The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2019

Volume 6, Pages 81-87

ICRES 2019: International Conference on Research in Education and Science

ParEncrypt: A Two-phased Encryption for Improved Security

Haneya KHAN
University of Texas at Dallas

Asim SIDDIQUI
University of Texas at Dallas

Fariha ISLAM
University of Texas at Dallas

Elham HAMID
University of Texas at Dallas

Ebru CELIKEL CANKAYA
University of Texas at Dallas

Abstract: The idea of encrypting text is alluring still as it is the most convenient and straightforward means to shuffle text so as to make it unintelligible to unintended third parties. We design and develop a novel symmetric encryption algorithm ParEncrypt that implements a cascaded structure of known encryption approaches: It first employs Feistel cipher to scramble the plaintext, then applies a further step of substitution implemented via left and right parentheses into the intermediate output to finalize the encryption. This two-phased encryption provides enhanced security with the cost of slower paced performance. We provide performance comparison of our design with common benchmark encryption algorithms and see that ParEncrypt outperforms all others with respect to the level of security. Additionally, our scheme offers an additional benefit as our special way of encryption provides a significant level of compression.

Keywords: Encryption, Lossless compression, Data security

Introduction

Security has become one of the most critical parts of the technological world. The increasing use of technology in almost every aspect of life mandates security and privacy of personal and sensitive data. One of the most basic, yet useful, forms of security are encryption algorithms. Various different types of encryption algorithms exist today, each with different properties and uses. Different subsets include symmetric key algorithms, asymmetric key algorithms, block ciphers, stream ciphers, etc.

Symmetric key algorithms are on subset of encryption algorithms and use a single private key to encrypt and decrypt sensitive data. Existing implementations include AES, DES, Blowfish, etc.

We propose a new symmetric key encryption scheme. We base our scheme on the Feistel cipher model, a symmetric key algorithm model, and introduce a nested implementation of it that produces a uniform ciphertext.We then evaluate the performance of our proposed encryption scheme and compare it to the performance of similar existing algorithms. Finally, we consider further work that can be done to better both the security and performance of our scheme.

Related Work

Research in the field of encryption nowadays is typically associated with the application of existing algorithms. For novel encryption algorithm research, much of it is done with a specific purpose in mind building off widely adopted cryptographic tools including the Advanced Encryption Standard, Rivest-Shamir-Adleman cryptosystem, and MD5 message-digest algorithm.

The authors of [1] seek to build a new encryption algorithm named DTT which utilizes computationally simple shifting procedures and "bitxor" to obfuscate messages. The algorithm itself incorporates environmental factors including temperature and time that strengthen it against cryptanalysis. The goal of the algorithm is to prevent hackers from decrypting the massage as it fails based on time and temperature. The simple shifting procedure is a key part of the Feistel Network Structure incorporated in our scheme, Nested.

Double-chaining encryption is another approach on improving existing encryption algorithms. In [2], a double chaining encryption algorithm is constructed with fixed 16-byte blocks. It functions similarly to Cipher Block Chaining mode commonly used in AES but does it in two steps instead of a single one. It performs Cipher Block Chaining once before reversing the order of blocks and performing it again. Because of the increased overhead, the number of rounds is shortened to 4. This is similar to our algorithm, as we reduced the number of rounds in order to compensate for the increased processing in each round. A key difference is that they double the chaining process where as we double the encryption itself.

Another work [3], develops an order-revealing encryption algorithm, which builds upon Order-Preserving Encryption (OPE) that supports comparisons over encrypted values. This algorithm is useful for encrypted databases, and data that is stored as it provides security and quick comparisons. A key weakness in OPE encrypted databases is the vulnerability to "inference attacks" which the Order-Revealing Encryption (ORE) algorithm strengthens against. Additionally, the ORE scheme speeds up the process significantly as it is over 60 times faster than conventional OPE schemes. Furthermore, OPE solutions were limited by the constraint that both the ciphertext and plaintext needed to be numbered and ordered. On the contrary, the proposed ORE scheme is not limited by the order constraint, and instead relies on pseudorandom functions in a tuple of three algorithms. The benefit is that searches can be executed on encrypted data which does not sacrifice functionality for security.

