Simplified DES with Noisy Ciphertext: A Cross-Layer Security Approach

Joseph Garro*, Savannah Rimmele†, and Nghi H. Tran‡
The University of Akron, Akron, OH USA

While security mechanisms, such as encryption, have been traditionally designed and implemented at higher layers of protocol stacks, recently, security schemes at the physical layer (PHY) have been proposed and studied. Nevertheless, despite several recent advances in PHY security, there is still no clear path to connect PHY security to conventional security approaches at higher layers. In this work, we propose a combination approach to investigate how the PHY security and traditional cryptographic methods interact with each other to enhance the security of the system. Specifically, we consider the simplified DES (SDES) encryption algorithms where the ciphertexts received by the adversary eavesdropping the communication channels are error-prone and analyze the impact of error in ciphertexts using simple brute-force and dictionary-based attacks. The obtained results provide important insights into the relationship between the efficiency of these attacks on the considered cryptosystem and the error rate in the ciphertext, which potentially leads to effective joint security mechanisms.

I. Introduction

Although security was originally viewed as a high-layer problem to be solved using cryptographic methods [1, 2], physical layer (PHY) security based on information theory has been gaining increasing research attention [3–5]. PHY security generally refers to exploiting the unique properties of the communication systems, such as noise, interference and fading, to provide secrecy and address security threats. While PHY security is very promising, the security of communication networks has traditionally relied on cryptographic schemes in the application layer or an upper layer. Over the years, while several efforts have been made to design variety of processes jointly in different layers (please see [6, 7]), security schemes in physical and application layers are still designed and executed separately.

In this work, we propose a combination approach to investigate how the PHY security and traditional cryptographic methods interact with each other to enhance the security of the system. Specifically, we will perform cryptanalysis on simplified DES (SDES) encryption algorithms where the ciphertexts received by the adversary eavesdropping the communication channels are error-prone. As a primarily result, we analyze the impact of error in ciphertexts using simple brute-force and dictionary-based attacks. Our numerical results show that there is an exponential increase in the amount of time for the eavesdropper to crack the encrypted message as they must attack each corrected version of the message until a result that they are satisfied with is achieved. The obtained results provide important insights into the relationship between the efficiency of these attacks on the considered cryptosystem and the error rate in the ciphertext, which potentially leads to effective joint security mechanisms [8].

II. Simplified DES (SDES) Encryption and Decryption with Error-Prone Ciphertext: System Description and Testing Methodology

A. SDES with Error-Prone Ciphertext

In this work, we consider a simplified DES (SDES) encryption scheme [2] for demonstration purposes, but the investigation can be extended to other symmetric-key and public-key cryptography. SDES uses a 10-bit key to encrypt an 8-bit block of plaintext data, which provides a key space of 2^{10} possible keys. SDES defines a symmetric block cipher that creates an 8-bit ciphertext from an 8-bit input plaintext utilizing numerous substitutions and permutations. To allow an encryption algorithm to work on input larger than its block size with a single key, a “block cipher mode of operation" needs to be used. Mode of operations specifies a manner to encrypt and decrypt multiple blocks by creating

*5th Year Undergraduate Student, Department of Electrical and Computer Engineering, The University of Akron, Akron, OH 44325
†5th Year Undergraduate Student, Department of Electrical and Computer Engineering, The University of Akron, Akron, OH 44325
‡Professor, Department of Electrical and Computer Engineering, The University of Akron, Akron, OH 44325

a repetitive process that relies on a single key. Blocks encrypted or decrypted later in the process will utilize an input that is included by the encryption or decryption of the previous blocks. The detailed operations are skipped here fore brevity of the presentation.

Now, let we assume that the ciphertext received contains errors due to noise and interference from physical layer channels. With SDES, the error presented in the ciphertext could be up to i bits of a b-byte ciphertext, where $i \leq 8b$. Hence, the ciphertext could have $8b$ choose i possible corrections for i bits of error. As the number of bits in the ciphertext increases, $\binom{8b}{i}$ grows exponentially large, and an attacker will have to consider to increase the number of corrections. As the length of the ciphertext increases, it becomes highly improbable for an attacker to correct the ciphertext and to obtain the plaintext it originally represented. Meanwhile, a savvy party possessing knowledge of the error could easily correct the ciphertext and decrypt it to obtain the correct result.

