Journal of Information Hiding and Privacy Protection

Tech Science Press

Doi:10.32604/jihpp.2025.067510

ARTICLE

Secure Text Mail Encryption with Generative Adversarial Networks

Alexej Schelle[1,2,*]

[1]Constructor University, Bremen gGmbH, Campus Ring 1, Bremen, 28759, Germany
[2]IU Internationale Hochschule, Juri-Gagarin-Ring 152, Erfurt, D-99084, Germany

*Corresponding Author: Alexej Schelle. Email: alexej.schelle.ext@iu.org

Received: 06 May 2025; Accepted: 07 July 2025; Published: 31 July 2025

ABSTRACT: This work presents an encryption model based on Generative Adversarial Networks (GANs). Encryption of RTF-8 data is realized by dynamically generating decimal numbers that lead to the encryption and decryption of alphabetic strings in integer representation by simple addition rules, the modulus of the dimension of the considered alphabet. The binary numbers for the private dynamic keys correspond to the binary numbers of public reference keys, as defined by a specific GAN configuration. For reversible encryption with a bijective mapping between dynamic and reference keys, as defined by the GAN encryptor, secure text encryption can be achieved by transferring a GAN-encrypted public key along with the encrypted text from a sender to a receiver. Using the technique described above, secure text mail transfer can be realized through component-wise encryption and decryption of text mail strings, with total key sizes of up to 10^8 bits that define random decimal numbers generated by the GAN. From the present model, we assert that encrypted texts can be transmitted more efficiently and securely than with RSA encryption, given an additional security component that users of the specific configuration of the GAN encryption model are unaware of the GAN encryptor circuit and configuration, respectively.

KEYWORDS: Artificial Intelligence; cyber security; generative adverserial networks; text encryption

1 Introduction

Modern hybrid models for secure text mail encryption are based on the connection of symmetric and asymmetric encryption techniques using algorithms such as the Rivest-Shamir-Adleman encryption algorithm (RSA) to send encrypted keys from a sender to a receiver [1]. Those methods usually rely on the definition of one public and one private key for secure key exchange, which a network may use to encrypt and decrypt a certain alphanumeric text mail structure [2]. Hybrid models do have the advantage of secure key transfer at the cost of lower computational efficiency; however, the generation of the key may be unsecured due to a lack of internal security. Additionally, the components of the secured key are encrypted with the same single key.

To examine hybrid encryption models and their vulnerabilities, including the efficiency trade-offs between symmetric and asymmetric encryption techniques, the concept of Generative Adversarial Networks (GAN) proves to be a well-suited technique to model data that approximates the original data or data objects from the generation and comparison of the data objects modeled by the GAN with the original object [3]. A large range of applications has been developed with GANs in information sciences, such as sound or color recognition [4,5], or more complex classification algorithms that serve to classify data structures and, in general, the elements of information theory [6–8].

Very fundamentally, Generative Adversarial Networks build an important component to understanding the functionality of biological and physical processes of neuronal networks. In the framework of basic field theories for the modeling of information processing, the very basic mechanisms of learning as described by simple mathematical models such as the single-layer perceptron (SLP) work equivalently to the stepwise convergence of a GAN that finds configurations of information compatible with already known or consistent results [9,10]. While technologies based on GANs have been developed quite extensively for applications in the context of artificial intelligence, fewer efforts have been made to enhance information security and data encryption techniques [11].

As a natural type of use, GANs can be applied to find random configurations of binary numbers that vary with the internal modeling parameters of the artificial network. In standard types of applications, the principal structure of a GAN is composed of a generator that models certain configurations of data structures and objects, respectively, which is connected to a differentiator that compares the generated data structures to pre-defined reference objects. In this configuration, a GAN primarily builds an artificial learning method that belongs to the class of supervised learning algorithms.

In the present work, we show how to avoid these weak points with key encryption models based on GANs that do not rely on factorization on the one hand. Developing an encrypting technique that defines different components (decimal numbers) for the encryption of the key itself, with a circuit encryptor configuration that is hidden from the users of the secure key, *additionally* enhances the security of the encryption technique. Applying this GAN-based secure text mail encryption algorithm, we find that up to 24-bit encryption corresponding to total key sizes of up to 10^5–10^6 bits for a standard text mail of a few hundred words can be processed on a standard personal computer on the scale of a few minutes of computation time. The security of text mail encryption with GANs is not obtained from the complexity of the key or the encryption algorithm *alone* but from the possibility of disconnecting (securing) the GAN encryptor from the parties that use the encryption model and prohibiting the access of external users to the actual specific random configuration of the (pre-defined) GAN encryptor.

