Hardware-Oriented Security and Trust
ECE 4156 HST / ECE 6156 HST
Spring 2024

Assoc. Prof. Vincent John Mooney III
Georgia Institute of Technology

Homework 7, 110 pts. (ECE 4156) 115 pts. (ECE 6156)
Due Friday March 29 prior to 11:55pm

1) (20 pts.) Read Section 5.2 and Appendix A of the Federal Information Processing Standard (FIPS) Publication 197, "ADVANCED ENCRYPTION STANDARD (AES)," in preparation for this question. For a 128-bit AES key please describe the Key Expansion step in response to the following two questions.

   a. (10 pts.) Describe how key expansion works using pseudocode. You may modify Figure 11 of the AES standard, but make sure **not** to include elements (i.e., variables and/or code) needed for 192-bit or 256-bit keys; instead, **only** provide pseudocode for the 128-bit key case. Please describe how the pseudocode works with a paragraph or two (or three but length is **not** required). Make sure to clearly define all aspects of your pseudocode so that a reader with a bachelor's degree in ECE or CS could follow your description without being required to read the AES standard.

Solution

Psuedo Code

Input: key of size 16 bytes denoted $k_0, k_1, \ldots, k_{15}$
Output: key of size 176 bytes denoted $w_0, w_1, \ldots, w_{175}$
Let there be 10 variables
Variables:
rc1 = 0x01000000, rc2 = 0x02000000, rc3 = 0x04000000, rc4 = 0x08000000,
rc5 = 0x10000000, rc6 = 0x20000000, rc7 = 0x40000000, rc8 = 0x80000000,
rc9 = 0x1B000000, rc10 = 0x36000000
Functions:
SubWord: 4 bytes → 4 bytes
$a_0, a_1, a_2, a_3 \mapsto Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3)$
RotWord: 4 bytes 4 bytes
$a_0, a_1, a_2, a_3 \mapsto a_1, a_2, a_3, a_0$
KeyExpansion: 16 bytes → 176 bytes
for (i=0 to 3) {$w_{4i} = k_{4i}$; $w_{4i+1} = k_{4i+1}$; $w_{4i+2} = k_{4i+2}$; $w_{4i+3} = k_{4i+3}$;}
for (i=4 to 43) {
if (i is divisible by 4) {
temp = SubWord(RotWord(w_{4i-4} , w_{4i-3} ,w_{4i-2}, w_{4i-1})) ⊕ ($rc_{(i/4)-3}$, $rc_{(i/4)-2}$, $rc_{(i/4)-1}$, $rc_{(i/4)}$);
} else { temp = (w_{4i-4} , w_{4i-3} ,w_{4i-2}, w_{4i-1}); }
(w_{4i} , w_{4i+1} ,w_{4i+2}, w_{4i+3}) = temp ⊕ (w_{4i-16} , w_{4i-15} ,w_{4i-14}, w_{4i-13});
}

In the pseudocode on the previous page, the first 128 bits of the expansion is simply the 128-bit key. The next 128 bits uses the rotation operation RotWord(), the nonlinear substitution operation SubWord() and the exclusive-or (XOR) operation. The nonlinear SubWord operation prevents compressing the expansion into a linear function.

b. (10 pts.) Now describe intuitively how entropy / randomness is maintained. In other words, assuming that the initial 128-bit number is a (truly) random number, comment on how an adversary might try to predict the individual key expansion step results given that the adversary does not know the initial 128-bit key. (Note that you are being asked to think about cryptanalysis in this question; due to lack of time, unfortunately this course does **not** cover cryptanalysis in general, although obviously this question is asking you to think about a very specific subcase of cryptanalysis, i.e., w.r.t. key expansion in AES.) Explain at least one clear reason why, intuitively, entropy / randomness is not decreased.

Solution

By far the most important reason that entropy or randomness is not decreased is the **nonlinear SubWord operation**. The RotWord and XOR operations diffuse the bit flips across expansion but are linear and therefore reversible.

Because the S-box used is known once the key is known so is the expansion to 128*11 = 1,408 bits. Therefore, **entropy is not increased either**.

2) (20 pts.) During your next job interview you mention that you learned the internals of AES, and so you are asked a series of questions as follows:

a. (5 pts.) What are the different key sizes supported and what is the advantage of maintaining the same plaintext to ciphertext size of 128 bits while supporting increasing key sizes?

Solution

AES supports key sizes of 128, 192 and 248. The main advantage of supporting the same plaintext to ciphertext size of 128 bits is compatibility and especially **backward compatibility**; in other words, for example, if in the future 128-bit keys are no longer secure (i.e., can be broken in a reasonable amount of time, e.g., less than a week), then the same applications with the same packet structures and data structures may continue to be used with only a change in key size used to 192 (or 248).

b. (5 pts.) In the final round MixColumns is not performed – why not?