It should be mentioned that an attacker can decrease the time to brute-force the corrections and keys for the ciphertext by assuming the number of errors in the ciphertext. This assumption allows the $\binom{8b}{i}$ property of the corrections to be exploited by interpreting the corrections as bit strings of length $8b$, where each bit with a value of '1' corresponds to a bit of error i. Assuming n bits of error, all bit strings of length $8b$ bits containing n '1's could be created to satisfy all n bits of error. These values could be saved into a dictionary that will supply the corrections that are to be applied to the ciphertext instead of trying all possible corrections. This will exponentially decrease the size of the correction space from 2^{8b} corrections to $\binom{8b}{i}$ corrections, which should correspond to a decrease in the amount of time required to try the desired corrections. The time required to create the dictionaries can be omitted as any dictionary of length b correcting up to n errors will be valid for any ciphertext of length b containing up to n errors. Hence, an attacker could possess many pre-generated dictionaries that they can use when necessary.

B. Testing Methodology

In the following, we shall describe the testing method we perform to determine the efficiency of brute-force and dictionary attacks on noisy SDES. To quantify the performance of each test, the time elapsed is recorded with the $\text{perf}_{\text{counter}}$ timer found within the time module. The chosen timer is system-wide and returns a value of the performance counter in seconds, including the time elapsed during sleep, which is represented in logarithmic scale. To further quantify the performance of each test, all tests were run on four different computers containing different hardware to observe differences in performance. The specifications of all four system are as follows:

- Microsoft Surface Pro 2017 with an Intel(R) Core i7-7660U CPU 2.50GHz with 2 Physical Cores
- Dell Precision T1700 with an Intel(R) Core i7-4790 CPU 3.60GHz with 4 Physical Cores
- A Custom PC with an Intel(R) Core™ i7-9700k CPU 4.90GHz with 8 Physical Cores
- Dell Precision 5820 Tower X-Series with an Intel(R) Core™ i9-10900x CPU 3.70GHz with 10 Physical Cores (20 Virtual Cores)

Each system has the test measuring the time to decrypt ciphertext with no error ran on it for ciphertexts with a length of 1 byte, 2 bytes, 3 bytes, and 4 bytes. When measuring the performance of both attacks, each system was originally going to test the decryption of up to 4 bytes of ciphertext containing up to 8 bits of error. It was observed that all systems would not be able to brute force the 3-byte and 4-byte plaintexts within a reasonable amount of time, so only 1-byte and 2-byte plaintexts were measured. When brute forcing the 3- and 4-byte plaintexts with 7 and 8 bits of errors using the dictionary attack, the performance of the systems varied. To compare the performance between all systems, the 3-byte plaintext test is set to have a maximum of 6 bits of error, the 4-byte plaintext has a maximum of 5 bits of error, and the 1- and 2-byte plaintexts have up to 8 bits of error.

For each system, the plaintexts, keys, IVs, and number of errors used for each test were kept constant. The actual error values are random, resulting in each test being run for a potentially unique ciphertext containing error. However, the tests are independent of the placement of error, allowing the error value to vary if the number of errors is constant. Error is added to each ciphertext by using a pseudo-random number generator seeded with the current system time to generate a number representing how many times to left shift the value '1'. The left shifted values are stored in a set to prevent duplicate errors from canceling each other out and are later split into bytes and exclusive OR'd with the bytes composing the original ciphertext. Despite not applying the error in the optimal manner described previously, this method ensures that the error is randomly spread throughout the ciphertext. This is acceptable as this research aims to answer the impact of the error when attacking ciphertext from a computational perspective rather than analyzing the resulting plaintexts.

III. Numerical Examples

In the following, several examples are provided to demonstrate the impact of errors on SDES.

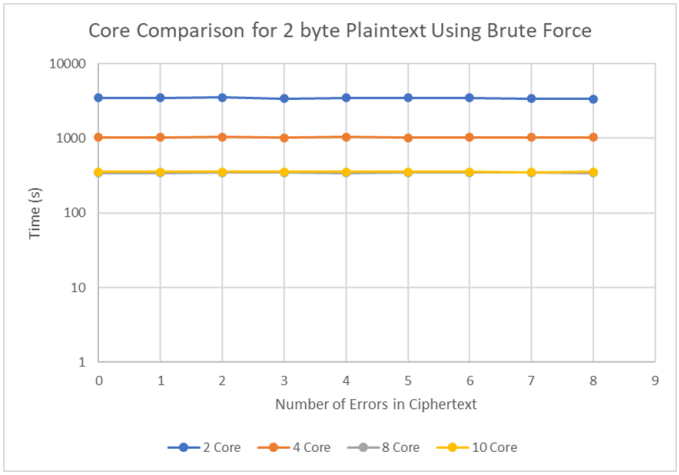

Fig. 1 Time Comparison on Each System for the 2-Byte Plaintext Using Brute Force.

