HW 5 - CNS Sapienza

Daniele Oriana 1457625

24 Nov 2016

2016-01-13

Q1

Perfect Cipher

Q1.1 [1/30] Define what a perfect cipher is. [If definition is wrong subsequent questions cannot be correctly answered]
Q1.2 [2/30] Prove that the property that defines what a perfect cipher is also holds if we exchange the roles of ciphertext and plaintext.
Q1.3 [2/30] Prove that a cipher cannot be perfect is the size of its key space is less than the size of its message space.
Q1.4 [2/30] Describe the Vernam Cipher (a.k.a. One Time Pad) and clearly state the property that must hold for it being a perfect cipher.

Answer

Q1.1

A cipher is said to be perfect if it doesn't reveal any information about the plaintext, we can write, equivalently:

Pr[P|C] = Pr[P], where P=plaintext and C=ciphertext.

Q1.2

For the Bayes theorem we have that:
$\Pr[P \wedge C]=\Pr[P|C]\Pr[C]=\Pr[C|P]\Pr[P]$
in addiction, when P and C are indipendent, we have that:
$\Pr[P \wedge C]=\Pr[P]\Pr[C]$

so we can write this chain of equivalence:

$\Pr[P \wedge C] = \Pr[C|P]\Pr[P]$
$\Pr[P \wedge C] = \Pr[P]\Pr[C]$
so, $\Pr[C|P]\Pr[P] = \Pr[P]\Pr[C]$, and finally:
$\Pr[C] = \Pr[C|P]$

Q1.3

The Shannon's theorem states that: a cipher cannot be perfect if the size of its key space is less than the size of its message space.
We can prove this theorem by contradiction:

- Assume #keys(l)<#messages(n) and consider ciphertext C0 such that $\Pr[C0]>0$
- For some key K, consider $P=D_K(C0)$. There exist at most l(#keys) such messages
- Choose message P0 such that it is not of the form $D_K(C0)$
- Hence $\Pr[C0|P0]=0$
- But in a perfect cipher $\Pr[C0|P0]=\Pr[C0]>0$. So we have a contradiction.

Q1.4

The one time pad is a cyptographic system based on the follinwg features:

- Plaintext space: $0,1^n$
- Key space: $0,1^n$
- The scheme is symmetric,key K is chosen at random
- $E_k(P) = C = P$ XOR K
- $D_k(C) = C$ XOR K = P XOR K XOR K = P

In the one time pad we can't obtain informations about the plaintext analyzing the ciphertext, in fact it provides no information about the original message or about probability distribution of it to a cryptanalyst. In this case we can say that the one time pad offers perfect secrecy.

Q2

Meet in the middle

With reference to a symmetric-key block-ciphering algorithm, answer the following:
Q2.1 [2/30] Describe the so called Meet-In-The-Middle attack (not to be confused with Man-In-The-Middle) as well as a possible scenario where Meet-In-The-Middle can be used by an attacker.
Q2.2 [2/30] Give a rough estimation of the max key-length that can be successful attacked by Meet-In-The-Middle, assuming an adversary using 2 hours computation time of a 64GB RAM machine (and negligible external storage), capable of encrypting/decrypting a block in 20ns (for any key of reasonable length) and of accessing a RAM word in 4ns. Ignore CPU time that you can assume involving simple arithmetics on cache; also assume that block size is 128b.

Answer

Q2.1

If we are in a situation in which we have a double DES encryption with two different keys, we can face a meet in the middle attack. In this situation, a double encryption of the plaintext doesn't guarantee a better level of security, but, intead, it leads to a completely lack of security.
How does this attack works?
The main idea of this attack provides that we can try all the possible encryption of the plaintext and all the possible decryption of the ciphertext, checking when a pair of keys transforms the plaintext in the ciphertext and viceversa. In this way we can avoid the $2^{2^{\mathrm{n}}}$ bruteforce attempts meeting in the middle of the plaintext and of the ciphertext, obtaining in this way a number of attempts equal to 2^{n} for encryption + 2^{n} for decryption.

Q2.2

RAM has $\frac{64 \cdot 10^9}{128} = 5 \cdot 10^8$ blocks.

Assuming that one block corresponds to one word in the RAM, the attacker can encryp $5 \cdot 10^8$ blocks in $(5 \cdot 24) \cdot 10^8 ns = 120 \cdot 10^8 ns$.

The attacker uses the RAM for two hours, so:

$\frac{7200 \cdot 10^9 ns}{12 \cdot 10^9 ns} = 600$ cycles of RAM.

Attacker can encrypt, in two hours, $(600 \cdot 5) \cdot 10^8 = 3 \cdot 10^{11}$ blocks.
Finally we have: $2^n = 3 \cdot 10^{11}$,
$n \approx log_2(3 \cdot 10^{11})$

Q3

The birthday attack

Q3.1 [3/30] Illustrate the so-called birthday paradox and describe what type of attack (birthday attack) can be prepared by exploiting this property.
Q3.2 [2/30] Is the birthday bound affected by the quality of the hash function (cryptographic or noncryptographic). Elaborate.
Q3.3 [2/30] Is keyed hashing making the birthday attack harder? Elaborate.

Answer

Q3.1

The birthday attack is based on the famous bithday paradox. This paradox states that: let h: D$-$ >R be any mapping,if we choose randomly 1.1774|R|$^{1/2}$ elements of D, then the probability of two of them are mapped into the same image is equal to 0.5. This attack can be used to find a collision when we are facing with not good hash function, in fact, in these cases, the paradox guarantees that we can find a collision in a bounded number of attempts.

Q3.2

The birthday bound is the approximative number of attempts to generate a collision using brute force. This value is affected by the quality of the hash function, in fact if we are considering a cryptographic hash function (which is an hash function strongly collision resistant) this birthday bound will be bigger than the value related to the noncryptographic hash function (which is an hash function that isn't strongly collision resistant).

Q3.3

It could make the birthday attack harder, but it depends on how you realize the keyed hashing, we consider the following cases:

- $\mathrm{MAC_k(m)} = \mathrm{h(k||m)}$: the adversary can add extra bits and compute the correct MAC
- $\mathrm{MAC_k(m)} = \mathrm{h(m||k)}$: the adversary can exploit the birthday paradox, in fact, with this configuration, if the adversary finds two colliding message, then he/she knows two messages with the same MAC for all keys
- $\mathrm{MAC_k(m)} = \mathrm{h(k||m||k)}$: it is a good solution
- $\mathrm{MAC_k(m)} =$ first bits of h(k||m) or h(m||k): it is a good solution.

Q5

Digital signatures

Q5.1 [2/30] Illustrate the (generic) processes for generating/verifying digital signatures.
Q5.2 [2/30] Define the existential forgery attack and point out the steps whose bad implementation.

Answer

Q5.1

Let $\mathrm{E_A}$ be the public encryption key of A and $\mathrm{D_A}$ be the private decryption key of A. To sign a message M, A first compute the hash y=H(M) and then sign the result with the private key $\mathrm{z=D_A(y)}$. After this first step, A sends to B the pair (z,M), at this point B can check the signature of A computing $\mathrm{y=E_A(z)}$ and verifying that is equal to y=H(M) (note that since B has received M, he/she can compute H(M)). Obviously H should be a strong collision resistant hash function, in order to avoid collisions that can lead to H(M)=H(M').

Q5.2

The existential forgery provides that the adversary can create any message/signature pair (m,θ), where θ was not produced by the legitimate signer. Here the adversary doesn't need to have control over m, in fact m doesn't need any particular meaning. If we don't use the "Hash First" solution (explained in the previous answer), we can incur in an existential forgery attack that can take place in the following way:

- Instead of Alice, Trudy take a random message R and compute the decryption of it with the public key of Alice, obtaining P=D_{PA}(R) and sending the pair (P,R) to Bob.
- Now Bob checks the signature applying the decryption with the public key of Alice on R and comparing the result with the received P. Obviously Bob will obtain as result that R has been signed by Alice.

So, a bad implementation characterized by the absense of an hash function can lead to an existential forgery.