622.755 – Advanced Topics in Cybersecurity 2 Winter Term, 2022/23

Project: SuperSAM Security Analysis

Francesco Bombassei De Bona - 12138677

1 Executive Summary

2 Specific points

2.1 Implementation Note: DH parameters and difficulty, Section 2.3 in page 3

Problem Description: The Diffie-Hellman parameters chosen in the SuperSAM implementation do not ensure a safe communication channel. Even if the parameter are chosen from page 3 of RFC 3526 [KK03], the parameters are not safe for the chosen security level.

Demonstrating this as a vulnerability: The Diffie-Hellman parameters are chosen from the 1536-bit MODP Group of RFC 3526 [KK03], which is a list of correct parameters for the Diffie-Hellman key exchange protocol. In the same RFC 3526, page 7, is shown that the 1536-bit MODP Group is not as strong as AES-128, which is the chosen security level for the SuperSAM implementation. Also, taking into account what presented in ..... (insert reference to the other paper), the 1536-bit MODP Group is on the edge to be considered safe for the chosen security level. As it is described in the implementation note, DH protocol is strictly responsible for the security of the whole system, since the encryption of the ticket is based on the parameters, and for this reason, the wrong choice of the parameters can have tremendous consequences for the safety of the system.

Fixing this vulnerability: The parameters should be chosen at least from the 2048-bit MODP Group of RFC 3526 [KK03](page 3), or from the 3072-bit MODP Group (page 4) if the hardware is capable of supporting it. Using a larger MODP Group increases the difficulty of resolving the discrete logarithm problem, and the security level of the DH increases and the system becomes more secure.

2.2 Impletation Note: DH usage, Section 2.4 in pages 3, 4, 5

Problem Description: The base version of DH, which is the one implemented in the SuperSAM system, is vulnerable to the "Man-in-the-middle" attack since it doesn't provide an authentication mechanism that the parties can use to prove who they are .

Demonstrating this as a vulnerability: Diffie-Hellman is a key exchange protocol that allows two parties to establish a shared secret key over an insecure channel. The protocol is based on the discrete logarithm problem, which is considered computationally infeasible for large prime numbers. Then, if the parameters are chosen correctly, the protocol is secure against any tentative of computing the shared key. The protocol is vulnerable to a "Man in the middle" attack, where an attacker can intercept the communication between the two parties and impersonate one of them.

In the scenario superSAM, the attacker computes two DH values, one for the reader and one for the token. When the token initializes the communication with "Hello", the attacker sends back the DH value computed for the token and the request to authenticate. Upon receiving the "secureID" from the token and its DH value, the attacker can start the communication with the reader: the attacker starts sends "Hello", receive the DH value of the reader and the request to authenticate, and sends the DH value computed for the reader and the "secureID" provided by the token. The attacker is now setup as a fake reader and can interact with the token by:

- Receiving an encrypted ticket from the token containing a series of valid building IDs that can be decrypted using the session key.
- Sending a new "ticket" to the token enabling entrance to buildings or denying it.

At the same time, the attacker interprets the token in the communication with the reader and interact with it by:

- Receiving a new valid master key from the token.

This fully enables an attacker to generate a new valid token enabling entrance to any building.

Proposed Improvement: The protocol can be made secure against this attack by ensuring authentication of the parties involved in the communication. Using a variation of the Password-Authenticated Key proposed in RFC 5683 [ZPFB10], DH can be made secure against the "Man-in-the-middle" attack.

The protocol is based on the idea of using a password to authenticate the parties involved in the communication and works as follows, in the setting of SuperSAM:

- The reader and the token agree on a password (System Master Key, smk) and a hash function.
- The token sends to the reader secureID$|H($secureID$|$smk$) \cdot g^T$, where H is the hash function.
- The reader computes $H($secureID$|$smk$)$, divides the received value with it, and recovers X_{TR}.