Solution
It was found that in the final round MixColumns only adds a **linear step** to the complexity of a brute force attack on AES. Specifically, as MixColumns consists of multiplication by a fixed & known matrix, the attacker can quickly reverse the MixColumns operation. Therefore, it was not considered advantageous to keep MixColumns in the final round of the AES algorithm published by NIST as a cryptographic standard.

c. (5 pts.) What cryptographic property do ShiftRows and MixColumns provide to AES, and why is this property important?

Solution
ShiftRows and MixColumns both provide the cryptographic property of **diffusion** which means that changes in bit values in specific plaintext locations impact a wide variety of ciphertext bit values. This is also known as the **avalanche effect**, i.e., all the ciphertext bit values depend on all the plaintext bit values, and so a change of any bit of plaintext has the potential to change any of the ciphertext bits.

d. (5 pts.) Are any of the AES operations (e.g., SubBytes or ShiftRows) non-invertible – i.e., without an inverse? If so, which operations do not have an inverse and why? If on the other hand all operations are invertible, why do you think that this is so?

Solution
The four operations repeated each round – SubBytes, ShiftRows, MixColumns and AddRoundKey – are all invertible.
The main reasons are the following:

a. all operations used (e.g., XOR) are **linear**
b. all operations are **bijective**

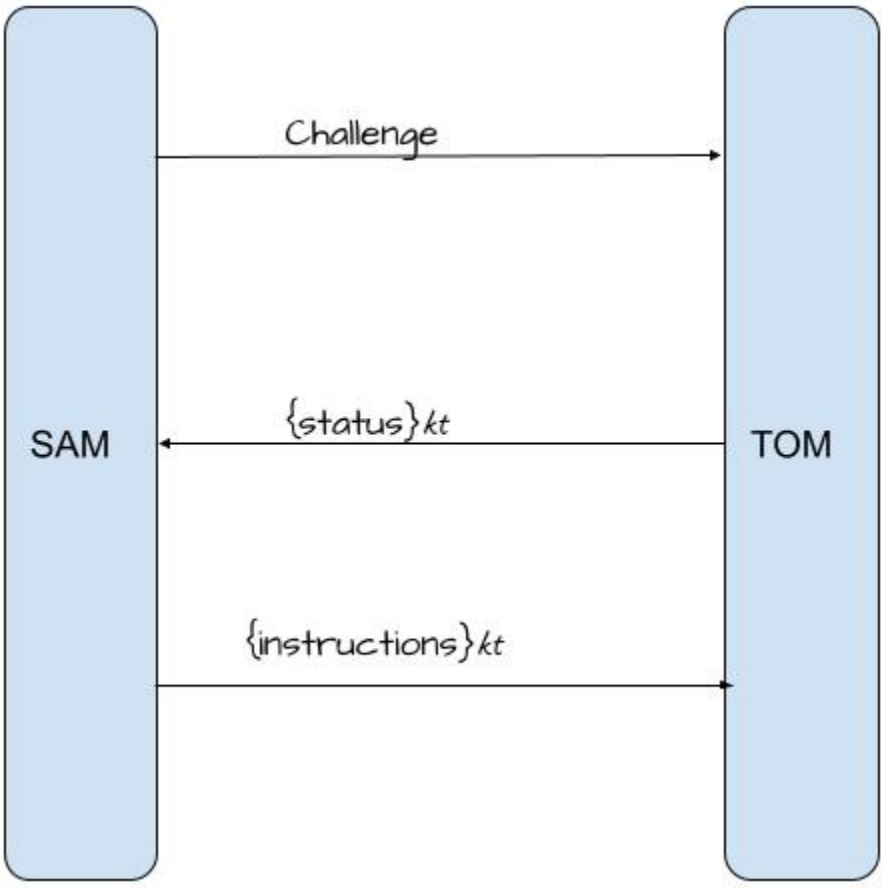
KeyExpansion is not bijective (i.e., the number of input and output bits at each step is not identical), but the first four words contain the key; thus, KeyExpansion is also invertible

3) (45 pts.) Consider the following scenario. A server SAM is communicating with an IoT device TOM. TOM has a PUF. SAM is provided with 2^{30} challenge-response pairs from the enrollment process for TOM's PUF. Each challenge is 128 bits and each response is 128 bits.

You are supervising a team where one of your superstar engineers, Alex, comes up with an idea. Alex proposes that PUF-based encryption can serve both as an encryption method as well as an authentication method at the same time. Specifically, SAM sends a challenge to TOM. TOM uses the 128-bit PUF response *kt* to encrypt a message m_{status}. Specifically, TOM sends SAM the following: $\{m_{status}\}_{kt}$. Based on the decrypted value of m_{status}, SAM sends TOM instructions $m_{instructions}$ as follows: $\{m_{instructions}\}_{kt}$.

a. (15 pts.) Describe the protocol using a diagram similar to what was shown in Authentication Part II. Explain each step in the protocol.

