ASCON80-pq Implementation Strategies

Vaikunth Jatinkumar Patel
Dept. of Systems & Computer Engineering
Carleton University
Ottawa, Canada
vaikunthpatel@cmail.carleton.ca

Yash Rajan Akruwala
Dept. of Systems & Computer Engineering
Carleton University
Ottawa, Canada
yashrajanakruwala@cmail.carleton.ca

Devang Hasmukhbhai Vasani
Dept. of Systems & Computer Engineering
Carleton University
Ottawa, Canada
devanghasmukhbhaivas@cmail.carleton.ca

***Abstract*— There is a very rapid increase in demand of the development of secure, low power cryptographic based solutions for protecting devices. There are families of various authenticated encryption and hashing algorithms and among those families, one such family is ASCON, a light weight and easy algorithm to implement which provides counter measures against side channel attacks. And these algorithms are being used to transmit messages between devices requiring low power consumption. ASCON was also one of the finalists in the National Institute of Standards and Technology Lightweight Cryptography Competition. It provides various features such as encryption of authenticated messages, hashing functionalities both in hardware as well as software, permutation techniques, error handling by using less computational power as well as memory. The problem is to reduce the overall power consumption during the encryption as well as the decryption process so that time and space complexity can be reduce by modifying one of the inline functions either during the encryption, decryption, or permutations. And the proposed methodology shows that there is a decrease of 31.31 percentage in overall power consumption during encryption process and a decrease of 24.31 percent during decryption process.**

Keywords— Hashing Algorithms, ASCON, Lightweight Cryptography

I. Introduction

There is a very huge increase in demand of devices such as IoT devices and these devices operates on low power and there are around 14.3 billion active connections in 2022 and it is estimated that it will increase to 16.7 billion by the end of 2023 according to [1]. So, there will be transmission of sensitive data on the internet by these devices. So, it is essential to secure these devices by applying efficient algorithms to reduce potential security threats.

The main threat to these devices is from side channel attacks. Side channel attacks are those which can break secure cryptographic algorithms and may lead to the leakage of information which seems easy to get by implementing the fault attack in the algorithms. And to secure the system from such attacks, light weight cryptography was introduced to provide security and privacy to the systems in the constrained environments i.e., operating in a limited resources environment and one such family that works on lightweight cryptography is ASCON and it works on low computational complexity. It is designed in such a way that it provides authenticated encryption with associated data (AEAD) and it also provides hashing functionalities. It uses 320-bit permutation techniques, bit-sliced into five 64-bit register words and provides 128-bit level security as mentioned in [2].

The ASCON family includes three Authenticated Encryption techniques namely ASCON-128, ASCON-128a, ASCON- 80pq. These modes use key initialization and finalization functions, and they use 12-modes permutation technique. Ascon-128 and Ascon-80pq uses 6-round permutations while Ascon-128a uses an 8-round permutation. The state size of an ASCON family is 320 bits while the key size of Ascon-128 and ASCON-128a is 128 bits while the key size of ASCON- 80pq is 160 bits [3].

The paper focuses on compiling the code for the ASCON-80pq from one of the libraries available online and running it on the ChipWhisperer Nano board using jupyter notebook and then getting the power traces for randomly generated 100 plain texts and calculating the total power consumption during the encryption as well as the decryption process.

Our focus was to optimize the inline functions in the file so that the total power consumption during the encryption as well as the decryption process decreases so that more efficient solution can be developed so that time complexity as well as the space complexity can be reduced, and more optimal solution can be obtained.

The rest of the paper is organized as follows. Section II will focus on the related work already being implemented in this algorithm. In section II there is a mention on working of ASCON algorithm. Section IV will mainly revolve around the proposed methodology. In Section V, we will show the results of changing the inner functions so that better solution can be obtained. And lastly, Section VI will focus on the conclusions and what future work can be done in this field so that the total power consumption can be reduced further.

II. Literature Review

There are various approaches mentioned on how to secure the device using ASCON's algorithm. In [1], the novel architecture of ASCON is being used and the operation i.e., Initialization, Encryption/Decryption Processes, Plain to Cipher and Cipher to Plain text conversion and finalization is being divided into three steps: Pre - permutation, Permutation and Post-Permutation. All these steps are being merged into single module which can handle both encryption and decryption modes. While in [2], the security is being compromised using Fault technique, a powerful Side Channel Analysis technique which is being used to reveal secret in- formation by inducing faults into the cryptographic algorithm. And to tackle these, a key-dividing strategy using double fault is being introduced to fetch the entire secret key.

