Dual-Method Side-Channel DES Key Recovery Using Timestamp and Machine Learning

K.M.C.A. Nimsara
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
it21285974@my.sliit.lk

P.C.K. Piyathilaka
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
it21195716@my.sliit.lk

J.A.R.A.K. Perera
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
it211992266@my.sliit.lk

T.P. Wijesinghe
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
it21325328@my.sliit.lk

K. Y. Abeywardena
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
kavinga.y@sliit.lk

Amila Senarathne
Faculty of computing
Sri Lanka Institute of Information) Technology (SLIIT)
Malabe, Sri Lanka
amila.n@sliit.lk

***Abstract*—This research presents two practical approaches to retrieve DES encryption keys through timing-based side-channel analysis. The first approach takes advantage of keys derived from determinable Unix timestamps and demonstrates that attackers can generate plausible keys and determine close matches based on similarity measures. The second approach employs a Long Short-Term Memory (LSTM) neural network to predict DES key bits based on features such as encryption time and Hamming distance. Both methods were implemented on a Raspberry Pi and shown to be viable in the real world—timestamp analysis cut down the key search space and the LSTM model could predict the majority of key bits. The findings demonstrate that weak key generation and timing leakage can be disastrous for cryptographic security even in embedded or lightweight environments.**

Keywords— Side Channel Attack, Timing Analysis, DES, LSTM, Cryptography, Raspberry Pi, Key Prediction, Timestamp Vulnerability, Machine Learning, Information Security

I. INTRODUCTION

Symmetric algorithms like the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES) are commonly applied in an effort to protect information in various digital applications, including secure communication and financial networks. The algorithms achieve protection through hiding information and integrity using secret keys [1]. However, as the methods of attack advance, traditional cryptographic defenses are increasingly being undermined by side-channel attacks (SCAs), and timing-based analysis is a particularly feasible and potent threat vector [2].

A timing side-channel attack takes advantage of differences in the amount of time a system requires to carry out cryptographic processes. By measuring the execution times accurately, attackers are able to deduce secret information like secret keys or internal computation states [3]. Although classical timing attacks tend to suffer from limitations for example being static-dataset-dependent, or not being integrated with other sources of leakage like power consumption and they remain a pragmatic reality, particularly in resource-limited environments [4]. Growing complexity and sophistication of cryptographic protocols call for more sophisticated, comprehensive tools that can perform multi-dimensional side-channel analysis.

As a reaction to this challenge, this project suggests creating a portable hardware platform that is explicitly intended to carry out time-based side-channel attacks on symmetric encryption algorithms. The device accommodates several timing analysis methods. First, it uses a time-critical data set and removes probable keys based on Euclidean distance measurements. Second, it uses deterministic timestamp-based key generation vulnerability modeling. Third, it uses machine learning in the form of Long Short-Term Memory (LSTM) neural networks to predict DES key bits based on features like encryption run time, Hamming weight, and Hamming distance. By combining these methods, the suggested system provides an end-to-end system for examining the cryptographic strength of symmetric keys and determining exploitable weaknesses in practice.

This research fills a significant research void wherein, even after Kocher's seminal contribution in 1996 on timing attacks [5], real, easily accessible, and recent implementations particularly on low-cost devices such as Raspberry Pi are limited. A lot of SCA research is focused on power and electromagnetic (EM) leakages [6], which are bound to be costly equipment and require technical expertise and thus are not readily accessible to new researchers or students. In addition, although machine learning solutions for power-based SCAs have received some attention [7], there has been fewer efforts on using sequence models such as LSTM on timing data, which is noisier and less deterministic.

By creating a low-cost, reproducible, and instructional hardware setup, this project closes the gap between theoretical knowledge and hands-on realization of timing-based side-channel attacks on DES. It offers an ML-driven real-time demonstration platform, giving useful insights to students, security practitioners, and researchers in cryptographic implementation security.

