An Adversarial Neural Cryptography Approach to Integrity Checking: Learning to Secure Data Communications

Korn Sooksatra* and Pablo Rivas†, *Senior, IEEE*
School of Engineering and Computer Science
Department of Computer Science, Baylor University
*Email: Korn_Sooksatra1@Baylor.edu, †Email: Pablo_Rivas@Baylor.edu

***Abstract*—Securing communications is an increasingly challenging problem. While communication channels can be secured using strong ciphers, attackers who gain access to the channel can still perform certain types of attacks. One way to mitigate such attacks is to verify the integrity of exchanging messages between two parties or more. While there are robust integrity check mechanisms currently, these lack variety and very few are based on machine learning. This paper presents a methodology for performing an integrity check inspired by recent advances in neural cryptography. We provide formal, mathematical functions and an optimization problem for training an adversarial neural cryptography architecture. The proposed neural architectures can adequately solve the problem. In our experiments, a receiver can verify if incoming messages are authentic or altered with an accuracy greater than 99%. This work expands the repertoire of integrity checking methodologies, provides a unique perspective based on neural networks, and facilitates data security and privacy.**

***Index Terms*—integrity check, neural cryptography, adversarial learning, cybersecurity, data privacy**

I. Introduction

In recent years, cryptography has been used for several applications, especially secure communications [1] which are being used worldwide for the safety and privacy of their participants (i.e., senders and receivers). Besides, some machine learning applications on a cloud platform (i.e., collaborate machine learning [2]–[4]) have applied the study of cryptography for participants' privacy. For example, [5]–[8] utilized homomorphic encryption [9] for performing operations between encrypted data.

Furthermore, the study of machine learning is also capable of being used in the field of cryptography as seen in [10]–[17]. On one hand, researchers improved encryption algorithms of secure communications only when they found potential strong attacks. On the other hand, machine learning models can add evolutionary adversaries to their schemes to strengthen themselves with the strong attacks' existence. Specifically, Generative Adversarial Networks (GANs) [18] has been utilized for the training in those schemes as being discussed in [19], [20] which considered the privacy of senders and receivers in secure communications. Those schemes are called *adversarial neural cryptography*.

This work was supported, in part, by Rivas.AI Lab.

In addition to the participants' privacy in secure communications, the integrity of their messages should be considered so that receivers can be sure that the incoming messages are not altered and from legitimate senders. Although it is challenging to perform the error correction for the altered messages [21], [22], the integrity check is a promising solution. We want to show that a machine learning model in the form of adversarial neural cryptography can learn how to perform the integrity check in the way of cryptography, although the cryptographic algorithms have been practically used.

Inspired by [19], in this paper we present a methodology based on adversarial neural cryptography for performing the integrity check between participants in a secure communication channel. In this scenario, the receiver can check incoming messages if they are altered with high accuracy. We begin discussing an integrity check among participants under an adversarial configuration; then, we pose an optimization problem for training the model's architecture. Our experiments and evaluation show that the proposed methodology is able to achieve our goal.

The paper is organized as follows: Section II explains the knowledge and motivation of this work; Section III describes the problem of integrity check and the model of the adversarial neural cryptography scheme; Section IV formulates the problem into mathematical functions and an optimization problem; Section V describes the architectures of all entities in the scheme; Section VI shows the detail of training refinement step; Section VII illustrates the evaluations during the training refinement step and the results; Section VIII further discusses the results and the remaining works after this, and everything is concluded in Section IX.

II. Background

This section describes the evolution of machine learning used in cryptography. At first, a machine learning scheme consists of a sender and a receiver so that the sender can learn how to encrypt a plaintext and the receiver can learn how to decrypt a ciphertext to communicate to each other, and this kind of scheme is briefly explained in Section II-A. Then, to strengthen the encryption scheme to be against a strong attack, an adversary was added to the learning schemes. This kind of

approach is described in Section II-B and is utilized in our approach.

A. Machine Learning Cryptography

On these years, machine learning models have been applied in the field of cryptography. The existing works [10]–[16] utilized neural network models to create encryption schemes for secure communications. They treated the parameters of neural networks as keys for the schemes which are not very secure since an adversary can create the substitute models that can resemble the target models as described in [23]–[25].

Moreover, Shruti et.al [17] exploited a deep learning model to create an encryption scheme including a key generation system. The whole system consists of a genetic algorithm [26] and a DNA computing [27]. They implemented the key generation system with the genetic algorithm where its fitness function can measure how random each population is. Then, the selected key is used in the DNA computing to encrypt a message and decrypt a ciphertext. Specifically, this DNA computing consists of transcription and translation [27].

Although these approaches can provide the security for messages in secure communications, they do not guarantee that it can be against strong attackers because there is always weaknesses in every encryption scheme. Further, learning models can find countermeasures against the strong attackers when only they adjust themselves with the attackers in their schemes. Therefore, some recent works have included attackers in their scheme and these are explained in Section II-B.