And it is a kind of trade-off between size of key search space and a total number of double-fault injection experiments.

There's a different application of ASCON mentioned in [4], which is implemented in Long Range (LoRa) communication setting enhancing security in private networks and it integrates all the devices such as Raspberry Pi Pico Controllers, LoRa RYLR896 radio modules, ESP32 micro controllers, temperature and other atmospheric pressure sensors.

There is an interesting application of ASCON in agriculture field mentioned in [5] which works on the real time data collected from the IoT devices or sensors and these data have agricultural parameters such as temperature, humidity, soil, moisture and this data is being encrypted using Ascon algorithm. Also, it is integrated with the mobile so that it becomes interactive for the users using it.

The disadvantages of [4] is that it cannot be deployed in Real world as the study is carried out under lab conditions and also it focuses on encryption's efficiency rather than the system. The paper in [1] provides encryption as well as the decryption security but the other parameters such as Power consumption, tests on different data sizes is not being conducted. Also, it is also just a theoretical approach, not being implemented in the real world. While in [3], there's a coverage of all aspects of hardware security such as signature, interleaved signature, cyclic redundancy checks and gives important data related to power, area when being implemented on FPGA but it is prone to limited scope of error detection and it does not do overhead checks.

III. ASCON ALGORITHM

A. Encryption

This is the ASCON-80pq`s Encryption block diagram and algorithm. It starts with initial creation of state S by using Initialization Vector, key of 160 bits and nonce of 128 bits. All three concatenated together and applied for permutation P(a) function which has 12 rounds. Now, associated data is processed in blocks. Every single blocks data is combined with a state using XOR operator. Then, it processed with permutation P(b) function which has 6 rounds. Plain text is also processed in similar manner. Every single block of plain text is processed with XOR operator with current state than passed to permutation P(b) function. Now, output is XOR with a constant. At the end, in the finalization state is processed with permutation P(a) function and then XOR with padding constant and key. At last tag(T) of 128 bits is produced from final state[6]. Encryption block diagram and pseudo code image is given in figure 1 and figure 2 respectively:

B. Decryption

This algorithm is design to decrypt the data which is encrypted by ASCON 80pq encryption algorithm. Here, inputs are Key K, Nonce N, Associated data A, cipher text C and Authentication Tag T. Outputs are Plain text P and a special symbol which indicates the authentication failure. The state S is initialized with key, nonce and initialization vector and then every state is updated with permutation P(a) which is 12 rounds. Associated data A is split into r-bit blocks. Every block is updating state using backward permutation P(b) which is 6 rounds and XOR operations. Then, cipher text is also break down into r-bit blocks. Every block is updating state and recover plain text using permutation P(b) and XOR operations. Final state is obtained by applying permutation P(a) and after that apply XOR on padding constant and key. Verification tag T* is produced from final state. If T* matches with T than plain text is considered as an authenticated. If tags doesn`t match than it returns special symbol which indicates authentication failure [6]. Decryption block is shown in figure 3 and the corresponding pseudo cods is shown in figure 4.

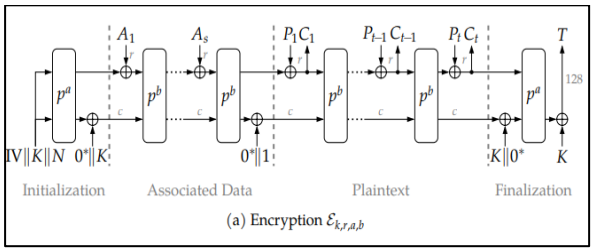

Figure 1 ASCON Encryption [6]

Authenticated Encryption $\mathcal{E}_{k,r,a,b}(K, N, A, P)$

Input: key $K \in \{0,1\}^k$, $k \leq 160$,
nonce $N \in \{0,1\}^{128}$,
associated data $A \in \{0,1\}^*$,
plaintext $P \in \{0,1\}^*$
Output: ciphertext $C \in \{0,1\}^{|P|}$,
tag $T \in \{0,1\}^{128}$

Initialization
$S \leftarrow \mathrm{IV}_{k,r,a,b} \,\|\, K \,\|\, N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \,\|\, K)$
Processing Associated Data
if $|A| > 0$ **then**
$A_1 \ldots A_s \leftarrow r$-bit blocks of $A\|1\|0^*$
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \,\|\, S_c)$
$S \leftarrow S \oplus (0^{319} \,\|\, 1)$
Processing Plaintext
$P_1 \ldots P_t \leftarrow r$-bit blocks of $P\|1\|0^*$
for $i = 1, \ldots, t-1$ **do**
$S_r \leftarrow S_r \oplus P_i$
$C_i \leftarrow S_r$
$S \leftarrow p^b(S)$
$S_r \leftarrow S_r \oplus P_t$
$\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$
Finalization
$S \leftarrow p^a(S \oplus (0^r \,\|\, K \,\|\, 0^{320-r-k}))$
$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
return $C_1 \,\|\, \ldots \,\|\, C_{t-1} \,\|\, \tilde{C}_t, T$

Figure 2 ASCON. Encryption Pseudo Code [6]

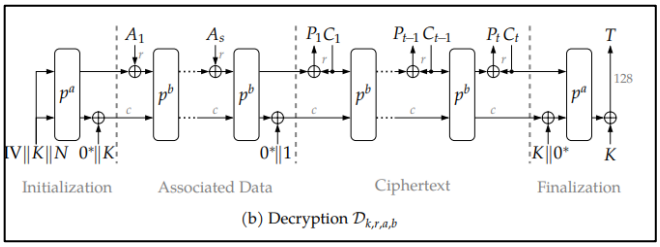

Figure 3 ASCON Decryption [6]

Verified Decryption
$\mathcal{D}_{k,r,a,b}(K, N, A, C, T)$

Input: key $K \in \{0,1\}^k, k \leq 160$,
nonce $N \in \{0,1\}^{128}$,
associated data $A \in \{0,1\}^*$,
ciphertext $C \in \{0,1\}^*$,
tag $T \in \{0,1\}^{128}$
Output: plaintext $P \in \{0,1\}^{|C|}$ or $\perp$

Initialization
$S \leftarrow \mathrm{IV}_{k,r,a,b} \| K \| N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$
Processing Associated Data
if $|A| > 0$ **then**
$A_1 \ldots A_s \leftarrow r$-bit blocks of $A \| 1 \| 0^*$
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \| S_c)$
$S \leftarrow S \oplus (0^{319} \| 1)$
Processing Ciphertext
$C_1 \ldots C_{t-1} \tilde{C}_t \leftarrow r$-bit blocks of $C, 0 \leq |\tilde{C}_t| < r$
for $i = 1, \ldots, t-1$ **do**
$P_i \leftarrow S_r \oplus C_i$
$S \leftarrow C_i \| S_c$
$S \leftarrow p^b(S)$
$\tilde{P}_t \leftarrow \lfloor S_r \rfloor_{|C_t|} \oplus \tilde{C}_t$
$S_r \leftarrow S_r \oplus (\tilde{P}_t \| 1 \| 0^*)$
Finalization
$S \leftarrow p^a(S \oplus (0^r \| K \| 0^{320-r-k}))$
$T^* \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
if $T = T^*$ **return** $P_1 \| \ldots \| P_{t-1} \| \tilde{P}_t$
else return $\perp$

Figure 4 Pseudo code for ASCON Decryption [6]

IV. PROPOSED METHODOLOGY

The proposed methodology revolves around implementing alternative strategies to reduce the overall power consumption so that better efficiency can be obtained and the time complexity and the space complexity can be decreased. There are multiple techniques to achieve this such as Avoid repetitive operations, Memory usage, Function inlining, optimize data loading and storing, Conditional compilation, Loop Unrolling.

In avoid repetitive operations, such technique can be deployed which prevent use of repeated calculations which are unnecessary and rather save their results if not changing over a period. In memory usage technique, the memory allocated space which are larger than necessary should be checked so that total program's memory can be reduced. In function inlining technique, the small functions which are being called frequently should be inlined so that the overhead reduces. Also, data storing and processing techniques should be addressed to reduce the space as well as time complexity. And if there is a fixed size block of data, loop unrolling technique should be used where loops are used for processing. Conditional compilation can also be used when there is a code which is called unnecessarily, and it should be avoided and should be called only when a certain condition is set to true.

Our focus was to implement all those utilization techniques and then we would compare those and find which one's better and adapt the code accordingly. On implementation, we found that by using inline functions we were able to achieve the reduce power consumption value for both the encryption as well as the decryption traces. The code for the same is being shown in figure 5.

```
static inline uint8_t get_encrypt(uint8_t* pt, uint8_t len) {
    return process_input(pt, len, 1); // Inline for efficiency
}

static inline uint8_t get_decrypt(uint8_t* c, uint8_t len) {
    return process_input(c, len, 0); // Inline for efficiency
}
```

Figure 5 Inline functionality for encryption and decryption.

