Algoritmi di Crittografia

Corso di Laurea Magistrale in Informatica

A.A. 2018/2019

1. Keyed hashing
   - General concepts
   - Classical designs
   - Designs based on Universal hash functions

Algoritmi di Crittografia

1. Keyed hashing
   - General concepts
   - Classical designs
   - Designs based on Universal hash functions

What keyed hashing is

- As the term suggests, *keyed hashing* is hashing with keys
- Mathematically, a *keyed hash function* h has two inputs: a fixed size key K and a (variable length) message M, and returns a fixed size output T
- Keyed hash functions have two main purposes
- First of all, they are used to warrant message *integrity*, i.e., a guarantee that the message has not been altered, as well as authenticity
- The second use is in the implementation of *Pseudo Random Functions* (*PRF*s)

Authentication and message integrity

- Here is a simple protocol
- Suppose Alice wants to send a message M to Bob
- Their concern here is message integrity (rather than confidentiality)
- In any case, Alice and Bob share a secret key K
- Alice uses key to compute (what is called) an *authentication tag* $T = h(K, M)$ and sends it to Bob, together with M
- Upon receipt, Bob computes $h(K, M)$ and check it against t
- If the comparison is successful, Bob knows that the message has not been altered and also that it came precisely from Alice

Message Authentication Codes (MACs)

- The above protocol is an example of *Message Authentication Code* (or, simply, *MAC*), a special cryptographic algorithm that protects message integrity and guarantees the receiver on the sender's identity
- MAC is also the term often used to refer to the computed tag, i.e., $T = MAC(K, M)$
- As a consequence, Bob's protocol becomes: "Compute the MAC of the value received and compare with the received MAC. Accept if equal, otherwise discard."
- Some important cryptographic protocols use MACs: notable examples are *TLS* and *SSH*
- Clearly, if confidentiality is also a goal, the protocol may be enriched with encryption (i.e., send M encrypted rather than in clear)

Pseudo random Functions

- A PRF is a function whose outputs cannot be distinguished from a true random mapping
- The exact definition is more technical and involves not just a single function but, rather, a family $\mathcal{F}$ of functions (since it does not make much sense to tag a single function as “pseudo-random”).
- We shall not dive into such technicalities
- PRFs (well, practical approximations of...) can be obtained using keyed hash functions or block ciphers
- In both cases, the secret key is precisely the ingredient that makes the output unpredictable to an attacker
- The other way around, PRFs can be used to generate cryptographic keys, usually from a password

Algoritmi di Crittografia

1. Keyed hashing
   - General concepts
   - Classical designs
   - Designs based on Universal hash functions

Building MACs from block ciphers

- The first MAC construction we consider is known as *CBC-MAC*
- CBC-MAC uses a block cipher in CBC mode
- However, since the purpose here is not to encrypt (or decrypt) a message, all the encrypted message is discarded but last block
- The algorithm is the following:

$$\begin{aligned} C_0 &= IV \\ C_{i+1} &= \mathbf{E}(K, C_i \oplus M_{i+1}) \quad i = 0, \ldots, n-1 \\ MAC(K, M) &= C_n \end{aligned}$$

  where, clearly, $M = M_1 || M_2 || \ldots || M_n$ is the message
- IV is usually fixed as 0, hence the first step becomes $C_1 = \mathbf{E}(K, M_1)$

CBC-MAC is insecure

- We will see two different attacks
- The first attack assumes the attacker can get the tags of two different (single block) messages, i.e., $T_1 = MAC(K, M_1)$ and $T_2 = MAC(K, M_2)$
- It is not difficult to show that $T_2 = MAC(K, M_1 || M_2 \oplus T_1)$
- In fact, applying the CBC-MAC algorithm, we get

$$\begin{aligned} C_1 &= \mathbf{E}(K, M_1) \\ &= T_1 \\ C_2 &= \mathbf{E}(K, T_1 \oplus (M_2 \oplus T_1)) \\ &= \mathbf{E}(K, M_2) \\ &= T_2 \end{aligned}$$

- Hence we have forged a valid message/tag pair without the knowledge of the key

CBC-MAC is insecure (cont.d)

- The second example attack is even simpler to explain
- Suppose we know $M_1 \neq M_2$ such that $T_1 = MAC(K, M_1)$, $T_2 = MAC(K, M_2)$, and $T_1 = T_2$.
- Now let m be one block message
- Then we have

$$
\begin{aligned}
MAC(K, M_1 || m) &= \mathbf{E}(K, m \oplus T_1) \\
&= \mathbf{E}(K, m \oplus T_2) \\
&= MAC(K, M_2 || m)
\end{aligned}
$$

- Again, we have forged a valid message/tag pair without the knowledge of the key
- Note that the initial colliding messages can be found in $2^{n/2}$ steps using the classical birthday attack

Building MACs from CS Hash Functions

- The second (and to a great extent the more “obvious”) strategy to build keyed hash functions is to use ... (unkeyed) hash functions!
- Keyed hash functions have two inputs (the key and the value to be hashed) while hash functions have just one, though ...
- The obvious solution is to somehow mix the key with the value
- We will consider three different constructions, namely: *Secret-prefix*, *Secret-Suffix*, and *HMAC* constructions

