CRYPTOGRU: LOW LATENCY PRIVACY-PRESERVING TEXT ANALYSIS WITH GRU

Bo Feng, Qian Lou, Lei Jiang, Geoffrey C. Fox

Indiana University Bloomington, Intelligent Systems Engineering, Bloomington, IN USA

ABSTRACT

Privacy issues have been raised up since the machine learning services are now popular to deploy on public clouds. Billions of private emails, personal text messages, users' online reviews are being scanned and sent to machine learning services for modeling inference. In this paper, we present our CRYPTOGRU, which is a hybrid homomorphic encryption and garbled circuits enabled framework for secure GRU network inference. Our project not only demonstrates secure working examples using GRU based neural network but also outlines techniques that map network layers to SIMD (single instruction multiple data) addition, SIMD multiplication, and two detailed techniques to improve the latency. We evaluate our project with a few well-defined GRU networks trained on 4 public datasets, and show CRYPTOGRU can generate the state-of-art accuracy but run up to 138 times faster on CPU with respect to the latency for secure inference.

Index Terms— Neural networks, GRU, Security and privacy

1. INTRODUCTION

Machine learning services are widely used in practice to perform predictive user behaviors. However, privacy issues have been raised up since the machine learning services are now popular to deploy on public clouds and collecting massive amount of user data. Billions of private emails, personal text messages, users' online reviews are being scanned and sent to machine learning services for modeling inference. Even though technology companies, such as Google, Facebook, AT&T, and Verizon all promised to keep users privacy, we know some well-known services are still using plain text prediction. For example, the smart reply generation service from Gmail still needs to scan users' plain email messages anonymously [1, 2].

Data encryption is a prominent method that can secure and protect users' privacy. Homomorphic encryption (HE) was proposed as an encrypted data calculation technique that can be integrated into neural networks to perform model inference without requiring to decrypt the input data. Secure model inference with HE has been developed and applied on image classification. CryptoNets [3] is a neural network applying encrypted image data on model inference. However, CryptoNets requires to change the neural network structures and replace activation functions with LHE based non-linear functions such as the $square$ function. However, homomorphic encryption requires significant amount computation due to the increased noise with the context. This leads to the non-linear computation on ciphertext is nearly impossible [4].

Secure two party computation, on the other hand, can mitigate this issue by performing a joint function without sharing values. Gazelle [5], as another example, developed convolutional neural networks for various image classification tasks with both homomorphic encryption and garbled circuits. However, machine learning secure text analysis is still challenging. Firstly, non-linear activation functions such as the $sigmoid$ and $tanh$ function cannot be implemented with fully homomorphic encryption schemes. Secondly, if these functions are replaced with linear functions such as the $square$ function, the result accuracy would be negatively impacted. Thirdly, if these functions are simulated by high degree polynomial functions, the varnishing gradient is another problem and its inference latency is high due to enormous cryptographic operations.

CRYPTOGRU is built based on Gazelle [5], which is low latency framework for secure network. Gazelle is the state-of-art cryptographic system tools for using linear algebra and linear layers for building neural networks. In this paper, we present the design of CRYPTOGRU, a secure gated recurrent unit which consists of homomorphic encryption algorithms for linear operations and garbled circuits for non-linear operations. Our contributions are summarized as follows:

- We firstly propose a hybrid solution of building a secure GRU cell by using both homomorphic encryption and garble circuits.
- We then improve the neural networks by replacing the $tanh$ activation function with a cheaper $ReLU$ function without sacrificing their accuracy.
- We further improve the inference latency of neural networks by quantizing both the $sigmoid$ and $ReLU$ functions.