Other encryption schemes seek to meet with new demands of technologies. The authors of [4] design a broadcast encryption scheme for verifying subscribers. They propose a new public-key cryptosystem by using efficient and computationally inexpensive public key cryptosystem Number Theory Research Unit (NTRU). The NTRU cryptosystem relies a on triple of integer parameters and four sets of polynomials with integer coefficients. Key Creation is achieved by randomly choosing two polynomials in a set which satisfies invertible modularity. For encryption, the encoder selects a public key and a random polynomial to compute cipher text. Decryption utilizes precomputed polynomial and inverts the encryption process. The algorithm is then applied to a broadcast encryption scheme that can detect traitors or pirate decoders. Upon detection it can revoke some users availability without redistributing a new secret key. The cryptosystem is computationally light with a logarithmic time complexity.

A growing area where integrity and confidentiality are key is in cloud computing. Researchers in [5] seek to optimize Fully Homomorphic Encryption for the purpose of securing cloud data from exploitation during computation. They use Rivest-Shamir-Adleman (RSA) to improve performance and Diffie-Hellman Algorithm for secure channel establishment in the Fully Homomorphic Encryption Scheme. The authors utilize inbuilt functions of MATLAB for implementation. Comparison between the RSA and Diffie-Hellman Algorithms is thorough and RSA is found to significantly cut down on execution time and probability of attacks at the cost of being less robust.

The advent of cloud computing has been in tandem with the development of the Internet of Things (IoT). One work [6], seeks to improve on the lightweight encryption algorithm (LEA) standardized in Korean IoT devices. The current LEA clock encryption algorithm is vulnerable to side-channel analysis attack. A masking technique is used to counter this vulnerability at the cost of performance. The authors develop and implement a new LEA block encryption algorithm which introduces a dummy operation of 4 bytes. This dummy operations does little to add to space overhead and increases each triple of 4 bytes to a quadruple of 4 bytes. It requires only a single operation to restore the original plain text value. Moreover, it protects against power consumption wave pattern outputs, so the secret key cannot be guessed easily. Upon testing, it was found to be roughly 17 times faster than existing masking algorithm that prevents against side-channel attack.

The authors of [7] don't really develop their own encryption algorithm. Rather they test the effectiveness of Attribute Based Encryption (ABE) which doesn't encrypt the whole data but attributes of the data. A key limitation in ABE is that the decryption process is expensive. They propose modifications to ABE which simplifies the process using digital signature and hash functions. The purpose of including these steps is to make it more robust against attackers. The secret key is generated using the input of the asymmetric public key into a hash function. Encryption takes in the public key and secret key and creates a digital signature. This only occurs if the encryption of the hash function matches with the access structure. Decryption is more complex. It takes an input of the public key, private key, secret key and access structure. If the private key matches the access structure, it will decrypt the hash function before checking the access structure.

Others incorporate a multitude of security mechanisms in designing a cryptosystem such as in [8]. The authors devise a mixed encryption algorithm utilizing the heterogenous capabilities of the networking components of IoT devices. The proposed solution combines AES for faster encryption speeds, ECC for digital signature and key management, MD5 to integrate it all together, forming a hybrid encryption algorithm. It uses the hard math problem elliptic curve to generate both the public and private key. Because the hybrid cipher algorithm is easy to calculate and key distribution easier to manage, it is seen as a potential solution to protect the authenticity and integrity during transmission for IoT devices. Nested is similar as it builds upon adopted security mechanisms.

ParEncrypt: Nested Encryption in Two Phases

In this section, we describe the design and implementation of our encryption scheme ParEncrypt. In essence, we propose a two phase encryption that cascades Feistel cipher in the first phase with substitution cipher to follow in the second phase. The symmetric scheme takes plaintext and a key as input, and then produces a double encrypted, uniform ciphertext in two steps, as shown in Figure 1; first, it doubly encrypts the inputted plaintext based on the Feistel cipher model using the inputted key, producing a ciphertext, and then second, it further encrypts the ciphertext into a series of left and right parentheses to obscure any patterns in the ciphertext. Moreover, the scheme is designed for the Unicode character set.

