

Project: SuperSAM Security Analysis

Ikram Ounadi

1 Executive Summary

The implementation note along with superAES foulder provides details on the design and concerns for SuperSAM, a secure key fob system. This solution is designed to give employees with secure access to facilities based on the access privileges contained on the key fob. The note discusses several facets of the system, including cryptographic primitives, communication protocols, and security mechanisms. I have made an effort using the previous given lectures and other sources to find vulnerabilities in the system. This overview summarises the main worries and provides a series of suggestions meant to strengthen the system's security against these found vulnerabilities.

- **Insufficient Security Level:** The system employs the 1536-bit MODP Group for Diffie-Hellman key exchange, which is deemed insufficient for the specified security level. Upgrade to a stronger MODP Group, such as the 2048-bit or 3072-bit Group, to ensure compatibility with AES-128 encryption.
- **Man-in-the-Middle Vulnerability:** The Diffie-Hellman key exchange protocol lacks authentication methods, rendering it susceptible to Man-in-the-Middle assaults. To remedy this, it is advised to use a secure protocol such as SSL/TLS or a password-authenticated key agreement mechanism for mutual authentication between the token and reader.
- **Simple encryption method :** The SuperSAM system's usage of AES in ECB mode creates a security risk by exposing plaintext patterns, allowing unauthorised access to duplicate code. To improve security, consider switching to a more secure mode, such as CBC or GCM.
- **Poor Security Evaluation of PRNG:** The pseudo-random number generator (PRNG) only passed the serial test, which is insufficient to ensure its cryptographic security. It is suggested that additional testing be performed using recognised standards, such as those outlined in NIST Special Publication 800-22, to guarantee that the PRNG is suitable for cryptographic applications.
- **Insecure Seed Initialization:** The LFSR seeds in the masked combined.c file are set to 0, resulting in a non-working mask and making the AES implementation vulnerable to first-order DPA attacks. To address this, the seeds should be loaded with random values from a secure PRNG.

- **Insufficient Number of Rounds:** The AES implementation in `masked combined.c` only executes 9 rounds of encryption, rather than the required 10 rounds provided by the AES algorithm. This divergence makes the implementation vulnerable to known key distinguisher attacks. It is suggested that the code be modified to include all ten rounds of encryption as required by the AES standard.
- **Fault Attack Vulnerability:** The AES implementation is vulnerable to fault attacks and adaptive selected plaintext attacks owing to the lack of fault-tolerant safeguards and usage of ECB mode. To address these vulnerabilities, include fault detection and repair procedures.

2 Specific points

2.1 Implementation Note: Simple Encryption method, Section 2.3. on Page 3

Problem Description:

It is stated in the implementation note that they employed AES with 128 keys as a cryptographic primitive, and because all of the core data is expected to be less than 128 bits, they used EBC.

The present SuperSAM system implementation of the AES encryption algorithm in ECB mode creates a severe security risk because it is the most basic block cypher mode of operation in existence. This method converts identical plaintext blocks into identical ciphertext blocks. As a result, plaintext patterns emerge in the ciphertext. While ECB mode is faster, simpler, and more parallelizable to create, it reveals information about the underlying message being encrypted. An eavesdropper may find similar blocks and extract the original plaintext merely by glancing at the ciphertexts (that are public information)[1].

Demonstrating this as a vulnerability:

An attacker might exploit this vulnerability by analysing encrypted data between key fobs and door readers. Duplicating access codes or signals from an authentic key fob can be done by examining patterns in the ciphertext and deducing the corresponding plaintext. This is because patterns are not adequately disguised when utilising the EBC mode because simply they can not be well hidden.

For example Let's say two key fobs have the identical four-digit access code, such as "1234". In ECB mode, identical plaintext blocks (i.e., access codes) generate identical ciphertext blocks.

An attacker intercepts encrypted access codes sent between key fobs and door readers. The attacker identifies repetitive blocks or patterns in the ciphertext. In this example, the attacker determines that the ciphertext blocks corresponding to the access code "1234" are identical on both key fobs.

