Tutorial 1

Shen Jiamin

Question 1

Alice was the Web administrator of the company VIC. A malicious attacker sent an email to Alice. The email instructed Alice to click on a link so as to login to the human resource (HR) system to view a report. In the email, information on the "sender" indicated it was from the HR manager in VIC. Alice wrongly believed that the email was indeed sent by the manager and followed the instructions. In doing so, she revealed her password to the attacker. Using Alice's password, the attacker logged-in to the web-server, and invoked many processes. As a result, the server was overloaded.

With respect to the security requirements mentioned in the lecture (confidentiality, integrity, authentication, availability, etc), discussed what aspects of security were compromised.

Answer The violated security aspects and offending actions are as follows.

1. (Authenticity) The email claims to be from the HR manager in VIC, but it was actually sent by an attacker.
2. (Authenticity) The website pretends to be the legitimate HR system, but it is actually controlled by the attacker.
3. (Confidentiality) Alice revealed her password to the attacker.
4. (Authenticity) The attacker was able to impersonate Alice and log-in to the web server.
5. (Integrity) The attacker invoked malicious processes on the web server, which compromised the server's process integrity.
6. (Availability) The web server was overloaded and could not serve legitimate users.

Question 2

Suppose it takes 512 clock cycles to test whether a 64-bit cryptographic key is correct, when given a 64-bit plaintext and the corresponding ciphertext (known plaintext attack).

(a) How long does it take to exhaustively check all the keys using a 4 GHz (single-core) processor?

(b) How long does it take on a cluster of 1024 servers, each with a quad-core 4 GHz processor.

(c) Bitcoin Network hash rate is the number of Tera cryptographic hashes carried out by all the bitcoin "miners" in the world in one second (so, hash rate of 1 means 1 T operations per second). Find out current hash rate. Suppose one cryptographic hash is equivalent to one test of the key, how long would the Bitcoin Network take to check all the 64-bit keys?

(Note: Let's approximate 1 year $\approx 2^{25}$ seconds. In this course, we follow the convention where $1\ K = 2^{10}$, $1\ M = 2^{20}$, $1\ G = 2^{30}$, $1\ T = 2^{40}$.)

> An important approximation: $2^{10} \approx 10^3$.

Answer

(a) Notice that a 4 GHz CPU core has $4 \cdot 2^{30} = 2^{32}$ cycles per second.
Checking 1 key takes $512 = 2^9$ cycles.
Checking 2^{64} keys would take $2^{64} \cdot 2^9 = 2^{73}$ cycles.
That is $2^{73}/2^{32} = 2^{41}$ seconds, or about 64 K years.

(b) Given 1024 servers, each with a quad-core processor, we have $1024 \cdot 4 = 2^{12}$ CPU cores.
The total time needed is now reduced by a factor of 2^{12} to become: $2^{16}/2^{12} \approx 2^4 \approx 16$ years.

(c) Current hash rate[1] is 944.9 EH/s. Let's take it as 1024 EH/s. That is, $2^{10} \times 2^{60} = 2^{70}$ H/s.
So, can break in $2^{64-70} = 2^{-6}$ seconds.

> What about 128-bit keys? The total number of keys is 2^{128}. It takes $2^{128-70} = 2^{58}$ seconds or approximately $2^{58-25} = 2^{33}$ years to check all keys.

[1] https://www.blockchain.com/explorer/charts/hash-rate

Question 3

Suppose it takes 512 clock cycles to test whether a 42-bit cryptographic key is correct, when given a plaintext m and the corresponding ciphertext c.

How long does it take to exhaustively check all the keys using a 4 GHz (single-core) processor?

Answer $2^{42} \cdot 512/(4 \cdot 2^{30}) = 2^{42+9-32} = 2^{19}$ seconds, which is approximately 145 hours.

A walkie-talkie system realtime Secure Walkie Talkie (rSWT) secures its communication using symmetric keys encryption. rSWT uses two encryption schemes, AES block cipher, and another fast stream cipher developed by the company called FAST. The cipher FAST is really fast, but its key length is only 42 bits.

To setup a group of walkie-talkies, the user enters k, a 128-bit key, into each walkie-talkie. The key k is called the long-term key. When a walkietalkie wants to broadcast a plaintext m, which is high quality audio, to other walkie-talkies, the followings are carried out.

(a) A 128-bit v is randomly chosen.
(b) Computes $t = \text{AES}_{\text{enc}}(k, v)$, where AES_{enc} is encryption of AES block cipher (without mode of operation).
(c) Obtains $\tilde{k}$, which is the first 42 leading bits of t. This $\tilde{k}$ is called the session key.
(d) Computes $c = \text{FAST}_{\text{enc}}(\tilde{k}, \mathbf{0}_{64} \| m)$, where $\mathbf{0}_{64}$ is a string of 64 zeros, $\|$ is string concatenation, and FAST_{enc} is the deterministic encryption of FAST. Note that c does not contain any IV (initialisation vector).
(e) Sends $(v \| c)$ over the air.