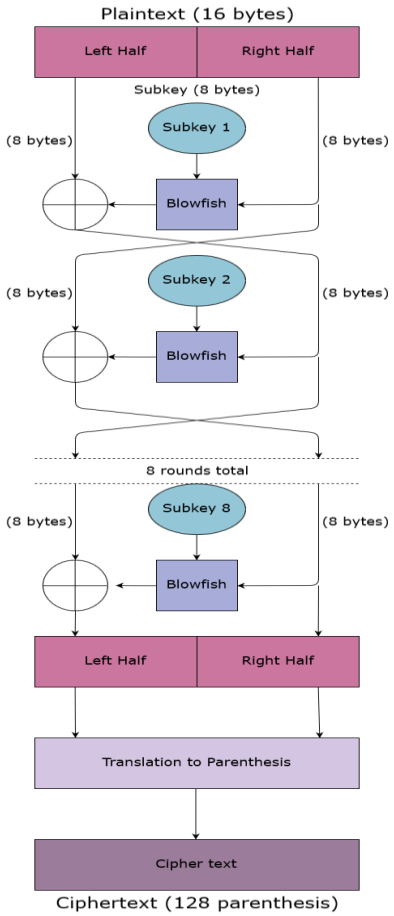

Figure 1. Encryption process diagram for ParEncrypt

Phase I: The Feistel Cipher

The first part of the scheme employs the Feistel cipher model that most existing and effective encryption algorithms, such as DES, are based on. We chose this block cipher model due to its iterative use of both substitution and permutation rounds that make it very difficult to break.

The algorithm uses a 16-char key (128 bits) for encryption. We chose a larger key size to prevent brute force attacks from breaking the cipher. The plaintext is split into 16-char (128-bit) blocks and incomplete blocks are padded so that the size of the blocks also cannot be used to break the cipher.

The scheme then employs eight rounds of substitutions and permutations on each block. Usually, 16 rounds are performed; however, we halved this due to other alterations we made so that the performance would not be degraded.

The Feistel cipher uses an internal function, the F function, for the substitutions and permutations. Generally, the F function is simply a one-way function; however, to increase security, we use the Blowfish cipher as the F function. Although the blowfish cipher is reversible, it is also based on the Feistel cipher and is very secure. This allows for the double encryption as the algorithm is essentially using nested Feistel ciphers. This is the main reason why we use 8 rounds instead of 16 rounds - to balance security against time. We use an existing, open-source implementation of the Blowfish cipher as this is not the focus of our algorithm - it is simply the F function inside it [9].

In the Feistel cipher, a different key is used for each round and is generally a variation of the original inputted key. For the subkeys, instead of a simple shift left or right, we shifted inwards: the two outer bits moved to the middle for every round. Finally, the blocks are appended together to produce a double encrypted ciphertext.

Phase II: Uniform Encryption

The second step in the scheme further enciphers the already encrypted test once more into a series of left and right parentheses. The characters in the encrypted text produced by the first step are broken into binary and then replaced with the left and right parentheses. We do this to hide any patterns in the original ciphertext that can be used to break the cipher.

Decryption

To decrypt the ciphertext, the algorithm simply performs the encryption steps in reverse order. First, the parenthesis is converted back into the ciphertext character. Then, the Feistel cipher algorithm runs with the subkeys in reverse order. Since the algorithm is symmetric, the same key is used to encrypt and decrypt.

Pseudocode

Our encryption scheme ParEncrypt can be summarized as a pseudocode as follows:

1. The *input* to the algorithm is a *16-char key* and the *plaintext* to encrypt.
2. The text is then split into *16-char blocks* and incomplete blocks are padded.
3. Each block is passed into the *Feistel cipher*, which permutes for *8 rounds*.

The *F function*, which is the encryption scheme used for each round, is the Blowfish cipher. *Subkeys* are the original key shifted in for each round.

4. The blocks are then appended together and passed into the *second encryption method,* which converts the ciphered text into *parentheses*.
5. The *output* is the *doubly encrypted, uniform ciphertext*.