B. Adversarial Neural Cryptography

This approach utilizes GANs [18] by adding an adversary into the learning scheme. Abadi et.al. [19] designed this approach by including Alice, Bob and Eve in their scheme. this starts by Alice encrypting message P with key K to ciphertext C. Then, Eve eavesdrops C, and Bob receives C whereas only Alice and Bob have key K. At last, Eve and Bob decrypt ciphertext C and respectively obtain P_{Eve} and P_{Bob}. Obviously, Eve is the adversary of this scheme. The ultimate goal is that Alice and Bob can fully communicate to each other while Eve cannot read the plain message P. The scheme can be demonstrated in 1.

The loss functions for Alice, Bob and Eve were carefully designed for the refinement step to make Bob able to decrypt ciphertext C and Eve unable to do it. As a result, Bob can almost fully recover plaintext P with the accuracy of greater than 95%. Furthermore, Eve cannot correctly decrypt ciphertext C with the accuracy of almost 50% which is not different to the random guess. Later, Coutinho et.al [20] empirically showed that the encryption between Alice and Bob in the scheme can be improved when Eve is stronger by providing more information to it. Specifically, they enhanced Eve by considering it as a chosen-plaintext adversary.

The aforementioned existing works of this kind of approach (i.e. adversarial neural cryptography) only focus on confidentiality which is one of security principles [28]. There is also another important principle that is not considered and on which

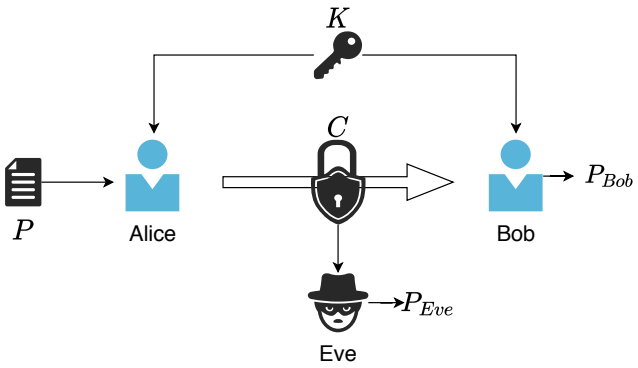


Fig. 1: Learning scheme for encryption with adversarial neural cryptography.

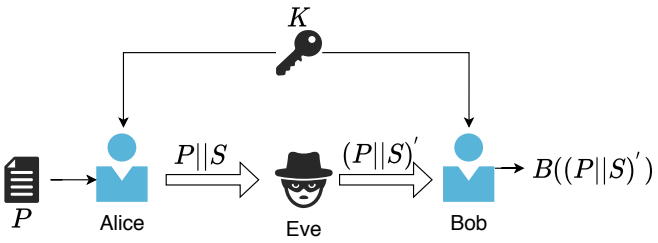


Fig. 2: Man-in-the-middle attack scheme.

cryptography can be applied. That is integrity. Therefore, our work focuses on how to design a scheme to let machine learning models learn how to perform the integrity check by using adversarial neural cryptography approach to be prepared for a strong adversary. To the best of our knowledge, this is the first work that utilizes adversarial neural cryptography for providing integrity of messages in secure communications.

III. Problem and System Model

The scheme of this work is similar to the one described in Section II-B which consists of Alice as the sender, Bob as the receiver and Eve as the adversary. Alice desires to send a message (P) to Bob; however, Eve is in the middle between them and performs man-in-the-middle attack. Specifically, Eve receivers P from Alice, is capable of alter some parts of P and relays this altered message $P^{'}$ to Bob. Therefore, Bob receives $P^{'}$ and does not know that this message has been altered.

To prevent the communication from this kind of attack, Alice then adds the signature S computed from the combination of P and the key K so that Bob can find if the received message has been altered by considering P, S and K. This solution has been done in many existing works in term of cryptography (e.g. CRC [29], HMAC [30] and digital signature technique [31]). Thus, the scheme can be illustrated by Fig. 2 where $\cdot||\cdot$ is the concatenation operator and $B(x)$ is the alteration-check function outputting 0 if x is a message from Alice and not altered and 1 otherwise.

The mechanism of this scheme works as follows:

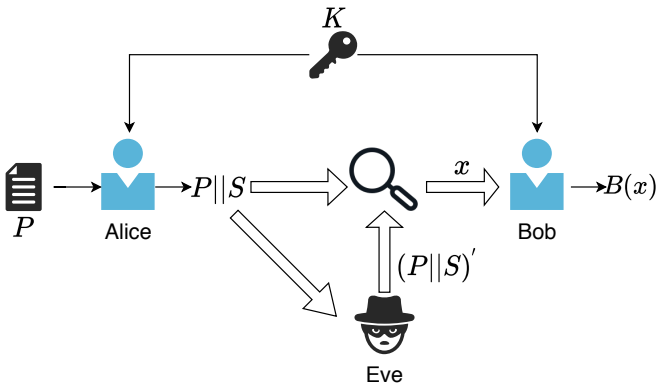


Fig. 3: Training model for integrity check.

