CRYPTOGRU: Low Latency Privacy-Preserving Text Model Inference

Bo Feng and **Qian Lou** and **Lei Jiang** and **Geoffrey Fox**
Intelligent Systems Engineering
Luddy School of Informatics, Computing, and Engineering
Indiana University Bloomington
{fengbo,louqian,jiang60}@iu.edu, gcf@indiana.edu

Abstract

Homomorphic encryption (HE) and garbled circuit (GC) provide the protection for users' privacy. However, simply mixing the HE and GC in RNN models suffer from long inference latency due to slow activation functions. In this paper, we present a novel hybrid structure of HE and GC gated recurrent unit (GRU) network, CRYPTOGRU, for low-latency secure inferences. CRYPTOGRU replaces computationally expensive GC-based $tanh$ with fast GC-based $ReLU$, and then quantizes $sigmoid$ and $ReLU$ to smaller bit-length to accelerate activations in a GRU. We evaluate CRYPTOGRU with multiple GRU models trained on 4 public datasets. Experimental results show CRYPTOGRU achieves top-notch accuracy and improves the secure inference latency by up to 138× over one of the state-of-the-art secure networks on the Penn Treebank dataset.

1 Introduction

Billions of text analysis requests are processed by powerful RNN models (Bahdanau et al., 2015; Kannan et al., 2016) deployed on public clouds everyday. These text analysis requests contain private emails, personal text messages, and sensitive online reviews. For instance, Gmail smart reply generation needs to scan users' plaintext email messages anonymously (Kannan et al., 2016).

Prior work (Juvekar et al., 2018) proposes a hybrid cryptographic scheme that uses homomorphic encryption (HE) to process linear layers and garbled circuits (GC) to compute activations in a convolutional neural network. Compared to convolutional neural networks (CNN), RNNs can achieve more competitive accuracy in text analysis tasks (Bahdanau et al., 2015; Podschwadt and Takabi, 2020; Bakshi and Last, 2020). Mixing HE and GC presents impressing results in secure classification tasks (Barni et al., 2011). However, mixing HE and GC in RNN will suffer from a long inference latency due to the slow GC-based activations. In contrast to a CNN, a RNN (Bahdanau et al., 2015) requires more types of activations such as $tanh$ and $sigmoid$. The GC protocol (Ohrimenko et al., 2016) has to use a huge garbled table to implement a $tanh$ or $sigmoid$ activation. Both garbling and evaluating such a large table add significant latency to RNN layers. Based on our experimental and theoretical analysis, the GC-based activations can occupy up to 91% of the inference latency in a HE and GC hybrid secure GRU.

To reduce the GC-based activation latency, we propose a novel secure gated recurrent unit (GRU) network framework, CRYPTOGRU, that achieves high security level and low inference latency simultaneously. We use SIMD HE kerel functions from Juvekar et al. (2018) to process linear operations in a GRU cell, while it adopts GC to compute activations. Our contributions are summarized as follows:

- We build a HE and GC hybrid privacy-preserving cryptosystem, CRYPTOGRU, that uses HE operations to process multiplications and additions, and adopts GC to compute activations such as $ReLU$, $tanh$, and $sigmoid$.
- We replace computationally expensive GC-based $tanh$ activations in a GRU cell with fast $ReLU$ activations without sacrificing the inference accuracy. We quantize GC-based $sigmoid$ and $ReLU$ activations with smaller bitwidths to further accelerate activations in a GRU.
- We implement all proposed techniques of CRYPTOGRU and compared CRYPTOGRU against state-of-the-art secure networks.

2 Background and Related Work

Text analysis using GRU. GRU and long short-term memory (LSTM) are two types of RNNs that can capture long term dependencies (Chung et al., 2014), which are important text classification and text generation (Bahdanau et al., 2015). A single LSTM cell has totally $4\times(n^2+nm+n)$ parameters,

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2052–2057
November 7–11, 2021.

while a single GRU cell has only $3\times(n^2+nm+n)$ parameters, where m means the dimension of the input and n for the dimension of the hidden state. Prior studies (Chung et al., 2014; Bahdanau et al., 2015) show GRU can the same level of inference accuracy as LSTM.