- The reader sends to the token $H(\text{secureID}|\text{smk})|H(\text{secureID}|\text{smk}|X_{TR}|g^T|{X_{TR}}^T)$
- The token computes $H(\text{secureID}|\text{smk})$, divides the received value with it, and recovers Y_{TR}.
- The token computes $H(\text{secureID}|\text{smk}|g^R|Y_{TR}|{Y_{TR}}^R)$, and compares it with the received value.
- The token sends to the reader $H(\text{secureID}|\text{smk}|g^R|Y_{TR}|{Y_{TR}}^R)$.
- The reader computes $H(\text{secureID}|\text{smk}|X_{TR}|g^T|{X_{TR}}^T)$, and compares it with the received value.

If at any time the computed value does not match the received one, the communication is aborted. Otherwise, the reader and the token use the shared secret $H(\text{secureID}|\text{smk}|X_{TR}|g^T|{X_{TR}}^T) = H(\text{secureID}|\text{smk}|g^R|Y_{TR}|{Y_{TR}}^R)$ as session key.

The variation in comparison to the original protocol of RFC 5683 is that we assume that the reader should be able to authenticate the token, but not the other way around. This scheme is also capable of supporting an old System Master Key since the reader can store also old keys and adapt its behavior accordingly.

It is important to note that in the case a System Master Key is compromised, an human operator must manually change all the tokens with the compromised key.

2.3 Implementation Note: PRNG test metods, Section 3.4 page 6

Problem Description: A PRNG to be suitable for cryptographic purposes must pass a series of tests as shown in "Handbook of applied cryptography" [MVOV97], and FIPS 140–2 [Nat01]. In this case, a serial test cannot be the only test to ensure that the PRNG is compliant to the standards.

Demonstrating this as a vulnerability: A "bad" PRNG can lead to the predictability of the number produced due to lack of entropy which can lead to a predictable key or a predictable IV. As it shown later in the document, the production of random numbers is crucial for the security of the AES implementation. A serial test (or two-bit test) is a test that checks if the PRNG produces sequence of bits were the distribution of two-bits sequence (00, 01, 10, 11) is not uniform (page 181 of [MVOV97]). This test is useful but not sufficient to detect if the PRNG is compliant to standards like the FIPS 140–1 or 140–2. For this reason, it's not sufficient to affirm that the PRNG distribution is uniform, but it's necessary to prove that the PRNG is compliant to the standards.

Proposed Improvement: The PRNG should be tested using the tests proposed in NIST SP 800–22 [BRS+10] to ensure that is compliant to the standards.

2.4 AES Implementation, rand.c and masked-combined.c

Problem Description: The right initialization of the two seeds is crucial for the security of the PRNG and the AES implementation. In the given code initially the two seeds are initialized to 0, which then leads to the vulnerability of the whole AES implementation to DPA attacks of first-order, as shown in the next paragraph. Also leaving unspecified the PRNG that should be used to generate the two seeds is a major security issue.

Demonstrating this as a vulnerability: Since the LFSR has the seeds initialized to 0, its output is always 0, and this leads to a non-working mask in *masked-combined.c*. Since the data in the implementation is not hidden in any way, an attacker can easily perform a DPA attack of first-order on the AES implementation and recover the key. A DPA attack is an attack that exploits the fact that the power consumption of the AES implementation is a function of the key and the input data. The first-order attack is the simplest attack that can be performed on the AES implementation, and relates one intermediate value to the whole power trace [MOP07](page 245).

To perform a demo of this attack, I decided to target the S-box of the AES implementation.

```
if (check == 0) {
    start_trigger();
}
CombinedSbox(state);
if (check == 0) {
    pause_trigger();
    check = 1;
}
```

After collecting the power traces, the plaintext and the ciphertext, I used a Python script to perform the DPA attack of first-order. The script was set up to perform a DPA attack using the Hamming Weight as distinguisher and the S-box output as intermediate value. Using the script, I was able to recover the key (0xb1, 0x96, 0x18, 0x9f, 0x4c, 0x52, 0xd1, 0x2c, 0x65, 0x6a, 0x4f, 0x54, 0xaf, 0x58, 0x06, 0xbf) of the AES implementation using only 50 traces, as shown in Figure 1.

Proposed Improvement: The two seeds should be initialized to a random value using a proper PRNG. By initializing the two seeds to a random value the AES implementation will be protected from DPA attacks of first-order since the initial values are protected by a mask [MOP07](page 245).