2. BACKGROUND AND RELATED WORK

2.1. Text Analysis using GRU

Recurrent neural network (RNN) plays an important role in deep learning based text analysis [6]. As shown in Figure 1,

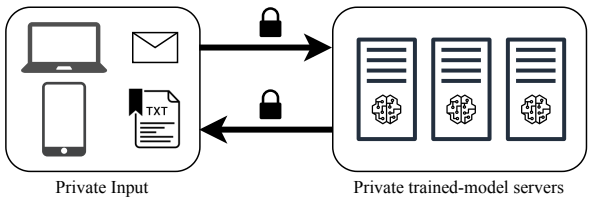

Fig. 1. An end-to-end overview of how CRYPTOGRU cell works in a secure inference system.

in a secure inference system, the input from text messages and emails are private and encrypted content will be sent to the server side. On the server side, the encrypted results are returned. Gated recurrent unit (GRU) and long short-term memory are neural network building blocks as recurrent neural networks for capturing long term dependencies [7]. They can be used for many types of architectures and various machine learning tasks, such as text classification and text generation. There are a total of $4 \times (n^2 + nm + n)$ number of parameters in a single LSTM cell and $3 \times (n^2 + nm + n)$ number of parameters in a single GRU cell, where m means the dimension of the input and n for the dimension of the hidden state. In CRYPTOGRU, less parameters mean cheaper computation for ciphertext. Various studies, e.g [7], have shown GRU has comparable performance with LSTM or even outperform LSTM in many cases. Bowman et al. [8] built RNN-based variational autoencoders to generate diverse sentences, in the model of which both encoder and decoder involved RNN.

2.2. Secure Inference

Fully homomorphic encryption (FHE) is a technique that enables to perform calculation on encrypted data without decrypting first. Many companies integrated the neural network prediction services into service clouds. Although the data trans-mission might be encrypted, user data still remain as plain texts which are public to the models of these neural network models.

Secure inference has been used in many image classification tasks. Chou et al. present their work about an image classification pipeline that uses homomorphic encryption [9]. In their work, the image features are extracted as plain content and representation features are encrypted.

2.3. Comparison with prior works

CRYPTOGRU is the first gated recurrent unit with hybrid HE and GC techniques, as far as we know. We still compare the technique details with some prior works, which can achieve same effect in secure inference.

Table 1 shows the technique differences between CRYPTOGRU with prior works. While our CRYPTOGRU is built on top of Gazelle, however Gazelle does not support RNN directly. More details of constructing a GRU with Gazelle is further illustrated in Section 3.1. SHE [10] is a secure inference system that is built with LHE-only protocols, of which the latecny is high for doing RNN-based text analysis. Bakshi et al. proposed CryptoRNN [4], which achieved RNN-based text analysis with high activation functions. However, these activations are replaced with approximation function of degree 2, which results in inefficient inference in term of the latency. Badawi et al. proposed PrivFT [12] for text classification task with homomorphic encryption methods. However, PrivFT does not support RNN architecture and requires full HE computation for secure inference.

Table 1. The comparison of secure text analysis techniques.

	RNN-based text analysis	Accurate Activation	Efficient Activation	No Plaintext
Gazelle [5]	✗	✗	✗	✓
SHE [10]	✓	✗	✗	✓
CryptoRNN [4]&HE-RNN [11]	✓	✓	✗	✗
PrivFT [12]	✗	✗	✗	✓
CRYPTOGRU	✓	✓	✓	✓

3. CRYPTOGRU DESIGN

3.1. Construct the baseline of CRYPTOGRU

Fully homomorphic encryption allows calculating an arbitrary function f for a plain input x on an encrypted input $[x]$ without decrypting $[x]$ from x. With only f and $[x]$, one can obtain the encrypted output $[y] = f([x])$ and on the client side, the plain text y can be decrpted from $[y]$. Conventional neural network inference uses depth-bounded arithmetic circuits (LHE). However, the computation cost is large for the LHE scheme. CRYPTOGRU adopts a simpler FHE scheme, namely packed additive homomorphic encryption (PAHE) scheme and garbled circuits (GC) by integrating the Gazelle library.