Threat model and cryptographic primitives. We consider semi-honest corruptions (Juvekar et al., 2018; Lou and Jiang, 2019; Chou et al., 2020) in our threat model, where a server S is hosting a model and many clients C are sending inputs for inference using S' model. The client and the server adhere the protocol, but attempt to infer information about the other party's input. Our protocol hides model weights, biases, and activations of a network model, which are likely to be proprietary.

HE (Gentry et al., 2009) is a cryptosystem that supports computation on ciphertext without decryption. GC enables two parties (Sender and Receiver) to jointly compute a function over their private data without revealing data beyond output from each other. A GC function is represented by a Boolean circuit with 2-input gates (e.g., XOR, AND, etc.). The Sender garbles the Boolean circuit and generates the garbled table. The Receiver receives the garbled table from an Oblivious Transfer (Juvekar et al., 2018) and then evaluates the table. The total GC communication overhead is proportional to the number of non-XOR gates in the garbling function (Rouhani et al., 2018; Riazi et al., 2018). For instance, a 12-bit $ReLU$ requires only 30 non-XOR gates, while a 12-bit $tanh$ needs $> 2K$ non-XOR gates.

Comparison with prior privacy-preserving inference. Prior studies create GC-only (Ohrimenko et al., 2016), HE-only (Chou et al., 2020; Badawi et al., 2019) and HE+GC hybrid (Juvekar et al., 2018) privacy-preserving neural networks for secure inferences. GC-only secure networks have to pay huge communication overhead and long inference latency, whereas the HE-only networks cannot accurately implement nonlinear activations by only homomorphic multiplications and additions. So, secure networks (Juvekar et al., 2018) implement linear layers with HE operations and nonlinear activations with GC operations. We compare CRYPTOGRU against prior related works in Table 1. PrivFT (Badawi et al., 2019), and FHE-Infer (Chou et al., 2020) are two HE-only secure neural networks. While Gazelle (Juvekar et al., 2018) is one of the first HE and GC hybrid convolutional neural networks, it does not support RNN cells. Although SHE (Lou and Jiang, 2019) uses an emerging HE protocol (TFHE), many TFHE-based activations greatly prolong its inference latency in text analysis tasks. Several prior works HE-RNN (Bakshi and Last, 2020; Podschwadt and Takabi, 2020) use HE to implement linear operations, and return the intermediate encrypted results to the client without non-linear operations.

	Text tasks	Accurate	Efficient	No decrypt
PrivFT	✗	✗	✗	✓
FHE-Infer	✗	✗	✗	✓
Gazelle	✗	✗	✗	✓
SHE	✓	✗	✗	✓
HE-RNN	✓	✓	✗	✗
CRYPTOGRU	✓	✓	✓	✓

Table 1: The comparison of secure models. ✓ means the scheme performs good under such condition or is friendly to the description and ✗ means the opposite.

Latency Bottleneck and Motivation. In a typical GRU cell, there are nine stages of linear operations and two non-linear operations. In out baseline implementation using Gazelle, the non-linear operations take up to 91.37%. Therefore GC-based non-linear operations are the bottleneck in this structure. This is further discussed in Section 3.

3 CRYPTOGRU

3.1 Constructing the base CRYPTOGRU

Conventional neural network inference uses depth-bounded arithmetic circuits (LHE). However, the computation cost is large for the LHE scheme. CRYPTOGRU adopts a simpler HE scheme, namely packed additive homomorphic encryption (PAHE) scheme and garbled circuits (GC).

