Secure Generative Adversarial Networks

Subhasis Thakur[1] and John Breslin[2]

[1] University of Galway, Galway, Ireland subhasis.thakur@universityofgalway.ie
[2] University of Galway, Galway, Ireland john.breslin@universityofgalway.ie

Abstract. Generative Adversarial Networks (GAN) can produce realistic synthetic data that can be used in many applications including training other neural networks. However, a malicious GAN can intentionally exclude specific data while training the GAN model and it can create an incorrect neural network model. In this paper, we propose a secure GAN model where we partition the GAN model among two entities (a) a GAN provider (only who knows the discriminator model of GAN) and (b) a GAN user (who builds the generator model of the GAN). The GAN provider does not share its training data for the discriminator model but allows the GAN user to choose the type of data to be used to train the discriminator model. The GAN provider and the GAN user engage in oblivious transfer and zero-knowledge proof protocol to verify (a) the correct discriminator model is developed by the GAN provider, and (b) the correct discriminator model is used to train the generator model. We prove the proposed protocols for building GAN are secure and privacy-preserving. We also present an experimental evaluation of the proposed GAN.

Keywords: Generative Adversarial Network · Zero-knowledge proofs · Oblivious transfer.

1 Introduction

Generative Adversarial Network (GAN) produces synthetic data with a probability distribution matching the real data. There are many applications of GAN as the synthetic data can be used as datasets to train various machine learning models. A GAN model has three main components, (a) a real dataset, (b) a neural network for Discriminator (D), and (c) a neural network for Generator (G). A GAN operates as follows: first D is trained with data from real data (labeled as 1) and fake data (labeled as 0) to efficiently determine if a data is real data. Next, the generator is trained as it takes a random input, and its outcome is used as input to D to decide if the discriminator will accept fake data generated by the discriminator. Usually, Binary Cross Entropy is used as a loss function to train D and G. In this paper, we build a privacy-preserving and secure GAN model where a GAN provider owns the training datasets for the discriminator model and does not want to share such data and the discriminator model with the GAN user, who builds the GAN generator neural network by sending the outcomes from the generator neural network to the discriminator network. The

GAN user does not control the execution of the discriminator model and it must verify outcomes from the discriminator neural is the result of the actual execution of a neural network constructed using the data types requested by the GAN user. In this paper, we build a secure and privacy-preserving GAN model where our main results are as follows: (1) We present an oblivious transfer-based protocol to allow the GAN user to securely choose data types to train the discriminator model and to choose sample datasets to be used to verify the correct training and usage of the discriminator neural network. (2) We present a zero-knowledge-proof-based protocol to verify the discriminator model is correctly trained with the requested data categories. (3) We present a zero-knowledge proof-based protocol to verify that a correct discriminator model is used to train the generator neural network. (4) We present an analytical and experimental evaluation of the proposed GAN model. The paper is organised as follows: in section 2 we discuss the privacy and security problems with GAN, in section 3 we present the secret GAN model, in section 4 we prove that the proposed GAN model is secure and privacy-preserving, in section 5 we present an experimental evaluation of the GAN, in section 6 we discuss related literature, and we conclude the paper in section 7.

2 Security problems with GAN

The overview of GAN is as follows: The neural network for the Discriminator D is trained with a combination of real data (labeled as 1) and fake data (labeled as 0) such that it can decide if the input data belongs to real data or not (outputs 1 or 0). After training D, the neural network for the generator is trained as it takes random input data to the neural network of G. The outcome of G is used as the input to the neural network of D. D classifies the inputs from G into either 1(classifies input from G as real data) or 0 (classifies input from G as fake data). Based on the classification of D, a loss function is used to train G. The interaction between G and D can be modelled as a minimax game. If the GAN user does not have access to train the discriminator model and parameters of the discriminator model then we have the following privacy and security problems: (1) A malicious GAN provider may not use the proper data to train the discriminator neural network. (2) A malicious GAN provider may not use the proper discriminator model to train the generator model. (3) Data to train the discriminator model and discriminator model can not be revealed to the GAN user to verify if proper discriminator model is developed and used. In this paper, we present a GAN model to address these security and privacy problems.

