Security Protocols and Application — Final Exam
Solution

Philippe Oechslin and Serge Vaudenay

28.6.2017

- duration: 3h00
- no document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will not answer any technical question during the exam
- the answers to each exercise must be provided on separate sheets
- readability and style of writing will be part of the grade
- do not forget to put your name on every sheet!

The exam grade follows a linear scale. In each exercise, each question has the same weight. Both exercises have the same weight.

1 Reconsidering generic Composition

This exercise is inspired from Namprempre-Rogaway-Shrimpton, Reconsidering Generic Composition*, EUROCRYPT 2014, LNCS vol. 8441, Springer.*

Q.1 Explain the following acronyms: AE, MAC, MtE, IV.
Say why some ISO 19772 readers have a flat nose.

Authenticated Encryption, Message Authentication Code, MAC-then-Encrypt, Initialization Vector.
Despite all studies, the ISO 19972 standard used the notion of "starting value" by failing to say if it is an IV or a nonce, by failing to say how it is synchronized (or transmitted), and by failing to say how to behave in the case of a padding error. On the presentation of this lecture, the speaker has shown a picture of a guy (Richard Stallman) facepalming after listing the above problems with the standard. Hence flat noses.

Q.2 Briefly explain what is the old Bellare-Namprempre result from 2000 and why it is not enough.

The Bellare-Namprempre (BN) result considers three generic compositions: encrypt-then-MAC, MAC-then-encrypt, and encrypt-and-MAC. It concludes by favoring the encrypt-then-MAC construction. However, this result assumes that the encryption selects his IV *securely. In practice,* IV *is an independent object which may somehow be controlled by the adversary and should be transmitted with the ciphertext. The BN result fails to capture this notion of* IV*. In addition to this, there could be nonces which are not necessarily like* IV*. An* IV *is just a random value. A nonce is any non-repeating value. It could be a counter.*

Q.3 We consider two keys K and L and two algorithms: one $\mathcal{E}_K(\mathsf{IV}, M)$ to encrypt a message M with IV (it produces ciphertexts of exact same length as M); one pseudorandom function $F_L()$ taking several inputs and producing a tag of fixed length. Consider the following constructions:

$$C = \mathcal{E}_K(F_L(N,A),M) \qquad T = F_L(N,A) \tag{1}$$

$$C = \mathcal{E}_K(F_L(N),M\|F_L(N,M)) \tag{2}$$

$$C = \mathcal{E}_K(F_L(N,M),M) \qquad T = F_L(N,C,A) \tag{3}$$

with nonce N, message M, and associated data A. For each construction, explain how to decrypt and why it is a bad construction.

For construction (1), we decrypt (N,A,C,T) by computing $M = \mathcal{D}_K(F_L(N,A),C)$ and checking $T = F_L(N,A)$. It is bad because authentication is too weak: an adversary who gets (N,A,C,T) can forge (N,A,C',T) by taking a random C'. The tag T will be correct as $T = F_L(N,A)$ and C' will decrypt to something random which is unlikely to be M.
For construction (2), we decrypt $M\|T = \mathcal{D}_K(F_L(N),C)$, extract T, and check $T = F_L(N,M)$. It is bad because A is not used, so not authenticated.
For construction (3), we can check $T = F_L(N,C,A)$ but we cannot decrypt C as IV is computed based on the result of decryption.

Q.4 With the same notations as above, consider the SIV mode (A4)

$$C = \mathcal{E}_K(F_L(N,A,M),M) \quad T = F_L(N,A,M)$$

Give a forgery attack of probability of success $1-(1-2^{-n})^q$, where q is the number of queries and n is the output length of F_L. (Assume an adversary who can make chosen plaintext and chosen ciphertext queries.)
Compare this scheme with the following one (A2), in terms of functional and security properties:

$$C = \mathcal{E}_K(F_L(N,A),M) \quad T = F_L(N,A,M)$$

We iterate q times what follows: we pick a random (N,A,C,T) and make a decryption query. Each fails with probability $1-2^{-n}$. So, at least one passes with probability $1-(1-2^{-n})^q$. This one which passes is a forgery.
The advantage of A2 is that we can decide to truncate T to the size we prefer. We cannot in A4 as we need the full T as an IV to decrypt. In addition to this, C and T can be computed in parallel in A2 but not in A4. One advantage of A4 is that it has a nonce-misuse resistance *property which was proven. Another advantage is that it needs only one F_L evaluation.*

2 Automotive Remote Keyless Entries

This exercise is inspired from Garcia-Kasper-Pavlides, Lock it and Still Lose it*, USENIX Security Symposium, 2016.*

Q.1 The most convenient car keys are the ones that you don't even need to take out of your pocket to open the car. They are called PKES or smart keys.

- Describe an attack to which this type of keys are vulnerable
- How can the owner of this type of keys prevent this attack.

