Security Protocols and Application — Final Exam

Solution

Philippe Oechslin and Serge Vaudenay

16.6.2015

- 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 Relay Cost Bounding

This exercise is inspired by the article *Relay Cost Bounding* by Chothia *et al.*, Financial Cryptography 2015. Questions Q.1, Q.2, and Q.3 are independent.

Q.1 Assume that a credit card holder is quietly taking the metro in Lausanne while someone is willing to order some drinks at Satellite — a popular bar — and make the payment through a relay attack.

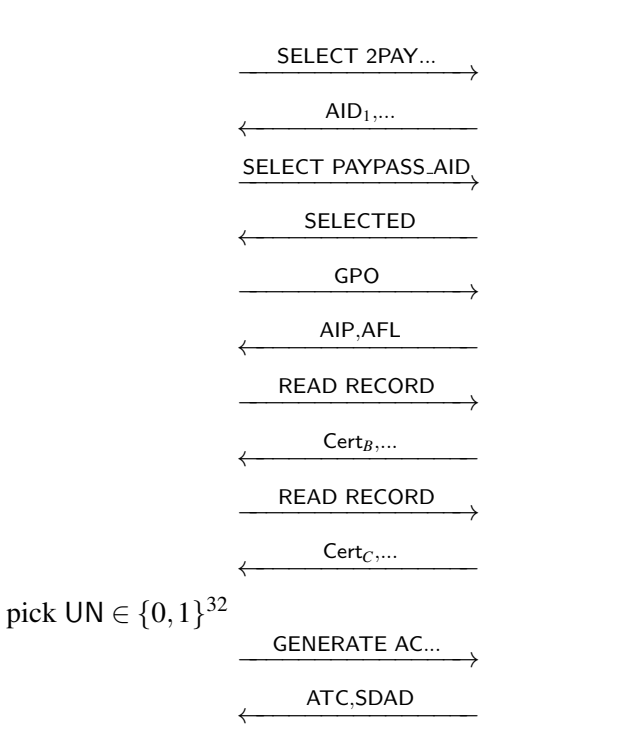
Q.1a Describe the attack scenario and how many people and devices are involved.

The guy at Satellite is paying with a smartphone or a fake NFC card which is communicating with the device of another malicious person in the metro. This person sticks to the victim with a NFC reader. Messages from the victim's card are relayed through these devices to the payment terminal at satellite and messages from the terminal are relayed back to the card.

Q.1b Assume that Satellite is located on the moon. Given that the Earth-Moon distance is of $384\,000\mathsf{km}$ and that the speed of light is of $300\,000\mathsf{km.s}^{-1}$, what communication delay would be introduced for each message from the card to the payment terminal and each message from the payment terminal to the card?

Assuming that communication follow the speed of light, it takes $\frac{384\,000}{300\,000} = 1.28\mathsf{s}$ *to travel from Earth to Moon or from Moon to Earth. So, a message takes* $1.28\mathsf{s}$ *more then usual to be delivered. On the terminal side, the additional visible delay is of* $2.56\mathsf{s}$.

Q.2 We consider the PayPass payment protocol. (The meaning of the messages is not important for this exercise.)



Given

- that standard equipments introduce a delay of 100ms on the top of the time of flight to relay a single message;
- that some cards take 637ms to complete all computations required on their side during the protocol;
- that payment terminals reject payments if the total protocol duration exceeds 1s;