Algorithm 1: CRYPTOGRU cell

Input: an input ciphertext $[x_t]$
Output: a ciphertext hidden state $[h_t]$
$[i_r],[i_i],[i_n] = MultPC([x],\Omega_i,b_i)$ // HE
$[h_r],[h_i],[h_n] = MultPC([h_{t-1}],\Omega_h,b_h)$ // HE
$[Gate_{reset}] = \mathbf{GCSig}(AddCC([i_r],[h_r]))$ // GC
$[Gate_{input}] = \mathbf{GCSig}(AddCC([i_i],[h_i]))$ // GC
$[Gate_{new}] = \mathbf{GCTanh}($ // GC
$AddCC([i_n], MultCC([Gate_{reset}],[h_n])))$
// HE
$[h_t] = AddCC([Gate_{new}],$ // HE
$MultCC([Gate_{input}],$ // HE
$AddCC([h_{t-1}],-[Gate_{new}])))$ // HE
return $[h_t]$

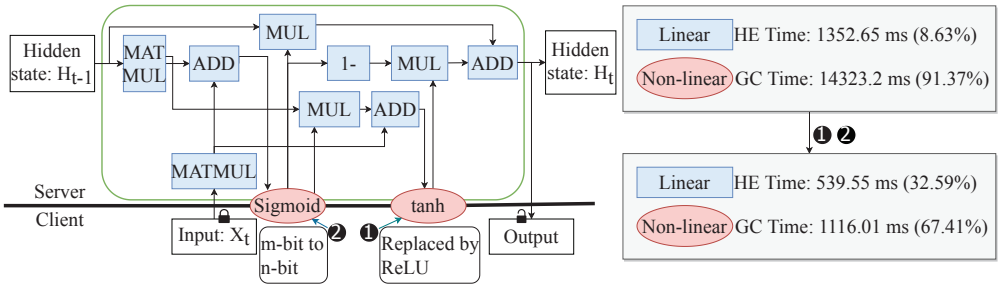

Figure 1: A cell of CRYPTOGRU with HE-based linear operations and GC-based non-linear operations.

Figure 1 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 1. In a GRU cell, linear operations include matrix vector multiplications (*MATMUL*), element-wise add, minus, and multiplications (*ADD*,*1-*,*MUL*). Element-wise minus function is implemented as adding negative elements. In CRYPTOGRU, we map the neural network layers to PAHE matrix-vector multiplication for these linear operations. The activation function $sigmoid$ and $tanh$ are non-linear, which are shown in red in Figure 1. For non-linear operations, we apply garbled circuits. The process of updating hidden states inside a GRU cell is shown in Algorithm 1. Here, the $MultPC$ is a matrix vector multiplication based on HE, where the matrix is plaintext and the vector is ciphertext. We use $[\,]$ to denote a ciphertext. In this function, $[i_r]$ is the product of $[x]$ with the first third of Ω_i, $[i_i]$ is the product of $[x]$ with the second third of Ω_i, and $[i_n]$ is the product of $[x]$ with the last third of Ω_i. This mechanism also applies to the product of $[h_{t-1}]$ with Ω_h. Here, the $AddCC$ and $MultCC$ are element-wise addition and multiplication respectively. In addition, the $GCSig$ and $GCTanh$ are the GC-enabled $sigmoid$ function and $tanh$ function respectively.

3.2 ❶ Replacing $tanh$ with $ReLU$

In this paper, we aim to build a privacy-preserving GRU network for text analysis. However, if we use the HE and GC hybrid technique (Juvekar et al., 2018) to implement a GRU network, the complex GC-based activations including $tanh$ and $sigmoid$ significantly prolong the inference latency. In GRU RNN, the activation nonlinearity function is typically $tanh$ but can also be implemented with the rectified linear unit $ReLU$ (Ravanelli et al., 2018; Chung et al., 2014). More importantly, we investigate that a $ReLU$ activation is more GC-friendly than a $tanh$ activation. A 8-bit $ReLU$ activation requires $\sim 4\times$ less latency than a 8-bit $tanh$ activation since a 8-bit $ReLU$ requires only 24 non-XOR gates, but a $tanh$ needs 95 non-XOR gates.