3 Secure GAN Model

Our solution to construct a secure GAN model is as follows: (1) First, the GAN provider and the GAN user will engage in an oblivious transfer-based protocol to securely choose an appropriate type of dataset to train the discriminator neural

network and collect dataset samples. It will allow the GAN provider to know the chosen data type but it will hide the information on data samples collected by the GAN user. (2) Next, the following sequence of training the discriminator and generator will be followed: (a) The GAN user generates random data as input to its generator model. It sends such random input data to the GAN provider. (b) The GAN provider chooses a training dataset to train the discriminator model. It labels the training data as 1 and it labels the random data from the GAN user as 0. It combines and shuffles these two datasets. The GAN provider uses this combined dataset to train the discriminator neural network. The GAN provider stores evaluation values for the functions in the discriminator neural network along with the corresponding input data. After completing the discriminator neural network training it informs the GAN user. (c) The GAN user may check if the discriminator neural network is properly trained or not. It can use the sample data or the random data it generated previously for such verification. The GAN user and provider will engage in the zero-knowledge proof protocol described in section 3.2. (d) Next, the GAN user will use the same random data generated in step 1 as input to its generator neural network. And, it sends this outcome to the GAN provider as input to the discriminator model. The GAN provider will execute the discriminator model with such input data and send its outcome to the GAN user. (e) The GAN user may again engage in the verification of neural network execution for the previous step. It will use the outcome from the discriminator to train its generator model. In section 3.1 we will describe the protocol for securely collecting data samples from the GAN provider, in section 3.2 we will describe zero-knowledge proof-based verification of deep neural network execution, in section 3.3 we will describe the protocol for verifying if the discriminator model is properly trained or not, and, in section 3.4 we will describe if proper discriminator model is used to train the generator neural network.

3.1 Secure sample data collection

The objective of this protocol is to securely collect a set of sample datasets from the GAN provider by the GAN user such that the GAN provider does not know the identity of the sample data collected from it. Let there be z datasets $d_1, \ldots, d_z$ such that each dataset d_i belongs to a category C_i, i.e., there are z categories of data. We assume that each dataset contains n data instances. The GAN provider assigns a unique index to each category between 1 to z. Each data instance for the dataset d_i gets a unique index between $((i-1)n+1)$ to (in). The GAN provider generates a Merkle tree from these data instances using the Hashes of these data instances by considering data instances with increasing indices, i.e., Hash of data instance with index 1 is the first (leftmost) leaf node of the Merkle tree, Hash of data instance with index 2 is the first (2nd leftmost) leaf node of the Merkle tree and so on. We will design an oblivious transfer-based protocol to choose an appropriate dataset to train the discriminator neural network as shown in figure 1(a).

3.2 Verification of Deep Neural Networks

Overview of KGZ-based zkSNARK The proposed secure GAN model needs to verify a deep neural network, i.e., it needs to verify that the outcome from a deep neural network is indeed the result of executing the deep neural network functions for a given input. We will use the deep neural network verification method developed in [8, 7]. We will use the KGZ polynomial commitment scheme [4] as the zero-knowledge proof protocol. In this verification process, the GAN provider acts as the prover, and the GAN user acts as the verifier. In this section, we will briefly discuss KGZ-based zero knowledge proof protocol. In this protocol, the prover has a polynomial $F(x)$ and it computes the value of $F(x)$ at $x = a$ as y. It wants to convince the verifier that y is indeed the result of computing $F(x)$ for the input y. We will use the following terms: (1) $\mathbb{G}_1, \mathbb{G}_2$ (Elliptic curves with bilinear pairing), (2) $\mathbb{E}()$ (Bilinear pairing function over $\mathbb{G}_1, \mathbb{G}_2$), (3)g_1, g_2 (Generators of $\mathbb{G}_1, \mathbb{G}_2$), (4) p_1, p_2 (Prime numbers for $\mathbb{G}_1, \mathbb{G}_2$), (5) $Add(), Mul()$ Adds elliptic curve points and multiplies elliptic curve points with integers.

Setup for common reference string: During this step, the prover and the verifier computes a set of random numbers. Let G_1 be an elliptic curve for the prime number $\mathbb{P}$ and g_1 be the generator of G_1. Let τ be a random positive integer less than $\mathbb{P}$. A set of l random numbers are generated from τ as follows:

$$p_1 = g_1, p_2 = M(\tau, g_1), , \ldots, p_l = M(\tau^{l-1}(Mod(\mathbb{P})), g_1) \tag{1}$$

where $M()$ is a function to multiply an positive integer with an elliptic curve point resulting a new elliptic curve point. We can use a ceremony to form the random number. The prover and verifier knows the above random points for both bilinear elliptic curves. We represent these random points for $\mathbb{G}_1, \mathbb{G}_2$ as follows:

$$p_l^1 = M(\tau^{l-1}(Mod(\mathbb{P}_1)), g_1), p_l^2 = M(\tau^{l-1}(Mod(\mathbb{P}_2)), g_2) \tag{2}$$

Commitment to a function: Let $F(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^n$ be the polynomial of degree n. We can create a commitment to $F(x)$ as follows:

$$Commit(F(x)) = Add(Mul(a_0, p_1^1), Mul(a_1, p_2^1), \ldots, Mul(a_n, p_n^1)) \tag{3}$$

Proof of Evaluation for a Polynomial: For a function $F(x)$, let $F(x = a)$ is y. The prover creates a proof of evaluation as a point in the elliptic curve in $\mathbb{G}_1$ as follows: Let,

$$\begin{aligned} Q(x) &= \frac{F(x) - y}{x - a} = r_0 + r_1 x + r_2 x^2 + \cdots + r_n x^n \\ Commit(Q(\tau)) &= Add(Mul(r_0, p_1^1), Mul(r_1, p_2^1), \ldots, M(r_n, p_n^1)) \end{aligned} \tag{4}$$

. *Verification of proof of evaluation*: The prover sends the verifier the following values: (1) The commitment to the function $F(x)$ as $Commit(F(\tau))$ in $\mathbb{G}_1$. (2) The set of values of function evaluation at a as y. (3) The set of commitment for proof of function evaluation to the functions as $\{Commit(Q(\tau))\}$ in G_1.

The verifier checks the validity of the commitment to the function evaluations by checking if the following equivalence holds:

$$\mathbb{E}(Commit(Q(\tau)), (\tau - a)) \equiv \mathbb{E}(Commit(F(\tau)) - y, g_2), \tag{5}$$

where $Commit(Q(\tau))$ and $Commit(F(\tau)) - y$ are computed using elliptic curve $\mathbb{G}_1$ and $(\tau - a)$ is computed using the elliptic curve $\mathbb{G}_2$. Note that the above equation checks if the commitment to the proof of function evaluation is correct, i.e., the polynomial division for equation 5 has no reminder. Note that verification requires multiplication of two elliptic curve points and hence we used bilinear pairing-based elliptic curves for this purpose.

Deep Neural Verification with KGZ zkSNARK Commitment to the DNN functions Let $F(x) = b_1^1 + w_{1,1}^1 x + w_{2,1}^1 x^2 + \cdots + w_{n,1}^1 x^n$, i.e., coefficients of $F(x)$ is the set of weights and bias for a node in the DNN. The prover creates a commitment for $F(x)$ as follows:

$$Commit(F_1^1(\tau)) = Add(Mul(b_1^1, \mathbb{P}_1), Mul(w_{1,1}^1, p_2^1), \ldots, Mul(w_{n,1}^1, p_n^1)) \tag{6}$$

For each DNN layer, the prover creates n elliptic curve points for the elliptic curve $\mathbb{G}_1$, and in total, it creates nk such points. The set of commitments for layer i will be denoted as the set $\cup_{j=1}^{n} Commit(F_j^i(\tau))$.

Function Evaluation A DNN function $f(x) = y_1^1 = b_1^1 + \cup_{i=1}^{n} a_i w_{i,1}^1$ is evaluated as $b_1^1 + a_1 x^1 + a_2 x^2 + \cdots + a_n x^n$. Hence we need to find a value for x that can represent the set of numbers $b_1^1, a_1, \ldots, a_n$. The prover and the verifier generate a number x as follows:

$$1 - \frac{1}{(a_1 + a_2 + \cdots + a_n)} = x \tag{7}$$

The above equation holds for large n. We assume that the DNN model is large and hence n is large. Using this value of x, the prover calculates DNN functions for $layer_1$. We denote such a set of evaluations as the set $\{y_i^1\}$. Note that, the set $\{y_i^1\}$ is the input to $layer_2$, and for $layer_2$ we calculate the value of x for DNN functions of $layer_2$ using the same procedure.

Proof of Evaluation Note that, value of evaluation for DNN functions $\{F_i^1(x)\}$ at $layer_1$ is the set $\{y_i^1\}$. For each function F_i^1, the prover creates a proof of evaluation as a point in the elliptic curve in $\mathbb{G}_1$ as follows: Let,

$$\begin{aligned} Q_1^1(x) &= \frac{F_1^1(x) - y_1^1}{x - a} = r_0 + r_1 x + r_2 x^2 + \cdots + r_n x^n \\ Commit(Q_1^1(\tau)) &= Add(Mul(r_0, p_1^1), Mul(r_1, p_2^1), \ldots, M(r_n, p_n^1)) \end{aligned} \tag{8}$$

The set of proof of evaluations for $layer_1$ will be denoted as the set $\{Commit(Q_i^1(\tau))\}$.

Verification of proof of evaluation The verifier has the following values: (1) The set of commitment to the DNN functions at $layer_1$ as $\cup_{j=1}^{n} Commit(F_j^i(x))$. (2) The set of value of DNN function evaluation at $layer_1$ as $\{y_i^1\}$. (3) The set

of commitment for proof of function evaluation to the DNN functions at $layer_1$ as $\{Commit(Q_i^1(\tau))\}$. The verifier checks the validity of the commitment to the function evaluations by checking if the following equivalence holds:

$$\mathbb{E}(Commit(Q_i^1(\tau)), (\tau - x)) \equiv \mathbb{E}(Commit(F_j^i(x)) - y_i^1) \tag{9}$$

Note that the above equation checks if the commitment to the proof of function evaluation is correct, i.e., the polynomial division for equation 5 has no reminder. Note that verification requires multiplication of two elliptic curve points and hence we used bilinear pairing-based elliptic curves for this purpose.

3.3 Training and verification of the discriminator neural network

The objective of this protocol is to ensure that the discriminator is trained with the data category requested by the GAN user. The protocol is shown in Table 2. Briefly, it is as follows: (1) The GAN user will generate a set of random data to be used as input data to its generator neural network. It will send this random data to the GAN provider. (2) The GAN provider will combine the chosen dataset in section 3.1 and the random data from the GAN user by labelling the chosen data as 1 and random data from the GAN user as 0. It will use this combined dataset to train the discriminator model. (3) During training of the discriminator model, the value of functions within the discriminator neural network will be stored and may be used to verify that the discriminator neural network is properly trained. The GAN provider will create a Merkle tree using these values of discriminator function evaluation. The GAN provider will send the roots of the Merkel trees to the GAN user after completing training of the discriminator model. The GAN provider will also send the commitments to the discriminator neural network functions (as shown in equation 6, section 3.2) (4) The GAN user will either choose a sample data as chosen in section 3.1 or the random data it has sent to the GAN provider in step 1. The GAN user and GAN provider will use the neural network execution verification using zero-knowledge proof as shown in section 3.2 with the input as the collected data samples or the random data the GAN user has generated. The verification will be as follows: (a) The GAN user will send the index of the sample dataset for verifying discriminator neural network verification. (b) The GAN provider will reveal the discriminator neural network function evaluation for the index of the sample dataset by revealing data in the Merkle tree whose root it has sent to the GAN user in the previous step. (c) The GAN provider will use the root of the Merkle tree to verify if the revealed value of discriminator neural network function evaluations is correct. (d) Next, the GAN provider and the GAN user will engage in the zero-knowledge proof explained in section 3.2 to verify that given the input data in step 1 and the function evaluation values revealed in the previous step correspond to the actual execution of the discriminator neural network.

3.4 Verification of Correct Discriminator Model Usage in Training the Generator

The objective of this protocol is to ensure that the correct discriminator is used to train the generator neural network. The protocol is as follows: (1) The GAN user generates a set of random data as input to the generator neural network (same data as generated in step 1 in section 3.3). (2) The GAN user uses this data as the input data to the generator neural network and sends its outcome to the GAN provider who uses it as the input to the discriminator neural network. The outcome for the discriminator neural network is sent back to the GAN user, who estimates the loss function using it to revise the parameters of the generator neural network. (3) The GAN user can engage in a zero-knowledge proof-based verification (as described in section 3.2) of the discriminator network as follows: (a) The GAN user and provider will use the random data as input to the functions of the discriminator neural network, the GAN provider will reveal the evaluation of discriminator functions for these given inputs and the outcome of the discriminator neural network. Using protocols shown in section 3.2 the GAN user can verify if these outcomes actually correspond to the execution of the discriminator neural network.