II. BACKGROUND AND RELATED WORK

Side-channel attacks (SCAs) are an essential threat to cryptographic systems as they target the physical or observable behavior of their implementations, as compared to trying to breach the fundamental mathematical algorithms. Of these SCAs, timing attacks stand out due to the invasive free aspect and capacity for remote execution. Timing attacks utilize differences in time taken for cryptographic processing to draw out sensitive data, such as secret keys. It was first

suggested by Paul C. Kocher in his groundbreaking research where he demonstrated that tiny differences in timing when performing modular exponentiation would disclose private keys of systems like RSA and Diffie-Hellman [5]. Kocher's finding initiated an outburst of studies on the topic of timing-based attacks for public-key and symmetric key schemes.

Subsequent developments demonstrated how timing attacks could be mounted against well-known block ciphers like DES and AES. Boneh and Brumley's work in 2003 demonstrated the practicability of such attacks by demonstrating a remote timing attack against OpenSSL's implementation of RSA [8]. Such findings reaffirmed that practical software systems in the real world were able to leak secrets based on timing differences in functions with key dependent control flow or memory access. Although initial activity centered on public-key algorithms, interest soon expanded to symmetric ciphers like DES where key and plaintext Hamming distance had an influence on running times in key scheduling and S-box usage.

Variations in DES occur because of bit-level operations and their interference with CPU caching and memory access latency and hence cause tiny delays in execution. Classical statistical methods were flummoxed by the noisy and minuscule nature of these timing variations. Yet machine learning and deep learning strategies provided new avenues for leveraging such leakages. Early machine learning approaches like decision trees, SVMs, and KNN worked well under clean trace environments [7]. Deep learning more recently in the form of Convolutional Neural Networks (CNNs) and Long Short Term Memory (LSTM) networks has been successful. CNNs work well with structured signals like power traces while LSTMs are better suited for time ordered data like timing sequences.

Zaid et al. made a large contribution to this field with the ASCAD dataset and first CNN-based profiling attacks on AES with better outcomes than traditional ML methods [9]. The majority of these works. However were performed on power side-channel data obtained through costly hardware implementations which restricted accessibility and reproducibility. Timing based SCAs and more specifically symmetric algorithm targets such as DES, on the other hand, are still relatively unexplored fields of research especially in noisy timing measurement real-world, embedded environments.

Concurrent with the above has been the longstanding criticism of deterministic key generation based on predictable like system times. Seeding random number generators (RNGs) with such predictable data makes them susceptible to key recovery if the time of generation is roughly known. Yilek et al. uncovered one such vulnerability in the Debian OpenSSL implementation wherein compromised entropy led to large-scale key duplication [10]. Biryukov et al. also examined RNG weaknesses for TLS keys with an emphasis on the general ramifications of low-entropy sources [11]. Yet the majority of this work deals with public-key cryptosystems while this paper applies the timestamp-based key recovery approach to symmetric systems DES filling a significant research gap.

This project combines these two potent attack techniques timestamp-based key recovery and deep learning-based timing analysis and implements them together in one modular framework on a Raspberry Pi. The Raspberry Pi was chosen because it is low-cost and widely used in education and embedded systems research. By capturing actual encryption timings, Hamming-based feature extraction and training an LSTM network for key prediction the platform offers a practical demonstration of timing based SCAs on DES. In contrast to other instructional or research platforms that are theoretical or restricted to power SCAs and AES this system illustrates real-time, noise immune side-channel analysis of symmetric ciphers with inexpensive hardware. The system is both a proof-of-concept and an open instructional platform for contemporary cryptographic vulnerabilities.

III. Research Objectives

The main objective of this research is to investigate the possibility of using time-based side-channel information to predict or retrieve DES encryption keys using two novel methods: (1) timestamp-based deterministic key generation analysis and (2) deep learning-based key bit prediction using timing information. By implementing the methods on a Raspberry Pi, this research intends to close the gap between theoretical side-channel weaknesses and real-world, practical exploitation particularly in educational, research and lightweight cryptographic settings.