Encryption of RTF-8 (text) data is realized by dynamically generating private decimal numbers that lead to the encryption and decryption of alphabetic strings in integer representation by simple addition rules, the modulus of the dimension of the considered alphabet [12,13]. Random decimal values for the encryption of alphabetic components for email strings are calculated by the randomization of binary numbers with definite dimensions N that generate decimal values with a random number generator in Python. The binary numbers for the private dynamical keys correlate with the binary numbers of public reference keys from a mapping defined by the specific GAN implementation. For reversible encryption with bijective mapping between dynamic and reference keys as defined by the GAN, the encoded text mail is transferred from a sender to a receiver together with the reference keys that build the basis for modeling decimal numbers from configurations of random binary keys. Using the technique described, secure text mail transfer can be realized by component-wise encrypting text mail strings with random decimal numbers obtained from the GAN.

The present GAN encryption model builds a new and innovative technique for text mail encryption that enables secure password and text string encryption with security levels that scale exponentially with the total key size, while the time for generating (partial) private keys for encrpytion scales only linearly with the number of bits used for calculating binary keys from specific random GAN configurations. From securing specific model configurations from the sender and receiver, security can be *enhanced* by excluding certain risks from internal security issues relating to the sender and the receiver.

2 Structure

Our encryption technology is built on two different components—the discriminator and the generator that exchange information in terms of N-bit binary number codes. From the standard definition of GANs, we formally simplify our system parameters from setting all weighting coefficients of the two interacting (SLP) neuronal networks to the binary basis [14]. Each component of the GAN thus defines a decimal number from the N-bit signal, which in particular enables the definition of a complex number from the sum

$$m(G,R) = \sum_{j=1}^{N} a_j 2^j + i \sum_{j=1}^{N} b_j 2^j, \tag{1}$$

where $a_j, b_j \in \mathbb{Z}_2$ and $G, R \in \mathbb{Z}_2^N$. The mathematical function $m : \mathbb{Z}_2^N \times \mathbb{Z}_2^N \to \mathbb{C}$ formally maps two N-bit configurations of the keys $G = [a_1, \ldots, a_N]$ and $R = [b_1, \ldots, b_N]$ to the complex number space $\mathbb{C}$ that describes the GAN configurations in a mathematically phenomenological way. As shown in Fig. 1, standard configurations of random numbers generated by the GAN generator can be modified with a circuit connected between the generator and the discriminator part of the encryption network. Each realization of a randomly generated N-bit signal at the generating part of the GAN is processed in the circuit and then transferred to the differentiator, which distinguishes the generated signal from predefined reference signals or data patterns (see Fig. 2).

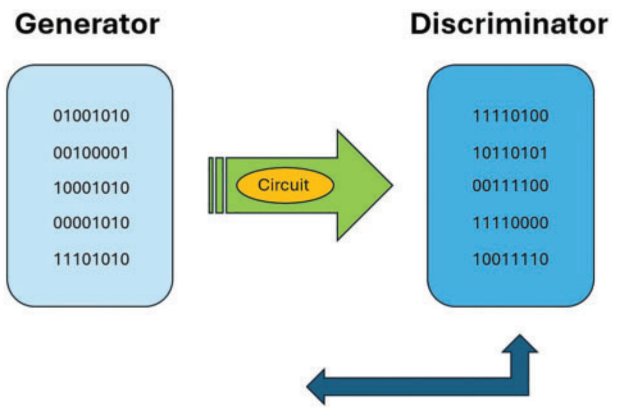

Figure 1: (Color online) The conceptual setup of the Generative Adversarial Network (GAN) is shown in the figure above. The generating part of the technology builds dynamic N-bit-sized key values G that are processed with the circuit of the GAN. From that configuration, reference keys R are obtained at the discriminator of the GAN. The transmitted reference keys are secure against hacking since the circuit technology is generated randomly and, in particular, hidden from external observers. After receiving the (dynamically) encrypted text message with the reference keys, transmitted text structures such as emails or passwords can be decrypted with the recalled dynamical keys obtained intrinsically from the submitted reference keys by applying the GAN circuit technology

Different circuit layers that are reversible concerning the mathematical transformation of the associated N-bit system can be configured for textual decryption. An important formal aspect is the circuit's reversibility, which is expressed by bijective mathematical mappings that describe the interaction of the generator with the discriminator of the GAN. Circuits can be built by connecting the standard logical NOT gate for reversible transformations and the logical gates AND, OR, NOR, and XOR, which leads to irreversible circuits for the generation of complex numbers (decimal number pairs) for the encryption and decryption of text structures [15]. Ensuring reversibility means that the GAN technology enables unique encryption and decryption of an associated RTF-8 text (compare Table 1).

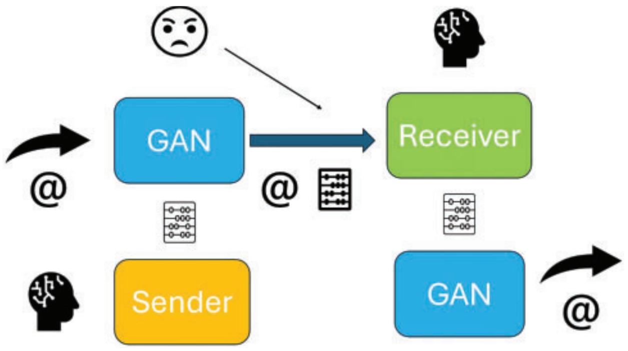

Figure 2: (Color online) The GAN encryption model is illustrated in the figure above. A sender provides the text message of length M that is sent from the sender to the GAN, which encrypts the message with an (exemplary) randomly generated configuration of a total number of logical NOT gates. The logical NOT gates connect the N-bits of a dynamical key at the generator with the N-bits of the reference keys at the discriminator of the neuronal network that tests the total configuration for deviation until the checksum of the deviation equals zero. The encrypted message, together with the M reference keys (that build a total key size of $M \times N$ bits) are sent to the receiver, which itself decrypts the message with M dynamical keys obtained from applying the GAN to generate them from the reference keys for decryption. From simply pishing the encrypted text and the reference keys, an external observer isn't able to decrypt the message within polynomial time since the number of possible keys scales exponentially with $2^{M\times N}$. The GAN encryptor technology is built from a Python source code that can be password protected to avoid information loss from security lack of the sender or the receiver of the message

Table 1: (Color online) Table summarizes the most important logical gates and the associated computation time for a GAN encryption model used to generate the (pairwise) circuit logic of the GAN for the encryption of a text message with a total of 3000 characters for in total M logical gates per GAN encryptor realization with N bits, i.e., a total key size of 24×10^3 bits. Logical gates AND, NOR, OR, and XOR define irreversible logical operations. From the logical NOT gate, reversible text encryption is realized. A specific GAN configuration is realized by randomly generating pairs of integer numbers that define connections of input bins of the GAN (similar to an SLP input signal), between which standard logical gates connect the different bitwise configurations. Pairwise bits (a, b) of either zero or one are then mapped to either $(a, f(a))$ for AND, OR, NOR, or XOR, or to $(f(a), f(b))$ for the NOT logical gate operation. Starting from partial key sizes with an N-bit configuration of $N = 32$ and larger, parallel processing techniques have to be integrated to build an encryption system of reliable computational complexity. While the partial key size is bound to this rather low value, the total key size may increase to a total key size of up to 10^6–10^8 bits.

A, B	AND	OR	NOR	XOR	NOT
0, 0	0	0	1	0	1, 1
0, 1	0	1	0	1	1, 0
1, 0	0	1	0	1	0, 1
1, 1	1	1	0	0	0, 0
C-time	316.87	314.37	321.07	318.14	533.46

Similar to the RSA algorithm, we assume one public and one private key, where the public key is transferred from a sender to a receiver, and the private key is used to encrypt and decrypt the text message. Instead of encrypting the text message or a transferred key with the (decrypted) combination of public and private keys (RSA), relying in particular on costly computation power, the security of the GAN approach makes use of the following technique. Relevant text messages are connected to a GAN encryptor that entails a total number of M reference keys (public keys) and M dynamical keys (private keys) of size N-bit, i.e., total key size of $M \times N$ bits, where M is the number of letters in the text. Each text message letter is encrypted and decrypted using a single character of the M-bit dynamic N-bit keys for encryption and decryption, respectively. Encryption and decryption are performed by mapping the relevant characters A_k of the text message to a decimal number B_k and adding the modulus of the decimal number for encryption and decryption, respectively, following the equation

$$B_k = (f(A_k) + Re[m(GAN_c(G, R))]) mod K \tag{2}$$

for encryption and

$$A_k = (f^{-1}((B_k) - Re[m(GAN_c(G, R))]) mod K \tag{3}$$

for decryption, where GAN_c is a function that maps the N-bit configurations of the Generative Adversarial Network for a pair of N-bit configurations at the generator and discriminator, respectively, to a specific (relative) reference key R′, i.e.,

$$(G, R') = GAN_c(G, R) \tag{4}$$

given a certain configuration C of the circuit technology, with K the number of characters of the considered alphabet. The function f: $\{a, \ldots, Z\} \rightarrow$ N maps alphabetical values from the alphabet to numeric values N (decimal codes). This way, each letter of the clear text is encrypted with a decimal number $Re[m(GAN_c(G, R))] \in \mathbb{N}$ obtained from the dynamic key, which is related to the reference key by the GAN that models each dynamic key $G \in \mathbb{Z}_2^N$ from a random reference key, intrinsically resulting in a related random number $Im[m(GAN_c(G, R))] \in \mathbb{N}$.

The GAN encryption model enables secure text mail transfer by allowing the sender to access only the reference keys and the clear and encrypted text for submission to a receiver. Encrypted text structures sent to the receiver are decrypted by connecting the reference keys with the GAN to generate the (same) dynamical key structure to decrypt the transferred text message. Thereby, the encryptor configuration is unknown to the users of the GAN (sender and receiver), while the current configuration of the random circuit is (assumed to be) hidden from all external parties. Compared to algorithms like RSA, besides a typically larger key size that is encrypted with the GAN encryptor, security is further enhanced by sending (a) decrypted reference key(s) on the one hand. The GAN setup itself allows the decryption of a text message only by decrypting the reference keys with the GAN encryptor that is hidden from the external observer. Ideally, the security setup can be extended by securing the GAN encryptor from the sender and the receiver, e.g., building simple password protection for the Python source code, making the encryption model secure against internal information leaks. Thus, an external observer cannot decrypt a text message by passing the password-protected key generation framework implemented in the Python programming language, since the current circuit configuration is still unknown. Passwords can be chosen to any complexity, and access to the system generating source code can be tracked, such as to change the encryptor configuration after access or after a certain period, with possibly several attempts to access the key-generating Python framework.

We have tested the algorithm against scaling as shown in Fig. 3 and find that M times 8-bit to 24-bit key encryption and decryption corresponding to a total key size of up to 10^5–10^6 digits for standard text messages of a few thousand ASCII signs are performed within a few seconds to minutes of computation time on a standard personal computer, depending on the key size of the component-wise GAN reference keys. Fig. 2 shows the scaling of the computational time against the number of bits used to generate keys for text encryption and decryption, respectively. The GAN algorithm works stably against failure modes and is available as a prototype software model on github.com [16].

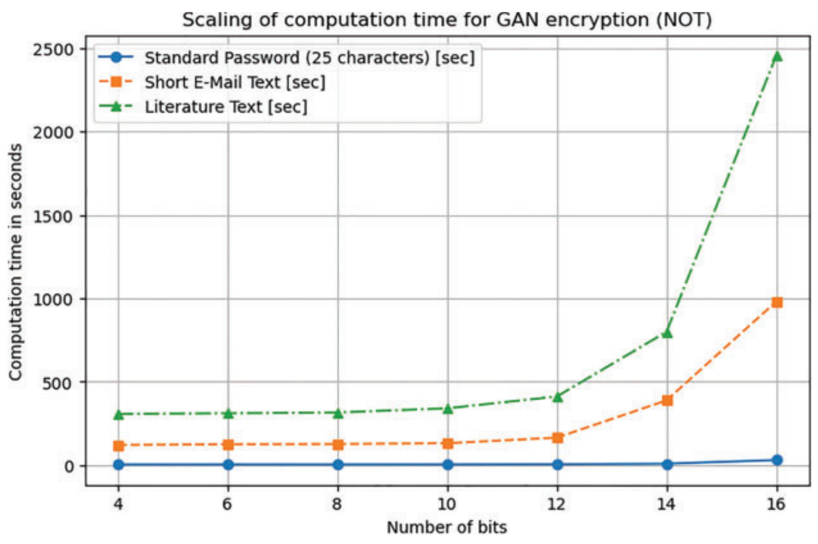

Figure 3: (Color online) Shown is the scaling of computation time as a function of the number of bits (partial key sizes, i.e., one key per character of the encrypted message) used to reversibly (upper figure) and irreversibly (lower figure) encrypt standard passwords of length 25 characters (circles), small email text structures (squares) and one-page literature texts (triangles)—for connections realized by random configurations of NOT and AND logical gates integrated into the GAN encryptor (AND logical gate shows the same scaling behavior, not shown explicitly)

3 Pseudocode

The following pseudocode describes the process flow of the GAN encryption model for encryption and decryption modes, respectively.

```
START THE ALGORITHMIC FLOW
SET KEY LENGTH (TOTAL LENGTH IS SUM OF LENGTH)
FUNCTION THAT MAPS LETTERS TO UNIQUE INTEGER NUMBERS
FUNCTION THAT MAPS VALUES TO UNIQUE LETTERS
FUNCTION THAT GENERATES BINARY KEYS
FUNCTION THAT DISTINGUISHES BINARY KEYS OF N-BIT SIZE
FUNCTIONS THAT ENTAIL CIRCUITS IN SEQUENTIAL LOGICS
GAN FUNCTION
    WHILE TRUE
    APPLY CIRCUIT FUNCTION
    IF FIRST ITERATION
        GENERATE INITIAL KEY VALUES
    ELSE
        GENERATE GAN KEY
    COMPUTE DISCRIMINATOR
        IF MATCHES PREDEFINED ACCURACY
            RETURN CURRENT KEY
```

```
RANDOMLY GENERATE BINARY KEYS
READ INPUT TEXT (FOR ENCRYPTION/DECRYPTION)
INITIALIZE RANDOM KEYS
ITERATE OVER EACH CHARACTER IN THE INPUT STRING
    FOR EACH LETTER
        GENERATE INITIAL KEY
        GENERATE REFERENCE KEY
        STORE REFERENCE KEY
        SET REFERENCE KEY = INITIAL KEY
        GENERATE ENCRYPTION/DECRYPTION KEY WITH GAN
      CONVERT LETTER TO DECIMAL VALUE (ENCRYPTION)
GENERATE DECIMAL VALUE FROM BINARY ENCRYPTION KEY
DECIMAL VALUE TO LETTER
ENCRYPT/DECRYPT DECIMAL (LETTER) VALUE
        CONVERT DECIMAL VALUE TO LETTER (DECRYPTION)
WRITE TO FILE
END
```

4 Results and Variations

The main applicational scope of the present encryption algorithm is the secure text transfer from a sender to a receiver using the GAN encryptor as described in the previous Section 2. Extensions for further applications, like irreversible text deletion or password generation, have been developed from the primary GAN encryption model, which has been tested to work successfully. Quantifying the computational performance of the routines in this applicational framework relies on testing different runtime variables as a function of the number of bits used to define the relevant decryption keys. For secure text mail encryption, scaling of computation time in terms of key complexity shows that a critical value of around a few hours of computation time is observed at component-wise key sizes of around 36 bits. Instead of gaining security from large component-wise key sizes used for encryption, it is the size of the total key that defines a secure framework for information security.

Protecting information in the present security model further originates from protecting the randomly and automatically generated GAN encryptor from all parties involved in the text transfer process, i.e., a sender, a receiver, as well as a potential external hacker. This is achieved by allowing access to the Python source code only for external third parties (to which current configurations of the randomly generated logical GAN circuit are unknown) rather than the actual users of the software, and by automating the generation of the encryptor. Although it is not necessary to secure information about the GAN configuration from the sender and the receiver for the GAN encryption model's security, it reduces the risk of internal information leaking.

The structure of a GAN is built of a generator and a discriminator that build an interacting system exchanging information as described above. While the generating part tries to model N-bit-wise dynamical keys that fit a certain set of randomly created reference keys stored in the discriminator part of the GAN, a circuit that is configured between the two components ensures non-symmetric key structures between dynamic and reference keys. Dynamic keys, which are pairwise connected to reference keys by the GAN encryptor, are used to encrypt and decrypt the relevant text in RTF-8 format.

In the present setup, we have defined logical operators mainly from combinations of NOT gates that ensure reversible transfer of N-bit logical signals as described by bijective functions defined on binary logical sets. Convergence is achieved by generating dynamic keys that match the reference keys at the discriminator after passing the logical circuit exactly, relating a pair of dynamic and reference keys in each convergent step of a multiple of several subsequent GAN iteration cycles. For a text string of a total of M letters and signs, equally many reference keys are generated within a few seconds to minutes in an N-bit GAN generator with typically $N = 18–24$ bits serving for information transfer between the sender and the receiver of an encrypted message. As realized in our case study, we have transferred the encrypted texts, such as passwords, by standard email from different senders to receivers, together with the reference keys for decryption over a wide distance of several thousand kilometers. Text decryption was possible within seconds to minutes using the decryption of encrypted text messages using the decryption mode of the GAN for the reference keys. An artificial memory intelligence has been built in the generator part of the GAN that was able to reduce the number of iteration steps for convergence, but not the computation time, to accelerate the approach of a zero-sum state (between the generator and the discriminator).

Mapping of characters to decimal number format from a function f has been realized for text and numbers in RTF-8 format with dynamic keys of size N-bit (with values of N up to $N = 32$). From an N-bit key structure, decimal numbers in the range of 0 to 10^8 are calculated by transforming binary key structures to decimal numbers. Our source code has also been extended to recognize special characters in text strings or integer numbers and to distinguish letters case-sensitively. Counting the number of case-sensitive large and small letters is an important feature of the N-bit GAN encryption model. In such a mode of operation, the GAN encryption model can be applied as a password validator and generator, respectively, that classifies passwords in terms of different complexities classes defined by the different variations of operation, i.e., class 1 for modes of special character recognition, class 2 for modes of number recognition, and class 3 for case-sensitive letter recognition. Most secure passwords can thus be generated, or detected, by randomizing character strings and combining randomly created strings that satisfy the conditions of classes 1 to 3. Passwords of lower complexity are obtained from requiring the constraints for class 1 and/or class 2 conditions only. Complexity 1 passwords of standard length (10 to 15 characters) have been generated and encrypted/decrypted within a few seconds of computation time, complexity 2 passwords within minutes, and finally, complexity 3 passwords within the time scale of less than an hour. Notably, the present approach assumes no external perturbations or internal error mechanisms that may lead to compromised GAN circuit configurations. To work successfully in the encryption and decryption mode, the GAN circuit has to match all bitwise configurations of the N-bit key size exactly. Only one wrong bit configuration leads to a wrong decimal number and therefore a falsely decrypted password or text message, respectively.

Finally, we have applied the GAN encryption model to perform non-reversible text deletion [17]. This mode of operation can be realized by implementing non-reversible circuit logic with the logical AND, OR, NOR, and XOR gates that are connected between the generator and the discriminator in the GAN encryptor. Since the forward direction (encryption) of this mode of operation generates an ideally disjoint set of reference keys, encryption of a text string results in an irreversible mapping after the dynamic keys are overwritten by the encrypted keys—since, indeed in such case, the set of dynamic keys cannot be remodeled from the remaining set of (non-bijectively relating) reference keys in the backward direction (decryption).

The GAN encryption algorithm works stably against error generation. Potential vulnerabilities are external attacks against the GAN configuration, or key generation errors caused by unstable chip technologies on the personal computer, or the runtime environment. Hacking of the corresponding text files that save the GAN encryptor configuration in bitwise format leads to failure of the GAN algorithm. Since decimal numbers are generated from mapping random N-bit configurations to decimal numbers from a standard number type transformation, decrypting text strings and messages from differently (wrongly) generated decimal numbers from wrong configurations of the GAN decryptor leads to errors in the final text string. Key generation errors from intrinsic failure mechanisms do build a source of error mechanisms, however, with only very low impact. Because errors may mainly only be generated from wrong chip configurations on a personal computer, or from wrong configurations of the runtime environment, not from the GAN algorithm, the percentage of this error mechanism is rather vanishingly small. Hence, as long as the GAN encryptor model works in the correct reversible mode, bijective mapping ensures errorless encryption and decryption of the GAN algorithm.

5 Discussion

In the present approach, the concept of Generative Adversarial Networks applies to generate dynamic keys of a large total size for encryption. Text messages that can be read and mapped to numeric values from a standard text file are decrypted by calculating and modifying random decimal numbers with a random number generator that entails a circuit as an encryption technology. The decrypted text message and the set of M binary keys are sent from the sender to the receiver, enabling the decryption of the encrypted message with the receiver's part. The sender and the receiver, as much as an external hacker using the encryption software or phishing the encrypted text with large-size reference keys attached, respectively, are unable to access the Python script, to which access is encrypted with a strong password, making the encryption model twofold secure against external hacking.

Reversible encryption can be applied for email text exchange, secure password transfer, or valid key generation. Text mail structures in RTF-8 format of up to a few thousand words can be encrypted and securely forwarded using the GAN encryption model within a few seconds to minutes of computation time. Special characters that are mapped from object types to numeric values can be implemented independently and individually. Password encryption of complex passwords containing up to 100 or more characters can be realized as an integrated part of more complex encryption models for banking and insurance environmental applications [18–20].

For such purposes, there are different approaches to integrating Python in a Java runtime environment that is suitable for programming user software applications [21]. Jython is an implementation of Python that allows for the integration of the programming language Python in a Java framework. While all Python routines are accessible in the Jython environment, it only supports Python up to version 2. Process builders or Java Native Interface with CPython do provide another application to call Python scripts from Java that is compatible with all versions of Python. The programming language Python can also be integrated into other languages such as C/C++, JavaScript, .NET, and Go.

Compared to the RSA algorithm, security is enhanced by first providing private and public keys that scale linearly with the number of digits used for password representation. Starting from password sizes of 20−25 digits, one may thus overbid the security of an RSA approach with partial key sizes of around $N = 16$ bits. Applying encryption with $N = 24$ bits at password sizes of a few hundred digits, one can securely transfer passwords with total key sizes of up to a few thousand bits with the proposed GAN encryptor. Different and alternative encrypting algorithms, such as ElGamal Encryption [22], Elliptic Curve Cryptography, or Lattice-Based Cryptography [23], do work based on other computation methods but approximately provide the same security measure as the RSA algorithm in terms of encryption and decryption key sizes.

Secondly, security is conceptually enhanced since the circuit logic of GAN isn't known to the users of the network, whereas the actual randomly generated configuration of logical gates is hidden from the developer of the software. That way, encrypted text can only be hacked if two parties (software user and developer) lack information security.

As illustrated in Table 2, the scaling of the security level as a function of the key size for the RSA, ECC, and GAN model algorithms (for a password string of size N = 24 characters) indicates that the GAN approach converges much faster to the same security level as a function of the required key size. The scaling of the security level with key size can be estimated as the square root of the key size for RSA, as a linear function for the ECC, and the GAN encryption model. Since the GAN model decomposed the total key into partial keys that, in total, form a general key of size N-bit, the computation time reduces by several orders of magnitude from the RSA to the GAN model. As compared e.g., to RSA and ECC encryption models, the GAN approach thus leads to the same security level for much lower required key sizes (per letter). In contrast, time complexity for hacking a generated private key scales exponentially for the GAN encryption model as a function of the total key size that is comparable to the total key sizes in RSA models.

Table 2: (Color online) The scaling of the security level as a function of the key size is shown in the table for RSA, ECC, and GAN model algorithms (for a password string of size N = 24 characters)

Security Level (bits)	RSA Key Size (bits)	ECC Key Size (bits)	GAN Single Key Size (bits)
80	1024	160	6–8
112	2048	224	8–10
128	3072	256	10
192	7680	384	16
256	15,360	512	22

6 Conclusion

In the present study, we have presented an encryption model for text encryption and decryption, respectively, based on a GAN encryptor that allows for secure RTF-8 text encryption. From modeling random keys that are unaccessible for decryption by any party of the GAN users without using the specific (protected) GAN encryptor model realization, i.e., sender, receiver, and software coordinator, the model implements the submission of encrypted text messages with reference keys from a sender to a receiver that build the foundation for decryption within the framework of the presented GAN prototype software. Encryption and decryption with total key sizes of up to 10^6–10^8 bits allow for secure text transfer and irreversible deletion of text structures or passwords in the framework of private as well as commercial applications and communication.

Integrating the GAN encryption model as a Python source code that encrypts and decrypts messages from a frontend of a standard email software may secure text messages that are sent from a sender to a receiver. Starting from the prototype as presented, this would require building a stable connection of the GAN encoder model to the runtime environment of the webmail software. To establish secure connections and to ensure correct encryption and decryption, challenges are to build the GAN model in a way that disables external hackers or malware from changing dynamically stored configurations of the GAN encryptor.

The model may build the foundation to develop commercial software technologies or integrated plugins in Python format from the presented basis model implemented in the Python 3 programming language. In the sequel to this paper, GAN encryptor configurations composed of NOT logical gates have been used mainly. More complex circuit configurations built from combinations of NOT and CNOT logical gates shall be subject to future work on the presented GAN encryption model.