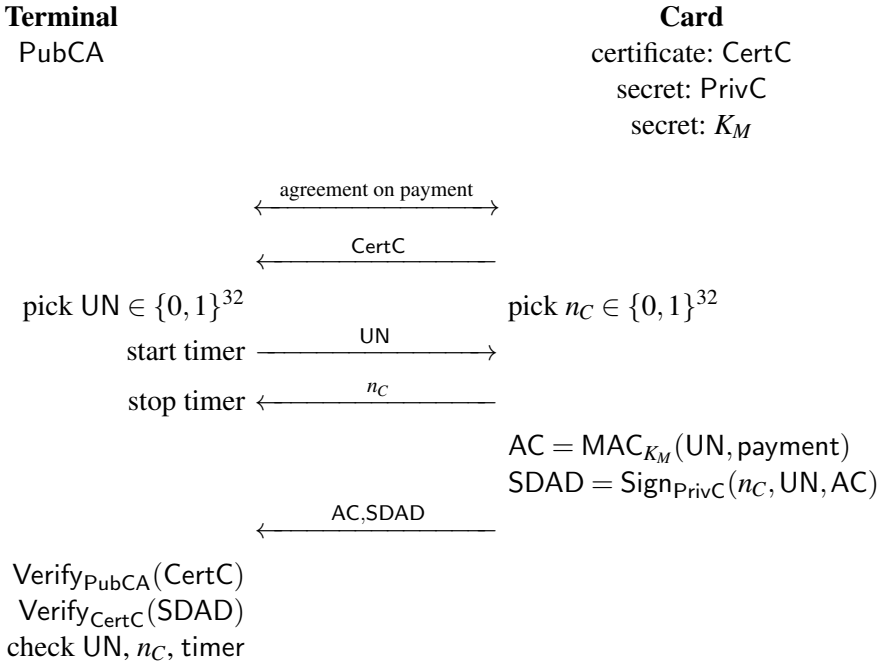
Q.2a show that the standard relay attack does not work.

Since we have 12 messages to exchange, the overhead will be of 1.2s*. So, the total time would be* $12 \times 100\text{ms} + 637\text{ms} = 1.837\text{s}$ *which exceeds the maximal duration. So, this passive relaying cannot work.*

Q.2b In their paper, how did Chothia *et al.* adapt the relay attack to make it work?

The answer to most of messages is easy to learn. So, the device at Satellite can just answer them at the same time as it relays them. It saves a lot of time. The delay will be needed only in a few critical messages which are unpredictable.
The authors also used tricks to reduce the overhead delay. For instance, they kept the WiFi interface working to prevent it from sleeping as waking it up introduces delays as well.

Q.3 We now consider a man-in-the-middle attack against the following payment protocol (which is a simplification of PaySafe). The terminal holds the root certificate PubCA of the Card's PKI. The card holds a certificate CertC (to be verified with PubCA) and a secret key PrivC which allows it to sign. Signatures are verified with CertC. The card also holds a secret key K_M which is shared with the bank. This key is used to authenticate a message by means of a MAC algorithm.



At the end, the terminal accepts the payment in SDAD if the certificate is valid, if the signature is valid, if the nonces in SDAD are correct, and if the elapsed time between sending UN and receiving n_C is lower than a given bound B.

Q.3a We consider any man-in-the-middle attack in which the man-in-the-middle sends some random UN to the card before he receives UN from the payment terminal. Show that the attack fails, except with small probability. (Make a detailed proof.)

The card will receive some UN *which is incorrect with probability* $1-2^{-32}$. *As this* UN *is included in the signature* SDAD*, the terminal will see that the* UN *is incorrect and abort.*

Q.3b We consider any man-in-the-middle attack in which the man-in-the-middle sends some random n_C to the payment terminal before he receives n_C from the card. Show that the attack fails, except with small probability. (Make a detailed proof.)

Similarly, the terminal will see in SDAD *that the* n_C *does not correspond to the one which was received, with probability* $1-2^{-32}$.

Q.3c Assuming that relaying a message with standard equipments introduces a delay of 100ms, adjust B and prove that no man-in-the-middle attack can break the protocol. (Make a detailed proof.)