Solution

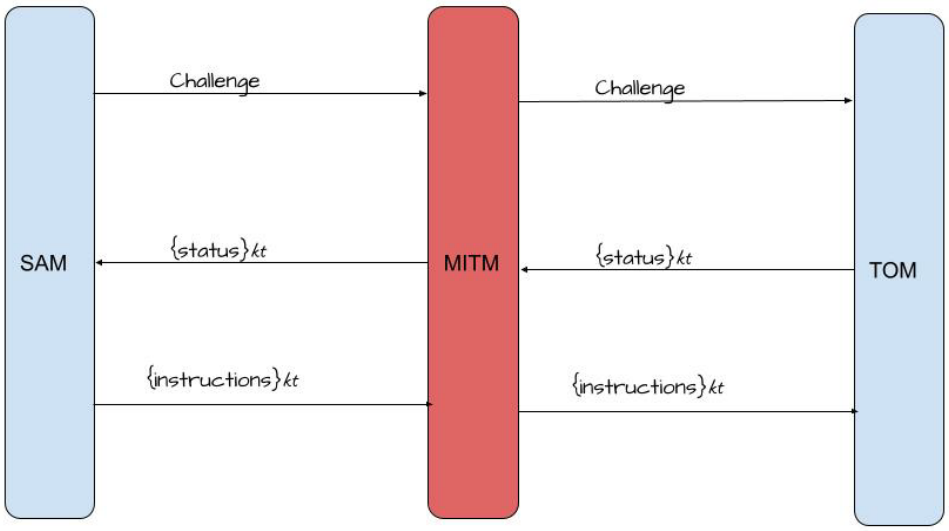


Protocol Steps

- The sequence starts with SAM sending a Challenge to TOM.
- Based on Challenge, TOM uses a PUF to calculate kt.
- TOM has an AES engine and uses kt to encrypt status and send the encrypted status to SAM.
- SAM decrypts status since SAM has a set of challenge-response pairs from TOM's enrollment process.
- SAM sends instructions to TOM encrypted with AES using kt.

b. (15 pts.) Is the protocol vulnerable to a Man-in-the-Middle (MITM) attack? Please redraw the diagram from your answer to part a. Show the best effort you can to attack using MITM. Explain whether the attack works or not with details.

Solution

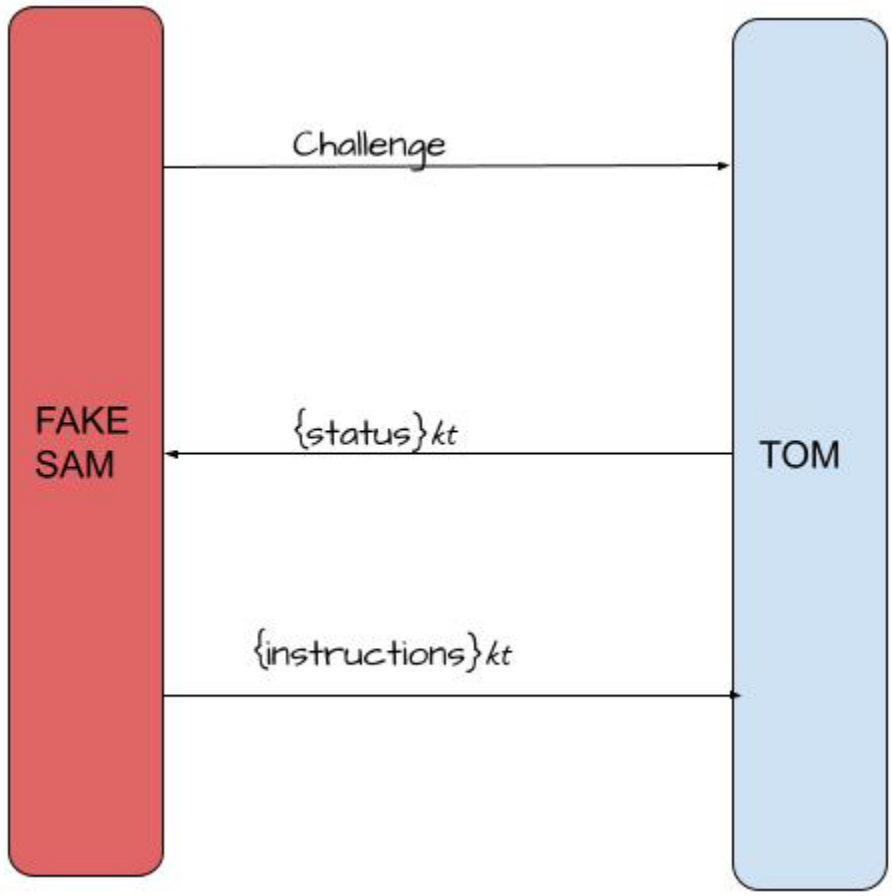


For the MITM attack to succeed, the Man-In-The-Middle needs to be able to pretend to be each person/entity and obtain key information to decrypt communications. An "attack" where all that is gained is encrypted traffic and no key information is not a **successful MITM attack**. The diagram above shows a MITM attempting to attack. Since there is no key exchange via messages, there is no opportunity for the MITM to obtain key information. Without key information, there is no MITM attack opportunity.