This research tackles the frequently neglected vulnerability of timing leak in symmetric cryptographic protocols that is gaining visibility given the prevalence of using insecure or legacy implementations in IoT and embedded systems. Through experimental presentations, the project illustrates how low entropy and timing differences can be exploited by attackers to obtain sensitive information.

The core objectives are as follows:

1. To simulate timestamp-based DES key generation, showing that deterministic or poorly seeded PRNGs can produce predictable keys.
2. To design a comparison engine that evaluates user-supplied keys against timestamp-generated keys using byte-level similarity metrics.
3. To collect timing data across encryption operations and generate a feature-rich dataset for training.
4. To develop and train an LSTM model to predict DES key bits from timing data with high accuracy.
5. To integrate both methods into a single, portable Raspberry Pi-based platform with real-time input, analysis, and output.
6. To validate system performance under resource constraints and ensure ease of use.
7. To ensure the system is modular, lightweight, and extensible, supporting future applications in teaching and research for other ciphers like AES.

IV. Research Methodology

The research is based on a multi-stage approach to examining weaknesses in cryptographic key generation, assessing predictive capability of machine learning models

for predicting encryption keys, and showing how these systems are implemented on an embedded hardware platform in real time. The approach is broken down into three phases: (1) research of timestamp-based cryptographic key generation and its weaknesses, (2) development of a controlled dataset to train machine learning algorithms to predict keys, and (3) hardware implementation to provide a standalone tool for real-time cryptographic analysis.

A. Timestamp-Based Cryptographic Key Generation and Analysis

Phase I of the methodology deals with the design of a cryptographic key generation system that relies on Unix timestamps as a source of randomness seed. This deterministic approach presents very serious vulnerabilities regarding timestamp predictability, which can be used by attackers to predict cryptographic keys.

1) System Design for Key Generation

A significant generation system in Python was developed, utilizing the random module available in Python's standard library. Central to the design is the seeding of the random number generator using Unix timestamps. The system accepts user-provided timestamps or system-generated time values as input. The randomness generated using this method is deterministic in nature, such that the same timestamps will always result in the same stream of random numbers. Such predictability greatly undermines the security of the keys that are generated.

The key generation process involves iteration over all seconds in a given range of timestamps. At every timestamp, an 8-byte (64-bit) key is generated. The *random.seed()* function is initialized with the Unix timestamp, and then the *random.randint(0, 255)* function is called eight times and with each integer representing one byte of the key. The individual bytes are then converted to a hexadecimal string, thus forming the final cryptographic key.

2) Vulnerability Analysis and Similarity Assessment

After the generation of the cryptographic keys, the next stage entails the analysis of the potential weaknesses of this timestamp-based method. The main problem here is the level of similarity between the generated keys and a user-supplied secret key since a high level of similarity increases the likelihood that an attacker can correctly infer the correct key. For measuring similarity, there were two approaches utilized: character-level similarity and binary-level similarity. While in character-level analysis, for each byte of the generated key, it is matched with the equivalent byte of the user key, hence the percentage of equivalent bytes is determined. This approach provides a direct measure of how alike the structural forms of both keys are.

For more detailed analysis, binary-level similarity is established by both the generated key and the user key being converted to binary strings of 64 bits each. The Euclidean distance between the binary strings is calculated and the raw distance is normalized against the maximum possible 64-bit key distance to return a percentage similarity score. This technique allows for a more precise measurement of how closely the keys generated are to the user key, recognizing any minor but necessary patterns that may exist.

A similarity threshold (65%) was established to report potentially vulnerable keys. Generated keys with a similarity higher than this threshold were flagged as vulnerable, representing a smaller attack search space. The result of this step was graphically illustrated by a bar chart of the top 20 generated keys with the highest similarity values to the user key. The visualization aids in the comprehension of how risky timestamp-based key generation is when realized in actual use.