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

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

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 datasets are used in binary classification tasks. In a one-layer GRU network, we compare the accuracy of using $tanh$ with $ReLU$ on the two datasets and compare the latency for a single sample inference. 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 "CG-❶" in all the following text.

3.3 ❷ Quantizing both $sigmoid$ and $ReLU$

During the computation of a full GRU cell, we identify the latency bottleneck is at non-linear functions. In Figure 1, we show the computation time for non-linear operations hold about 91.37% for a typical case. This is mainly due to the computational complexity for ciphertext is significantly proportional to the underlying bit-length (Riazi et al., 2018). As shown in Table 2, the latency of $ReLU$ is significant less than that of $tanh$ due to simpler computational complexity (Rouhani et al., 2018).

Then, we quantize the all default bit-length from 20 to 8 in activations. First, the design of garbled circuits is proportional simpler after the quantiza-

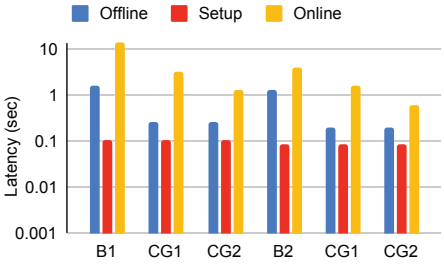

Figure 2: Latency comparison for two sets of experiments. "B1", the first "CG1", and the first "CG2" represents a model with the input size set at 10 and the hidden size set as 128. "B2", the second "CG1", and the second "CG2" represents a model with the input size set at 100 and the hidden size set as 64. "CG1" and "CG2" represents the CG-❶ and CG-❷ respectively. For both baseline cases, the online latency is significant. CG-❶ can reduce the online and offline latency and CG-❷ can further reduce the online latency while maintains the same level of the offline and setup.

tion since the garbled circuits are sensitive to the bit length. Second, since 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 and we compare the latency shown in Figure 2. This benefit is further discussed in Section 4. We label this version as "CG-❷" in all the following text.

4 Experiments and Results

4.1 Cryptographic settings

We develop the CRYPTOGRU with the Gazelle SIMD Homomorphic operations in C++ (Juvekar et al., 2018). 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-circuit scheme. For the homomorphic encryption, we use Brakerski-Fan-Vercauteren (BFV) scheme (Brakerski, 2012). Yao's garbled circuits scheme is used for a two-party secure computation (Yao, 1986). We set the bit-length to 20 for plaintext and 60 for ciphertext in the BFV scheme as explained in Section 3.1.

4.2 Ablation studies from ❶ and ❷

We test our three versions of CRYPTOGRU for the performance with respect to the latency. The results are shown in Table 3 and Figure 2. 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. We use 30 time steps in the default settings. Here we illustrate 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, CG-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 $ReLU$. Benchmark results show that the $ReLU$ function can use 6.4 times less circuit gates for ciphertexts. This version is about 77% faster than the baseline. By contrast, the CG-❷ has the same setup and offline latency as the CG-❶, but the online latency is only about 1290.06ms, resulting that the total latency is 1655.56ms, which is about 54% faster than the CG-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 CG-❶ use a total of 1851.32ms, and the CG-❷ use a total of 913.32ms. In this setting, the CG-❶ shows an improvement of about 66% respect to the latency of the baseline and CG-❷ shows a further improvement of about 51% compared to the CG-❶.

Applying two techniques ❶ and ❷ can decrease the total latency and online message size for GC. Compared with related work, CRYPTOGRU can achieve low latency in a secure inference system and maintain the same level of accuracy. There are some limitations due to the nature of homomorphic computing complexity. In addition, the recurrent computation would raise noise in homomorphic encryption. We mitigate the noise by bootstrapping the ciphertext (Chillotti et al., 2020).