1) Alice takes P and K as its input and computes signature S.
2) Alice concatenates P and S and tries to send $P||S$ to Bob; nonetheless, Eve is in the middle between them.
3) Eve receives $P||S$ and relays $(P||S)^{'}$ to Bob.
4) Bob receives $(P||S)^{'}$ and computes $B((P||S)^{'})$ where $B : \mathbb{R}^{|P|+|S|} \rightarrow \mathbb{R}$. If $P||S = (P||S)^{'}$, $B((P||S)^{'}) = 0$ and $B((P||S)^{'}) = 1$ otherwise.

In this work, Alice, Bob and Eve are neural networks, and the goal is to let Alice and Bob learn how to create the signature and identify whether the received message has been altered respectively. Further, to make Alice and Bob strong, they have to be against a strong adversary. Hence, Eve also needs to learn how to alter $P||S$ so that Bob cannot identify that $P||S \neq (P||S)^{'}$.

The training model is shown in Fig. 3 where x is selected from either $P||S$ or $(P||S)^{'}$. The algorithm for training Alice, Bob and Eve and their architectures are mainly adopted and modified from generative adversarial networks (GANS) [18] and the adversarial neural cryptography in [19]. These all will be explained in the subsequent sections.

IV. Problem Formulation

Alice's and Bob's goal is to maximize the alteration probability ($B(\cdot)$) when $(P||S)^{'}$ is different to $P||S$ and minimize it when $(P||S)^{'}$ is not different to $P||S$. Note that $A(\cdot)$, where $A : \{0,1\}^{|P|+|K|} \rightarrow \{0,1\}^{|P|} \times \mathbb{R}^{|S|}$, is the signature function of Alice and $E(\cdot)$, where $E : \mathbb{R}^{|P|+|S|} \rightarrow \mathbb{R}^{|P|+|S|}$, is the alteration function of Eve. Therefore, Alice's and Bob's loss function can be formulated as

$$L_{AB}(x) = log(1 + B(A(x))) + log(2 - B(E(A(x)))), \quad (1)$$

where x is a plaintext. The first part of (1) is to make $B(\cdot)$ close to 0 when the input is directly from Alice, and the other part is to make $B(\cdot)$ close to 1 when the input is from Eve. Additionally, $1+B(A(x))$ and $2-B(E(A(x)))$ are to ascertain the valid value since $log(0) = -\infty$. Further, the expected value of (1) over the training dataset D is formulated as

$$\mathbb{L}_{AB,D} = \mathbb{E}_{x \in D}(L_{AB}(x)), \quad (2)$$

where each data x in D is a message in form of P which is Alice's input.

On the other hand, Eve's goal is to alter $P||S$ to $(P||S)^{'}$ as much as possible; at the same time, it needs to make Bob not able to realize that it alters $P||S$. Hence, its loss function can be formulated as

$$L_E(x) = log(1 + B(A(x))) - c \cdot d(A(x), E(A(x))), \quad (3)$$

where x is a plaintext, $d(X,Y) = \Sigma_{i=1}^{|P|}|X_i - Y_i|$, X_i is the value of X at index i and c is its hyperparameter. The first part of (3) is to fool Bob that $P||S = (P||S)^{'}$ and the last part is to alter $P||S$ as much as possible. Note that if c is large, Eve will focus on increasing $d(P||S, (P||S)^{'})$, and thereby Eve may not be successful fooling Bob.

Additionally, the expected value of (3) over training dataset D can be formulated as

$$\mathbb{L}_{E,D} = \mathbb{E}_{x \in D}(L_E(x)). \quad (4)$$

Further, it is worth noting that every value in S and $(P||S)^{'}$ is in the range $[-1, 1]$.

V. Neural Network Architecture

Since we want Alice to create S with respect to P and K, we apply the architecture from [19]; however, we use only a part of it because creating S does not need the whole network. Specifically, Alice's architecture starts with the input layer which is $P||K$, and then the next layer is a fully-connected(FC) layer to mix them together. After that, the subsequent layers are two 1-dimension convolutional layers to spatially transform them to the signature, and the last layer is a FC layer to make sure that the size of the output is $|S|$. All the layers are followed by *tanh* activation functions.

Bob's architecture is similar to Alice's except that after the two convolutional layers, there are two more FC layers followed by *ReLU* activation functions and one more layer followed by a *sigmoid* activation function. These additional FC layers are for the classification task which is to detect the alteration of the incoming message.

Lastly, Eve's architecture is also similar to Alice's except that Eve has two extra FC layers after the convolutional layers because it lacks of K as its input. The architecture of all entities are shown in Table I where the values in Dense columns are the numbers of neurons in the particular layers, and the values in 1D-Conv are the numbers of filters, the sizes of the particular filters and the strides respectively.