Fig. 1 shows the time to brute-force a ciphertext containing an error by applying all possible corrections to it, independent of the number of errors. Instead, the time required depends on the number of cores that each system has, where the systems with more cores can divide the work among the cores, reducing the total work each core must do. The overall time required per system is largely the same, with the higher-core count systems being able to brute force the ciphertext much faster than the lower-core count systems.

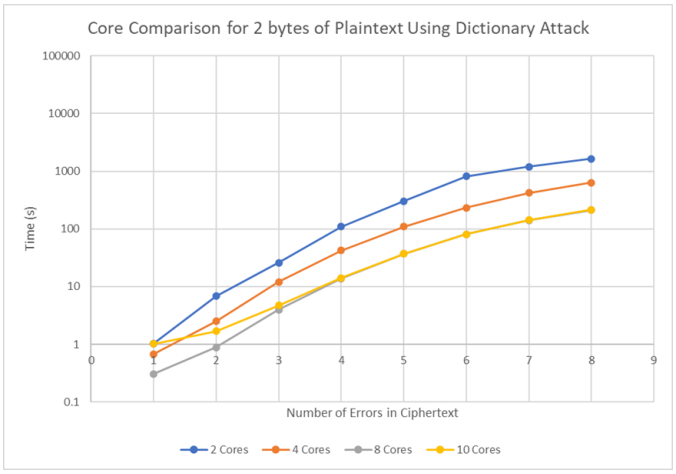

Fig. 2 Time Comparison on Each System for the 2-Byte Plaintext Using Dictionary Attack.

Fig. 2 compares the performance of each of the systems specified in Section II.B when using a dictionary to correct ciphertexts with a known number of errors. Since the dictionary attack is to be used when an error is assumed, there is no dataset when no error occurs. Each dictionary can handle the zero error case as the first value in every dictionary is 0, but the test will only complete once every correction in the dictionary has been tested. Hence, the time to complete a test with zero error is going to be dependent on the size of the dictionary used.

For all ciphertexts, it is observed that the use of a dictionary reduces the amount of time to brute-force the ciphertext. This result is expected, as the dictionary limits the correction space for each ciphertext, reducing the amount of work required. As the length of the ciphertext increases, the dictionary attack required exponentially more time to complete. As expected, the size of the dictionary grows in response to the number of assumed errors in the ciphertext and its length, following the n choose k property previously explained. Notice that the time required by the dictionary attack

begins to approach the corresponding brute-force time for each system as the assumed number of errors increases, as predicted. This occurs since the correction space defined by the dictionary is bound by the brute-force correction space and its size grows towards it as the number of errors increases.

References

[1] Marić, I., Shamai, S., and Simeone, O., *Information Theoretic Perspectives on 5G Systems and Beyond*, Cambridge University Press, 2022.

[2] Stallings, W., *Cryptography and Network Security: Principles and Practice*, 6th ed., Prentice Hall Press, USA, 2013.

[3] Zhou, H., and El Gamal, A., "Network Information Theoretic Security With Omnipresent Eavesdropping," *IEEE Transactions on Information Theory*, Vol. 67, No. 12, 2021, pp. 8280–8299.

[4] Li, G., Luo, H., Yu, J., Hu, A., and Wang, J., "Information-Theoretic Secure Key Sharing for Wide-Area Mobile Applications," *IEEE Wireless Communications*, 2023.

[5] Zhou, H., and El Gamal, A., "Network information theoretic security," *2020 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2020, pp. 978–983.

[6] Zhou, L., Wu, D., Zheng, B., and Guizani, M., "Joint Physical-Application Layer Security for Wireless Multimedia Delivery," *IEEE Commun. Mag.*, Vol. 52, No. 3, 2014, pp. 66–72.

[7] Wang, J., Huang, P., Wang, X., and Yang, Y., "Cross-Layer Scheduling for Physical Layer Secrecy and Queue Stability in a Multi-User System," *Globecom 2013 - Wireless Networking Symposium*, 2013.

[8] Sadig, T., Maleki, M., Tran, N. H., and Bahrami, H. R., "An Encryption-Aware PHY Security Framework for 4-Node Gaussian Wiretap Channels With Joint Power Constraint," *IEEE Transactions on Communications*, Vol. 68, No. 12, 2020, pp. 7837–7850. https://doi.org/10.1109/TCOMM.2020.3024825.