4.3 Results

We test the latency and accuracy of CRYPTOGRU against public datasets and compare the performance with the state-of-art prior related works. We use Enron emails (Klimt and Yang, 2004) and Penn Treebank datasets (Le et al., 2015) that are common to machine learning tasks for text to evaluate the performance of the CG-❷ (referred as CRYPTOGRU in this section) as well as the IMDB and Yelp datasets from Section 3.2. Experiments cov-

Schemes	Input Size	Hidden Size	Total Latency	GC Msg Size
Baseline	10	128	15675.85 ms	213.4 MB
CG-❶	10	128	3590.65 ms	19.4 MB
CG-❷	10	128	1655.56 ms	7.7 MB
Baseline	100	64	5382.91 ms	106.7 MB
CG-❶	100	64	1851.32 ms	9.7 MB
CG-❷	100	64	913.32 ms	3.9 MB

Table 3: The benchmark results of CRYPTOGRU.

Datasets	Neural Networks	Accuracy(%)	Latency
Enron Emails	PrivFT	-	7.95s∗
	CryptoGRU	84.2	**2.03s**
Penn Treebank	SHE	89.8ppw	∼576s
	CryptoGRU	79.4ppw	**4.14s**
IMDB	HE-RNN	86.47∗	70.6s∗
	PrivFT	91.49∗	7.90s∗
	CryptoGRU	84.6	**2.07s**
Yelp Review	PrivFT	96.06∗	7.88s∗
	CryptoGRU	91.3	**0.91s**

Table 4: Results from CRYPTOGRU and related work.

ered in Table 4 are typical classification or regression tasks for text datasets. We use the perplexity per word (PPW) as the target for the Penn Treebank dataset, which means the average log-probability per word. This is a common regression task for this dataset. Enron Emails is a dataset collection consisting of 500,000 emails with subjects and body messages.For Enron email datasets, we classify emails as spam or ham. This is a binary classification task. We perform the binary classification task for the IMDB dataset that labels the reviews either as positive or negative (Maas et al., 2011). For Yelp reviews dataset [1], we also perform the binary classification task. Reviews with a star greater than and equal to 3 are regarded as positive.

We summarize the comparison results in Table 4. For the Penn Treebank dataset, our CRYPTOGRU can infer a sample in 4.14s, which is about 138 times faster than the SHE (Lou and Jiang, 2019). 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 (Podschwadt and Takabi, 2020). The CRYPTOGRU can infer a sample from Enron Emails in 2.03 and Yelp reviews in 0.91s.

5 Conclusion

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. We propose CRYPTOGRU, a secure inference building block for gated recurrent unit that emphasises on text-like or time series models. We elaborate all the improvements based on theoretical analysis and confirm the legitimacy for all optimization means. CRYPTOGRU improves a GRU with homomorphic encryption, share secrets, and garbled circuits heterogeneously to achieve low latency as well as high accuracy.

Code Availability

CRYPTOGRU code is available at: https://github.com/bfeng/CryptoGRU. The public repository also includes software dependencies like the 'cryptoTools' and the 'Gazelle' code. Used datasets from all experiments are downloadable from the internet as described in the text.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work was partially supported by the National Science Foundation (NSF) through awards CCF-1908992, CCF-1909509, and CCF-2105972. This work is partially supported by the National Science Foundation (NSF) through awards CIF21 DIBBS 1443054, SciDatBench 2038007, CINES 1835598 and Global Pervasive Computational Epidemiology 1918626.

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. 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*.

[1] https://www.ics.uci.edu/~vpsaini/

Maya Bakshi and Mark Last. 2020. Cryptornn-privacy-preserving recurrent neural networks using homomorphic encryption. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 245–253. Springer.

Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. 2011. Privacy-preserving ecg classification with branching programs and neural networks. *IEEE Transactions on Information Forensics and Security*, 6(2):452–468.

Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer.

Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91.

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

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

Craig Gentry et al. 2009. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing*, volume 9, pages 170–178.

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

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

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious multi-party machine learning on trusted processors. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 619–636.

Robert Podschwadt and Daniel Takabi. 2020. Classification of encrypted word embeddings using recurrent neural networks. In *PrivateNLP@ WSDM*, pages 27–31.

M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. 2018. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102.

M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721.

Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6.

Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE.