electronics

MDPI

Article

WGAN-E: A Generative Adversarial Networks for Facial Feature Security

Chunxue Wu [1], Bobo Ju [1], Yan Wu [2], Neal N. Xiong [3] and Sheng Zhang [1,*]

1 School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China; wcx@usst.edu.cn (C.W.); xh11407130@Outlook.com (B.J.)
2 O'Neill School of Public and Environmental Affairs, Indiana University Bloomington, Bloomington, IN 47405, USA; yanwu8910@gmail.com
3 Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74464, USA; xiongnaixue@gmail.com
* Correspondence: zhangsheng_usst@aliyun.com; Tel: +86-021-55271311

Received: 17 February 2020; Accepted: 10 March 2020; Published: 15 March 2020

Abstract: Artificial intelligence technology plays an increasingly important role in human life. For example, distinguishing different people is an essential capability of many intelligent systems. To achieve this, one possible technical means is to perceive and recognize people by optical imaging of faces, so-called face recognition technology. After decades of research and development, especially the emergence of deep learning technology in recent years, face recognition has made great progress with more and more applications in the fields of security, finance, education, social security, etc. The field of computer vision has become one of the most successful branch areas. With the wide application of biometrics technology, bio-encryption technology came into being. Aiming at the problems of classical hash algorithm and face hashing algorithm based on Multiscale Block Local Binary Pattern (MB-LBP) feature improvement, this paper proposes a method based on Generative Adversarial Networks (GAN) to encrypt face features. This work uses Wasserstein Generative Adversarial Networks Encryption (WGAN-E) to encrypt facial features. Because the encryption process is an irreversible one-way process, it protects facial features well. Compared with the traditional face hashing algorithm, the experimental results show that the face feature encryption algorithm has better confidentiality.

Keywords: facial feature; generative adversarial networks; Wasserstein GAN; face recognition; neurocryptography

1. Introduction

The general methods of facial feature extraction are: Gabor features, Haar-like features (HAAR), Histogram of Oriented Gradient (HOG), and Local Binary Pattern (LBP) [1]. The advantage of the traditional face recognition method is that it runs more quickly under the CPU. The disadvantage is that the recognition rate is relatively low, because features need to be specified manually and are not "autonomous" as in deep learning. Recently, the face recognition process based on deep learning mainly uses convolutional neural networks [2,3]. The disadvantage is that it runs very slowly under the CPU [4,5].

In recent years, biometric technology has made great progress and has gradually penetrated all aspects of human life. However, with the continuous deepening of the application of biometric technology, its inherent security and privacy protection issues have gradually been exposed. The biological characteristics of the human body are fixed. Once they are lost and used by criminals for

illegal purposes, it can have unimaginable consequences. Therefore, Biometric Encryption (BE) came into being [6].

Cryptography pays extensive attention to the confidentiality and integrity of information. Cryptographic mechanisms are often described as programs or Turing machines. In these terms, an attacker is also described as the boundary between its complexity (e.g., limited to polynomial time) and chance of success (e.g., limited to probability). A cryptographic mechanism is considered secure if it can reach polynomial time and cannot be deciphered. For example, if no one can extract information from encrypted facial features, the encryption algorithm is secure. This definition is more standardized by modern cryptography [7].

Biometric cryptography is a process that binds keys and biometrics together. Therefore, after the biometrics are submitted to the system, both the keys and biometrics should be guaranteed to not be decipherable from the template stored in the system [6]. This concept was first proposed by Tomko et al. in 1994 and was patented in combination with fingerprint feature encryption. In 1998, Nichols et al. first proposed the concept and implementation method of key and biometric binding. In China, Wang et al. introduced the concept of encryption to biometrics in 1999 and conducted groundbreaking research. In 2001, Ratha et al. introduced the concept of revocable biometric authentication. Adjustable single-parameter conversion capabilities are used to exchange raw biometric data. If the conversion template is stolen, only the parameters are changed and a new template can be generated.

With the development of computers, electronics, and modern communication technologies, the issue of security and confidentiality of information transmission and storage has become an important research field. A powerful measure to ensure information security is to encrypt information using cryptographic algorithms. There are multiple data encryption algorithms. In cryptography, serial cipher is a very important encryption method. The serial cipher is based on the idea of "one time and one secret". The linear shift register and circuit combination are used to generate a pseudo-random sequence to encrypt the information. Therefore, the serial password has the characteristics of fast encryption. At present, the proposed sequence cipher algorithm basically uses the XOR of the key sequence and the plaintext to obtain the ciphertext. It is difficult for cryptanalysts to decrypt using the plaintext redundancy. The sequence ciphers also have the characteristics of disturbing the statistical characteristics of the plaintext and the absence of data expansion and error transmission, and they are suitable for secure communication. Therefore, serial ciphers maintain a unique advantage in practical applications, especially in military, diplomatic, and important confidential transmissions, and are considered the mainstream of current international cryptosystem applications. Since the neural network can perform nonlinear function mapping on the input space and the output space according to different training conditions, the neural network technology can be used to generate a key sequence for data sequence encryption. Neural network-based sequence encryption can achieve face feature encryption [8].

The rest of the paper is arranged as follows. Section 2 introduces the existing contributions of the predecessors in neural cryptography and the advantages and disadvantages of these methods, and derives a novel method to encrypt re-encoded facial features using generative adversarial neural networks. In Section 3, we detail our experimental model. Section 4 gives the relevant experimental process in detail and analyzes the reliability of the experimental results. Finally, in Section 5, we explain the conclusions and future work.

2. Related Work

Manual authentication is a security task that restricts access to a physical location or computer network to only authorized personnel. This can be done by equipping authorized users with a password, token, or using their biometric technology. Unfortunately, the first two are poorly secured because they are easily forgotten and stolen; even biometrics suffer from some inherent limitations and specific security threats. A more practical approach is to use a combination of two or more factor authenticators for the benefits of security or convenience or both. Compared with only biometric

technology, tokenized pseudorandom numbers and user-specific biometrics (BioHashing) have obvious functional advantages, namely zero equal error rate. The main disadvantage of the basic BioHashing method is that when "impersonator B" steals A's pseudo-random number and attempts to authenticate to A, it shows lower performance. Lumini et al. introduced some improvements [9]. This research uses the basic BioHashing method to maintain a very low equal error rate when no one steals the hash key, and can achieve good performance when the "impersonator" steals the hash key.

Kuan et al. introduced a novel method that can use BioPhasor hybrid and 2N discretization technology to securely calculate the biometric hash on a dynamic hand signature [10]. Using BioPhasor as a hybrid process provides a one-way transformation, making it impossible to accurately recover biometric vectors from damaged hashes and stolen tokens. The results show that, for stolen tokens (worst-case scenario) with random and skilled forgery methods, the proposed method can produce stable and distinguishable bit strings, and the equal error rate (EER) is 0% and 9.4%, respectively. The counterfeit has an equal error rate of 0% in the token (best) scheme.

A consistent encryption key from noisy data, such as a biometric template can be derived with the help of some additional information called a sketch. The main difficulty is that many biometric templates are represented as points in a continuous domain with an unknown distribution, and known results can only work in the discrete domain, or lack a rigorous analysis of entropy loss. Qiming Li et al. suggested not to study methods to directly solve these problems, but instead to study the relative entropy loss of any given scheme, which would limit the additional number of bits that can be extracted if the best parameters are used. They gave a general scheme and showed that the relative entropy loss due to suboptimal discretization is at most nlog3, where n is the number of points and the boundaries are tight [11].

With regard to deep learning and neural networks for image, text, and speech encryption, a great deal of research has been done by predecessors [12–14]. Their research confirms the effectiveness and advancement of neural networks for encryption algorithms from various angles.

As a proof of concept, Chen He et al. proposed a method based on deep learning to attack chaotic-based image encryption algorithms [15]. The method firstly projects the chaotic encrypted image into the low-dimensional feature space, and retains a large amount of basic information of the original image in the low-dimensional feature space. Using a low-dimensional feature, a deconvolution generator is used to generate a perceptually similar decrypted image, thereby approximating the planar image in a high-dimensional space. Compared with the traditional image encryption attack algorithm, this method does not require time-consuming manual analysis and inference of the key.

The Stacked Auto-Encoder (SAE) is a deep learning algorithm for unsupervised learning. It has a multi-layered structure that projects a vector representation of the input data into a lower vector space. These projection vectors are a dense representation of the input data. Therefore, SAE can be used for image compression. Using chaotic logic mapping, the compressed logical map can be further encrypted. Fei Hu et al. recommend using SAE and chaotic logic mapping for image compression and encryption [16].

Deep learning as a service (DLaaS) has become a promising method for promoting deep neural networks (DNNs) for various purposes. However, using DLaaS can also lead to potential privacy leaks from clients and cloud servers. This privacy issue has led to research interest in the privacy protection reasoning of DNN models in cloud services. Peichen Xie et al. proposed a practical solution called BAYHENN's secure DNN reasoning [17]. It protects both client privacy and server-side privacy. The key strategy of this research solution is to combine homomorphic encryption and Bayesian neural networks. Specifically, the study uses homomorphic encryption to protect raw client data and Bayesian neural networks to protect DNN weights in cloud servers.

Théo Ryffel et al. detailed a new framework for protecting deep learning privacy and discussed its advantages [18]. The framework values data ownership and security processing and introduces valuable representations based on command chains and tensors. This abstraction allows for complex privacy protection structures such as federated learning, secure multiparty computing, and differential privacy.

When deep learning is applied to sensitive datasets, there are many privacy-related implementation issues. These issues are particularly evident in the medical, financial, legal, and government industries. Homomorphic encryption allows the server to infer the input of the client's encryption [19]. For any model depth, there is no complete implementation of the generic deep learning operation using homomorphic encryption. Related scholars have proposed a new method to effectively implement multiple deep learning functions by using bootstrap homomorphic encryption. The findings provide a promising direction for user-regression of privacy-protected representation learning and data control [20].

With the rapid development of current Internet networks and online applications, network traffic classification becomes more and more important. Much of the research on this topic has led to many different approaches. Most of these methods use predefined features extracted by experts to classify network traffic. In contrast, Mohammad Lotfollahi et al. proposed a deep learning-based approach that integrates feature extraction and classification into a single system [21]. The proposed solution, called Deep Packet, can handle two traffic characteristics, one is to classify network traffic into major classes (such as FTP (File Transfer Protocol) and P2P (peer to peer lending)), and the other is to require end-user application identification (such as BitTorrent).

As neural networks are applied to increasingly complex tasks, they are often trained to meet the ultimate goal of transcending simple functional specifications and address multi-agent problems. Through these efforts, Google Brain discovered that neural networks can learn to protect communication security [22]. However, neural networks are not good at cryptography. It is worth noting that, under neural network encryption, a single neuron cannot learn the XOR, but a multilayer network can do it. However, neural networks can learn how to protect the confidentiality of data from other neural networks: they discover how to encrypt and decrypt forms without having to teach a specific algorithm for these goals.

Martin Arjovsky et al. introduced an algorithm of Wasserstein GAN (WGAN), which is an alternative to traditional GAN training [23,24]. In this new model, they demonstrated that WGAN can improve learning stability, get rid of pattern collapse and other issues, and provide meaningful learning curves for debugging and hyperparametric search. In addition, the study demonstrates that the corresponding optimization problem is reasonable and provides a wide range of theoretical work, highlighting the deeper connections to other distances between distributions.

WGAN has made progress in stabilizing GAN, but sometimes it can only generate bad samples or fail to converge. Ishaan Gulrajani et al. found that these problems were usually due to the use of weight-cutting in WGAN to impose Lipschitz constraints on critics, which could lead to bad behavior. Thus, Gulrajani et al. proposed an alternative method of weight reduction: penalizing the critics' gradient specification relative to their input [25]. The proposed method performs better than the standard WGAN and can stably train multiple GAN architectures with almost no hyperparameter adjustment, including 101-layer ResNet (Residual Neural Network) and a language model with continuous generators.

This study introduces face feature extraction. The re-encoded facial features are then encrypted using a model trained against the neural network WGAN-E.

3. Face Feature Description and Neural Network Encryption Model

3.1. Face Feature Description

The LBP primitive operator introduced by Ojala et al. is a powerful texture description method. The operator marks the pixels of the image as a 3×3-neighbor of each pixel and considers the result as a binary number. LBP can also be used as a biometric description. The principle of the basic LBP operator is shown in Figure 1. Later, the method was extended to allow the use of circular areas of different sizes and bilinear interpolation of pixel values, allowing for any radius and number of

pixels nearby. For neighboring regions, the study uses the symbol LBP(P, R), which means that the P-sampling point on the circle of the radius of (see Figure 2) is an example of a circle (8, 2) [1].

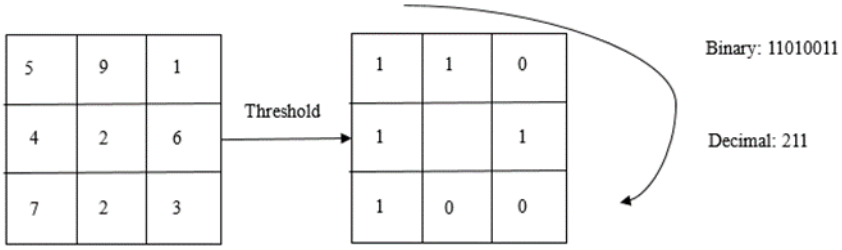

Figure 1. LBP principle. The center is 2; the numbers that are larger than 2 become 1, while the numbers that are smaller than 2 become 0.

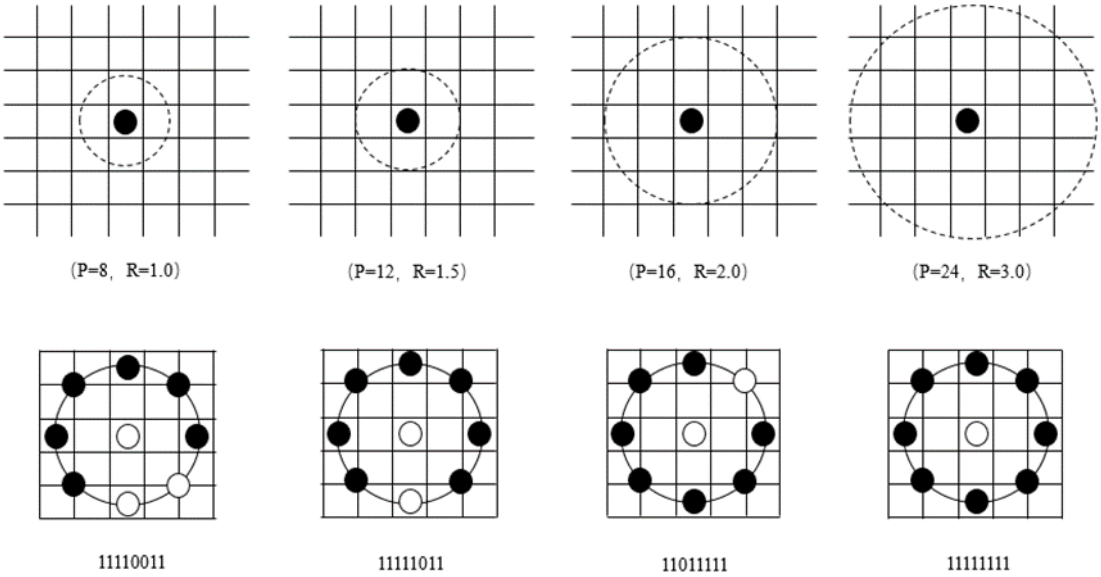

Figure 2. Improved LBP(P, R). The above section draws an example diagram of the selection in (P,R). The bottom half of figure is an example of (P,R) = (8,2). It is easier to understand Figure 2 when combined with Figure 1.

This study uses the following notation for the LBP operator: $\mathrm{LBP}^{u^2}_{P,R}$. The subscript indicates the use of operators near (P, R). Superscript u^2 means that only the unified mode ((P,R) = (8,2) is used uniformly to calculate LBP and all remaining modes are marked as one tag.

The histogram of the marked image $f_I(x, y)$ can be defined as

$$\mathrm{H}_i = \sum_{x,y} I\{f_I(x, y) = i\}, i = 0, \cdots, n-1 \tag{1}$$

where n is the number of different tags generated by the LBP operator

$$I\{\mathrm{A}\} = \begin{cases} 1 & \mathrm{A\ is\ true} \\ 0 & \mathrm{A\ is\ false} \end{cases} \tag{2}$$

This is a distribution of edges, spots, and planar regions over the entire image. For valid facial representations, it should also keep spatial information. For this purpose, the image is divided into regions $R_0, R_1, \cdots, R_{m-1}$ and the spatially enhanced histogram is defined as

$$H_{i,j} = \sum_{x,y} I\{f(x,y) = i\} I\Big\{(x,y) \in R_j\Big\}, \quad i = 0, \cdots, n-1, \; j = 0, \cdots, m-1 \tag{3}$$

In this histogram, the effective description of the face is: the histogram of the label contains pixel-level information, and the information generated by the label and region level is added together. This information establishes a globally described face description of the regional histogram.

3.2. Principles of Encryption and Decryption

A complete cryptosystem includes plaintext space, key space, and algorithms. Using the associative memory neural network for encryption, this study only uses the iteration of the network, regardless of whether it can be added to the stable point of the network. This study only needs to extract a given face feature input network and train it with a specified number of iteration steps. It not only can be encrypted, but is also simple and easy. The disadvantage is that there is a certain key redundancy along with the improvement of security.

Since there are many noise-added samples and redundant attractors in the associative memory, if these data were separated from a specific neural network and number of iterations, the data would be meaningless. Therefore, this study proposes a method for neural network encryption using noisy samples and redundant attractors.

Encryption principle: There are two types of cryptosystems, single key and dual key. This article uses the traditional single key system, that is, the confidentiality of the system depends on the security of the key. Of course, if the encryption is not publicly disclosed, then the security level of the algorithm and key system is high. In the traditional encryption system, the operations of the encryption algorithm and the decryption algorithm are performed under the control of a set of keys, which are called encryption key and decryption key, respectively.

Encryption process: Assume that there is a plaintext of length 256. The packets are encrypted and *XOR*ed according to the secret key generated by the above process, or other higher encryption algorithms are used.

Decryption process: The receiver intercepts the vector after the plaintext according to the protocol and compares the network parameters transmitted through the relatively secure channel. The 24th bit of the vector is intercepted, and then 4 bits of information are intercepted according to the protocol (each vector has three vectors). The algorithm needs to calculate the weights of the network and iterate the positive samples in groups of four for five iterations to obtain the results. The numbers after the 24th one (or $4 \times 33 + 24$) are intercepted. A set of inputs is decomposed into the five five-dimensional samples starting from the 55th bit, and the result of iterating three times is the key. The key and the ciphertext are *XOR*ed to obtain the plaintext.

3.3. WGAN-E Encrypts Facial Feature Data

This section discusses how to use shared keys to protect the confidentiality of facial features. It describes the organization of the system in this study and the role of various parts of the system. It also illustrates the architecture of the neural network model used in this study.

The classic security scenario involves three aspects: Users, Servers, and B competitors. Generally, A's facial features need to be stored securely on the server, while B wants to steal A's facial features. Therefore, the required security attribute is confidentiality rather than integrity. Define an adversary as a "passive attacker" who can intercept communications. However, beyond that, the adversary's permissions are very limited: it cannot start a session, inject messages, or modify messages during transmission [25].

This study begins with a particularly simple example of this scenario (Figure 3), in which *A* wants to send a separate secret face feature *FF* (representing the Facial Feature, "Face Feature") to the Server. The face feature Facial Feature is the input to *A*. When A processes this input, it produces an output *EFF* (representing Encrypt Facial Feature, "Encrypted Face Features"). Both Server and B receive the *EFF*, process it, and attempt to recover the face feature FF. This study uses P_B and P_S to represent their calculation results. *A* and *Server* have an advantage over *B*: they share a *Key*. This study uses K as an additional input to A and Server. Assume that each face feature *FF* has a new *Key*, but, at least on this abstract level, this study does not mandate that the *Key* and FF are the same length.

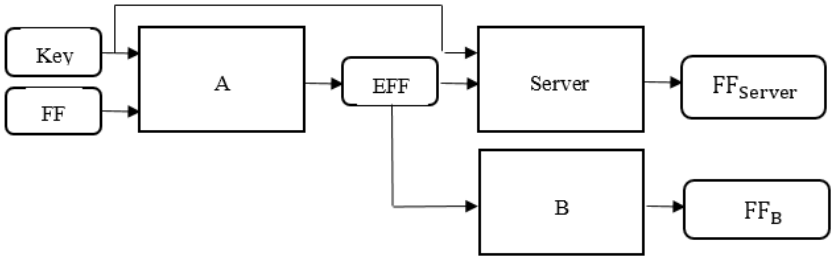

Figure 3. A, Server, and B password system. The A network acts as a Generator here, and the B and Server networks act as Discriminators. Facial features use the key and A network to obtain the encrypted face features, and B and Server are used to restore the encrypted face features to face features.

For the face feature encryption process, *A*, *Server*, and *B* are neural networks. Their structure is described in Section 3.4. They all have parameters, which are recorded as ω_A, ω_{Server}, and ω_B, respectively. Because ω_A and ω_{Server} do not need to be equal, even if *A* and *Server* have the same structure, encryption and decryption are not necessarily the same function. For neural networks, *A*, *Server*, and *B* are all in a tuple of values, not a sequence of bytes. In other words, *Key*, *FF*, FF_{Server}, FF_B, and *EFF* are all tuples of values. Note that with this formula, *EFF*, FF_B and FF_{Server} can be composed of arbitrary values, even if *FF* and *Key* consist of 0 and 1. In practice, the implementation of this study limits these values to the range, but allows intermediate values.

Although this setting is basic, it is sufficient for the basic plan. Especially considering that *A* and *Server* use *Key* as a one-time key between facial features *FF* and encrypted facial features *EFF*. However, this study does not require A and Server to operate in this way. In fact, in the three-part experiment, other options were found. For the sake of simplicity, this study not only ignores the process of generating keys, but also ignores the randomness of probability encryption (Goldwasser and Micali, 1984) [25]. These enhancements may be the subject of further work.

The goal of *B* is simple: accurately reconstruct *FF* (in other words, minimize the error between *FF* and FF_{Server}). A and Server want to communicate clearly (minimize the error between *FF* and FF_{Server}), as well as hide their communication with B. Note that, according to the modern password definition (Goldwasser and Micali, 1984), this study does not require the encrypted face feature *FF* to "seep random" to *B*. Encrypted face features may even contain explicit metadata to identify it. Therefore, for B, distinguishing between *FF* and random values extracted from a distribution is not a goal. In this regard, B's goal is in stark contrast to the opponents of GANs [26]. On the other hand, this study can try to redefine the goal of B to distinguish the encrypted face features composed of two different files.

With these goals in mind, this study does not separately train A and Server to implement some known cryptosystems (Dourlens, 1996), but jointly trains A and Server to allow them to communicate successfully without pre-designated concepts. In the case of defeat for , as with the definition of GANs, this study hopes that A and Server defeat the most likely *B*, not a fixed *B*. Of course, *A* and *Server* may not win every plaintext and every keyword, because knowledge of certain face features and keys may be fixed in *B*. (For example, *B* can output the same facial features, at least once is correct.) Therefore, this study assumes a distribution on the face features and keys, and describes the goals of *A* and *Server* based on the expected values [27].

This study uses $\mathrm{A}(\omega_{\mathrm{A}}, \mathrm{FF}, \mathrm{Key})$ to represent the output of *A* on input *FF* and *Key*, $\mathrm{S}(\omega_{\mathrm{Server}}, \mathrm{EFF}, \mathrm{Key})$ to represent the output of Server on input *FF* and *Key*, and $\mathrm{B}(\omega_{\mathrm{B}}, \mathrm{EFF})$ to represent the output of *B* on input *C*. At the same time, the distance function d is introduced in the face feature, although the exact choice of this function may not be important. For the specific operation, this study takes the L2 distance $\mathrm{d}_2(\mathrm{FF}, \mathrm{FF}') = \sqrt{\sum_{i=1,\mathrm{N}} \left(\mathrm{FF}_1^n - \mathrm{FF}_2^{\mathrm{n}}\right)^2}$ (Euclidean distance), where *N* is the length of the face feature. This study defines the loss function for each instance for *B*:

$$\mathrm{L}_{\mathrm{B}}(\omega_{\mathrm{A}}, \omega_{\mathrm{B}}, \mathrm{FF}, \mathrm{Key}) = \mathrm{d}(\mathrm{FF}, \mathrm{B}(\omega_{\mathrm{B}}, \mathrm{A}(\omega_{\mathrm{A}}, \mathrm{FF}, \mathrm{Key}))) \tag{4}$$

Intuitively, $\mathrm{L}_{\mathrm{B}}(\omega_{\mathrm{A}}, \omega_{\mathrm{B}}, \mathrm{FF}, \mathrm{Key})$ represents how much B is wrong when the face feature is *FF* and the key is *Key*. This study also defines the distribution of a loss function for the face feature and key through the expected value for B:

$$\mathrm{L}_{\mathrm{B}}(\omega_{\mathrm{A}}, \omega_{\mathrm{B}}) = \mathrm{E}_{\mathrm{FF,Key}}(\mathrm{d}(\mathrm{FF}, \mathrm{B}(\omega_{\mathrm{B}}, \mathrm{A}(\omega_{\mathrm{A}}, \mathrm{FF}, \mathrm{Key})))) \tag{5}$$

This study obtains the "best B" by minimizing this loss:

$$\mathrm{O}_{\mathrm{B}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}}) = \mathrm{argmin}_{\omega_{\mathrm{B}}}(\mathrm{L}_{\mathrm{B}}(\omega_{\mathrm{A}}, \omega_{\mathrm{B}})) \tag{6}$$

Similarly, this study defines a sample reconstruction error for Server and extends it to a distribution of face features and keys:

$$\mathrm{L}_{\mathrm{server}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}}, \mathrm{FF}, \mathrm{Key}) = \mathrm{d}(\mathrm{FF}, \mathrm{Server}(\omega_{\mathrm{Server}}, \mathrm{A}(\omega_{\mathrm{A}}, \mathrm{FF}, \mathrm{Key}), \mathrm{Key})) \tag{7}$$

$$\mathrm{L}_{\mathrm{Server}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}}) = \mathrm{E}_{\mathrm{FF,Key}}(\mathrm{d}(\mathrm{FF}, \mathrm{B}(\omega_{\mathrm{Server}}, \mathrm{A}(\omega_{\mathrm{A}}, \mathrm{FF}, \mathrm{Key})))) \tag{8}$$

This study defines the loss function of A and Server by combining the optimal values of $\mathrm{L}_{\mathrm{Server}}$ and L_{B}:

$$\mathrm{L}_{\mathrm{AServer}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}}) = \mathrm{L}_{\mathrm{Server}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}}) - \mathrm{L}_{\mathrm{B}}(\omega_{\mathrm{A}}, \mathrm{O}_{\mathrm{B}}(\omega_{\mathrm{A}})) \tag{9}$$

This combination reflects *A* and Server's desire to minimize *Server* rebuild errors and maximize "optimal B" rebuild errors. The use of simple subtraction is somewhat arbitrary, the following study will describe useful variants.

By minimizing $\mathrm{L}_{\mathrm{AServer}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}})$, this study obtains "Best A and Server":

$$(\mathrm{O}_{\mathrm{A}}, \mathrm{O}_{\mathrm{Server}}) = \mathrm{argmin}_{(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}})}(\mathrm{L}_{\mathrm{AServer}}(\omega_{\mathrm{A}}, \omega_{\mathrm{Server}})) \tag{10}$$

We write "optimal" in quotes because there is no need for a global minimum. In general, there are many optimal solutions for *A* and Server. As a simple example, assume that the key has the same size with the face feature and the encrypted face feature, *A* and *ServerXOR* may obtain face features and encrypted face features, respectively, the key to the arrangement, and all the permutations, is: the same is good, as long as *A* and *Server* are the same. By way of building the network of this study (see Section 3.4), all permutations may occur.

Training begins with random initialization of the A and Server networks. The purpose of training is to go from this state to $(\mathrm{O}_{\mathrm{A}}, \mathrm{O}_{\mathrm{B}})$, or close to $(\mathrm{O}_{\mathrm{A}}, \mathrm{O}_{\mathrm{B}})$. Next, we explain the training process [25,26,28]. Algorithm 1 describes the implementation principle of WGAN-E.

Algorithm 1 WGAN-E, a neural cryptography algorithm. All experiments in the study used default values $\alpha = 0.001$, $c = 0.04$, $m = 2000$, $n_{critic} = 25$, $\lambda = 10$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: c, the clipping parameter. m, the batch size. $\beta_1\,\beta_2$, hyperparameters parameters. n_{critic}, the number of iterations of the critic per generator iteration. The gradient penalty coefficient λ, Adam hyperparameters α (the learning rate). w_0, initial critic parameters. θ_0, initial generator's parameters. x, face image data vectors. z, face LBP data vector.

while θ has not converged **do**
  for $t = 0, \ldots, n_{critic}$ **do**
    for $i = 1, \ldots, m$ **do**
      Sample $\left\{x^{(i)}\right\}_{i=1}^{m} \sim \mathrm{P}_r$ a batch from the real data.
      Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
      a random number $\varepsilon \sim U[0,1]$.
      $\overline{x} \leftarrow G_\theta(z)$
      $\hat{x} \leftarrow \varepsilon x + (1-\varepsilon)\overline{x}$
      $L^{(i)} \leftarrow D_w(\overline{x}) - D_w(x) + \lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1)^2$
      $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w\left(x^{(i)}\right) - \frac{1}{m}\sum_{i=1}^{m} f_w\left(g_\theta\left(z^{(i)}\right)\right)\right]$
      $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$
      $w \leftarrow \mathrm{clip}(w, -c, c)$
    end for
    $w \leftarrow \mathrm{Adam}\left(\nabla_w \frac{1}{m}\sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$
  end for
  Sample $\left\{z^{(i)}\right\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w\left(g_\theta\left(z^{(i)}\right)\right)$
  $\theta' \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$
  $\theta \leftarrow \mathrm{Adam}\left(\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} -D_w(G_\theta(z)), \theta', \alpha, \beta_1, \beta_2\right)$
end while

3.4. WGAN-E Neural Network Structure

A, Server, and B Architecture: This study aims to create a neural network architecture that fully learns *XOR* and other hybrid functions, but does not strongly encode any particular form of the algorithm [29,30].

To this end, the study selected the following "mixing and conversion" architecture. It has a first fully connected (FC) layer where the number of outputs is equal to the number of inputs. Face features and keys are entered into this FC layer. Because each output bit can be a linear combination of all input bits, this layer is enabled—but not enforced—to mix between key and face features. In particular, this layer can swap bits. Before the FC layer is a series of convolutional layers, the last layer producing an output suitable for the size of the face feature or the encrypted face feature. These convolutional layers for learning apply some functions to the bit groups that are mixed by the previous layer without specifying what the function should be. It is worth noting that, in image processing applications, the reverse order (convolution and FC) is more common. Neural networks developed for these applications often use convolution to take advantage of spatial locations. For neurocryptography, this study specifically wants localization. Although it will definitely manually pair each input face feature manually using the corresponding key bits, this study suggests that doing so would be boring [22,31]. WGAN-E Neural network structure is shown in Figure 4.

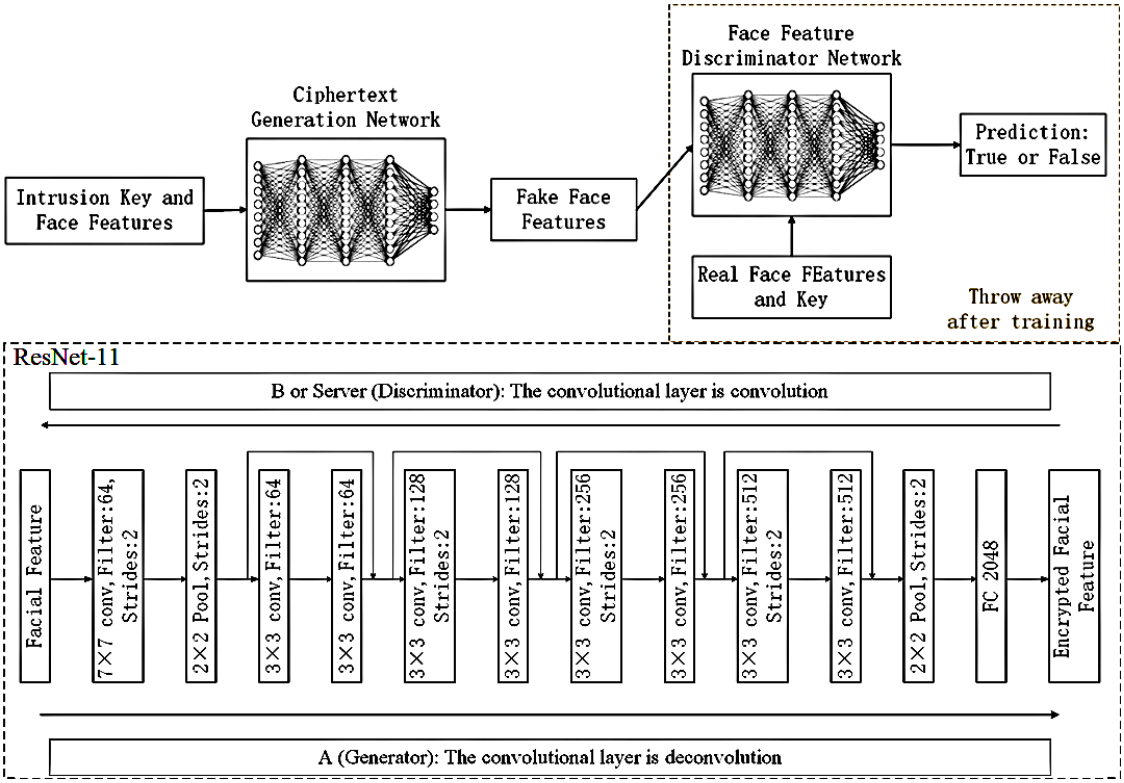

Figure 4. WGAN-E neural network structure. We used the ResNet-11 network to implement the generation network and discriminator network. The face feature encryption process is trained by the deconvolution operation of ResNet-11. With the convolution operation of ResNet-11, the encrypted face features can be restored.

This study avoids imposing more restrictions to simplify the problem. For example, we do not associate *A* with the *Server* parameter, as we thought both *A* and *Server* should learn the same function, such as *XOR*.

4. Experiment and Analysis of Results

The training and test dataset used is the LFW (Labeled Faces in the Wild) dataset. This is a commonly used test set for face recognition. The face pictures provided in it are all derived from natural scenes in life, so the recognition difficulty is increased, especially due to factors such as multiple poses, lighting, expressions, age, and occlusion. Photos of the same person also vary widely. In some photos, more than one face may appear. For these multi-face images, only the center coordinate face is selected as the target, and other areas are regarded as background interference. There are 13,233 face images in the LFW dataset. Each image gives the corresponding person name. There are 5749 people, and most of them have only one picture. The size of each picture is 250 × 250, most of which are color images, but there are also some black and white face pictures.

4.1. Data Preprocessing

The original LBP face features are not directly training. We need to improve these facial features (as shown in Figure 5). The improved methods of this study are as follows. Improvements to LBP face features: We note that LBP face features are composed of 8-bit binary tuples, thus it is only necessary to insert random 8-bit random between these tuples to generated key. In particular, the face features mentioned below refer to the modified face features.

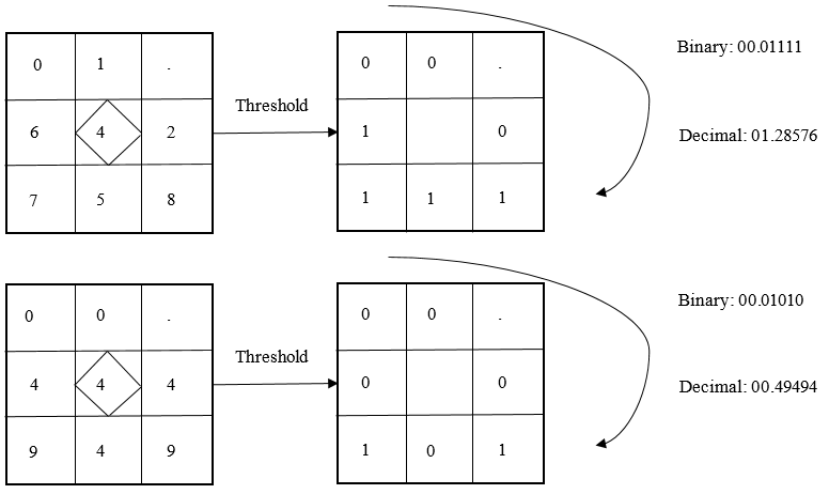

Figure 5. Improvements to shape-indexed face features.

Our training method is based on Adam (Adaptive Moment Estimation). In practice, our training methods are very different from those of GANs [32,33]. Our training methods have been improved in several ways, and some improvements have been made to the goals of Section 3.3. Next, we introduce these improvements and give further details.

First, training relies on hundreds of "small-batch" calculations. We do not calculate "best B" for given datasets. We simply approximate it, alternating B's training with A's and Server's. Intuitively speaking, training may be roughly as follows. A may initially create some encrypted face features, but Server and B do not understand at all. By training a few steps, A and Server may find a way to communicate so that Server can at least partially decrypt A's facial features, but B cannot understand this. In particular, A and Server may find some trivial conversions. However, after some training, B may begin to learn to crack part of the facial feature data. With more training, A and Server may find ways to improve encryption, especially that can better utilize key facial features for encryption. B finally finds that it is impossible to easily crack the data of these key facial features. This kind of alternation is a typical confrontation game; the theory of continuous adversarial game involves the results of balancing, which may apply to our problem [34].

In addition, in the training of *A* and *Server*, we do not try to maximize the reconstruction error of *B*. If *B* is completely wrong, *B* can flip all output bits completely at the next iteration. A more realistic and useful goal for *A* and *Server* is to minimize the mutual information between B's guess and real face features. In the case of symmetric encryption, this goal is equivalent to making the answer generated by *B* indistinguishable from random guessing. This approach is somewhat similar to the approach designed to prevent overtraining of current opponents (Salimans et al., 2016) [33]. In addition, we can also adjust the loss function so that it does not have a big impact on B, nor does it fix some minor errors in Server.

Finally, once we stop training *A* and *Server*, they choose their own cryptosystem. This study verifies that this work is good and tries to break the cryptosystem instance by training many Bs. Some of these examples may come from the early stages of training.

4.2. Experimental Process

As a proof of concept, this study implements *A*, *Server*, and *B* networks that use N-bit random face features and encrypted face features and generate N-entry floating-point encrypted face features, N = 8, 16, 32, and 64 (in the previous study, two face feature formats have been normalized to an 8-bit "0-1" format). Face features and keys are evenly distributed. The key is not intentionally reused, but reappears due to random selection. The experiment considers a more interesting detail which allows for different sizes of face features and keys. This study used TensorFlow (Abadi et al., 2016a; b) to

perform our experiments. We used a Nvidia V100 GPU Server with memory size 32 GB for training. The specific computing platform does not affect the results of the experiment.

The network of this study follows the "mix and convert" model, described in Section 3. The network connects two n-bit inputs (face features and keys) to a 2N entry vector, using −1 (−1 for 0) and 1 to represent the bit value. The convolutional layer is described in terms of window size, input depth, and output depth. Each step has a "step"—the amount of movement of the window. Intuitively, the layer slides out a size 2 window from the 2N output elements of the FC layer and outputs two numbers: output depth and step. In addition to the last layer, we use a sigmoid nonlinear unit. In the last layer, after reducing the output to N elements, we use a tanh nonlinear unit. tanh bringing the value back to a range that can be mapped to a binary value. The *Server* network is the same as the *A* network, and the *B* network only takes the encrypted face feature as input.

To train the network, this study used a "mini-batch" size, with one batch having 2000 facial features. We used TensorFlow's Adam (Kingma and Ba, 2014) optimizer to learn at a rate of 0.001 [31]. The entire training did not slow down the learning rate because we want *A*, *Server*, and *B* to continue to react strongly to changes in other components until *A* and *Server*—hopefully—implement a solution that has enough support for additional changes to *B*.

Training is alternated between *A*/*Server* and *B*. *A* and *Server* train a small batch and then train for next small batches. We choose this ratio to give the adversary *B* a slightly calculated advantage without over-training it, leading to it becoming too strong for the current parameters of A and Server. From an empirical point of view, this value is very effective [35].

The the loss function of *B* is as described above: the L2 distance between the guess of *B* and the face feature of the input. The loss functions of *A* and Server are more complex, as shown in Sections 3.2 and 3.3. This function has two components, related to the server's refactoring errors and the eavesdropper's success. The first component is the L2 distance between the *Server* output and the input face feature. On the other hand, the latter is $(\mathrm{N}/2 - \mathrm{B\,L2\,error})^2/(\mathrm{N}/2)^2$. This definition expresses the goal described in Section 3, i.e., *B* should not do better than random guessing. Therefore, when half of the messages are wrong and half are correct, this component is minimized. We chose a quadratic formula to emphasize *B* made a big mistake, and gave a few penalties when *B* correctly guessed a few bits, occasionally, even if B's guess is random. Using this formula allows us to have a meaningful loss function for each example (rather than looking at larger batch statistics) and improve the robustness of the training. The cost is that our last well-trained *A* and *Server* allow *B* to get more information than a simple random guess. We have not yet achieved satisfactory results because the linear (rather than quadratic) loss function is linear in B's reconstruction error. The best solution is still an open question [36].

The WGAN-E face feature encryption and decryption system working process is shown in Figure 6.

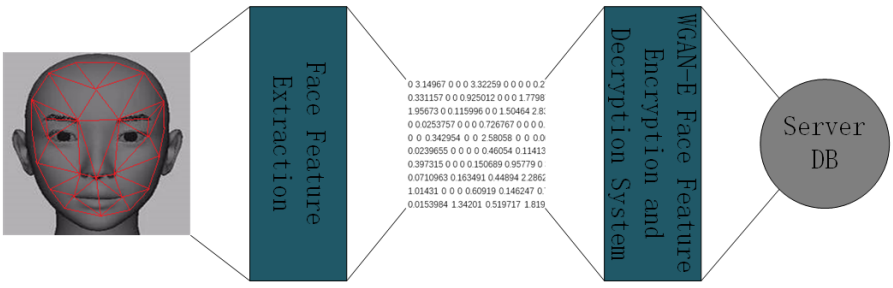

Figure 6. WGAN-E face feature encryption and decryption system. Note: DB is database. A facial feature is generated by LBP feature extraction to generate a series of codes for representing feature values. These face feature codes are stored in the server after being encrypted by WGAN-E face features. When face information needs to be identified, the facial features are extracted from the server and the original facial features can be obtained by WGAN-E decryption.

4.3. Analysis of Experimental Results

After successful training, the output result passes through a precision threshold (e.g., up to 0.05 bits of reconstruction error A and Server, rather than random guess prediction), we reset the B network, train it five times from scratch, each step is 250,000, and record the best results through B. If A/Server combination failed to achieve the target below threshold at 150,000 steps, and at the same time if retraining B gains a substantial advantage, the solution to this problem is non-robust. Otherwise, we think this is a successful training result.

Figures 7 and 8 show the number of successful runs, server rebuild errors, and B rebuild errors with N = 8 face features and key training steps, using a small batch of 2000. Each point in the graph is the average error in 2000 examples. The ideal result is that the server's rebuild error drops to zero, and B's rebuild error reaches 8 (half the error). In this example, both refactoring errors are high. After some time, A and Server began to communicate effectively, but, to a certain extent, B could gradually improve their understanding. Then, in step 10,000, A and Server oppose B's progress. By 15,000 steps, the training objectives were achieved. The remaining steps only slightly increased the B reconstruction error.

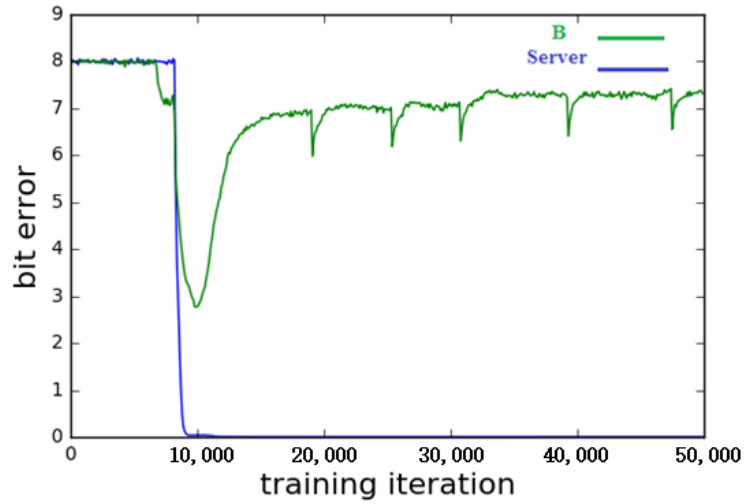

Figure 7. Evolution of the reconstruction errors of Server and B during training.

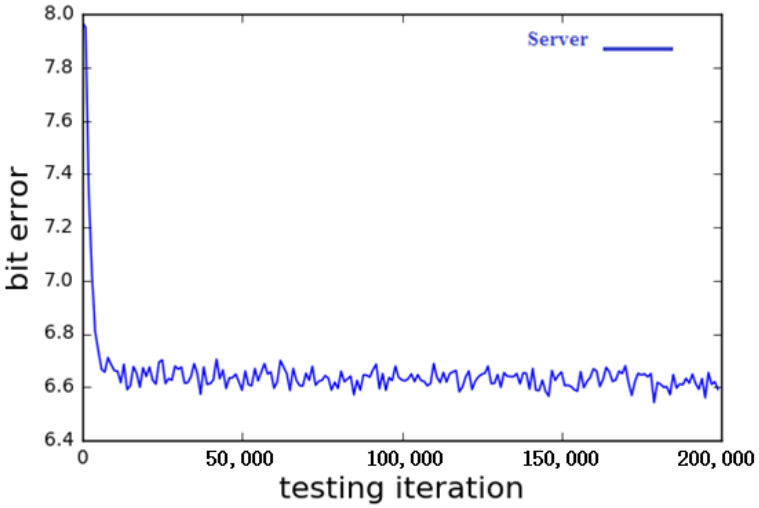

Figure 8. Evolution of Server Reconstruction Errors during Training.

The training diagram in Figure 9 does not appear to be a typical success result of a neural network. In this process, from the beginning to the end oscillation of the steps is not usually good. The dynamics of this confrontational training seem to be more reminiscent of the evolutionary process. These dynamics seem to depend on some random variations that make the pieces slightly mixed, but, once there is some mixing, the gradient drop can push it further forward.

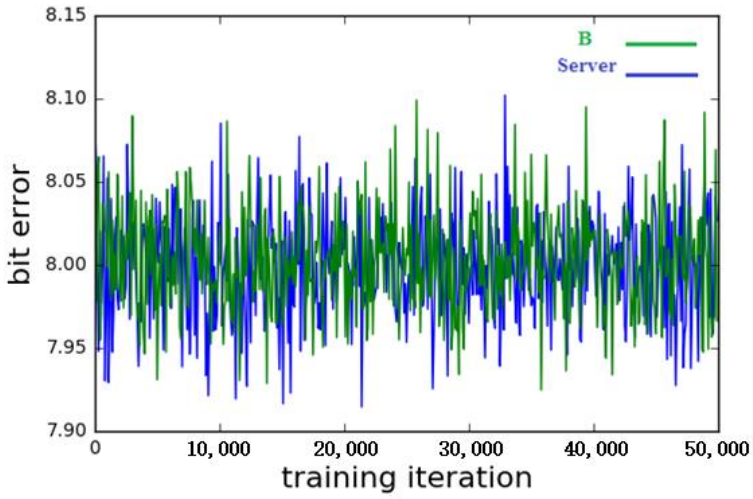

Figure 9. The initial failure of the training process.

The argument behind this interpretation is that training is not always successful. At N = 8, 3 out of 10 initial runs failed, and none of these failures caused a reconstruction error of S at the 0.05 threshold, nor did the B reconstruction error exceed 7.3 bits (8 bits). To test the robustness of the other 14 A/B combinations, we conducted five trainings on B and obtained B reconstruction errors from 4.67 to 6.97 with an average of 6.1. If we arbitrarily define success as keeping the server at 0.05-bit reconstruction error and asking B to have at least six errors, on average, half of the training is successful (5 of 10 cases).

Although training with opponents is often unstable (Salimans et al., 2016), some additional engineering of neural networks and their training might increase the overall success rate. For example, with only 512 small batches, our success rate is only 1/3 (we use a small batch size of 200). In the future, it may be worthwhile to study the effects of small batch sizes, as well as other parameters such as learning rates.

For a successful training, we studied the changes in encrypted facial features caused by different facial features/key pairs. Although we did not perform a detailed analysis of the human feature encryption method, we did make some observations. First, it is key-dependent: changing the key and keeping the result of the face feature constant leads to different encrypted face feature outputs. It is also a necessary for successful communication. However, it is more than just XOR. In particular, the output values are usually floating point values, not 0 and 1. In addition, the effect of changes to the key or face feature bits propagates across multiple elements in the encrypted face feature, rather than being limited to a single bit as in XOR. In a key, the flipping of a unit usually results in a significant change in 3–6 of the 8 elements in the encrypted face feature, and the other elements will change less.

To evaluate the merits of a cryptographic algorithm, we should first consider whether it meets certain security contraints, and then, the efficiency of the algorithm and the key distribution and confidentiality.

Algorithm security analysis (Table 1): The face feature encryption algorithm based on the GAN proposed in this paper uses a different key sequence for each round of information, even if one of the keys is known using the exhaustive key algorithm. That is, even though it obtained a round of information, it could not obtain any other information. The key sequence of the algorithm is generated according to the GAN and has more security features. Therefore, it has a more secure encryption effect, further improving the security of the key, and improving the confusion of the key generation algorithm to effectively resist password attacks.

Table 1. Algorithm security analysis (L, Low; M, Medium; H, High).

	MD5	SHA1	HMAC	AES	DES	RSA	ECC	WGAN
LBP	M	H	L	H	L	H	H	H
Gabor	L	M	L	M	L	M	H	H
ASM	L	M	M	H	M	H	H	H
SeetaFace	M	H	L	H	L	H	H	H
FaceNet	H	H	M	H	M	H	H	H
OpenFace	M	H	L	H	L	H	H	H

Sensitivity analysis of keys (Table 2): Initial sensitivity is the basic nature of the key system. Minor changes are made to the initial conditions of the key system. The values of the bit change rate are relatively large, and the initial sensitivity of the key system is very large. Due to the sensitivity of the key system to the initial conditions, many uncorrelated, random generated key sequences can be obtained.

Table 2. Sensitivity analysis of keys (L, Low; M, Medium; H, High).

	MD5	SHA1	HMAC	AES	DES	RSA	ECC	WGAN
LBP	M	H	L	H	L	M	M	H
Gabor	M	H	L	H	L	M	M	H
ASM	M	H	L	H	L	M	M	H
SeetaFace	M	H	L	H	L	M	M	H
FaceNet	M	H	L	H	L	M	M	H
OpenFace	M	H	L	H	L	M	M	H

Anti-cryptanalysis (Table 3): For this encryption algorithm, the password attacker may mainly adopt the reconstruction algorithm/GAN method, so that it must know the initial state and the connection matrix of the algorithm/GAN. If the plaintext total method is used to imitate the key system, then it must solve how to use plaintext–ciphertext to solve the algorithm/GAN parameters, the initialization state, and the external input initial value. If differential analysis and linear analysis are used to decipher, it is very difficult to linearly approximate a nonlinear key system.

Table 3. Anti-cryptanalysis (L, Low; M, Medium; H, High).

	MD5	SHA1	HMAC	AES	DES	RSA	ECC	WGAN
LBP	L	H	M	H	M	M	H	H
Gabor	L	M	M	H	M	M	M	H
ASM	L	H	L	H	L	M	H	M
SeetaFace	L	H	M	H	M	M	H	H
FaceNet	L	H	M	H	M	M	H	H
OpenFace	L	H	M	H	M	M	H	H

Efficiency analysis (Table 4): Since the algorithm can use parallel operations in the encryption process, such as WGAN against neural network operations, in general, parallel operations are faster than serial operations, especially for binary calculations. Under normal circumstances, the length of the ciphertext is longer than that of the plaintext. This is mainly because the encryption of the plaintext packet sometimes needs to be filled, and the encryption efficiency of the entire system is also very high. Since the improved algorithm enhances the security factor, there is a certain decline in the execution efficiency with the conventional encryption algorithm. Therefore, the improved algorithm exchanges the execution efficiency for the algorithm security.

Table 4. Efficiency analysis (S, Slow [>1]; M, Medium [0.1–0.9]; F, Fast [0.01–0.09]; FR, Faster [<0.001]).

	MD5	SHA1	HMAC	AES	DES	RSA	ECC	WGAN
LBP	FR	S	M	FR	F	M	S	F
Gabor	FR	M	F	FR	F	M	S	F
ASM	FR	M	F	FR	M	F	S	F
SeetaFace	F	S	M	F	F	S	S	M
FaceNet	F	S	S	F	FR	S	S	M
OpenFace	M	S	M	F	F	M	S	M

Deciphering complexity analysis (Table 5): Let us first analyze the complexity of deciphering the ciphertext. First, we must know the length of the ciphertext, so that we can get the network sample. The decipher cannot know the dimension of the neural network and the number of samples. To calculate only in the simplest case, only the confrontation network is used. We assume that the total number of network samples is n. Considering that there are at least two samples of the network, the sample dimension of the network should be between 2 and [n/2]. Since this study does not use the stability point of the WGAN network, the cracker cannot know the number of times our sample needs to be iterated, so we think that the calculation is very expensive and unsolvable in a limited time without knowing the key. For the parameters of the network, we transmit several pairs of such sequences, which are continuous during transmission. These parameters are the starting position of the network, the network dimension, and the number of iterations required for the sample to generate the key. If the transmission is dispersed, the parameters of the network are greatly increased.

Table 5. Deciphering complexity analysis (L, Low; M, Medium; H, High).

	MD5	SHA1	HMAC	AES	DES	RSA	ECC	WGAN
LBP	M	H	L	H	L	H	H	H
Gabor	M	H	L	H	L	H	H	H
ASM	M	H	L	H	L	H	H	H
SeetaFace	M	H	L	H	L	H	H	H
FaceNet	M	H	L	H	L	H	H	H
OpenFace	M	H	L	H	L	H	H	H

Equal error rate (EER) is a biometric security system algorithm used to predetermine the threshold values for its false acceptance rate and its false rejection rate. When the rates are equal, the common value is referred to as the equal error rate. The value indicates that the proportion of false acceptances is equal to the proportion of false rejections. The lower is the equal error rate value, the higher is the accuracy of the biometric system. When the key is not known by the impostor, the EER is as showm in Table 6. When the key has been hacked by the impostor, EER is as shown in Table 7.

Table 6. When the key is not known by the impostor, equal error rate.

	BioHashing	BioPhasor	SecureSketch	WGAN-E
LBP	0.02	0	0	0
Gabor	0.01	0.02	0	0
ASM	0	0.01	0.01	0
SeetaFace	0	0	0	0
FaceNet	0	0	0.01	0
OpenFace	0.01	0	0	0

Table 7. When the key has been hacked by the impostor, equal error rate.

	BioHashing	BioPhasor	SecureSketch	WGAN-E
LBP	0.71	0.85	0.80	0.56
Gabor	0.72	0.82	0.82	0.58
ASM	0.71	0.86	0.82	0.58
SeetaFace	0.70	0.81	0.77	0.52
FaceNet	0.70	0.82	0.78	0.53
OpenFace	0.70	0.81	0.77	0.53

5. Conclusions and Future Work

This paper uses neural networks to learn to protect the communication between face features and servers. Learning is not limited to a specific set of cryptographic algorithms: it is based solely on the privacy rules that the training objectives represent. The attacker is simulated by a neural network. In addition, this study might be able to implement the process of replacing analog encrypted data with reinforcement learning. Finally, we conclude that neural networks are useful not only for face feature protection, but also for attacks. Although neural networks seem unlikely to be good at protecting facial features (3 failures in 10 initial runs), they may be very effective in understanding metadata and traffic analysis.

Using multi-threaded code will greatly speed up the model efficiency because separate weighted sum, neuron, multiply, or add operations can be performed on separate threads. However, even without multithreading, on a processor with n cores, the study can run n repeated models on n input instances and expect a flattening cost of n = 1. Implementing an efficient matrix multiplier can also improve batch processing time and may allow for faster convolution operations. In addition to multithreading, this study is also dedicated to designing virtual full adders and using GPU to accelerate existing network models. Implementing homomorphic logic gates on GPU is valuable because each of the research system's operations can be paralleled in some form. Plaintext weights are encrypted without noise to improve performance. In practice, the latest deep learning model uses many additional features, such as residual network modules or a self-normalization activation function. We should be able to extend our encryption system with most of the functions. This study will also do some work to optimize the deep learning encryption model of embedded hardware, review the problem scenario of this study and create a powerful analysis of execution. Deep learning solutions could ensure user privacy. Over time, this may lead to a privacy-centric software development industry that coexists with the expanding role of big data.

In this work, we demonstrate the problem of weight reduction in WGAN-E and introduce an alternative in the form of a penalty clause in the critics' loss, which does not present the same problem. Using our approach, we demonstrate powerful modeling performance and stability across various architectures. Another interesting direction is to adapt the penalty term to the standard GAN objective function, which encourages discriminators to learn smoother decision boundaries and stabilize training.

In fact, we considered using AlexNet, VGG series, and ResNet series to conduct ablation experiments for the selection of neural networks A, B, and Server in the experiment, and compare their performance in the experiment. However, no matter how delicately the facial features are extracted or how accurate the encryption and decryption are, the former two are not comparable to ResNet.

The server that decrypts the neural network does not guarantee that 100% of the decryption is correct. This is a small defect in this system. However, when the data volume base is quite large, such as the LFW dataset used in this study, Server and A learn each other slowly and finally achieve decryption accuracy of about 98.7%. In real life, commercial face recognition systems have difficulty accurately identifying non-Caucasian people or twins, and the recognition accuracy is also affected by light (e.g., when the light is strong, it is difficult for the Alipay face scan payment system to recognize the face). However, comparing the face recognition accuracy error caused by the above reasons, we think that a slight error in the decryption accuracy of Server is acceptable. In actual use, if a face is canned many times or a commercial face database is used for training, this error would be smaller.

We believe that it is necessary to discuss whether encryption and decryption affect the recognition speed of the FR system. In fact, in the process of encryption and decryption, it is substantially equivalent to the time required for the A network and the Server network with trained facial features. In actual work, this time is usually related to the depth of the network. In our system, this time is approximately equal to 0.02 s. For the entire face recognition process, this time is already quite small.

Author Contributions: Conceptualization, B.J., S.Z., and C.W.; methodology, B.J. and C.W.; software, B.J. and N.N.X.; validation, B.J. and Y.W. formal analysis, C.W., Y.W., and N.N.X.; investigation, B.J. and Y.W.; resources, C.W. and N.N.X.; data curation, B.J.; writing—original draft preparation, B.J.; writing—review and editing, C.W., S.Z., Y.W., and N.N.X.; visualization, B.J.; supervision, C.W. and N.N.X.; and project administration, C.W., S.Z., and N.N.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (Nos. 2018YFC0810204 and 2018YFB17026), National Natural Science Foundation of China (No. 61872242), Shanghai Science and Technology Innovation Action Plan Project (17511107203), and Shanghai key lab of modern optical system.

Acknowledgments: The authors would like to appreciate all anonymous reviewers for their insightful comments and constructive suggestions to polish this paper in high quality.

Conflicts of Interest: All authors declare no conflicts of interest in this paper.

References

1. Ahonen, T.; Hadid, A.; Pietikäinen, M. Face Recognition with Local Binary Patterns. In *Proceedings of the Advances in Visual Computing*; Springer Science and Business Media LLC: Berlin, Germany, 2004; Volume 3021, pp. 469–481.
2. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
3. Jiang, H.; Learned-Miller, E. Face Detection with the Faster R-CNN. In Proceedings of the 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, USA, 30 May–3 June 2017; pp. 650–657.
4. Dayan, P.; Hinton, G.E.; Neal, R.M.; Zemel, R.S. The Helmholtz Machine. *Neural Comput.* **1995**, *7*, 889–904. [CrossRef] [PubMed]
5. Wu, S.; Kan, M.; He, Z.; Shan, S.; Chen, X. Funnel-structured cascade for multi-view face detection with alignment-awareness. *Neurocomputing* **2017**, *221*, 138–145. [CrossRef]
6. Cavoukian, A.; Marinelli, T.; Stoianov, A.; Martin, K.; Plataniotis, K.N.; Chibba, M.; DeSouza, L.; Frederiksen, S. Biometric Encryption: Creating a Privacy-Preserving 'Watch-List' Facial Recognition System. In *Security and Privacy in Biometrics*; Springer Science and Business Media LLC: London, Britain, 2013; pp. 215–238.
7. Goldwasser, S.; Micali, S. Probabilistic encryption. *J. Comput. Syst. Sci.* **1984**, *28*, 270–299. [CrossRef]
8. Guo, D.H. ASIC Design of Chaotic Encryption Based on Neural Networks. *Chin. J. Comput.* **2000**.
9. Lumini, A.; Nanni, L. An improved BioHashing for human authentication. *Pattern Recognit.* **2007**, *40*, 1057–1065. [CrossRef]
10. Teoh, A.B.; Ngo, D.C. Biophasor: Token Supplemented Cancellable Biometrics. In Proceedings of the 2006 9th International Conference on Control, Automation, Robotics and Vision, Singapore, 5–8 December 2006; Institute of Electrical and Electronics Engineers (IEEE); pp. 1–5.
11. Sutcu, Y.Q.; Memon, N. Secure Sketch for Biometric Templates. *Comput. Vis.* **2006**, *4284*, 99–113.
12. Hong-Yu, L.I. A Stream Cipher Encryption Method Based on Neural Network. *Sci. Technol. Inf.* **2008**.
13. Jiang, L.; Shanghai, L. A multistage chaotic encryption algorithm based on neural network. *Comput. Appl. Softw.* **2006**.
14. Yi, L.I.; Pan, X.J. AES Based on Neural Network of Chaotic Encryption Algorithm. *Sci. Technol. Eng.* **2010**.
15. He, C.; Ming, K.; Wang, Y.; Wang, Z.J. A Deep Learning Based Attack for The Chaos-based Image Encryption. *arXiv* **2019**, arXiv:1907.12245.
16. Fei, H.; Pu, C.; Gao, H.; Tang, M.; Li, L. An image compression and encryption scheme based on deep learning. *arXiv* **2016**, arXiv:1608.05001.
17. Xie, P.; Wu, B.; Sun, G. BAYHENN: Combining Bayesian Deep Learning and Homomorphic Encryption for Secure DNN Inference. *arXiv* **2019**, arXiv:1906.00639.
18. Ryffel, T.; Trask, A.; Dahl, M.; Wagner, B.; Mancuso, J.; Rueckert, D.; Passerat-Palmbach, J. A generic framework for privacy preserving deep learning. *arXiv* **2018**, arXiv:1811.04017v2.
19. Alon, B.; Oren, E.; Ran, G.-B. Low Latency Privacy Preserving Inference. *arXiv* **2018**, arXiv:1812.10659.
20. Le, T.P.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1333–1345.
21. Lotfollahi, M.; Siavoshani, M.J.; Zade, R.S.H.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2019**, *24*, 1999–2012. [CrossRef]
22. Coutinho, M.; Albuquerque, R.D.O.; Borges, F.; Villalba, L.J.G.; Kim, T.-H. Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography. *Sensors* **2018**, *18*, 1306. [CrossRef]
23. Li, J.; Yang, Y.; Ge, H.; Wang, Y.; Zhao, L. Generative adversarial nets in laser-induced fluorescence spectrum image recognition of mine water inrush. *Int. J. Distrib. Sens. Networks* **2019**, *15*. [CrossRef]
24. Kinzel, W.; Kanter, I. *Neural Cryptography*; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2004; pp. 1351–1354.
25. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs. *arXiv* **2017**, arXiv:1704.00028v3.
26. Denton, E.; Chintala, S.; Szlam, A.; Fergus, R. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *arXiv* **2015**.

27. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv* **2017**, arXiv:1701.07875v2.
28. Rouhani, B.D.; Riazi, M.S.; Koushanfar, F. DeepSecure: Scalable Provably-Secure Deep Learning. *arXiv* **2018**, arXiv:1705.08963.
29. Xie, P.; Bilenko, M.; Finley, T.; Gilad-Bachrach, R.; Lauter, K.; Naehrig, M. Crypto-Nets: Neural Networks over Encrypted Data. *Comput. Sci.* **2014**.
30. Ke, Y.; Zhang, M.; Liu, J.; Su, T. Generative Steganography with Kerckhoffs' Principle based on Generative Adversarial Networks. *arXiv* **2017**, arXiv:1711.04916.
31. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *Comput. Sci.* **2014**.
32. Barone, A.V.M. Low-rank passthrough neural networks. In Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP, Melbourne, Australia; 2018.
33. Lahiri, A.; Jain, A.; Nadendla, D.; Biswas, P.K. Improved Techniques for GAN based Facial Inpainting. *arXiv* **2018**, arXiv:1810.08774.
34. Boemer, F.; Costache, A.; Cammarota, R.; Wierzynski, C. nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data. In Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, London, UK, 11–15 November 2019.
35. Nowozin, S.; Cseke, B.; Tomioka, R. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016.
36. Shaon, F.; Kantarcioglu, M.; Ranise, S.; Swarup, V. A Practical Framework for Executing Complex Queries over Encrypted Multimedia Data. In *Proceedings of the Advances in Visual Computing*; Springer Science and Business Media LLC: Cham, Germany, 2016; Volume 9766, pp. 179–195.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).