Algorithm 1: HE and GC in CRYPTOGRU cell

Input: an input ciphertext $[x_t]$
Output: a ciphertext hidden state $[h_t]$
$[Gate_x] = MultPC([x], \Omega_i, b_i)$
$[Gate_h] = MultPC([h_{t-1}], \Omega_h, b_h)$
$[i_r], [i_i], [i_n] = chunk - split([Gate_x], 3)$
$[h_r], [h_i], [h_n] = chunk - split([Gate_h], 3)$
$[Gate_{reset}] = sigmoid(AddCC([i_r], [h_r]))$
$[Gate_{input}] = sigmoid(AddCC([i_i], [h_i]))$
$[Gate_{new}] =$
$tanh(AddCC([i_n], MulCC([Gate_{reset}], [h_n])))$
$[h_t] = AddCC([Gate_{new}],$
$MulCC([Gate_{input}], MinusCC([h_{t-1}], [Gate_{new}])))$
return $[h_t]$

We start with a straightforward approach to build the CRYPTOGRU. We construct our base version of CRYPTOGRU on top of Gazelle linear algebra kernel functions and garble circuits layers. Figure 2 illustrates the details of an internal view of a full GRU cell, which consists of both linear and non-linear operations. The linear operations are in blue, as shown in Figure 2. In a GRU cell, linear operations include matrix vec-

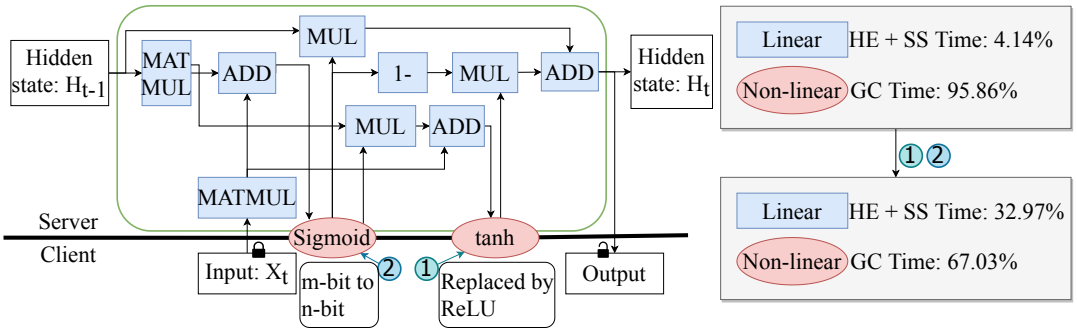

Fig. 2. CRYPTOGRU cell with labeled linear and non-linear cryptographic operations. Annotations are referred in Section 3.1, 3.2, and 3.3.

tor multiplications (*MATMUL*), element-wise add, minus, and multiplications (*ADD*,*1-*,*MUL*). In CRYPTOGRU, we apply the algorithms provided by Gazelle by mapping the neural network layers to homomorphic encrypted matrix-vector multiplication for these linear operations. Non-linear operations are in red, as shown in Figure 2. In a GRU cell, the activation function $sigmoid$ and $tanh$ are non-linear, which are . For non-linear operations, we apply garble circuits to build non-linear operations. The process of updating hidden states inside a GRU cell is shown in Algorithm 1.

3.2. ❶ Replace the $tanh$ activation with $ReLU$

In GRU RNN, the activation nonlinearity function is typically $tanh$ but can also be implemented with the rectified linear unit $ReLU$ [13, 7]. However, $tanh$ is a more expensive non-linear computation than $ReLU$ in term of the implementation using garbled circuits.

We use some tests against two public datasets to demonstrate this motivation. One dataset is the IMDB, which consists of 50,000 movie reviews. The other dataset is the Yelp review. Both of the datasets are used in binary classification tasks. To quantify their performance in a simple RNN neural network, we test the accuracy of using $tanh$ with $ReLU$ on the two datasets. The results are summarized in Table 2. From this table, the GRU model with $ReLU$ can gain almost the same accuracy as that uses $tanh$ but trade off its training time for significant shorter latency during the inference stage. We label this version as "CRYPTOGRU-❶" in all the following text.

Table 2. The $tanh$ activation vs. $ReLU$ activation.

	Accuracy		Latency	
Datasets	$tanh$	$ReLU$	$tanh$	$ReLU$
IMDB	84.8%	84.6%	14860ms	**3779ms**
Yelp Reviews	77.3%	78.1%	5383ms	**1852ms**

3.3. ❷ Quantize both $sigmoid$ and $ReLU$ activations

During the computation of a full GRU cell, we identify the latency bottleneck is at non-linear functions (a.k.a activation functions in this case). In Figure 2, we show the computation time for non-linear operations hold about 95.86%. This is mainly due to the computational complexity for ciphertext is significantly proportional to the underlying bit-length [x]. As shown in Table 2, the computational complexity for $tanh$ is around 2^N and $5N$ for $ReLU$, where N is the bit-length of numbers [x].

In the baseline version of CRYPTOGRU, the default number of bits is 20, and here we quantize the activation functions to 8-bit. The benefits of using the quantization are from two folds. First, the design of garbled circuits is proportional simpler after the quantization since the garbled circuits are sensitive to the bit length. Second, because these activation functions do not have any weight parameters, the overall accuracy of the neural networks with quantized activation functions can still hold. We summarize the results of testing CRYPTOGRU in Table 3. This is further discussed in Section 4. We label this version as "CRYPTOGRU-❷" in all the following text.

4. EXPERIMENTS AND RESULTS

In this section, we describe the basic cryptographic primitives, performance comparison of a single GRU cell using micro-benchmarks, and inference test results comparing with some prior related works.

Cryptographic primitives and evaluation setup. CRYPTOGRU is integrated with the Gazelle encryption library, which is implemented in C++. Two main sets of cryptographic primitives are used for the CRYPTOGRU inference. One set is for homomorphic encryption and the other set is for a garbled-circuits scheme. For the homomorphic encryption, we use Brakerski-Fan-Vercauteren (BFV) scheme [14]. Yao's garbled circuits scheme is used for a two-party secure computation [15]. We use the default settings for the BFV scheme which consists of 20-bit plaintext and 60-bit ciphertext modulus from the Gazelle library [5] as it is explained in Section 3.1. All benchmarks and inference tests are conducted on a workstation which includes a 12-thread Intel Core i7-7800X CPU 3.50GHz CPU and 32GiB total system memory with a

Table 3. The benchmark results of CRYPTOGRU. Baseline represents a version of CRYPTOGRU built with the Gazelle library.

Schemes	Input Size	Hidden Size	Offline+Setup Latency	Online Latency	Total Latency
Baseline	10	128	1651.2+107.5 ms	13917.15 ms	15675.85 ms
CRYPTOGRU-❶	10	128	258+107.5 ms	3225.15 ms	3590.65 ms
CRYPTOGRU-❷	10	128	258+107.5 ms	1290.06 ms	1655.56 ms
Baseline	100	64	1305.6+84 ms	3993.31 ms	5382.91 ms
CRYPTOGRU-❶	100	64	204+84 ms	1563.32 ms	1851.32 ms
CRYPTOGRU-❷	100	64	204+84 ms	625.32 ms	913.32 ms

Table 4. Results comparison between CRYPTOGRU with related work.

Datasets	Neural Networks	Accuracy	Latency
Enron Emails	PrivFT [12]	-	0.63s∗
	CryptoGRU	84.2	**2.03s**
Penn Treebank	SHE [10]	89.8	∼576s
	CryptoGRU	79.4	**4.14s**
IMDB	HE-RNN [11]	86.47∗	70.6s∗
	PrivFT [12]	91.49∗	0.65s∗
	CryptoGRU	84.6	**2.07s**
Yelp Review	PrivFT [12]	96.06∗	0.65s∗
	CryptoGRU	91.3	**0.91s**

customized Ubuntu 18.04 server LTS operating system.

