Information Systems Security
Correction Series 6 - Hash Functions and MACs

November 24th, 2021

Exercise 1 : Hash Functions

- The first hash function does not have any of the three properties :
  If we have a given hash y, finding a preimage is obvious : we just take $x = y$. Then $h(x) = y \mod n = y$. We don't have the preimage resistance.
  Finding a second preimage is also easy : given a pair $(x, h(x))$, then $x' = x + n$ is a collision ($h(x') = x + n \mod n = x \mod n = h(x)$).
  And since we do not have the second preimage resistance, we do not have the collision resistance (since the collision resistance implies the second preimage, then not having the second pre image means we don't have the collision either).

- This one does not have any of the properties either :
  Since n is prime, we can compute $\Phi(n) = n - 1$. Then we can compute e, the inverse of d, and find $x = (x^d)^e \mod n$. We do not have the preimage resistance.
  We do not have the second preimage resistance either : For a given pair $(x, h(x))$, $x' = x + n$ is a collision ($h(x') = (x + n)^d \mod n = x^d \mod n = h(x)$).
  And since we do not have the second preimage resistance, we do not have the collision resistance either.

  In this case, if n was a product of two primes p and q kept secret, as in RSA, then we would have the preimage resistance (if we can't factorize n, we can't find e), but still not the second preimage and collision.

- Yet again, none of the three properties are respected :
  If $y = h(x)$, then an obvious preimage is $x = x_1 x_2 ... x_n$ avec $x_1 = y$ and $x_2 = ... = x_n = (00)_{16}$ (i.e. all bytes are 0 except the first one which is y). For any length of n, this would be a preimage (So $x = x_1 = y$ is an even easier example).
  If we have $(x, h(x))$, then $x' = x \parallel (00)_1 6$ is a collision : $h(x') = x_1 + ... + x_n + (00)_1 6 = h(x) + (00)_1 6 = h(x)$.
  Once again, no second preimage resistance implies no collision resistance.

- And this time... Still none of the three properties !
  If we have $y = h(x)$, then $x = x_1 x_2 ... x_n$ with $x_1 = x_2 = ... = x_{n-1} = (00)_{16}$ and $x_n = k$ is a preimage.
  If we have $(x, h(x))$, then $x' = (00)_16 \parallel x$ is another preimage for the same hash (with n the number of bytes of x) : $h(x') = (n+1) * (00)_16 + ((n+1)-1) * x_1 + ... + 1 * x_n = (00)_16 + h(x) = h(x)$.
  Which also implies no collision resistance.

Exercise 2 : Message Authentication Codes

- We have $m = m_1 \parallel m_2$ and $t = t_1 \parallel t_2$. Then we can build the following message :

  $$m_{fake} = (m_2 \oplus t_1) \parallel (t_2 \oplus m_1)$$

  Which has the following MAC :

  $$t_{fake} = E_k(m_2 \oplus t_1) \parallel E_k((t_2 \oplus m_1) \oplus E_k(m_2 \oplus t_1)) = t_2 \parallel E_k(t_2 \oplus m_1 \oplus t_2) = t_2 \parallel E_k(m_1) = t_2 \parallel t_1$$

  Which we can build from t_1 and t_2. We have built this MAC without the key.

- First, we can see that the formula given for the MAC means that the MAC is simply equal to c_n.

  1. If we have m and c, we can build a fake pair (m', MAC') similarly as before :

     $m_1^{'} = m_n \oplus c_{n-1}$,
     $m_i^{'} = c_{n+2-i} \oplus m_{n+1-i} \oplus c_{n-i}$ for $i \in \{2, ..., n-1\}$,
     and $m_n^{'} = c_2 \oplus m_1$.

     If we compute the corresponding ciphers, we have :
     $c_1^{'} = E_k(m_1^{'}) = E_k(m_n \oplus t_{n-1}) = c_n$,
     $c_i^{'} = E_k(m_i^{'} \oplus c_{i-1}^{'}) = c_{n-i+1}$ for $i \in \{2, ..., n-1\}$,
     $c_n^{'} = E_k(m_i^{'} \oplus c_{n-1}^{'}) = E_k(c_2 \oplus m_1 \oplus c_2) = E_k(m_1) = c_1$.

     This ciphers are just the same as the ones we already have in c, just in reverse order, which means we can built this encryption, and the MAC (which is just equal to the last cipher block, which means $MAC' = c_n^{'} = c_1$) without the key.

  2. Even without the message, if we intercept just an encrypted message (which means we have the ciphers, c, and obviously the MAC since it's equal to the last cipher), we can modify it without anyone noticing.
     Since the MAC is equal to the last cipher, by construction, then it

means we can modify all ciphertexts, except the last one c_n, and the MAC will still be valid. So we can completely change the ciphers, and the receiver will find a message very different that what was originally sent, but with a MAC still valid.

3. We can simply use two different keys, one for the encryption, and one for the MAC.

4. We would not have the collision resistance : Given any pair of messages and MACs (m, t) and $(m^{'}, t^{'})$, we can build a collision with message $m^{'}$, with the following message :

   $$m^{''} = m \parallel (m_1^{'} \oplus t) \parallel m_2^{'} \parallel ... \parallel m_n^{'}$$

   Which will give the MAC t' again, as at the step $m_1^{'} \oplus t$, we will obtain :

   $$t_1^{''} = E_k(m_1^{'} \oplus t) \oplus t_n = E_k(m_1^{'} \oplus t \oplus t) = E_k(m_1^{'}) = t_1^{'}$$

   So each following block will give the same chain of ciphers as $m^{'}$, which will then finish with the same cipher, which means the same MAC.

   (There's a few ways to solve this, for example by having different keys for different lengths, or by including the length of the message somewhere in the message as padding).