c. (15 pts.) Is the protocol vulnerable to a replay attack? Please redraw the diagram from your answer to part a. Show the best effort you can to attack using replay. Explain whether the attack works or not with details. Assume that TOM does not store all past challenges received.

Solution

There is a major assumption that must be made, and which changes the answer to this question. The assumption is regarding TOM: does TOM keep a record of all challenges received so far or not? If the assumption is that **TOM lacks sufficient memory** to record all possible challenges received to date, then **a replay attack is possible**. Specifically, an adversary who eavesdrops on communication can collect Challenge and instructions encrypted with AES using kt and replay them as follows:



How can this be useful? One possibility is that if there are a **reasonably finite number of status possibilities**, a codebook can be collected by the replay attacker where an encrypted status response is mapped against TOM's objective status. For example, in WWII the U.S. figured out an encrypted form of "Midway" by pretending to be low on water at Midway. In any case, if TOM is not aware that the challenge was already used previously, then TOM does not realize that communication was not with SAM.

4) (25 pts.) During class Professor Mooney explained why it is possible to model-build the Arbiter PUF. Use your own words to explain how an adversary can "learn" the challenge-response space of a 100-bit Arbiter PUF. As in the previous homework question, you are on a job interview where there is great interest in probing your understanding of PUFs in general with the Arbiter PUF as a specific example. You are asked to provide a detailed explanation of why far less than 2^{100} challenges suffices to learn the challenge-response space. Your answer should include (i) what is the underlying source of entropy or randomness, (ii) how long a typical adversary may be in possession of the microchips each of which has an Arbiter PUF to be learned, (iii) what is the overall technique applied by the adversary, and finally (iv) a few specific examples of challenge pairs that the adversary may apply to learn some specific parameter or parameters of the Arbiter PUF challenge-response model being built.
(Note: Try to write down your best answer in two pages or less please!)

Solution

(i) The underlying source of entropy is manufacturing variations in the switch elements, specifically, transistor delays primarily with metal as a much more minute impact (less than 1% of the impact of the transistor variations). Since each switch element contains four connections each with a variable delay, for n switch elements there are at most $4n$ entropy sources.

(ii) We stated in class that we assume that the fabrication facility is distinct from the company that designed the PUF and placed it on the chip being fabricated. We further stated that the fabrication usually takes several months to produce a chip after receiving the physical GDSII design files, so a reasonable assumption is that the adversary has access to the chip for a month.

A lot of students have answered that a short period of time is enough. But a short period is relative and needs to be explained as to how many days/ hours/ weeks are needed for the adversary to learn the Arbiter PUF.

(iii) The overall technique is to provide pairs of challenges to gain information regarding relative magnitude delays of specific switch elements.

Many students have answered using Machine Learning techniques they can understand the PUF. As long as the PUF's response is being learnt and understood the answer is valid.

(iv) For example, consider two challenges which differ only in the very first bit. In this case, the response will compare the same overall path except for the paths through the first switch element will have been reversed

5) [ECE 6156 only!] (5 pts.) In Lecture 19, some of the NIST tests for randomness were described. What does the following statement mean (quoting from the abstract of the NIST document): "However, no set of statistical tests can absolutely certify a generator as appropriate for usage in a particular application, i.e., statistical testing cannot serve as a substitute for cryptanalysis." Please explain in your own words what this means; do not quote from any other source but instead do your best to express the concerns being raised here. You may not give an answer longer than 10 sentences; if you do, please circle the 10 sentences you want graded or else you will receive a zero.

Solution

NIST wants to protect itself from a lawsuit where a random number generator technology has passed all the NIST tests yet is broken by an adversary leading to a successful cyber-attack due to the broken random number generator.

Some students said that the goal of cryptanalysis is to attack random number generators. It is more accurate to say that the goal of cryptanalysis is to attack a cryptosystem; one way to help achieve this goal is if the "random numbers" used are not in fact truly random resulting in the cryptanalyst having a way to at least partially predict the values of the "random numbers."

Additionally, some students said that while NIST cannot guarantee that a random number generator is true, cryptanalysis can; I am not convinced. If you know of ways to improve upon NIST's tests for randomness, you can probably become famous very quickly, so please come talk to me as I would like to join you in this if you will let me do so!!