4 Analysis

Theorem 1. *The protocol for secure data sampling procedure is correct and secure.*

Proof. The objective of this protocol is to hide the identity of samples collected from the GAN provider. Note that we used an oblivious transfer protocol for securely collecting such data samples. As shown in Table 1, the GAN user makes random choices over the data instances for a specific data category as shown in Step 3. Such a choice is hidden by creating an elliptic curve point from the index of the chosen data instance. To reveal the choices made by the GAN user, the GAN user must solve the discrete logarithm problem to know such a choice of data samples from the corresponding elliptic curve. The protocol is correct because the following equivalence holds: $Decryption.Key = Mul(b_i, A)$.

$$\begin{aligned} Encryption.Key &= Mul(a, R) - Mul(i, T) \\ &= Mul(a, (Mul(c_i, A) + Mul(b_i, g_1))) - Mul(i, mul(a, A)) \\ &= Decryption.Key \end{aligned} \tag{10}$$

The above equivalence holds if $c_i = i$, this means the GAN user can only decrypt the data samples which it has chosen.

Theorem 2. *The protocol for training the discriminator is secure.*

Proof. The objective of this protocol is to ensure that the discriminator neural network is trained with the datasets by the GAN user. Note that the training

procedure includes the following: (1) The GAN provider proves that the data instance to be used to train the discriminator belongs to the dataset chosen by the GAN user. As mentioned in section 3.1, the GAN provider uses the procedure to check the membership of a leaf node in a Merkle tree to prove that the data instance belongs to the chosen dataset. Note that, before collecting sample data instances and revealing the chosen data category, the GAN user receives the Root of the Merkle tree (section 3.1). Hence the GAN provider cannot change data instances or use data from a different data category. (2) We claim that the chosen dataset is used to train the discriminator neural network according to the protocol mentioned in Table 2 because: (a) as shown in Line 8, Table 2, the GAN provider trains the discriminator neural network and sends the commitment to the neural network function of the discriminator to the GAN user along with the valuation of the neural network functions. (b) The GAN user can verify the valuation is indeed the result of the discriminator neural network model execution if the input is part of sample datasets collected according to the protocol shown in Table 1. The neural network verification protocol can be used to securely execute such verification. (c) It is possible that the GAN provider may not use correct data in Step 8. However, the above verification can prevent such manipulation by the GAN provider. As the GAN provider does not know the identity of the data samples collected according to the protocol in Table 1, it does not know if it will go through the verification process of step 9 (Table 2), it will be reluctant to use the wrong data to train the discriminator neural network. (3) Further, the GAN provider sends commitments to the trained discriminator neural network to the GAN user after every training cycle (Steps 8 and 10). The GAN user can check if such commitments mismatch with each other as proof that neural network parameters are changed after every training iteration.

Theorem 3. *The protocol for training the generator is secure.*

Proof. The protocol for training the generator is secure because: Note that, inputs to the discriminator model are created by the GAN user, hence it cannot be manipulated by the GAN provider. We use the neural network verification protocol shown in section 3.2. Such a zero-knowledge proof protocol is secure and cannot be manipulated by the GAN provider. The committed discriminator model (as discussed in step 3, section 3.3) will be used in the above verification; hence the GAN provider cannot use a different discriminator model. The GAN user computes the loss function from the outcome of the discriminator model and uses it to train the generator model.

5 Experimental Evaluation

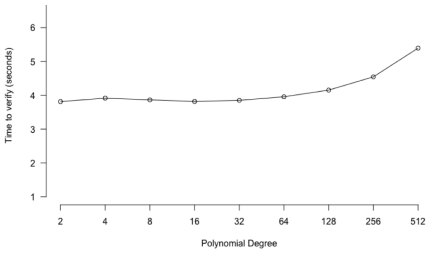
We evaluate the secure GAN platform in terms of the time it takes to verify the neural network of the discriminator. We implement the KGZ-polynomial commitment scheme-based verification in Python. We use an Apple Macbook Pro with M1 Pro processor and 16 GB memory. We measure the time it takes to verify the discriminator neural network by increasing the degree of polynomials

representing DNN functions. We increase the polynomial degree from 2 to 512. We observe the time it takes to verify polynomials is increased from 3.8 seconds to 5.3 seconds. In the paper [19], we have shown that we need three verifications per neural network layer. We claim that the increment in the time to verify is very small as we increased the polynomial degree from 2 to 512. Degree of such a polynomial indicates size of a neural network, i.e., number of weights and a bias as formulated in page 7. Hence the proposed verification method can be used in practical applications. In the figure below, we plot the time it takes to verify the DNN functions.