VI. Training Refinement

From the above formulations, we can train Alice, Bob and Eve as follows. We design to run the training in η epochs. In each epoch:

1) Set Eve to be untrainable since Alice and Bob is trained first, and they need to use Eve during their training part.
2) Train Alice and Bob one time with dataset D.
3) Set Alice and Bob to be untrainable because training Eve needs to use Alice and Bob, and set Eve trainable.

TABLE I: The architecture of our proposed adversarial neural cryptography.

Entity ↓, Layer →	Dense	Dense	1D-Conv	1D-Conv	1D-Conv	Dense	Dense	Dense
Alice	$\|P\|+\|K\|$	-	2,4,1	4,2,2	-	-	-	$\|S\|$
Bob	$\|P\|+\|K\|$	-	2,4,1	4,2,2	4,1,1	$\|P\|+\|K\|$	$(\|P\|+\|K\|)/2$	1
Eve	$\|P\|+\|K\|$	$\|P\|+\|K\|$	2,4,1	4,2,2	-	-	$\|P\|+\|K\|$	$\|P\|+\|K\|$

4) Train Eve twice since Eve has to be strong so that Alice and Bob can be trained again a strong adversary.

Note that a gradient-based method (e.g. *SGD* [32], *Adam* [33] and *RMSprop* [34]) is applied to each training. This procedure can be described in Algorithm 1.

VII. Experiments and Results

The following paragraphs describe the experimental setup and the results of our experiments.

A. Setup

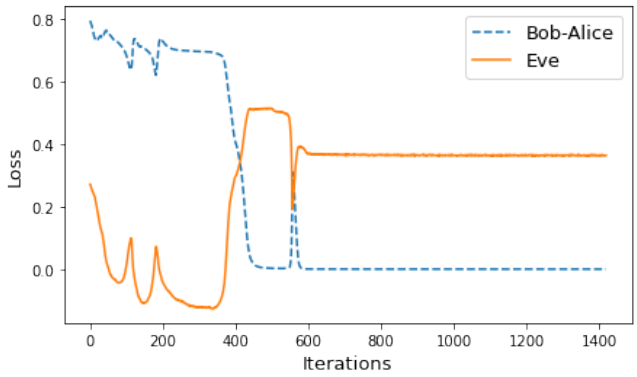
The integrity check of adversarial neural cryptography is implemented and simulated by Python 3 and Tensorflow 2. The plaintext's and key's sizes are 16. The number of training data is $2^{|P|}$, and the batch size is 512. Further, all the training data is randomly generated, and the optimizer for training the models is *Adam* [33] with the learning rate of 0.001. The experiments and evaluations include:

- The explanation of Alice's, Bob's and Eve's behaviors during the training.
- The effect of hyperparameter c and $|S|$ on the convergence time during the training.
- The statistical score of the results on different hyperparameters.

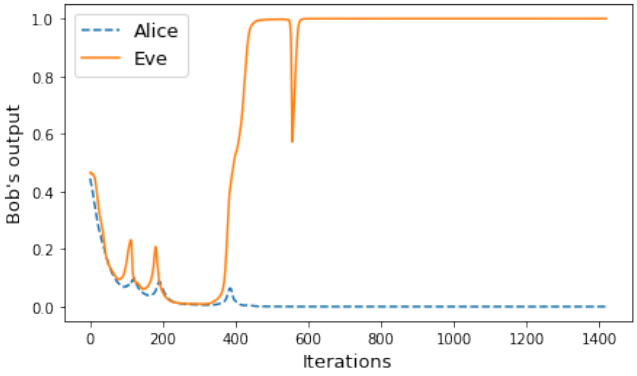
B. Alice's, Bob's and Eve's behaviors

With $c = 0.01$, $|S| = 5$ and $|P| = |K| = 16$, Eve can fool Bob from iteration 0 to 400 as seen in Fig. 4. Fig. 4a shows the loss values of Alice, Bob and Eve during the training and illustrates that Eve's loss is very low before iteration 400 Bob's loss is significantly high. However, Eve's loss goes higher and higher after iteration 400; in contrast, Bob's becomes much lower, and their losses are converge around iteration 600. At last, Eve cannot fool Bob after iteration 600 as demonstrated in Fig. 4b which shows Bob's output when the input is from Alice and Eve.