Due to the two previous questions, the man-in-the-middle must wait for UN *from the terminal before he sends it to the card and he must wait for* n_C *from the card to send it to the terminal. He further must relay the correct* UN *and* n_C. *So, he must introduce a delay of* 200ms. *Hence, by taking* $B = 150$ms *the attack is defeated by the protocol. Normally,* 150ms *is enough for a slow card to receive 32 bits and send 32 bits which are already prepared in a register.*

2 Heartbleed

Q.1 Below is the source code of the OpenSSL library that includes the Heartbleed bug.

```
1455 dtls1_process_heartbeat(SSL *s)
1456   {
1457   unsigned char *p = &s->s3->rrec.data[0], *pl;
1458   unsigned short hbtype;
1459   unsigned int payload;
1460   unsigned int padding = 16; /* Use minimum padding */
1461
1462   /* Read type and payload length first */
1463   hbtype = *p++;
1464   n2s(p, payload);
1465   pl = p;
1466
1467   if (s->msg_callback)
1468         s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
1469               &s->s3->rrec.data[0], s->s3->rrec.length,
1470               s, s->msg_callback_arg);
1471
1472   if (hbtype == TLS1_HB_REQUEST)
1473         {
1474         unsigned char *buffer, *bp;
1475         int r;
1476
1477         /* Allocate memory for the response, size is 1 byte
1478          * message type, plus 2 bytes payload length, plus
1479          * payload, plus padding
1480          */
1481         buffer = OPENSSL_malloc(1 + 2 + payload + padding);
1482         bp = buffer;
1483
1484         /* Enter response type, length and copy payload */
1485         *bp++ = TLS1_HB_RESPONSE;
1486         s2n(payload, bp);
1487         memcpy(bp, pl, payload);
1488         bp += payload;
```

Q.1a Mark the lines were the error occurs and explain what is wrong with it.

The amount of data copied into the response at line 1487 is equal to the payload length declared in the received request (variable payload on line 1464. If this is more than the actual length of the request, extra data will be read from memory and copied into the response.

Q.1b Heartbleed leaks memory. Which part of memory is being leaked ?

The memory area where the incoming request are being stored. This is basically the heap (memory managed by malloc) plus some cache area tha is managed by OpenSSL itself. .

Q.2 Exploiting Heartbleed:

Q.2a Describe a scenario in which an attacker makes use of Heartbleed to access a victim's e-banking application even if that application uses two-factor authentication.

The attacker find a valid session cookie in the memory of a web server and use that cookie to access the victim's session. Thus, the attacker does not need to go through authentication.

Q.2b Describe a scenario which explains why the private key of a web server can be retrieved using Heartbleed even if a Diffie-Hellman key exchange is used and thus the private key is not used for the key exchange.

The private key is used by the server to prove that he is the legitimate holder of the server certificate. When the TLS connection is negotiated, the server signs some information with the private key. The private key thus ends up in memory.

Q.2c Why is it not sufficient to overwrite the private key after it has been used, to remove it from memory ?

The library that is used to calculate the signature makes intermediate copies of the key. Even if the program overwrites the buffer in which it stores the key, there can still be buffers used by the library that are not being overwritten.

Q.3 Preventing Heartbleed:

Q.3a Administrators are told that they need to update their software regularly to avoid security issues. Explain why in the case of Heartbleed, people who update regularly were more impacted than others.

Heartbeat is a new feature that has only existed for two years. People who did not update in the last two years were not vulnerable.

Q.3b Several groups of developers have decided to fork their own version of OpenSSL. Cite the name of two such projects or groups:

- *LibreSSL (OpenBSD)*
- *BoringSSL (Google)*

Q.3c Do you believe that having different versions of OpenSSL that are developed separately will increase its security ? Give a justification for your answer.

If a error is detected, it may only exist in one version of the library. The impact would thus be smaller. If it turns out that the different versions of OpenSSL have a different level of quality, the users could chose the better of the three.

Q.4 What good is it for anyways ?

Q.4a Give one in example in which using a heartbeat request is more useful than doing a simple ping at the network level.

A ping packet only creates activity at the network level. The main goal of the heratbeat is verify that a DTLS conenction (e.g. TLS over UDP) is still alive and keeping it alive to prevent a timeout which would make a new TLS negociation necessary.

Q.4b Give one example in which it is useful to have a mechanism for sending heartbeat packets of *variable* length.

One reason for sending variable sized heartbeat messages is to discover the largest message that can be sent without being truncated (path MTU discovery) in datagram (UDP) based TLS.
Another reason could be to test encryption and decryption of data of variable size to detect issues with padding.