	Prover (GAN Provider)	Verifier (GAN User)
1	Send indices of data categories to the verifier. Create the Merkle tree of the dataset and send the root of the Merkle tree to the verifier.	
2	Create a random positive integer $a \in Random(1, p_1)$ where p_1 is the order of the elliptic curve G_1. Generate the elliptic curve points $A = Mul(a, g_1)$ and $T = Mul(a, A)$ where g_1 is the generator of the elliptic curve G_1. Send A to the verifier.	
3		Choose a data category $c \in (1, 2, \ldots, z)$. Choose random data instances for the category c by choosing random numbers between $((c-1)n+1)$ and (cn). Let such numbers are $(c_1, c_2, \ldots, c_k)$
4		Choose a set of k random positive integer b_i less than the order of the elliptic curve G_1 and generate the elliptic curve points $R_i = Add(Mul(c_i, A), Mul(b_i, g_1))$. Send the set of elliptic curve points R_i to the prover. Send c to the prover.
5	For each R_i , generate n keys $K_1, \ldots, K_n$ such that $K_i = Sub(Mul(a, R), Mul(i, T))$ where i is the ith element in the set $\{((c-1) \times n+1), \ldots, (cn)\}$, where Sub function subtracts two elliptic curve points for the elliptic curve G_1.	
6	Encrypt ith data instance for the category c with the key K_i with a symmetric key encryption protocol.	
7	Send the encrypted data instances to the verifier.	
8		The verifier decrypts the data instances with the keys $\{Mul(b_i, A)\}$.

Table 1. Protocol for secure data sampling.

(a)



(b)

Fig. 1. Protocol and Results

6 Related Literature

GAN was introduced in 2014 in the paper [2]. There have been many advancements in GAN research in the last few years. In [3] the authors have investigated optimization problems in GAN. In [6] the authors developed GAN for convolutionary neural networks. In this paper, we used zero-knowledge proof for DNN verification. In [1] the authors have developed a zero-knowledge proof protocol that allows the prover to prove that it still knows a number. We use Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) which reduces the size of proof and complexity of proof verification considerably. We use the KGZ polynomial commitment scheme developed in [4, 8]. Other constructions of zkSNARKs developed in [5] use quadratic arithmetic program-based construction of zkSNARK. In this paper, we advance the state of the

art in GAN as we propose a secure GAN model. We have investigated GAN manipulation at various stages including appropriate training data selection for discriminators, and correct training of the discriminator neural networks.

7 Conclusion

In this paper, we proposed a secure GAN model. We consider the scenario where a malicious GAN provider does not choose and use appropriate data to train the discriminator model that is hidden from the GAN user. Also, a malicious GAN provider may not correctly use the discriminator model to train the generator model. We solved these security problems with GAN with oblivious transfer protocol and zero-knowledge proofs. We proved that the proposed verification protocol is efficient. Further, we will extend the results of this paper in developing a secure GAN as a service platform.

Acknowledgment

This publication has emanated from research supported by grants from Science Foundation Ireland (SFI) under Grant Numbers 21/FFP-A/9174 (Sustain) and 12/RC/2289_P2 (Insight).

References

1. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
2. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014), https://arxiv.org/abs/1406.2661
3. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation (2018), https://arxiv.org/abs/1710.10196
4. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
5. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252 (2013). https://doi.org/10.1109/SP.2013.47
6. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016), https://arxiv.org/abs/1511.06434
7. Thakur, S., Breslin, J.: Secure coalition formation for federated machine learning. In: Fred, A., Hadjali, A., Gusikhin, O., Sansone, C. (eds.) Deep Learning Theory and Applications. pp. 238–258. Springer Nature Switzerland, Cham (2024)
8. Thakur, S., Breslin, J.: Verification of deep neural networks with kgz-based zksnark. In: Arai, K. (ed.) Intelligent Systems and Applications. pp. 79–95. Springer Nature Switzerland, Cham (2024)