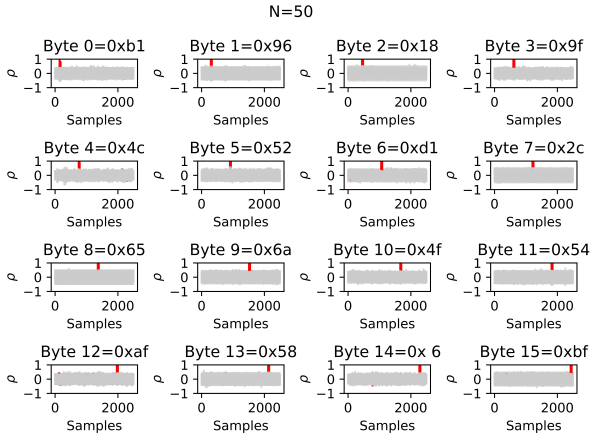

Figure 1: DPA attack of first-order on the AES implementation

2.5 AES Implementation, number of rounds in masked-combined.c

Problem Description: The number of rounds of the AES implementation is 9, which is not compliant to the AES standard that specifies 10 rounds for a 128-bit key.

Demonstrating this as a vulnerability: In the AES implementation the number of rounds is 9. This type of implementation was proven to be vulnerable to known key distinguisher attacks [**?**]. In this setting

2.6 AES Implementation, masked-combined.c

Problem Description: The implementation of the AES algorithm is not protected against fault attacks in which the memory can be corrupted (i.e. bit flipping), and against adaptive chosen plaintext attacks.

Demonstrating this as a vulnerability: Assumed that an attacker can get physical access to the device, either the reader or the token, he can perform a fault injection on the AES implementation. To do so, the attacker can use a fault injection tool like a laser beam to flip a bit in the memory of the device or a voltage glitching tool to obtain the same result. Since AES is implemented in ECB mode, using the same key and the same plaintext produce the same ciphertext evey time. By altering the plaintext, the ciphertext or the key, accordingly to the scenario of the attack, the attacker can recover the key of the AES implementation. The bit flipping attacks aim to show a relation between the change of the value in one of the secrets and the generation of an output value different from the one previously recorded without the modification.

The adaptive chosen plaintext attack is a variant of the chosen plaintext attack where the attacker can choose a specific well-known plaintext to prepend to the chosen plaintext to attack and encrypting them with the key already stored in the device. By shifting to

the left the well-known plaintext by one byte at the time, the attacker can brute force the original plaintext by observing the ciphertexts.

Proposed Improvement: The AES implementation should be protected against fault attacks by using a fault-tolerant mechanism like the one proposed in [**?**]. This would prevent especially the token from being vulnerable to tampering by an attacker. To prevent the adaptive chosen plaintext attack, the AES implementation should be implemented in CBC mode. Important to notice that the CBC mode is not much more secure than the ECB mode. If the hardware is sufficiently powerful, a version of AES with authenticated encryption (GCM or CCM) should be used, to increase even more the security of the AES implementation.

References

[BRS+10] L E Bassham, A L Rukhin, J Soto, J R Nechvatal, M E Smid, E B Barker, S D Leigh, M Levenson, M Vangel, D L Banks, N A Heckert, J F Dray, and S Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical Report NIST SP 800-22r1a, National Institute of Standards and Technology, Gaithersburg, MD, 2010.

[KK03] Mika Kojo and Tero Kivinen. More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). Request for Comments RFC 3526, Internet Engineering Task Force, May 2003.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: revealing the secrets of smart cards*. Springer, New York, 2007. OCLC: ocm71541637.

[MVOV97] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. CRC Press, Boca Raton, 1997.

[Nat01] National Institute of Standards and Technology. Security requirements for cryptographic modules. Technical Report NIST FIPS 140-2, National Institute of Standards and Technology, Gaithersburg, MD, May 2001.

[ZPFB10] Zachary Zeltsan, Sarvar Patel, Igor Faynberg, and Alec Brusilovsky. Password-Authenticated Key (PAK) Diffie-Hellman Exchange. Request for Comments RFC 5683, Internet Engineering Task Force, February 2010.