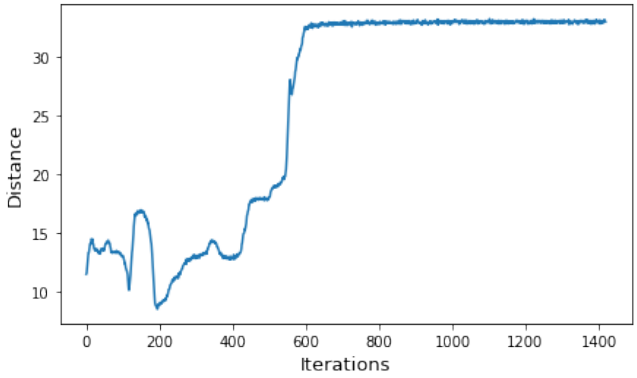
Additionally, Fig. 4c shows the distance between Alice's and Eve's outputs (i.e. $P||S$ and $(P||S)'$ respectively) during the training. It can be noticed that Eve is very strong because the distance is very low around iteration 200. Nevertheless, Eve decides to increase the distance at iteration 400 since it realizes that it cannot reduce its loss anymore if it reduces the distance because Bob is becoming stronger and stronger. Finally, the distance is very high and converge at iteration 600.



(a) Loss of Bob and Alice and Eve.



(b) Bob's output with Alice's and Eve's outputs as input.



(c) The distance between Alice's and Eve's output.

Fig. 4: The evaluation values over training iterations with $c = 0.01, |S| = 5$ and $|P| = 16$.

C. The effect of c and $|S|$ on the convergence time

As being discussed, c is the regularizer to balance between fooling Bob and how messy the message is in (3). Hence, when setting c to 0, Eve does not change the message received from Alice and simply relay the same message to Bob. In contrast, when setting c to a large real number, Eve changes the received message a lot; however, it does not care if Bob will realize that the message is altered. Consequently, c should not be set too low or too high. Further, c impacts on the convergence time as demonstrated in Fig. 5 which shows Bob's output during

Algorithm 1: Gradient-based optimization algorithm for integrity check.

Input: Alice, Bob, Eve, η
for $i = 1 : \eta$ **do**
  Set Alice and Bob into a trainable state;
  Set Eve to be untrainable, allowing Alice and Bob to learn to communicate;
  Randomly generate a set of plaintext P and unique key K as D;
  Train Alice and Bob by using $\nabla \mathbb{L}_{AB,D}$;
  Set Alice and Bob untrainable, allowing Eve to learn to break the learned encryption protocol;
  Set Eve trainable and let the model train for the next two iterations;
  for $j = 1 : 2$ **do**
    Randomly generate a set of P and K as D;
    Train Eve by using $\nabla \mathbb{L}_{E,D}$;
  end
end
Result: Parameters of Alice, Bob and Eve

the training with different c. Fig. 5a shows that Bob's output can be convergence at iteration 50 which is much earlier than Bob's outputs in Fig. 5b and 5c because it has the least c. Hence, Eve does not focus much on fooling Bob, and thereby Bob can easily beat Eve very early. Similarly, Bob's output in Fig. 5b is convergence earlier than the one in Fig. 5c with the same reason. Nonetheless, Bobs with $c = 0.01$ and $c = 0.005$ are comparatively struggled fighting with Eve as seen in Fig. 5b and 5c. Thus, Bobs in those settings can be stronger than the one with $c = 0.1$.

Further, Fig. 6 illustrates that Bob's output is always 0 no matter its input is from Alice or Eve. Hence, this can be claimed that with comparatively small $|S|$, Bob cannot find the difference Alice's output and Eve's output.

D. Statistical scores of the results

The test data is generated to evaluate Alice and Bob in several settings and has three part: 10000 Alice's outputs, 5000 Eve's outputs and 5000 randomized values. We assume that it is very difficult to randomize a value to mimic Alice's output. Therefore, randomized values are not legitimate for Bob. In the other words, there are 10000 legitimate data and 10000 altered data; hence, this test data is balanced. In addition, Eve generating those test data was trained with $c = 0.005$. Moreover, in each setting, the optimal threshold is obtained from the threshold that achieves the maximum difference between true positive rate (TPR) and false positive rate (FPR).

Table II shows the confusion matrix of the result in the setting of $c = 0.005$ and $|S| = 6$ and demonstrates that there is only 12 messages from Alice that are misclassified as from others (false positives). Also, only 19 messages from others are misclassified as from Alice, and most of them are randomly generated test data which can be outliers. The rest of this section shows the empirical score indicating the performance of our approach.

According to Table III, Bob's accuracy scores are greater than 0.99 in most of the settings. In the other words, it can accurately detect that its input is from Alice, Eve or other sources. Further, this metric is good for the test data because the test data is balanced.

TABLE II: Confusion matrix in the setting of $c = 0.005$ and $|S| = 6$.

		Predicted		
		Alice	Others	Total
True	Alice	9988	12	10000
	Others	19	9981	10000
	Total	10007	9993	N

TABLE III: Accuracy score for Bob under several training settings.

$\|S\| \downarrow, c \rightarrow$	0.1	0.01	0.005
4	0.723	0.998	0.996
5	0.989	0.998	0.997
6	0.996	0.994	0.998

TABLE IV: F1 score for Bob in several training settings.