V. RESULTS

Now that the Ascon-80pq code has been implemented in simpleserial-base file, we must use the ChipWhisperer platform to do power analysis on cryptographic algorithms. This involves creating random plain text, logging power traces during encryption and decryption, and analyzing these traces by visualizing them with Matplotlib. That python code collects power consumption statistics across several cryptographic processes by building for loop for plain text and producing random cipher text. Those plotted data traces help in an analysis of power consumption patterns and provides information about the operational features of the cryptographic processes. We use NumPy's array structure and manipulation features after importing it to efficiently keep and handle the power traces collected during the encryption and decryption procedures. Now we also need to transfer voltage traces into power traces for better analysis of results and for that we can use equation V2 / R (where R is resistance and, in our case it 51 for ChipWhisperer).

As shown in figure 6, encryption of power traces of 100 random plain texts are being generated through the original ascon-80pq code which was found on the GitHub repository. And figure 7 shows power traces for 100 generated corresponding cipher texts.

And after the implementation of optimized parameters, the figures 8 and 9 show the power traces of the 100 random plain texts and 100 random corresponding decryption texts.

Our goal of optimization is to reduce power consumption and when we analyse power consumption of original code, we got “0.003144980859835643 mJ” for encryption and “0.002833995431354796 mJ” for decryption. But after optimizing of our code, power traces were collected again and power consumption of “0.002833995431354796 mJ” is achieved for encryption and “0.0021425365211406406 mJ” for decryption.

And when calculated reduction in power consumption in percentage, the results showed that there is “31.311073996955415 %” decrease in power consumption for encryption and “24.398730589470365 %” power consumption for decryption and that data is shown in figure 10.

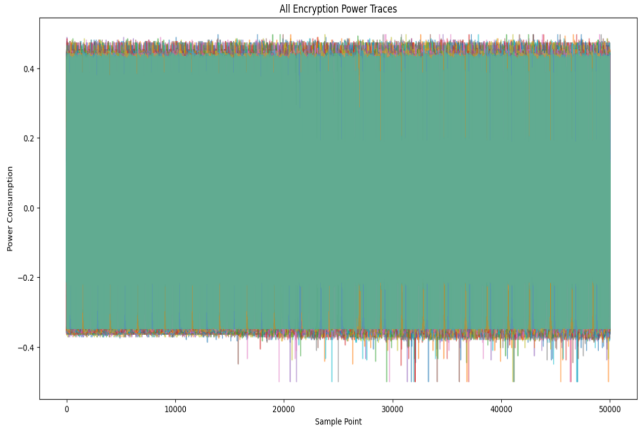

Figure 6 Power consumption for Encryption.

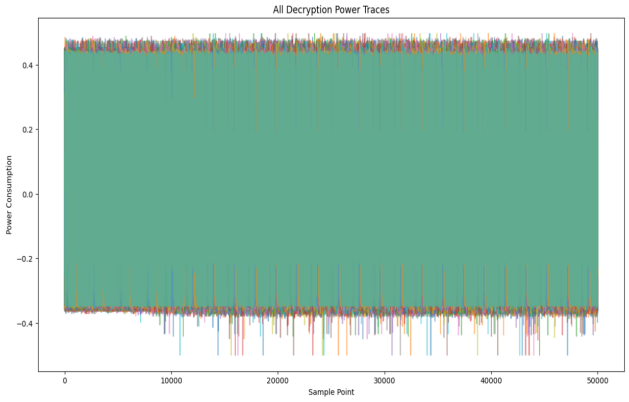

Figure 7 Power consumption for Decryption.

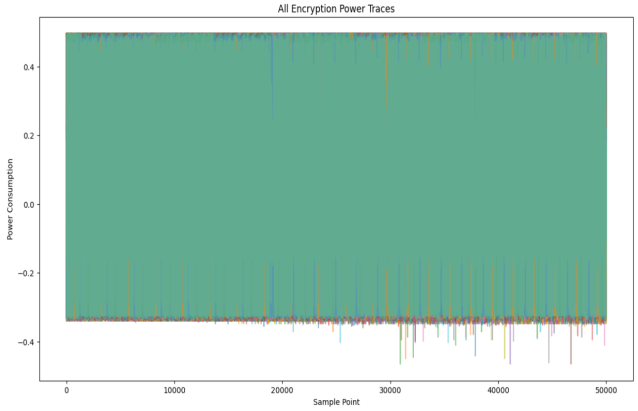

Figure 8 Optimized Power consumption for Encryption.