Using this information, the attacker concludes that both key fobs use the identical access code, "1234". Using this information, the attacker may make a clone of a legal key fob by recreating the intercepted ciphertext blocks. The cloned key fob can then be used to obtain unauthorised entry to the system since the door reader recognises it as a genuine key fob with the access code "1234".

The attacker successfully replicates access codes or signals from legitimate key fobs by evaluating ciphertext patterns and deducing the matching plaintext. This demonstrates the weakness of SuperSAM's usage of ECB mode, as patterns are not sufficiently masked, making it very easy for an attacker to exploit.

Proposed Improvement:

To address this issue, the system should abandon AES in ECB [2] mode in favour of a more secure mode, such as CBC or, even better, GCM [3], which uses IVs and provides stronger pattern obfuscation.

When considering the SuperSAM system's needs, the decision between CBC and GCM is determined by variables such as performance, security.

Compared to ECB, both variants give stronger pattern obfuscation. First I will give both the characteristics of both modes :**The Cypher Block Chaining (CBC) Mode** :[4] XORs the current plaintext block with the previous ciphertext block before encryption, Protects against patterns and repetition in plaintext, Encryption requires an initialization vector (IV) , Suitable for encrypting communications longer than a block.

Galois/Counter mode(GCM) :[5] combines encryption with authentication, ensuring secrecy and integrity in a single process, It utilises a unique IV for each encryption, similar to CBC mode, resulting in different ciphertext blocks even when the plaintext blocks are identical, GCM mode has integrated authentication, which helps prevent tampering and unauthorised modifications.

Since the SuperSAM system demands both secrecy and integrity in an efficient manner, GCM mode is the superior option. However, also the system's primary requirement is secrecy and integrity without integrated authentication, CBC mode can be an acceptable option, provided an extra authentication method is implemented independently.

In my opinion Even yet, GCM implementation might be more difficult due to its complexity and particular needs. Compared to CBC, I would choose it because it is the better alternative, or because I have trust issues and I'd chose the better option even if it is more complex .

2.2 Implementation Note:Insufficient Security Level, Section 2.3. on Page 3

Problem Description:

The SuperSAM system's Diffie-Hellman key exchange parameters are obtained from the 1536-bit MODP Group or we can simply designate this group by group 5, which is specified on page 3 of the given [6].

Despite the fact that these settings are appropriately implemented, they do not meet the system's needed security level, which is comparable to AES-128 encryption because it is too weak. This vulnerability compromises the system's encrypted communications.

Demonstrating this as a vulnerability:

As noted on page 8 of the previously mentioned reference [6] in the security consideration, the vulnerability appears because the 1536-bit MODP Group settings are not equivalent in strength to AES-128 encryption. Given that the DH protocol is a critical component of the system's security, incorrect parameter selection puts encrypted communications at risk.

If an attacker successfully exploits the DH key exchange process, they may be able to carry out cryptographic attacks like man-in-the-middle, impersonation, or even gain the shared secret key. This might lead to unauthorised entry to buildings and threaten the overall security of the system.

Furthermore, the SuperSAM system is based on proximity based RFID technology, with the SuperSAM token storing information regarding access privileges. If an attacker can exploit the weak DH settings, they may be able to tamper with the token's information, get unauthorised access to buildings, or alter the token's access privileges.

Proposed Improvement:

The modifications I recommend are:

- Either changing the DH parameter to the 2048-bit MODP Group, as defined on page 3 of [6] which can also be referred to as group 14. And this is to ensure conformity with the expected AES-128 security level, which is considered the minimum acceptable by [7].
- I'd also recommend installing the 3072-bit MODP Group or also referred to as group 15 which is specified on page 4 of [6] for greater security, especially if hardware characteristics allow.

The proposed changes to the SuperSAM system's Diffie-Hellman (DH) settings require an in-depth examination of their implications. The implementation of the 3072-bit MODP Group necessitates careful evaluation of computing resources, ensuring that prospective increases in demand are in line with the system's hardware capabilities while maintaining overall performance. Stronger DH parameters may have an influence on key exchange performance, potentially resulting in longer transfer time; therefore, an evaluation is required to ensure that system responsiveness stays within acceptable ranges. To achieve smooth integration, the entire system compatibility, including client devices, servers, and access control systems, must be checked and any possible concerns resolved. Key management techniques, such as secure creation, storage, distribution, and rotation protocols, should be evaluated and reinforced to ensure cryptographic activities' integrity and secrecy. Robust testing and validation after installation are critical for identifying

and resolving unexpected difficulties, assuring a seamless transition, and proving the effectiveness of suggested upgrades. In this way, the SuperSAM system may effortlessly integrate enhanced security measures while maintaining maximum performance and compatibility across the system .