Effectiveness of improvements using ❶ and ❷. To quantify the performance of the baseline CRYPTOGRU version and the effectiveness of two proposed techniques, we build benchmarks to compare the internal latency of all linear and non-linear operations in a CRYPTOGRU layer. We test our three versions of CRYPTOGRU for the performance respect to the latency. The results are shown in Table 3. In a single GRU cell, there are 12 operations, 9 of which are linear and 3 are non-linear as discussed in Section 3.1. For each operation, we calibrate its setup latency, offline latency, online latency. In addition, the complexity of these operations are proportional to the size of input and configured hidden size. Here we set two sets of input and hidden sizes. Given the input size is 10 and hidden size is 128, for the baseline case, the offline latency is 1651.2ms, the setup latency is 107.5ms, and the total latency is 1567.85ms. Compared to this case, CRYPTOGRU-❶ can finish with a 258ms offline latency, 107.5ms setup latency, and 3225.15ms online latency, resulting a total of 3590.65ms latency. The offline latency and online latency are improved due to the simpler computational complexity of using $ReLU$. From the benchmark results, $ReLU$ function can use 6.4 times less circuit gates for ciphertexts. This version is about 77% faster than the baseline version. In contrast, the CRYPTOGRU-❷ has the same setup and offline latency as the CRYPTOGRU-❶, but the online latency is only about 1290.06ms, resulting the total latency is 1655.56ms, which is about 54% faster than the version ❶. The two techniques show the same effect when the input and hidden sizes are 100, and 64 respectively. The baseline version use a total of 5392.91ms, the CRYPTOGRU-❶ use a total of 1851.32ms, and the CRYPTOGRU-❷ use a total of 913.32ms. In this setting, the CRYPTOGRU-❶ shows an improvement of about 66% respect to the latency of the baseline and CRYPTOGRU-❷ shows a further improvement of about 51% compared to the CRYPTOGRU-❶.

Comparison to prior works. To compare the CRYPTOGRU against the state-of-art prior related works, we test the latency and accuracy against public datasets. We use two more public datasets that are common to machine learning tasks for text to evaluate the performance of the CRYPTOGRU-❷ (referred as CRYPTOGRU in this section) as well as the IMDB and Yelp datasets from Section 3.2. Enron Emails is a dataset collection consisting of 500,000 emails with subjects and body messages. We summarize the comparison results of the inference latency using CRYPTOGRU with some prior works in Table 4. Because we do not have code from the PrivFT, the latency results are estimated only from their shared online document and these results are tested from benchmarks performing on GPU [12]. For the Penn Treebank dataset, our CRYPTOGRU can infer a sample in 4.14s, which is about 138 times faster than the SHE [10]. For the IMDB datset, our CRYPTOGRU can finish one sample inference within 2.07s on CPU, which is about 33 times faster than the HE-RNN [11]. The CRYPTOGRU can infer a sample from Enron Emails and Yelp reviews in 2.03s and 0.91s respectively.

Summary. We clearly present the improvements of applying two techniques ❶ and ❷ over a baseline approach. Compared with related work, CRYPTOGRU can achieve low latency in a secure inference system and maintain the same level of accuracy.

5. CONCLUSION AND FUTURE WORK

Machine learning as a service attracts interest from many aspects in industry. Public cloud companies already launched prediction services. However, sending plaintext to model servers for inference raise attentions to user privacy issues. Hence, we propose CRYPTOGRU, a secure inference building block for gated recurrent unit that emphasises on text-like or time series models. CRYPTOGRU adopts homomorphic encryption, share secrets, and garbled circuits heterogeneously to achieve low latency as well as high accuracy.

6. REFERENCES

[1] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al., "Smart reply: Automated response suggestion for email," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 955–964.

[2] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[3] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[4] Maya Bakshi and Mark Last, "Cryptornn-privacy-preserving recurrent neural networks using homomorphic encryption," in *International Symposium on Cyber Security Cryptography and Machine Learning*. Springer, 2020, pp. 245–253.

[5] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1651–1669.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[7] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[8] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[9] Edward J Chou, Arun Gururajan, Kim Laine, Nitin Kumar Goel, Anna Bertiger, and Jack W Stokes, "Privacy-preserving phishing web page classification via fully homomorphic encryption," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2792–2796.

[10] Qian Lou and Lei Jiang, "She: A fast and accurate deep neural network for encrypted data," in *Advances in Neural Information Processing Systems*, 2019, pp. 10035–10043.

[11] Robert Podschwadt and Daniel Takabi, "Classification of encrypted word embeddings using recurrent neural networks.," in *PrivateNLP@ WSDM*, 2020, pp. 27–31.

[12] Ahmad Al Badawi, Luong Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung, "Privft: Private and fast text classification with homomorphic encryption," *arXiv preprint arXiv:1908.06972*, 2019.

[13] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[14] Zvika Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Annual Cryptology Conference*. Springer, 2012, pp. 868–886.

[15] Andrew Chi-Chih Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.