Acknowledgement: I'd like to thank Adrian Dahl, Sven Engels, Fritz Fischer, Mert Köktürk, Renars Miculis, Sarah Rosa Werner, and Betül Yurtman for their contributions and discussions on the content of the presented work on a text encryption model with Generative Adversarial Networks.

Funding Statement: The author acknowledges financial support of IU International University of Applied Sciences and Constructor University for the lecturer positions at the universities.

Availability of Data and Materials: The data that support the findings of this study are openly available in TextMailEncryption at https://github.com/alexej-schelle/TextmailEncryption (accessed on 6 July 2025).

Ethics Approval: This study was conducted in accordance with the ethical standards of IU International University of Applied Sciences.

Conflicts of Interest: The author declares no conflicts of interest to report regarding the present study.

References

1. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. 1978;21(2):120–6. doi:10.1145/359340.359342.
2. Boneh D, Franklin M. Identity-based encryption from the Weil pairing. SIAM J Comput. 2003;32(3):586–615. doi:10.1137/s0097539701398521.
3. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. Commun ACM. 2020;63(11):139–44. doi:10.1145/3422622.
4. Donahue C, McAuley J, Puckette M. Adversarial audio synthesis [Internet]. Paper presented at: International Conference Learn Representations (ICLR); 2019 [cited 2025 Mar 22]. Available from: https://arxiv.org/abs/1802.04208.
5. Zhang R, Isola P, Efros AA. Colorful image colorization [Internet]. Paper presented at: The European Conference on Computer Vision (ECCV); 2016 [cited 2025 Mar 22]. Available from: https://arxiv.org/abs/1603.08511.
6. Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. IEEE Trans Pattern Anal Mach Intell. 2021;43(6):1957–72. doi:10.1109/TPAMI.2020.2976433.
7. Zhu J, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks [Internet]. Paper presented at: IEEE International Conference on Computer Vision (ICCV); 2017; Venice, Italy [cited 2025 Mar 22]. Available from: https://arxiv.org/abs/1703.10593.
8. Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. Paper presented at: International Conference on Learn Representations (ICLR); 2019; New Orleans, LA, USA [cited 2025 Mar 22]. Available from: https://arxiv.org/abs/1809.11096.
9. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev. 1958;65(6):386–408. doi:10.1037/h0042519.
10. Minsky M, Papert S. Perceptrons: an introduction to computational geometry. Cambridge, MA, USA: MIT Press; 1969.
11. Hu W, Tan Y, Wang X. Generative adversarial networks for cybersecurity: attacks, defenses, and future trends. ACM Comput Surv. 2022;55(3):1–36. doi:10.1145/3490237.
12. Katz J, Lindell Y. Introduction to modern cryptography (4. Aufl.). Boca Raton, FL, USA: Chapman & Hall/CRC; 2025.
13. Stinson DR, Paterson MB. Cryptography: theory and practice. 4th ed. Boca Raton, FL, USA: CRC Press; 2023.
14. Mehlig B. Machine learning with neural networks: an introduction for scientists and engineers. Cambridge, UK: Cambridge University Press; 2021.
15. Mano MM, Ciletti MD. Digital design with an introduction to the Verilog HDL, VHDL, and systemVerilog. 6th ed. Hoboken, NJ, USA: Pearson; 2023.
16. Dahl A, Engels S, Fischer F, Köktürk M, Miculis R, Werner SR, et al. TextmailEncryption [Internet]. GitHub repository; [cited 2025 Mar 22]. Available from: https://github.com/alexej-schelle/TextmailEncryption.

17. Shannon CE. A mathematical theory of communication. Bell Syst Tech J. 1948;27(3):379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
18. Smith J, Brown R. Core banking systems: architecture and integration. Cham, Switzerland: Springer; 2020.
19. Williams M. Modern insurance software solutions: trends and technologies. Hoboken, NJ, USA: Wiley; 2021.
20. Kumar A, Zhang L. Cybersecurity in financial and insurance software. J Fintech Secur. 2022;15(3):45–67.
21. Doe J, Smith J. A comparative study of Python and Java in modern software development. J Program Lang. 2020;15(3):45–60.
22. Ranasinghe R, Athukorala P. A generalization of the ElGamal public-key cryptosystem. J Discrete Math Sci Cryptogr. 2022;25(8):2395–403. doi:10.1080/09720529.2020.1857902.
23. Regev O. On lattices, learning with errors, random linear codes, and cryptography. arXiv:2401.03703. 2024. doi:10.48550/arXiv.2401.03703.