCs costs much lower than 75 DS). Figure 5 [Right] shows that about 7-bit precision produces almost optimal model quality.

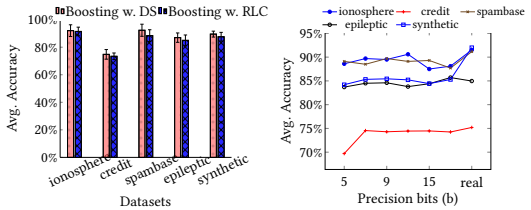

Figure 5: Model quality: boosting with RLCs vs. boosting with DSs (Left). Bit precision vs. model accuracy (Right)

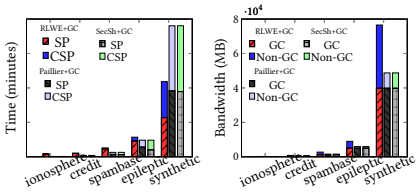

Figure 6: Overall cost distribution

Cost breakdown. Figure 6 [Left] shows the computation cost breakdown between SP and CSP for the three constructions. Figure 6 [Right] compares the GC portion of communication with that of non-GC. Overall, RLWE+GC construction is favorable computation-wise whereas Paillier+GC and SecSH+GC[2] communication-wise.

[1]These results can be reproduced and verified with the scripts we have uploaded to *https://sites.google.com/site/testsboost/*.

[2]We use the Paillier cryptosystem in SecSh+GC as the required AHE scheme

Comparing with other methods. We compare SecureBoost with a recently proposed privacy-preserving logistic regression (LR) and neural network (NN) protocols based on secret sharing [5]. We set a two five-node hidden-layered NN [3]. The NN converges in 160 iterations whereas the LR in only 10. Figure 7 shows that all the methods (SecureBoost, SecureML LR, and SecureML NN) perform similarly except for the non-linearly separable synthetic data, for which LR fails significantly. Figure 8 shows all the SecureBoost constructions being much more efficient than the SecureML NN.

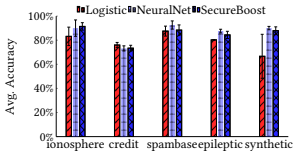

Figure 7: Comparison of model accuracy: Secure-Boost vs. SecureML - Logistic Regression and Neural Network.

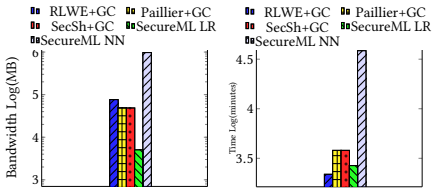

Figure 8: Overall cost comparison: SecureBoost constructions vs. SecureML protocols for the synthetic dataset.

4 CONCLUSION

We develop the SecureBoost protocol for service providers to learn high-quality boosted classification models from encrypted or randomly partitioned users' data with practical costs. 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 and evaluated, using a novel combination of homomorphic encryption, garbled circuits, and randomized secret sharing to protect privacy and achieve efficiency.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under Grant 1245847.

REFERENCES

[1] A. J. Duncan, S. Creese, and M. Goldsmith. Insider attacks in cloud computing. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

[2] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. *Conference on Computer and Communications Security*, page 1322, 2015.

[3] 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*, ICISC'12, pages 1–21, Berlin, Heidelberg, 2013.

[4] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2000.

[5] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[6] 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*, pages 334–348, Washington, DC, USA, 2013. IEEE Computer Society.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. 2016.

[8] L. Unger. Breaches to customer account data. *Computer and Internet Lawyer*, 32(2):14 – 20, 2015.

[3]The minimal shape to satisfactorily handle the non-linearly separable synthetic dataset