Using a larger MODP Group makes it more difficult to solve the discrete logarithm problem, but it also increases the DH's security level and makes the system safer.

2.3 Implementation Note:Man-in-the-Middle Vulnerability , Section 2.4. on Page 4

Problem Description:

To establish session keys between tokens and readers, the SuperSAM system uses the fundamental Diffie-Hellman Key Exchange (DHKE) protocol. This protocol version, however, does not validate the communication entities, leaving it open to a Man-in-the-Middle (MitM) attack.

Demonstrating this as a vulnerability:

Diffie-Hellman is a key exchange protocol that enables two parties to establish a shared secret key using an unsecure channel. The procedure is based on the discrete logarithm problem, which is computationally infeasible for big prime numbers. If the settings are specified appropriately, the protocol is safe against any attempts to compute the shared key. The protocol is vulnerable to a "man in the middle" attack, which allows an attacker to intercept communication between two parties and impersonate one of them. In the scenario superSAM it works as follows :

- The attacker generates a fake token with its own DH value and secureID.
- When the actual token says "Hello" to the reader, the attacker intercepts the communication and answers with the fake token's "Hello" message, as well as the DH value and secureID.
- The attacker impersonates the real token by relaying its DH value and secureID to the reader.
- The reader and the attacker (pretending to be the true token) exchange DH keys to agree on a shared secret key (sk).
- The attacker sends the reader a fake token that is encrypted with the shared secret key.
- The reader confirms the fake token and allows access to the attacker, thinking it to be the real token.

Once the attacker has correctly impersonated both the token and the reader, they can communicate with each other as if they were real parties. The attacker can obtain valid building IDs and the master key from the token. This enables the attacker to create new valid tokens granting access to any building and impersonating legitimate users.

Proposed Improvement:

To defend against MiTM attacks, we can utilise the SSL/TLS security protocol described in [8].

This protocol encrypts data, validates the server's identity via digital certificates, and adds a digital signature to assure data integrity. The idea behind implementing it in SuperSAM is that when the token is entered into the reader, they exchange certificates to authenticate each other's identities. If both certificates are real, they create a secure SSL/TLS connection. All communication between the reader and the token is encrypted, making it impossible for an adversary to intercept or tamper with it.

Another thing I thought about is that in cryptography, a password-authenticated key agreement method is an interactive technique allowing two or more parties to establish cryptographic keys based on one or more of their passwords. An key characteristic is that an eavesdropper or man-in-the-middle cannot gather enough information to brute-force guess a password without engaging in further contacts with the parties after each (few) guess. This implies that weak passwords can still provide good security.[9]

After reviewing the protocol proposed in [10], I tried to talk about an idea a protocol which is more or less adapted to SuperSAM, or so I believe, and varies from the original PAK agreement only in its assumption of one-way authentication—whereas PAK concentrates on mutual authentication, here, the reader authenticates the token. The idea is the reader and token begin by agreeing on the System Master Key (smk) and selecting a hash function (H) to protect the transaction. The token starts the authentication process by providing the reader a value obtained by hashing its identification (secureID) with the smk. The reader then authenticates the token by validating the hash and returning a separate hashed value including additional information from the first transaction. The token confirms the reader's reliability by confirming this new hash. Following successful mutual verification, they generate a session key from the exchanged values, which is afterwards used for secure communication.

2.4 Implementation Note:Insufficient Side-Channel attacks Resistance, Section 3.2 on Page 4-5

Problem Description:

While the SuperSAM token has a strong implementation of Advanced Encryption Standard (AES) with a higher-order masking method meant to prevent Differential Power Analysis (DPA) attacks, it is vulnerable to other side-channel attacks. The existing de-

fence, although successful against DPA, is ineffective against larger side-channel threats, including Electromagnetic Analysis (EMA) assaults.