Results

Our experiments record the execution time of encrypting and decrypting various sizes of plaintext and ciphertext using ParEncrypt. We compare our runtimes to those of existing algorithms. Figure 2 shows a sample run of our scheme.

```
user@system #python nested.py --encrypt '7sU4b$jsKs^y9R5g' 'plaintext to encrypt'

()()))(()()))((()(((()()((())(()()()(()))))()(((()))(())()(()()((((((((()(())(
)(())((((()))()))()))(((()((((()(()()))))(()(())((((((()))(((()(())()((()(()))
))()))))))((()(()(()()(()())()(()(()))((()((()(()(()()()()()(()(()()()))(())()(
(())((()))

user@system #python nested.py --decrypt '7sU4b$jsKs^y9R5g' '()()))(()()))((()(((
()()((())(()()()(()))))()(((()))(())()(()()(((((((()(())()((()))((((()))()))())
(((()((((()(()()))))(()(())((((((()))(((()(())()((()(())))()))))))((()(()(()(
)(()())()(()(()))((()((()(()(()()()()()(()(()()()))(())()((())((()))'

plaintext to encrypt............

user@system #
```

Figure 2. Sample Run of ParEncrypt

Performance of ParEncrypt

We randomly generated a 16-char Unicode string to use as the key. We then ran our algorithm on three different sized plaintext input. The small plaintext input had 15 characters. The medium plaintext input had 500 characters. The large plaintext input had 1500 characters. For each size, we ran 10 rounds of both encryption and decryption and captured the runtime of the algorithm. Table 1 describes the average runtime for encryption and decryption for the different sized plaintexts.

Table 1. Average Runtimes of ParEncrypt on Different Plaintext Sizes

	Encryption Time	**Decryption Time**
15 char	.010 sec	.009 sec
500 char	.015 sec	.011 sec
1500 char	.021 sec	.026 sec

We then compared our scheme's performance to that of existing algorithms, including Blowfish, AES, and DES. We used the same key and 1500-character plaintext for each algorithm and collected runtimes for 10 rounds. Table 2 compares the average encryption and decryption runtimes. Figures 3 and 4 show this in graphs.

Table 2. Runtime of ParEncrypt vs. Existing Schemes

	Encryption Time	**Decryption Time**
Blowfish	.003 sec	.002 sec
AES	.003 sec	.003 sec
DES	.003 sec	.002 sec
ParEncrypt	.020 sec	.024 sec

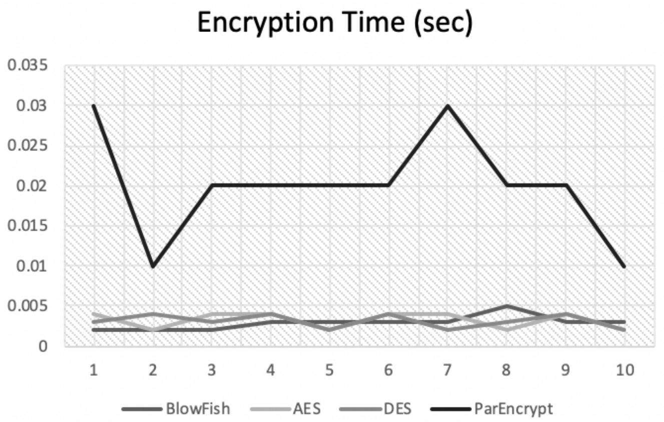

Figure 3. Encryption Runtime of ParEncrypt vs. Existing Schemes

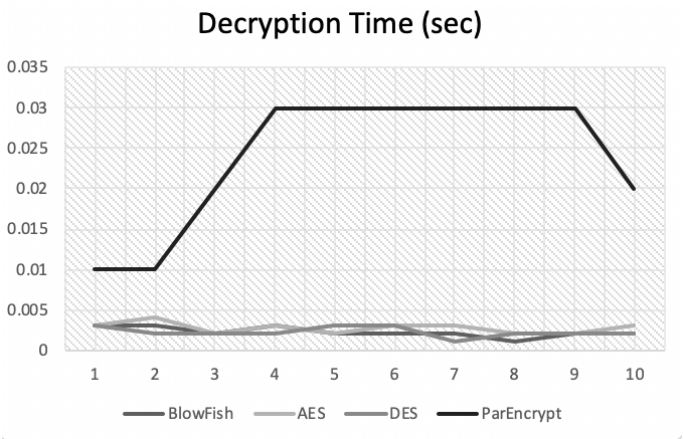

Figure 4. Decryption Runtime of ParEncrypt vs. Existing Schemes

The figures above illustrate the disparities in encryption and decryption time between our encryption scheme and other existing algorithms. As illustrated by the graphs, our encryption scheme performs much slower than Blowfish, AES, and DES, as expected. Our scheme performs a double encryption, whereas the other three simply encrypt once. There is a trade-off between security and performance, and our scheme prioritizes security.

Conclusion

In this work, we propose a novel symmetric encryption scheme that cascades two conventional encryption approaches in two phases: The Feistel cipher and substitution cipher. Our algorithm is simple, yet powerful and outperforms the security provided by conventional encryption schemes that we compare with. The unavoidable trade-off remains still that ParEncrypt performs slower than any one-phase counterpart, as expected. Yet another benefit of ParEncrypt is that it offers a significant level of compression.

As part of future work, we plan to make a comprehensive comparison of ParEncrypt with benchmark encryption tools such as AES and DES to reveal its security level among the others. Additionally, we plan to elaborate ParEncrypt's fringe benefit: lossless compression.Though not perfect, our scheme does provide a level of compression which serves as an added benefit. We will explore ways to improve this level of compression to make it comparable to other tools.

We also plan on running our encryption algorithm on standard corpora from different source languages so as to investigate language specific discrepancies. Moreover, we plan on expanding our implementation by replacing each of the two phases with various alternatives, such as DES or AES for phase one, and transposition cipher for phase 2, etc.

Acknowledgements

The authors would like to thank to students Teena Jagan and Rohit Ravi for their help in the initial design of this project.

References

D. Kahil, M. A. Lebdeh, M. S. E. Dine and A. E. Rafhi, "Innovation of a secured transmitter/ receiver chain by creating a new encryption algorithm," 2017 Sensors Networks Smart and Emerging Technologies (SENSET), Beirut, 2017, pp. 1-3. doi: 10.1109/SENSET.2017.8125037

D. H. Kurniawan and R. Munir, "Double Chaining Algorithm: A secure symmetric-key encryption algorithm," 2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA), George Town, 2016, pp. 1-6. doi: 10.1109/ICAICTA.2016.7803097

Lewi, K., & Wu, D. J. (2016, October). Order-revealing encryption: New constructions, applications, and lower bounds. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1167-1178). ACM.

Zhang, W., Lü, X., & Li, H. (2010). Efficient broadcast encryption scheme based on number theory research unit. Wuhan University Journal of Natural Sciences, 15(3), 247–250. https://doi.org/10.1007/s11859-010-0313-7

Negi, A., & Goyal, A. (2018). Optimizing Fully Homomorphic Encryption Algorithm using RSA and Diffie-Hellman Approach in Cloud Computing.

J. Choi and Y. Kim, "An improved LEA block encryption algorithm to prevent side-channel attack in the IoT system," 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, 2016, pp. 1-4.

Kumar, N. S., Lakshmi, G. R., & Balamurugan, B. (2015). Enhanced attribute based encryption for cloud computing. Procedia Computer Science, 46, 689-696.

M. Xin, "A Mixed Encryption Algorithm Used in Internet of Things Security Transmission System," 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, 2015, pp. 62-65. doi: 10.1109/CyberC.2015.9

"Blowfish," Blowfish on GitHub. Michael Gilfix, Jun-2018. https://gist.github.com/eigenein/a56ce4d572484a582e14

Author Information

Haneya Khan
University of Texas at Dallas
Department of Computer Science
800 W Campbell Rd.
Richardson, TX 75080 USA
Contact E-mail: *hkk150030@utdallas.edu*

Asim Siddiqui
University of Texas at Dallas
Department of Computer Science
800 W Campbell Rd.
Richardson, TX 75080 USA

Fariha Islam
University of Texas at Dallas
Department of Computer Science
800 W Campbell Rd.
Richardson, TX 75080 USA

Elham Hamid
University of Texas at Dallas
Department of Computer Science
800 W Campbell Rd.
Richardson, TX 75080 USA

Ebru Celikel Cankaya
University of Texas at Dallas
Department of Computer Science
800 W Campbell Rd.
Richardson, TX 75080 USA