$\|S\| \downarrow, c \rightarrow$	0.1	0.01	0.005
4	0.625	0.998	0.996
5	0.990	0.998	0.997
6	0.996	0.994	0.998

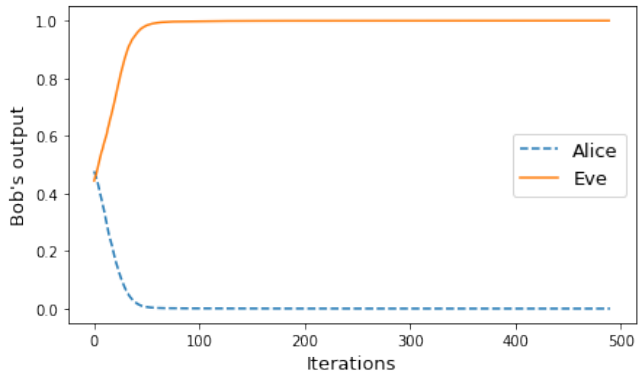
TABLE V: AUC score for Bob in several training settings.

$\|S\| \downarrow, c \rightarrow$	0.1	0.01	0.005
4	0.717	1.000	1.000
5	0.999	1.000	1.000
6	1.000	0.999	1.000

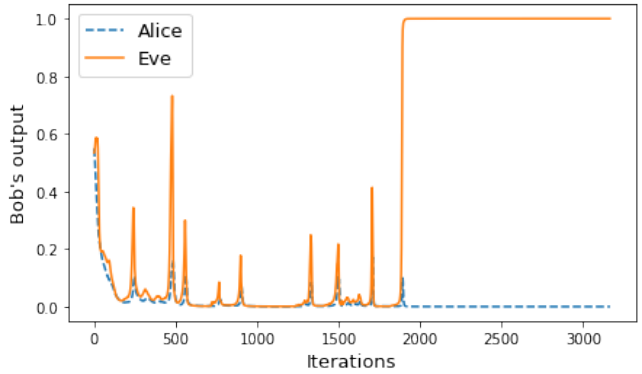
Table IV shows that Bob's f1 scores are also very high in most of the settings. This can be indicated that there are very low false negatives and false positives.

Additionally, Table V indicates that Bob can achieve high values (i.e. almost 1 or 1) of area under the ROC curve (AUC) score which shows how good the models are in all possible thresholds.

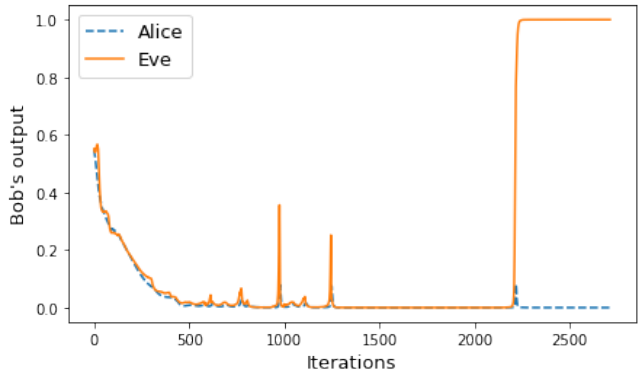
In conclusion, as demonstrated in Table III, IV and V, Bob's performance tends to be better as $|S|$ increases. The reason is that Alice can keep more information when $|S|$ is larger; hence, it is hard for Eve to find a way to alter the



(a) $c = 0.1$



(b) $c = 0.01$



(c) $c = 0.005$

Fig. 5: Bob's output over training iterations with $|S| = 4$ and $|P| = 16$.

information in the message without Bob's perceive. Further, the only setting that makes Bob result the significantly low performance is $|S| = 4$ and $c = 0.1$. The reason is that it never adjusted itself with strong Eve as shown in Fig. 7 in which most of the time, the distance between Alice's and Eve's outputs are clearly high compared to the distance in Fig. 4c.

VIII. Discussion and Future Works

As can be seen in Section VII, our approach in the form of adversarial neural cryptography can check the integrity of messages by the high score of accuracy. F1 and AUC indicate a good performance of our approach as well. However, some of our results are not satisfying because Eve is weak during their training, as demonstrated in the setting of $c = 0.1$ and $|S| = 4$. Furthermore, the combination of encryption and integrity check is considered capable of being used in the real world by two schemes: *integration* and *mixture*.

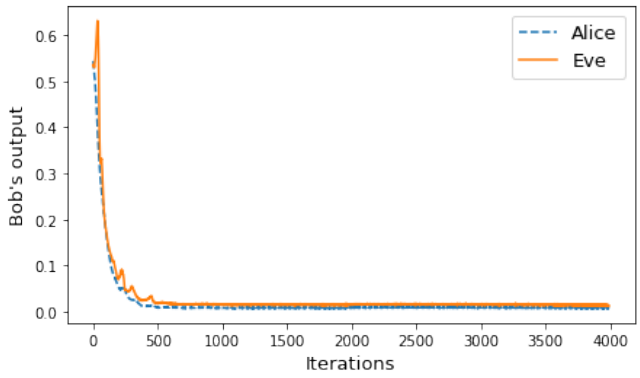


Fig. 6: Bob's output over training iterations with $c = 0.01$, $|S| = 1$ and $|P| = 16$.

