MAS 433 Tutorial 5

A+

Wang Xueou (087199E16)

October 12, 2011

Question 1 Solution:
1.1. The probability that 2 of 4 randomly selected people have the same birthday is

$$p = 1 - (1 - \frac{1}{365})(1 - \frac{2}{365})(1 - \frac{3}{365}) \approx 0.0164$$

1.2. The probability that 2 of 32 randomly selected people have the same birthday is

$$\begin{aligned} p &= 1 - (1 - \tfrac{1}{365})(1 - \tfrac{2}{365}) \cdots (1 - \tfrac{31}{365}) \\ &= 1 - \textstyle\prod_{i=1}^{31}(1 - \tfrac{i}{365}) \\ &\approx 1 - e^{\frac{-32\times(32-1)}{2\times365}} \\ &\approx 0.743 \end{aligned}$$

1.3. It is impossible for any two of them have the same matriculation number, so the probability is 0.

Question 2 Solution:
2.1. If only 0's are padded, there will be ambiguity in the last partial block. For exmaple, 2 different last partial blocks $m_{p1} = 10101$ and $m_{p2} = 1010100$ will be the same after padding. So if 2 messages are different in only the last partial blocks, the chance of partial blocks being exactly the same after padding increases. Consequently, the complexity of find a collision decreases.

2.2.
If the message length is 200 bits, it should be padded to 512 bits, constituting 1 message block. So 1 compression function operation is needed.
If the message length is 0 bit, it should be padded to 512 bits, constituting 1 message block. So 1 compression function operation is needed.
If the message length is 1 bit, it should be padded to 512 bits, constituting 1 message block. So 1 compression function operation is needed.
If the message length is 447 bit, it should be padded to 512 bits, constituting 1 message block. So 1 compression function operation is needed.
If the message length is 448 bit, it should be padded to 512×2 bits, constituting 2 message blocks. So 2 compression function operations are needed.
If the message length is 511 bit, it should be padded to 512×2 bits, constituting 2 message

blocks. So 2 compression function operations are needed.
If the message length is 512 bit, it should be padded to 512×2 bits, constituting 2 message blocks. So 2 compression function operations are needed.
If the message length is 960 bit, it should be padded to 512×3 bits, constituting 3 message blocks. So 3 compression function operations are needed.

2.3. The message digest of "Tutorial5.tex" is
"2c7c8f75f7dfe2877643b73a2698a6ff93754ac Tutorial5.tex"

2.4. The message digest of "Tutorial6.tex" is the same as "Tutorial5.tex".

2.5. After changing the submission deadline, the message digest becomes
"471c57334ce514589d7ff58858e30a1a16016f71 Tutorial5.tex".

Question 3 Solution:
3.1. The property is preimage resistance. Let $y = hash(password)$, and y is stored in the computer. For this y, it is hard to find a preimage m such that $h(m) = y$.

3.2. The benefit of using a salt is making a lookup table assisted dictionary attack against the stored values impractical, provided the salt is large enough. In other words, an attacker would not be able to create a precomputed lookup table of hashed values (password + salt) because it would take too much space. A simple dictionary attack is still very possible, although much slower since it can't be precomputed.

3.3. If the password is perfectly random and long enough, the probabilty of any guessed password can be hashed to the hashed value is $\frac{1}{2^{160}}$, as the output of SHA-1 is 160. So the complexity is 2^{160}.
If the password is not random, for example, an English word, the complexity should be lower.

Question 4 Solution:
This game makes use of the second-preimage resistance of hash function, i.e., for any given m, it is hard to find a different m' such that $h(m) = h(m')$. In this question, if Alice wants to cheat, she needs to find another r' such that $H(1||r') = H(1||r)$ or $H(0||r') = H(0||r)$, but this is difficult.
The proper size of r should be $512 - 1 - 64 = 447$ bits, as SHA-1 has a block size of 512 bits, and we just consider i function operation.

or 128 bits

Question 5 Solution:
5.1.
5.1.1. Let $H_i = H(H_{i-1}, m_i), m = m_1||m_2$. We have $H_0 = IV, H_1 = H(H_0, m_1), H_2 = H(H_1, m_2)$. We apply the following algorithm:

1. **for** $i = 1$ to $2^{\frac{n}{2}}$ **do**
2. Choose an aribitrary block m_2^i, and compute H_1^i such that $H(H_1^i, m_2^i) = H_2$
3. Store m_2^i in the table T, indexed by H_1^i, i.e., let $T[H_1^i] \leftarrow m_2^i$.
4. **end for**
5. **repeat**
6. Choose an arbitrary message block m_1.
7. Compute $H_1 = H(H_0, m_1)$.
8. **until** $H_1 \in T$.
9. **return** $m_1||T[H_1]$

The returned $m_1||T[H_1]$ is a preimage of this hash function. The complexity of line2-4 is $2^{\frac{n}{2}}$. The complexity of line6-9 is also $2^{\frac{n}{2}}$. The complexity of this meet in the middle attack is thus $2^{\frac{n}{2}+1}$.

5.1.2. Let $H_i = H(H_{i-1}, m_i), m = m_1||m_2$. We have $H_0 = IV, H_1 = H(H_0, m_1), H_2 = H(H_1, m_2)$. Since m is known, we can compute H_2. Then, we apply the algorithm in **5.1.1.** to find a preimage m'. If $m' \neq m$, we are done. The probability that they are equal is $\frac{1}{2^n}$ and if they are indeed equal, we need to perform another time the aforementioned algorithm. Thus the complexity is $(1 - \frac{1}{2^n}) \times 2^{\frac{n}{2}+1} + \frac{1}{2^n} \times (2 \times 2^{\frac{n}{2}+1}) = 2^{\frac{n}{2}+1} - 2^{1-\frac{n}{2}} + 2^{2-\frac{n}{2}} \approx 2^{\frac{n}{2}+1}$.

5.1.3. There is no shortcut, and the complexity is $2^{\frac{n}{2}}$.

5.2.

5.2.1. Let $H_1 = H(H_0, m_1)$.

Given H_1, since H_0 can be chosen arbitrarily, we can randomly choose an H_0, and then compute m_1'. The complexity is 1.

5.2.2. Given m_1 and H_0, we can choose an H_0' that is different from H_0 and use it to compute m_1' as the second preimage. The complexity is 1.

5.2.3. We randomly choose an H_0 and an m_1 to get H_1. Then we choose another H_0' and then compute the corresponding m_1' such that $H(H_0, m_1) = H(H_0', m_1')$. The complexity is thus 2.

Question 6 Solution:

6.1. This MAC algorithm is only for protecting the last block, so the attacker can modify any blocks except the last one and send the modified message together with the tag t without being detected.

6.2. This MAC algorithm doesn't provide any protection for the order of message blocks.

Question 7 Solution:

7.1.

	CBC	CBC-MAC
Purpose	encryption	message authentication
IV	a random and public stream	fixed to 0
Output	ciphertext	MAC

7.2. The attackers may have 2 kinds of attack:
Attack 1. Use the MACs of 2 messages, the MAC of a new message can be generated without knowing the secret key.

For example $M = m_0||m_1$ CBC-MAC(M)=t

$M' = m_0'||m_1'||m_2'$ CBC-MAC(M')=t'

The attacker can forge a new message: $m_0||m_1||(m_0' \oplus t)||m_1'||m_2'$, and the forged tag is t'.

Attack 2. With message length being appended to mesage, message can still be forged.
For example:
Let M_1' represents M_1 with length padding
Let M_2' represents M_2 with length padding
Let $M_3' = m_1'||a_0||a_1||a_2$, where each a_i represents one message block.
Suppose now an attacker knows that the MACs of M_1, M_2, M_3 are t_1, t_2, t_3, then an attacker can generate the MAC for the following message without knowing the secret key:

$$M_4 = M_2'||(a_0 \oplus t_1 \oplus t_2)||a_1||a_2, \text{ and the MAC is } t_3$$

7.3. CMAC strengthens the CBC-MAC: CMAC uses an additional key (K_1 or K_2) for th elast message block to thwart the attacks on CBC-MAC. Specifically, K_1 is used when the last block is a full block, K_2 is used if it is partial block, which is padded bit 1 followed by some zero bits. Without K_1, CMAC is the same as CBC-MAC.