- *Relay attack. One attacker can stand close to the victim while another one is close to the car. They can then relay the signals transmitted by the car or the key. For the car the situation is thus exactly the same as if the key was close to the car.*
- *The user could store the card in a metallic envelope. This would prevent any unwanted radio communications of the key. Metallic wallets are already used for contactless credit cards. It somehow defeats the purpose of smart keys, as the driver needs to take the card out of the envelope every time he or she wants to enter the car.*

Q.2 A typical rolling code contains the following elements:

$$\langle UID \parallel btn \parallel MAC_{keyUID}(btn, ctr) \rangle$$

- For each parameter, describe what it represents and explain why it is needed.

- *UID identifies the car, such that the cars know which one is addressed by the messages*
- *btn describes which button was pressed, such that the car knows what function it has to execute (lock, unlock, open trunk, ...)*
- *$keyUID$ is a cryptographic key used to calculate the a message authentication code. This proves that the message was generated by a car key that knows the correct cryptographic key.*
- *ctr is a counter that is incremented with each button press. If the car keeps track of the last value of ctr that was used, it can reject messages that are being replayed.*

Q.3 Now assume your car has two different keys that use rolling codes of the format described above.

- Describe a method with which the car can accept messages from both keys and still not be vulnerable to a replay attack.

- *If the counter space is large enough, the keys could start with two different values for the counters such that one counter could never catch up with the other one. The car then needs to remember the last value of the counter that it saw for each key.*
- *The btn parameter could be different on each key. E.g. values 0,1,2 belong to key 1, 3,4,5 to key 2. The car keeps two counters, one for each set of buttons.*

Q.4 The VW-1 scheme uses the following message structure: $\langle f(UID) \parallel g(ctr) \parallel btn \rangle$
Its security was mainly based on the fact that f and g were unknown. Once the functions are known, it is claimed that the car can be opened after a single message has been eavesdropped.
Assume that the f and g functions are a modern cryptographic hash function H like SHA-2. You have eavesdropped a message while your neighbor used his key to open his car.

- Explain how you can use the eavesdropped message $\langle H(UID) \parallel H(ctr) \parallel btn \rangle$ to create a new message that will enable you to open the car.
- What condition must be met for your attack to succeed ?

- *Given $H(ctr)$ we can try to recover ctr by bruteforce. We can then send $\langle H(UID) \parallel H(ctr+1) \parallel btn \rangle$*
- *The domain of ctr must be small enough to be bruteforced.*

Q.5 The correlation attack on HiTag2 is based on the fact that partial guesses of the key can be classified according to a score. The function f takes 20 bits as inputs and outputs a single bit b.
The attacker guesses the first 8 inputs to the function f and calculates b for all possible combinations of the remaining 12 bits of input of the function.

- If the guess was wrong, how many times, in average, will the output b be correct for all possible values of the 12 remaining bits ?
- If the guess was right, how many times, in average, will the output b be correct for all possible values of the 12 remaining bits ?

- *If the first 8 inputs are not correct, then no combination of these bits with the remaining 12 bits will be correct. Each combination of 12 bits will thus yield an arbitrary value which can be expected to be correct 50% of the time. We thus expect to have $\frac{2^{12}}{2} = 2^{11}$ correct outputs.*
- *If the first 8 inputs are correct, one combination of the remaining 12 bits will create the correct 20 bit input. For this combination the output bit must be correct. For the remaining combinations we expect 50% of correct outputs. The results is thus $1 + \frac{2^{12}-1}{2} = 2^{11} + 0.5$*

Q.6 From the previous question we see that the difference in correlation scores between a correct guess and an incorrect guess is quite small. The authors say that the attack succeeds with as little as 4 captured messages.

- Using four messages, in how many different ways can a guess of 16 bits of the key be scored ? Explain briefly.
- Are all the scores of the same quality ? Explain briefly.

- *With each message we start with the 16 guessed key bytes being on the left side of the LSFR. After scoring the guess with the output b we can operate the shift register to create new output and score this output. Thus the 4 messages give us 16 possibilities each, for a total of 64.*
- *As we shift to the left, there are fewer input bits of the f function that depend on the guessed key bits. The difference of the score between a correct and a false guess becomes smaller.*

Q.7 HiTag2 uses keys of 48 bits. Such keys can be bruteforced with powerful computers.

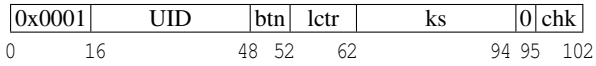

Fig. 1. message structure for Hitag2

Figure 1 shows the the different elements of a HiTag2 message and the number of bits of each element.

- How many messages do you need to eavesdrop in order to be able to run a bruteforce attack ?
- Describe your bruteforce attack.
- Give an estimate of the complexity of your attack

- *Each message only gives 32 bits of keystream. Thus we need two messages to recover 48 bits of the key plus the most significant bits (MSB) of the counter.*
- *For each possible value of the higher MSB of the counter and of the 48 bits of the key, we calculate the keystream using the least significant bits of the counter which are present in the messages. We search until we find a combination that yields the correct 32 bit keystream for both messages.*
- *According to the authors of the paper the higher order bits of the counter are never greater than 10. Thus there is a maximum number of 10 values for the MSB of ctr and 2^{48} possibilities for the key for a total complexity of $10x2^{48}$.*