Demonstrating this as a vulnerability:

In this case Attackers intend to exploit the electromagnetic radiation released during the SuperSAM token's cryptographic procedures, notably AES computations. The primary purpose is to extract sensitive information, such as encryption keys, by analysing electromagnetic emissions.

Proposed Improvement:

Applying electromagnetic attacks to the SuperSAM token includes inspecting its implementation for probable electromagnetic leakage spots during cryptographic processes. The concentration is on discovering flaws that might allow adversaries to monitor electromagnetic emissions and get sensitive cryptographic data. The investigation seeks to determine how these emissions may provide information about the token's cryptographic procedures. To address these possible dangers, countermeasures like as shielding, filtering, or algorithm changes may be devised and then tested for their efficiency in reducing electromagnetic leakage while preserving system performance. The findings should be properly recorded, including detected vulnerabilities, analytical results, recommended countermeasures, and references to relevant literature on electromagnetic assaults in cryptographic systems to provide a complete overview.[11]

2.5 Implementation Note: Poor security evaluation of PRNG, Section 3.4 on Page 6

Problem Description:

The cryptographic implementation of the SuperSAM system claims to have tested the PRNG (Pseudo-Random Number Generator) to ensure that it passes the serial test, which looks at the distribution of two-bit sequences. However, depending only on the serial test may not guarantee the PRNG's cryptographic security.

Demonstrating this as a vulnerability:

Even if a PRNG passes the serial test, it might present problems such as predictable produced numbers due to insufficient security. The unpredictability provided by a strong PRNG is critical for the security of cryptographic operations such as key generation and initialization vector (IV) production. Relying entirely on serial testing may miss underlying weaknesses that might be exploited in cryptographic scenarios.

Proposed Improvement:

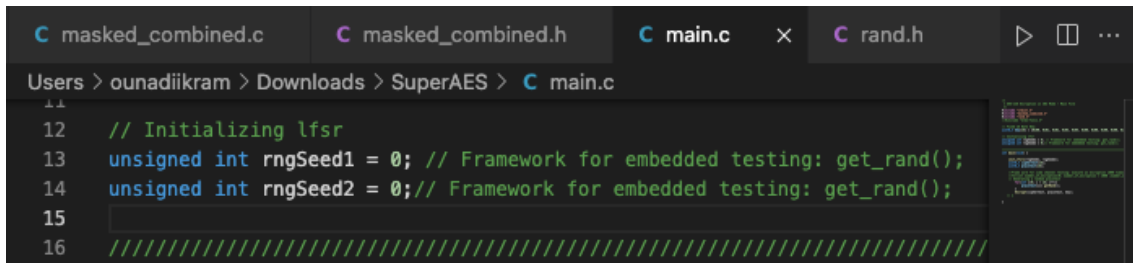
To overcome this issue, it is proposed that the PRNG pass additional testing, especially utilising known standards such as those defined in [12]. This larger testing technique

assures that the PRNG not only passes simple statistical tests but also meets challenging cryptographic standards, offering a more solid proof of its suitability for cryptographic applications.

2.6 SuperAES: Insecure Seed Initialization, main.c

Problem Description:

The given code initialises both seeds, "rngSeed1" and "rngSeed2", to 0. This results in a non-working mask in **masked_combined.c**, leaving the AES implementation open to first-order DPA attacks.



```
C masked_combined.c  C masked_combined.h  C main.c  x  C rand.h  ▶ □ ...
Users > ounadiikram > Downloads > SuperAES > C main.c
12  // Initializing lfsr
13  unsigned int rngSeed1 = 0; // Framework for embedded testing: get_rand();
14  unsigned int rngSeed2 = 0; // Framework for embedded testing: get_rand();
15
16  //////////////////////////////////////
```

Figure 1: Insecure Seed Initialization

Demonstrating this as a vulnerability:

Given that the LFSR's seeds are set to 0, its output is always 0, resulting in a non-working mask in masked-combined.c. Since the data in the implementation is not disguised in any manner, an attacker may easily execute a first-order DPA attack on the AES implementation and retrieve the key. A DPA attack takes use of the fact that the AES implementation's power consumption varies with the key and input data. To demonstrate the vulnerability theoretically, I thought about using unsecured SuperAES implementation. Establish a DPA attack environment. Collect power traces when running SuperAES. Pre-process the power traces. Select an appropriate DPA attack algorithm. Perform a first-order DPA assault. and finally analyze the results to recover the secret key.[13]

Proposed Improvement:

To prevent this problem, the two seeds should be initialised with random values from a secure PRNG. This ensures that the initial values are protected by a mask, making it more difficult for attackers to exploit them. By initialising the seeds with random values, the AES implementation becomes more resistant against first-order DPA attacks.

2.7 SuperAES:The vulnerability of fault attacks , masked_combined.c

Problem Description:

The AES implementation in the **masked_combined.c** file is subject to fault attacks, which include manipulating memory to flip bits, as well as adaptive selected plaintext attacks.

Demonstrating this as a vulnerability:

A fault attack can be carried out by flipping a bit in the state array during the AddRoundKey operation. This can be accomplished with a fault injection tool, such as a laser beam or a voltage glitching tool.

By flipping a bit in the state array, the attacker can cause the ciphertext to deviate from what would have been generated without the fault. The attacker has the possibility to retrieve the secret key by analysing the relationship between the bit flip and the ensuing ciphertext.

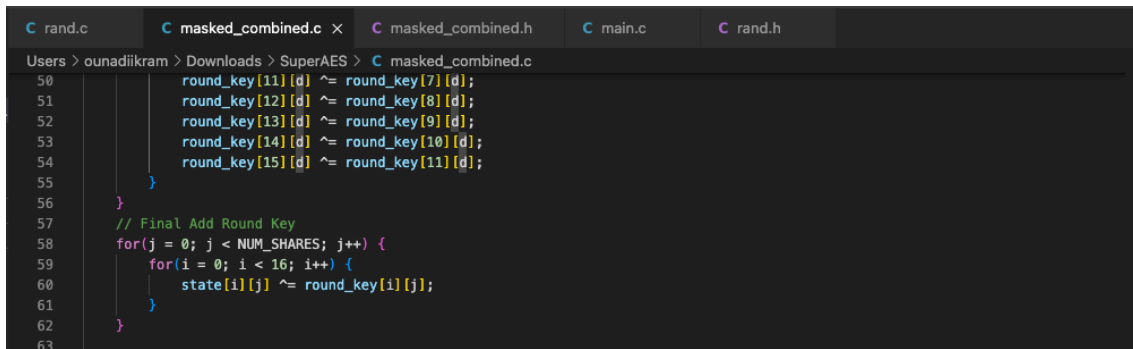
A screenshot of a code editor with a dark theme. The editor shows the file 'masked_combined.c' open. The code is in C and shows a function for the final AddRoundKey operation. It includes a loop for 'j' from 0 to NUM_SHARES-1, and an inner loop for 'i' from 0 to 16. Inside the inner loop, the state array is updated: 'state[i][j] ^= round_key[i][j];'. Above this, there are several lines of code that assign values to 'round_key' array elements, such as 'round_key[11][d] ^= round_key[7][d];'. The editor's tab bar at the top shows 'C rand.c', 'C masked_combined.c x', 'C masked_combined.h', 'C main.c', and 'C rand.h'. The left sidebar shows a file tree with 'Users > ounadiikram > Downloads > SuperAES > C masked_combined.c'.

Figure 2: State array

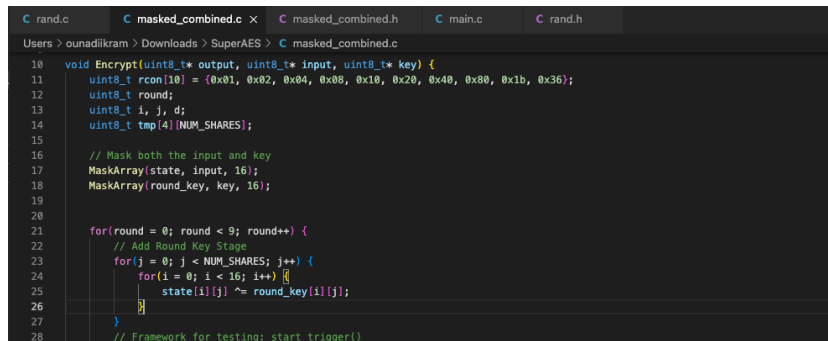
Proposed Improvement:

A number of changes can be made to improve the security of the AES implementation and protect it from fault attacks and adaptive chosen plaintext attacks, such as using a fault-tolerant mechanism to detect and correct bit flips in the state array during encryption, thereby mitigating the impact of fault attacks. This can be accomplished by employing a coding system that detects and corrects faults, such as Hamming or Reed-Solomon codes. Another thing I discovered that I think can be useful in this circumstance is to build a bit-wise scrambling mechanism to improve the resilience of hardware AES implementations against fault attacks. This method involves scrambling the individual bits of the state bytes across the two data routes in a balanced manner. The scrambling can be implemented at several points in the AES algorithm, including the ShiftRows transformation.[14]

2.8 SuperAES:Insufficient Number of Rounds, masked_combined.c

Problem Description:

The AES implementation in the **masked_combined.c** file does only 9 rounds of encryption, rather than the typical 10 rounds provided by the AES algorithm for a 128-bit key.



```
10 void Encrypt(uint8_t* output, uint8_t* input, uint8_t* key) {
11     uint8_t rcon[10] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36};
12     uint8_t round;
13     uint8_t i, j, d;
14     uint8_t tmp[4][NUM_SHARES];
15
16     // Mask both the input and key
17     MaskArray(state, input, 16);
18     MaskArray(round_key, key, 16);
19
20
21     for(round = 0; round < 9; round++) {
22         // Add Round Key Stage
23         for(j = 0; j < NUM_SHARES; j++) {
24             for(i = 0; i < 16; i++) {
25                 state[i][j] ^= round_key[i][j];
26             }
27         }
28     }
29     // Framework for testing: start_trigger()
```

Figure 3: Number of rounds

Demonstrating this as a vulnerability:

This divergence from the standard exposes the implementation to known key distinguisher attacks. An attacker with knowledge of the decreased number of rounds can use this issue to distinguish between the susceptible and standard 10-round implementations. This may lead to the recovery of the secret key used in the encryption procedure.

Proposed Improvement:

To address the issue, the AES implementation should be changed to conduct all ten rounds of encryption as mandated by the AES standard. To do this, change the loop method to iterate 10 times.

References

- [1] Ubiq Security. ECB vs CBC Block Cipher Mode Differences, publication year, e.g., 2022.
- [2] EDUCBA. ECB vs CBC, 2022.
- [3] IBM. Galois/Counter Mode (GCM), 2022.
- [4] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson, 8th edition, 2017.
- [5] National Institute of Standards and Technology (NIST). Galois/counter mode (gcm) spec, Year of the document.
- [6] IETF. More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), 2003.
- [7] Cisco Community. Diffie-Hellman Groups, N/A.
- [8] Dionisie Gitlan. How does ssl prevent man-in-the-middle attacks? *SSL Dragon Blog*, 2023.
- [9] Wikipedia contributors. Password-authenticated key agreement — Wikipedia, the free encyclopedia, 2023. [Online; accessed Day-Month-Year].
- [10] A. Brusilovsky, I. Faynberg, Z. Zeltsan, and S. Patel. Password-authenticated key (pak) diffie-hellman exchange. RFC 5683, RFC Editor, February 2010. Informa-tional.
- [11] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Mea-sures and countermeasures. In *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings*, pages 259–270. Springer, 2003.
- [12] National Institute of Standards and Technology. NIST Special Publication 800-22 Revision 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, 2010. Accessed: `{date}`.
- [13] Josh Jaffe. A first-order dpa attack against aes in counter mode with unknown initial counter.
- [14] Marc Joye, Pascal Manet, and Jean-Baptiste Rigaud. Strengthening hardware AES implementations against fault attacks. *Thomson R&D France, Technology Group, Corporate Research, Security Laboratory*.