7.4. If K_1 is set as $E_k(0)$, then when the message block is 0, we have:

$$W = E_k(0)$$
$$t = E_k(0 \oplus K_1 \cancel{\oplus W}) = E_k(0) = W$$

when the message consists of two message blocks m1 = m2 = 0, ...

The MAC is always W if the message block is 0.

Question 8. Solution:
8.1. Key-prfix method: MAC$_K(M)$ =Hash $(K||M)$
An attacker can extend the message and generated new MAC for the extended message (if there is no finalization stge in the Merkle-Damgard construction, such as SHA-1 and SHA-2). Suppose that $(K||M)$ after padding becomes $(K||M||p)$, and the attacker knows the MAC value Hash$(K||M||p)$. then an attacker can compute Hash$(K||M||p||x)$ for any x due to the iterated nature of hash function.

8.2.
800 bits: Let m denote the 800-bit message, then m should be padded to a 1024 bits message, i.e., $m' = pad(m) = m_1||m_2$. The HMAC is then given by:

$$\text{MAC}_k(m) = \text{Hash}((K \oplus \text{opad})||\text{Hash}((K \oplus \text{ipad})||m_1||m_2))$$

the padding is not that correct: the message length in the padding is 1024+512 bits

Since there are 4 block to be hashed with another hashed K, 5 compression function operations are needed.

0 bit: Let m denote the 0 bit message, then m should be padded to a 512 bits message, i.e., $m' = pad(m) = m$. The HMAC is then given by:

$$\mathrm{MAC}_k(m) = \mathrm{Hash}((K \oplus \mathrm{opad})||\mathrm{Hash}(K \oplus \mathrm{ipad})||m)$$

Since there are 3 block to be hashed with another hashed K, 4 compression function operations are needed.

960 bits: Let m denote the 960-bit message, then m should be padded to a 512×3 bits message, i.e., $m' = pad(m) = m_1||m_2||m_3$. The HMAC is then given by:

$$\mathrm{MAC}_k(m) = \mathrm{Hash}((K \oplus \mathrm{opad})||\mathrm{Hash}(K \oplus \mathrm{ipad})||m_1||m_2||m_3)$$

Since there are 5 block to be hashed with another hashed K, 6 compression function operations are needed.

Question 9 Solution:
9.1.

CBC mode	IV must be random
CFB mode	All IVs must be different for the same key
OFB mode	All IVs must be different for the same key
CTR mode	IVs are different for each message, but remains the same for each message, and should be different for different keys.
synchronous stream cipher	All IVs must be different for the same key
asynchronous stream cipher	All IVs must be different for the same key
hash function	IV is set to a fixed constant
CBC-MAC	IV is set as 0
CMAC	IV is set as 0

9.2.

CBC mode	Then there is no randomness in the first ciphertext block, so the first block can be attacked in the same way as the attack on EBC mode,i.e., dictionary attack. Since there is a lot of redundency in the meaningful plaintext, the attacker can collect a lot of plaintext-ciphertext pairs of the first block, and then recover the first plaintext blocks of other ciphertexts by comparing the ciphertext blocks with the plaintext-ciphertext pairs collected.
CFB mode	The value of $E_K(IV)$ can be recovered once we know the first block of plaintext. So all the first blocks of plaintexts of other ciphertexts can be recovered now.
OFB mode	The value of $E_K(IV)$ can be recovered once we know the first block of plaintext. Then the OFB mode becomes ciphertext=plaintext $\oplus E_K(IV)$. We can now attack it in the same way as EBC mode, i.e., dictionary attack. not only the first message block
CTR mode	The same keystream is used to encrypt more than one message, so it is susceptible to dictionary attack.
synchronous stream cipher	The same keystream is used to encrypt more than one message, so it is susceptible to dictionary attack.
asynchronous stream cipher	The same keystream is used to encrypt more than one message, so it is susceptible to dictionary attack. **similar to CFB mode**
hash function	IV is set to a fixed constant, and there is no secret key.
CBC-MAC	IV is set as 0
CMAC	IV is set as 0