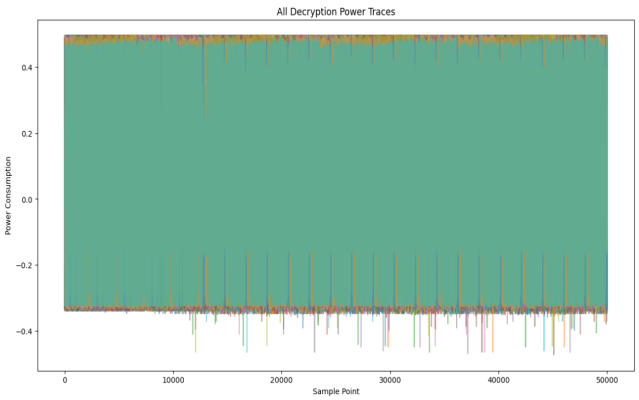

Figure 9 Optimized Power consumption for Decryption.

```
old_power_consumption_enc = 0.003144980859835643
new_power_consumption_enc = 0.0021602535756224204

percentage_change_enc = ((old_power_consumption_enc - new_power_consumption_enc) / old_power_consumption_enc) * 100

print("percentage change in encrption : ",percentage_change_enc)

old_power_consumption_dec = 0.002833995431354796
new_power_consumption_dec = 0.0021425365211406406

percentage_change_dec = ((old_power_consumption_dec - new_power_consumption_dec) / old_power_consumption_dec) * 100

print("percentage change in decryption : ",percentage_change_dec)


percentage change in encrption :  31.311073996955415
percentage change in decryption :  24.398730589470365
```

Figure 10 Comparison of Percentage Values of Encryption and Decryption.

VI. CONCLUSION

In conclusion, methodology focuses on ASCON80-pq algorithms` implementation. This represents thorough analysis of low power usage of encryption and decryption methods for low power consumption devices. Our team effectively reduced power consumption by 31.31% during the encryption process and 24.31% throughout the decryption process by implementing various optimization techniques such as inline functions and loop unrolling. ASCON80-pq is best for low power and provide security to side channel attacks. Our work shows usefulness of optimizing cryptographic algorithm for power efficiency.

REFERENCES

[1] A. Koppuravuri, H. Pasupuleti, S. Gvk and J. Bapat, ”A High Throughput ASCON Architecture for Secure Edge IoT Devices,” 2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID), Kolkata, India, 2024, pp. 486-491, doi: 10.1109/VLSID60093.2024.00087.

[2] J. Kaur, M. Mozaffari Kermani and R. Azarderakhsh, ”Hardware Constructions for Error Detection in Lightweight Authenticated Cipher ASCON Benchmarked on FPGA,” in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 4, pp. 2276-2280, April 2022, doi: 10.1109/TCSII.2021.3136463.

[3] D. Chang, D. Hong, J. Kang and M. S. Turan, ”Resistance of Ascon Family Against Conditional Cube Attacks in Nonce-Misuse Setting,” in IEEE Access, vol. 11, pp. 4501-4516, 2023, doi: 10.1109/ACCESS.2022.3223991.

[4] M. Nooruddin and D. Valles, ”An Advanced IoT Framework for Long Range Connectivity and Secure Data Transmission Leveraging LoRa and ASCON Encryption,” 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2023, pp. 0583-0589, doi: 10.1109/AIIoT58121.2023.10174401.

[5] R. R, R. Venkatesan and T. J. Jebaseeli, ”Smart Farming with Improved Security using Ascon Encryption and Authentication,” 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024,pp. 365-373, doi: 10.1109/IDCIoT59759.2024.10467361.

[6] “Ascon 80 pq algorithm implementation pdf - Google Search,”https://www.google.com/search?q=Ascon+80+pq+algorithm+implementation+pdf&rlz=1C1CHBF_enIN889IN889&oq=ascon+&gs_lcrp=EgZjaHJvbWUqCAgAEEUYJxg7MggIABBFGCcYOzIGCAEQRRg7MgYIAhBFGEAyBggDEEUYOzIGCAQQRRg8MgYIBRBFGDwyBggGEEUYPDIGCAcQRRg8qAIAsAIA&sourceid=chrome&ie. [Accessed: 11-Apr-2024].

[7] Baudrin, Jules & Canteaut, Anne & Perrin, Léo. (2022). Practical Cube Attack against Nonce-Misused Ascon. IACR Transactions on Symmetric Cryptology. 120-144. 10.46586/tosc.v2022.i4.120-144