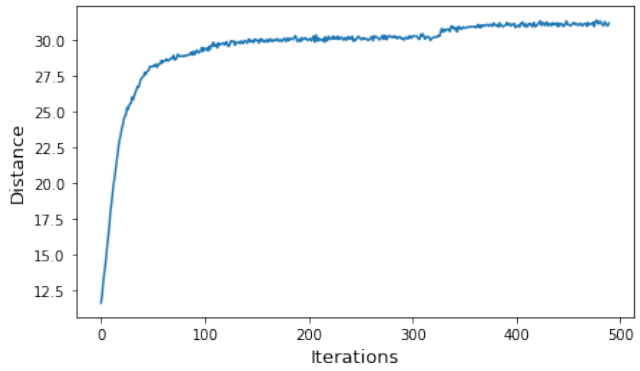


Fig. 7: Distance between Alice's and Eve's outputs during the training with $c = 0.1$, $|S| = 4$ and $|P| = 16$.

First, in the *integration* scheme, we can integrate our approach to the one in [19] which is discussed in Section II-B. Specifically, plaintext P is extended with signature S by Alice in our scheme, and then $P||S$ is encrypted by Alice in [19] to obtain ciphertext C. Then, C is sent to Bob. Hence, although Eve can eavesdrop C, it cannot obtain plaintext P. Even in the worst case where Eve can perform a man-in-the-middle attack, alter some information in C and relay it to Bob, Bob can check its integrity by the procedure explained later. After Bob receives C, he can decrypt it by Bob in [19] to obtain $P||S$ and then check if $P||S$ was altered by Bob in our approach. At last, if $P||S$ is clean, Bob can obtain P. Fig. 8 shows the mechanism of this scheme.

Second, in the *mixture* scheme, Alice's and Bob's architectures are massive since Alice has to encrypt plaintext P and add signature S; at the same time, and Bob needs to simultaneously decrypt ciphertext C and check its integrity.

We leave the study and comparison of those two schemes

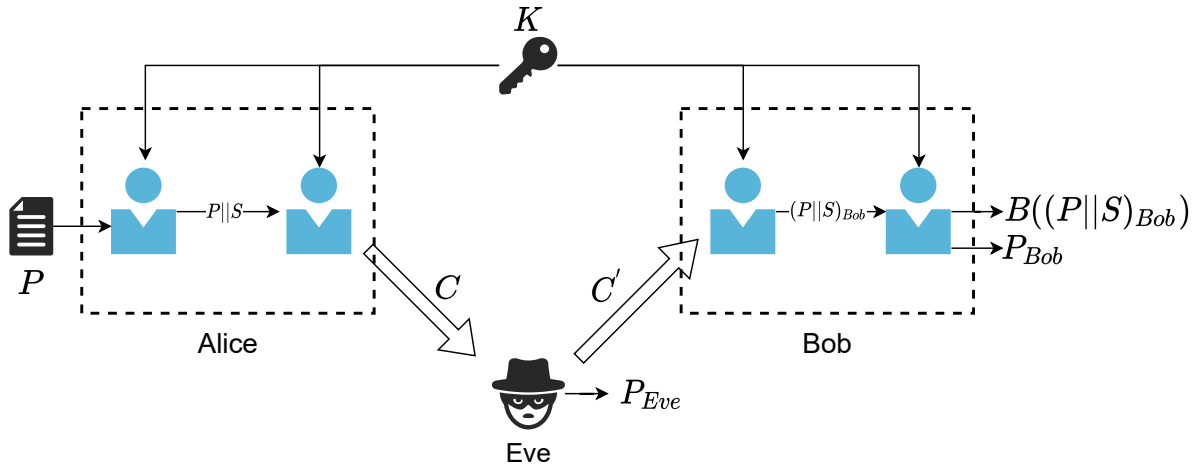


Fig. 8: This is the integration scheme where the model in [19] is integrated with our approach. That is, Alice in our model receives P and passes $P||S$ to Alice in [19] to produce C. Further, Bob in [19] receives $C^{'}$ and passes $(P||S)_{Bob}$ to Bob in our model to check if it is legitimate. Eventually, after training, this scheme can provide the confidentiality and integration principles for the participants.

and the architecture and training refinement of the *mixture* scheme for future works.

In addition, to make the approach feasible to be practically used, it has to be robust to adversarial examples as described in [35]–[38] because if this approach is vulnerable to adversarial examples, an adversary can change the semantics of the plaintext to what he/she wants where the sender and the receiver do not notice about it. Such study is still an open problem and is left for future work on adversarial neural cryptography.