B. Machine Learning-Based Key Prediction

Second phase deflects attention to machine learning and more precisely, predicting DES key bits based on side-channel features like encryption time, Hamming measurements and binary-encoded input strings. The research sought to examine the possibility of predicting encryption key bits via the exploitation of features such as timing information, bit-level structure and Hamming distance.

1) Data Collection and Feature Engineering

The initial step in this case was to set up a controlled experiment environment to gather encryption data. To do this, a virtual machine of Windows 10 with a Ryzen 5 processor, 8GB of RAM and an NVIDIA GTX 1660TI GPU were utilized. A Python script using the *PyCryptodome* library was used to carry out DES encryption. Both encryptions had the same plaintext message and manually input 8-byte hexadecimal key so that both encryption procedures would be identical.

Timing information was gathered by repeating the encryption process many times each using a different key and plaintext and averaging the times to reduce noise and transient idiosyncrasies. This yielded a dataset with three relevant features for every encryption occurrence: the plaintext, the encryption key and the average encryption time. Along with the raw timing information and several other features were also pulled out of the plaintext and key. These include the Hamming weight of the plaintext and key which is the count of 1s in each byte, and the Hamming distance between the plaintext and key which is the bitwise difference. These features reflect the inherent complexity and structural patterns of the data which can be essential in unlocking encryption behavior.

2) Model Development and Training

The machine learning solution involved using a long short term memory (LSTM) network, a recurrent neural network (RNN) structure. Which is well-suited to modeling sequential dependencies. This is important because the encoding phase and side channel information contain temporal patterns that the model must learn.

To preprocess the data for training the model, all features were scaled by *StandardScaler* so that they were approximately the same scale. The data was padded to the same sequence length and an additional special padding value of -1 was used for padded locations. This was necessary

because an LSTM model requires a fixed-length input sequence. The final model architecture uses a masking layer to exclude padded values form influence model training. A single LSTM layer with 128 units processed sequential dependencies in the data and followed by a dense layer with sigmoid activation for bitwise binary classification. This model was optimized by using Ada, over 50 epochs with binary cross entropy as the loss function and a batch size 32.

3) Model Evaluation and Testing

After training the model experimented on an independent test set in order to estimate its performance when predicting the most important bits of the key. The performance of the model was evaluated by examining the predicted key bits against the actual key bits. The output indicated whether or not the model was able to correctly infer the key bits from the given side-channel features and thereby prove that machine learning may be used for predicting cryptographic keys from side-channels.

C. Hardware Integration for Real-Time Cryptographic Analysis

The last challenge in the research is to merge the timestamp based key generation and machine learning based key prediction subsystems into a real time hardware platform. Raspberry Pi 4 Model B was used here because it is cheap, compact-sized and Python-based machine learning libraries are supported. The aim was to develop a portable and self-sustained device that can generate cryptographic keys and predict them without using a complete computer system.

1) Hardware Setup and System Deployment

Raspberry Pi was configured with libraries required like *Python 3, NumPy, pandas, Keras or PyTorch* for machine learning, and other auxiliary libraries such as datetime and random to create keys based on timestamps. Setup was put in place to create keys based on timestamps and predict DES key bits from the trained LSTM model. Raspberry Pi was also connected to a 16x2 LCD display through GPIO pins to display live results of cryptographic analysis.

Remote access was supported by the Raspberry Pi Connect utility which allowed users to remotely control the device without the need for any external peripherals. This supported effective testing and experimentation with both the key generation and prediction process, allowing easy changes and real-time model performance tracking.

2) Real-Time Cryptographic Evaluation

The LCD screen showed the closest generated DES key to a user provided key in Hamming distance and the predicted key bits from the LSTM model. This integration showed the possibility of carrying out cryptographic analysis on a low-profile, low-cost hardware platform directly. The system was converted into an active, autonomous tool that could conduct real-time cryptographic key generation, prediction and side-channel analysis and serve as a goldmine for researchers, security experts, and instructors.

By integrating both the machine learning based prediction models and timestamp based key generation within a single device. This stage of the research gives a real-case scenario of the testing of cryptographic systems in real world applications. The portability of the system is perfect for use in constrained environments or educational purposes and shows the feasibility of machine learning in cryptography and the necessity of best practices in secure key generation.

V. Results and Disscusion

A. Timestamp-Based Key Generation and Analysis

The first approach showed how DES keys generated from timestamps of a system are predictably deterministic in figure 1. If a range of times is given by the user, a pseudo-random number generator that is seeded with Unix-style timestamps will always generate the same 64-bit DES key for a specific timestamp. This was illustrated with graphical plots that verify the predictability of such keys. This deterministic result is a powerful cryptographic weakness: if an attacker is able to guess or deduce the rough time interval in which a key was created, the keyspace can be significantly narrowed, and brute-force key recovery becomes very practical.

Figure 1- DES keys generated using timestamp-based seeding

Subsequent analysis was to cross-compare the generated keys with a known target key utilizing byte-level and bit-level measures of similarity. A top 20 rank of similar keys was created, illustrating how easily the set of potential keys could be narrowed when the window of timestamps is exact in figure 2.

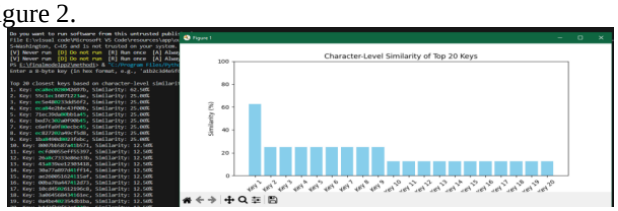

Figure 2- Top 20 timestamp-generated keys closest to the target

This illustrates an actual side-channel vulnerability utilizing simple statistical measures of similarity, it's possible to significantly narrow the key search space. In addition, the research used Euclidean distance in the binary space to determine the nearest key at the bit level in figure 3. This was more sensitive than comparisons at the byte level and found minor structural similarities which would not be evident in hexadecimal representation. These partial approximations of the key are alarming in systems with poor authentication needs or weak against replay and injection attacks.

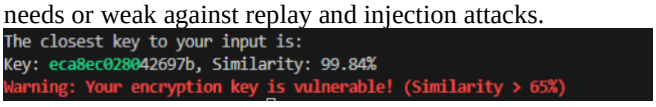

Figure 3-Closest regenerated key based on bitwise distance)

B. LSTM-Based DES Key Bit Prediction

The second approach involved the use of a Long Short-Term Memory (LSTM) neural network to predict DES key

bits from side-channel features. A properly formatted dataset was generated by repetitive encryption of plaintexts using varying keys. Encryption time, Hamming weight, and Hamming distance were calculated for every encryption process. The dataset was used as the input for training and testing the LSTM model under the setup of a simulated environment.

Training outcomes demonstrated promising generalization accuracy. The model was converging within over 50 epochs with negligible indications of overfitting and test accuracy of around 96% for predicting single key bits as seen in figure 4. This was far better compared to the baseline random forest model which was only able to reach 56% accuracy. Additional exploration involved examining the effect of single input features on model performance. Results showed that encryption time alone possesses predictive ability but that its ability is greatly enhanced using Hamming measures. This serves to highlight the necessity of using more than a single side-channel dimension in order to effectively make cryptographic inference.

Figure 4- LSTM model accuracy and loss over 50 training epochs

A graphical representation of a model output confirmed the success of the LSTM approach. In a test prediction the model correctly predicted 47 of 64 bits which73.44% accuracy as shown in figure 5. Which is much greater than random guessing. The predicted bits were superimposed over the real key with correct bits in green and incorrect bits in red and clearly showing the model's predictive power. This finding highlights the real danger of even partial key recovery from timing data in compromised systems.

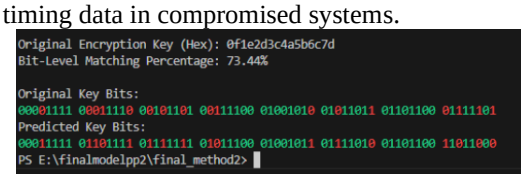

Figure 5-Bitwise comparison of predicted and actual DES key

While the technical prowess of the LSTM model in recovering most of the DES key bits is impressive, the security implications run much deeper. Recovering most, if not all, of the key would greatly reduce the work for a brute-force attack. For example, if the model can guess most of the bits and only a very small subset is unknown, what is left of the key space can be easily searched with current computing power. In practice, the computational complexity of finding the key is limited to a number of affordable possibilities, implying that full key recovery is very feasible. Even partial key recovery can help attackers launch more sophisticated cryptographic attacks, especially in systems with reused keys or additional side-channel information. This serves to emphasize how serious a threat time leak is to the confidentiality of encryption algorithms like DES when paired with machine learning.

C. Integrated Findings and Practical Implications

The joint findings from both methods strongly validate the hypothesis that time-based side channel information can lead to severe security vulnerabilities for symmetric key cryptosystems. Timestamp based key derivation, although low overhead and simple introduces deterministic structure that can be leveraged by adversaries. The feasibility of key regeneration and comparison within a small-time window is a realistic threat model for legacy systems and embedded devices using low-entropy sources like system clocks.

The LSTM inference model takes this threat scenario further by showing that software-based side-channel data without invasive hardware manipulation can be used to predict significant parts of cryptographic keys. When running on energy-limited hardware like the Raspberry Pi 4, both methods continued to be feasible and effective, showing that sophisticated side-channel attacks are now no longer the domain of high-end laboratory environments. This find has significant ramifications for the cybersecurity world as it demonstrates that cryptographic attacks are progressively becoming accessible for students and amateur attackers alike.

D. Educational and Security Development Impact

From the educational perspective, this book connects theoretical cryptanalysis with real, provable attacks. The researchers and students can directly see how small implementation flaws such as the use of system time for key generation or non-constant time encryption functions can have catastrophic implications. For developers and designers of systems, the findings reaffirm the need for following secure key generation practices and side-channel-secure cryptographic implementations. These involve using true random number generators and constant-time execution to prevent timing-based anomalies.

E. Limitations and Ethical Considerations

Though successful, the research did have several limitations. Foremost among these was the requirement of a controlled setup for the training of the LSTM. The timing data from the encryption were gathered in a virtualized environment under minimal interference, something that is not so easily achieved in actual implementations where hardware heterogeneity and multitasking are present. The performance of the LSTM model could suffer in unstructured or noisy environments. Likewise, the success of the timestamp-based attack depends on precise knowledge of the key generation window. In the absence of that knowledge, the brute-force space is unrealistically huge. On the ethical side the dual nature of these tools requires their responsible use. Although the research was carried out within ethical confines the methods outlined can be abused if used with malicious intent. This calls for ethical education in addition to technical schooling and open security policies in cryptographic development.

VI. Conclusion and future work

This research shows that time-side-channel attacks against symmetric block ciphers such as DES are not merely theoretically feasible, but also practically viable using low-cost, off-the-shelf hardware. Two complementary methods were realized: a timestamped deterministic key generation attack and an LSTM-based machine learning key bit prediction approach using timing features. Both were shown to function in a Raspberry Pi 4 Model B, and hence real-time key recovery is feasible even for resource-constrained settings.

The timestamp-based approach took advantage of predictable randomness in key generation, wherein Unix timestamps were used as seeds for PRNG. In scenarios of simulated real-world attacks where an attacker knows or guesses the time of key generation, the system was able to regenerate candidate keys and rank them based on bitwise and Euclidean similarity measures. This attack demonstrates the risk of utilizing weak sources of entropy, which significantly lower the search space for keys and makes brute-force recovery easier. The second approach utilized a side-channel-trained deep learning model to predict DES key bits from side-channel information of encryption time, Hamming weight, and Hamming distance. The LSTM network learned the temporal patterns extremely well and could recover a large percentage of the encryption key. In practice, even partial key recovery reduces the brute-force effort to guess the entire key significantly. This demonstrates that even quite modest timing leaks, in combination with machine learning, may be able to seriously undermine cryptographic confidentiality without exposure to internal system states.

The broader implication of this research is its accessibility. In contrast to side-channel attacks in the past based on high-cost laboratory devices but our platform demonstrates how advanced cryptographic attacks can be staged by anyone with modest programming skills and low-cost hardware. This reduces the barrier of entry for prospective attackers and enabling students, enthusiasts or attackers to actually exploit these vulnerabilities. Consequently, systems that depend on insecure randomness or do not have constant-time execution are more vulnerable than ever. Educationally, the work is also a convenient hands-on manual. The platform serves as a bridge between abstract cryptanalysis and practical exploitation, hence an excellent teaching tool in education on cybersecurity. The modular nature provides the potential for additional experimentation, visualization and exploration of attack vectors and defense mechanisms.

Future work can make this platform more robust by giving an analysis on other ciphers such as AES or light-weight cryptography protocols commonly used in IoT devices. Noisy real-world data with variability across a range of different hardware platforms can make the machine learning models more robust and generic in nature. Additional side-channel features such as cache usage patterns or timing information can be incorporated in order to give multimodal attacks. On the defensive side, the system can also be employed to verify countermeasures like noise injection, random delay, masking method or secure pseudorandom number generators (PRNGs). Finally, this paper recalls an important lesson in contemporary cryptography: implementation faults can be as harmful as poor algorithms, and security needs to be imposed on all levels from algorithm development to actual deployment.

Acknowledgment

We would like to thank our supervisor, Mr. Kavinga Yapa, most cordially for his valuable guidance, encouragement, and support throughout this research. His expertise and everlasting motivation were invaluable at all times. We thank our co-supervisor, Mr. Amila Senaratne, for his valuable remarks and technical guidance, which served towards converting concepts to viable outcomes. We are grateful to academic and technical staff for providing a good learning environment. Finally, we thank our peers and colleagues for their relentless support and friendship. This project would not have been possible without the support and input of all these individuals.

References

[1] R. MacDonald, "What Is Symmetric Encryption, How Does It Work & Why Use It," 1kosmos, 14 August 2023. [Online]. Available: https://www.1kosmos.com/blockchain/symmetric-encryption/.

[2] Z. B. A. A. S. Z. Sri Harsha Mekala, "Cybersecurity for Industrial IoT (IIoT): Threats, countermeasures, challenges and future directions," sciencedirect, 2023.

[3] geeksforgeek, "What is a Side-Channel Attack? How it Works," geeksforgeek, 21 June 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-a-sidechannel/.

[4] A. S. E. T. Dag Arne Osvik, ""Cache Attacks and Countermeasures: the Case of AES," Weizmann Institute of Science, 2005.

[5] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," paulkocher, 1996.

[6] E. O. T. P. Stefan Mangard, "Power Analysis Attacks: Revealing the Secrets of Smart Cards," iacr, 2007.

[7] R. P. O. M. F.-X. S. Liran Lerman, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis," lerman, 2018.

[8] D. B. David Brumley, "Remote Timing Attacks are Practical," ePrinter Archive, 2011.

[9] J. W. X. T. S. Z. Z. L. B. Y. Hai Huang, "Deep learning-based improved side-channel attacks using data denoising and feature fusion," plos one, 2025.

[10] E. R. H. S. B. E. S. S. Scott Yilek, "When private keys are public: Results from the 2008 Debian OpenSSL vulnerability," hovav, 2008.

[11] J. W. B. S. P. V. V. Alex Biryukov, "Random number generators: Pitfalls and improvements," IACR ePrint Archive, 2020.