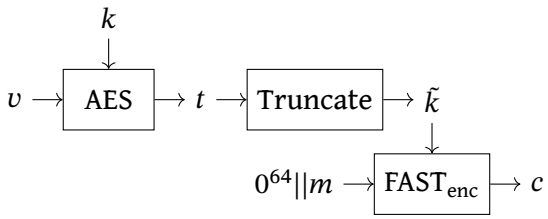
When a new message is to be sent, the above will be repeated. As such, the session key likely to be different for different messages.

The receiver extracts $\tilde{k}$ from v using the long term key k, and then decrypts using FAST. If the leading 64 bits of the decrypted message is not all zeros, then the receiver plays an error message. Otherwise, the receiver plays the decrypted audio.

In practice, an attacker can eavesdrop signal transmitted over the air. Hence, any reasonable threat model should assume that the attackers can eavesdrop and can obtain $v \| c$.

Now, with $v \| c$ and knowledge that the leading 64 bits are zeros, can the attacker carry out exhaustive search to find the key?

Answer Yes, the attacker can carry out an exhaustive search to find the key. Given $v \| c$, the attacker can try all possible values of $\tilde{k}$ (which is 42 bits) and compute $\hat{m} = \text{FAST}_{\text{dec}}(\tilde{k}, c)$ for each $\tilde{k}$ until the leading 64 bits of $\hat{m}$ are all zeros.



We know that 42-bit is too short and thus the key can be broken. However, as calculated earlier, it would still take very long time by a laptop. In their marketing efforts, rSWT claims that 42-bit is sufficient for realtime applications. This is what appeared in their advertisement:

> *"42-bit is sufficient. By the time the message is maliciously decrypted, the message becomes useless".*

In this question, you play the role of an attacker. You want to design a hand-held device that is able to crack and obtain the plaintext in realtime. Specifically, when given the v and c, the device should derive the 42-bit session key readily within 0.1 second. The hand-held device can have computing resource comparable to a high-end laptop.

(a) Suggest a way to get the session key, assuming that the device has huge storage, say 100 TB.

(b) (optional, techniques required would be covered later) Give another method where the device has lower storage, say 1 TB.

(Hint: Use a pre-computed table.)

Answer Tradeoff the time with space, i.e. solving the problem in less time by using more storage space.

(a) Construct a table of $\text{FAST}_{\text{enc}}\left(\tilde{k}, \mathbf{0}_{64}\right)$ for all possible 2^{42} values of $\tilde{k}$. There are 2^{42} entries in the table, where each entry is 64-bit long as the result of encrypting the 64-bit of zeros. Hence, the derived table will take $2^{42} \cdot 64$ bits $= 2^{42} \cdot 8$ bytes $= 32$ TB.

Algorithm 1 Construct Lookup Table

```
Initialize table T with size 2^42
for k̃ ∈ {0, 1, ... , 2^42 − 1} do
    T[k̃] ← FAST_enc(k̃, 0_64)
end for
```

Algorithm 2 Lookup Table Attack

Input: $v \| c$, T

```
for k̃ ∈ {0, 1, ... , 2^42 − 1} do
    if T[k̃] = c[0..64] then
        return k̃
    end if
end for
```

To break a SWT communication, first extract the first 64 bits of the captured ciphertext, then perform a lookup operation on the constructed table in order to determine the employed $\tilde{k}$.

Given $\tilde{k}$, the attacker can perform a decryption process using the stream cipher, thus recovering the plaintext in realtime.

Remark: We still need a data-structure to lookup the key. One method is to employ hash-table that can have "constant-time" We omit the details (algorithm & data-structure course).

(b) Rainbow table[2].

[2] https://en.wikipedia.org/wiki/Rainbow_table

Question 4

Lecture 1 mentioned that Winzip encrypts the compressed file. Why it is meaningless to carry out the two operations in the other way, that is, encrypts the file, and then compresses the encrypted file?

(Hint: Consider the effectiveness of compression on "random" sequences, and a requirement of encryption scheme.)

Answer Compressing an encrypted file will yield very little or no compression gain. This is since the encrypted file will resemble a "random" sequence (due to a requirement of a good encryption scheme).

A compression algorithm, which takes advantage of repeating patterns, therefore will not work well on an encrypted file.

Question 5

Bob encrypted a music mp3 file using Winzip, which employs the 256 bit key AES. He chose a 6-digit number as password. Winzip generated the 256-bit AES key from the 6-digit password using a (deterministic) function, say SHA1[a].

Alice obtained the ciphertext. Alice also knew that Bob used a 6-digit password and knew how Winzip generated the AES key.

(a) Given a 256-bit string, can Alice determine whether this string was indeed the correct AES key?

(b) How many guesses did Alice really need in order to get the mp3 file?

[a] We haven't introduce SHA1 yet. Here, just treat it as some routine that take in 6-digit as input, scramble them and output a 256-bit string.

Answer

(a) Yes.

Given a 256-bit string, Alice can use it to decrypt the ciphertext and get a plaintext. She can then check whether the decrypted plaintext follows mp3 format, or simply use an mp3 player to play it. If the plaintext is indeed a mp3 file, then the 256-bit string is very likely the correct AES key.

(b) The total number of guesses needed is only 10^6.

(Note: Why not 2^{256}?)