IX. Conclusion

This paper presents a methodology for performing an integrity check inspired by neural cryptography [19]. In particular, we use adversarial neural cryptography for performing an integrity check between participants in a secure communication channel. As a result, the receiver can test if incoming messages to see if they are altered; these tests are successful with an accuracy of greater than 99%.

We can summarize our contributions as follows:

- We provide a thorough, rigorous discussion of an integrity check mechanism between participants in a secure communication channel.
- We provide formal, mathematical functions and an optimization problem for training an adversarial neural cryptography architecture.
- The architectures of the sender, receiver, and adversary are designed by utilizing state-of-the-art models [19], and trained with the refinement strategies discussed in the literature [18].
- We provide detailed, reproducible instructions on the evaluation process, the training refinement, and the experiments that led to the results achieved.

While there is more work to do in some areas mentioned before, we believe these types of schemes can transform the way we will use neural networks in the security and privacy of data communications.

Acknowledgements

The authors thank the Department of Computer Science for their support under the Excellence Fund. This research was also funded, in part, by Rivas.AI Lab.

References

[1] R. E. Blahut, *Cryptography and secure communication*. Cambridge University Press, 2014.

[2] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

[3] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5904–5914.

[4] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2537–2551, 2017.

[5] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.

[6] O.-A. Kwabena, Z. Qin, T. Zhuang, and Z. Qin, "Mscryptonet: Multi-scheme privacy-preserving deep learning in cloud computing," *IEEE Access*, vol. 7, pp. 29 344–29 354, 2019.

[7] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[8] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[9] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford university Stanford, 2009, vol. 20, no. 9.

[10] E. Volna, M. Kotyrba, V. Kocian, and M. Janosek, "Cryptography based on neural network." in *ECMS*, 2012, pp. 386–391.

[11] H. Noura, A. E. Samhat, Y. Harkouss, and T. A. Yahiya, "Design and realization of a new neural block cipher," in *2015 International Conference on Applied Research in Computer Science and Engineering (ICAR)*. IEEE, 2015, pp. 1–6.

[12] W. Kinzel and I. Kanter, "Neural cryptography," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, vol. 3. IEEE, 2002, pp. 1351–1354.

[13] T. Godhavari, N. Alamelu, and R. Soundararajan, "Cryptography using neural network," in *2005 Annual IEEE India Conference-Indicon*. IEEE, 2005, pp. 258–261.

[14] A. Ruttor, W. Kinzel, L. Shacham, and I. Kanter, "Neural cryptography with feedback," *Physical Review E*, vol. 69, no. 4, p. 046110, 2004.

[15] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[16] A. Ruttor, W. Kinzel, and I. Kanter, "Neural cryptography with queries," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 01, p. P01009, 2005.

[17] S. Kalsi, H. Kaur, and V. Chang, "Dna cryptography and deep learning using genetic algorithm with nw algorithm for key generation," *Journal of medical systems*, vol. 42, no. 1, p. 17, 2018.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[19] M. Abadi and D. G. Andersen, "Learning to protect communications with adversarial neural cryptography," *arXiv preprint arXiv:1610.06918*, 2016.

[20] M. Coutinho, R. de Oliveira Albuquerque, F. Borges, L. J. Garcia Villalba, and T.-H. Kim, "Learning perfectly secure cryptography to protect communications with adversarial neural cryptography," *Sensors*, vol. 18, no. 5, p. 1306, 2018.

[21] S. W. McLaughlin, D. Klinc, B.-J. Kwak, and D. S. Kwon, "Secure communication using error correction codes," Jul. 9 2013, uS Patent 8,484,545.

[22] G. Kaddoum and F. Gagnon, "Error correction codes for secure chaos-based communication system," in *2010 25th Biennial Symposium on Communications*. IEEE, 2010, pp. 193–196.

[23] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[24] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[25] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *arXiv preprint arXiv:1804.08598*, 2018.

[26] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[27] Q. Limin, "The study of dna-based encryption method [d]," *Zheng Zhou: Zheng Zhou University of Light Industry*, 2008.

[28] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[29] T. V. Ramabadran and S. S. Gaitonde, "A tutorial on crc computations," *IEEE Micro*, vol. 8, no. 4, pp. 62–75, 1988.

[30] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," 1997.

[31] X. Serret-Avila and G. Boccon-Gibod, "Methods and systems for encoding and protecting data using digital signature and watermarking techniques," Aug. 31 2004, uS Patent 6,785,815.

[32] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Technical Report*, 2012.

[35] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[36] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[37] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *2018 ieee security and privacy workshops (spw)*. IEEE, 2018, pp. 36–42.

[38] P. Tabacof, J. Tavares, and E. Valle, "Adversarial images for variational autoencoders," *arXiv preprint arXiv:1612.00155*, 2016.