Secret-prefix MACs

- Let H be a CS hash function. We define

$$MAC(K,M) = H(K||M)$$

- Such construction is vulnerable to the length-extension attack, i.e., it allows an attacker to compute $MAC(K, M_1||M_2)$ starting from the knowledge of the tag $MAC(K, M_1)$
- Also, if keys of different lengths are allowed, the resulting concatenated value (to be hashed) could be obtained in many different ways. For instance:

$$\begin{array}{llcl} K = \texttt{Crypto} & M = \texttt{graphy} & \Rightarrow & K||M = \texttt{Cryptography} \\ K = \texttt{Cryptogra} & M = \texttt{phy} & \Rightarrow & K||M = \texttt{Cryptography} \end{array}$$

- A simple fix here consists of including the key length ℓ:

$$MAC(K,M) = H(\ell||K||M)$$

Secret-suffix MACs

- We now define
$$MAC(K, M) = H(M||K)$$
- This "simple" modification makes the length-extension attack impossible since no prefix of $M_1||M_2||K$ coincides with $M_1||K$
- However, the secret-suffix construction is insecure if collisions can be found for the internal hash function (also known as *internal collisions*)
- Given a message M, the attacker can perform an offline search for a message M' that collides with M on the internal hash function
- An internal collision implies that the intermediate hash state before the key in involved is the same in the two cases (M and M') thus leading to identical authentication tags

HMAC

- The *HMAC* (*Hash-based MAC*) is defined as follows:

$$HMAC(K, M) = H((K_p \oplus a)||H((K_p \oplus b)||M))$$

where a and b are well defined constants of the same size as the blocks of the underlying hash function H

$$\begin{aligned} a &= '\backslash x5c\backslash x5c \ldots \backslash x5c' \\ b &= '\backslash x36\backslash x36 \ldots \backslash x36' \end{aligned}$$

- The constants a and b are also referred to as the *opad* (outer padding) and *ipad* (inner padding), respectively
- The key K_p (where p stands for “padded”) is derived from K so has to have size equal to opad/ipad size (i.e., possibly reducing size by hashing and possibly increasing size with trailing 0s)

Why HMAC is better

- Length extension attacks are not critical to HMAC since the application of the second (i.e., the outer) Hash function “destroys” the result of the first (i.e., the internal) one
- In $HMAC(K, m_1||m_2)$ the hash function H is applied to the two block message

$$\underbrace{(K_p \oplus a)}_{1\ \text{block}} || \underbrace{H((K_p \oplus b)||(m_1||m_2))}_{1\ \text{block}}$$

- On the other hand, in $H(HMAC(K, m_1)||m_2)$ the hash function is applied to the message

$$\underbrace{H((K_p \oplus a)||H((K_p \oplus b)||m_1))}_{1\ \text{block}}||m_2$$

- *opad* and *ipad* “should” have large Hamming distance

Algoritmi di Crittografia

1. Keyed hashing
   - General concepts
   - Classical designs
   - Designs based on Universal hash functions

Universal hash functions

- *Universal hash functions* come in families (there is no such thing as a universal hash function)
- Intuitively, $\mathcal{H}$ is a family of universal hash functions over a given domain $\mathcal{U}$ if, for any two values $x, y \in \mathcal{U}$, the probability that $h(x) = h(y)$, for a randomly chosen $h \in \mathcal{H}$, is negligible
- Let's consider a simple example. Suppose the elements of $\mathcal{U}$ are 16 byte long (say) and define $x = (x_1, x_2, x_3, x_4)$, where x_i can be regarded as an integer in the range $[0 : 2^{32} - 1]$, $i = 1, 2, 3, 4$
- The $\mathcal{H}$ family includes all functions defined as follows

$$h(x) = \sum_{i=1}^{4} a_i x_i \bmod M$$

where M is a sufficiently long prime number ($\geq 2^{32}$) and $a_i \in \mathbf{Z}_M$

Universal hash functions (cont.d)

- Depending on the size of the domain, M is fixed.
- Then, to "randomly choose" a function from $\mathcal{H}$ simply means to select the four numbers $a_1, \ldots, a_4$ uniformly at random
- Assume $x \neq y$, and assume h is randomly chosen
- Suppose now the a_i are "uncovered" in sequence. There is just one requirement, namely that, if a_j is the last value revealed, then $x_j \neq y_j$.
- Note that, since $x \neq y$, one such index j must exist
- For simplicity, suppose $j = 4$

Universal hash functions (cont.d)

- Now, after a_1, a_2 and a_4 have been uncovered, for $h(x) = h(y)$ to hold we must have:

$$a_4(x_4 - y_4) \equiv \sum_{i=1}^{3}(y_i - x_i)a_i \pmod{M}$$

- Since M is prime, the multiplicative inverse of $x_4 - y_4 \neq 0$ does exist; hence

$$a_4 = \left(\sum_{i=1}^{3}(y_i - x_i)a_i\right)(x_4 - y_4)^{-1} \pmod{M}$$

- But since a_4 is chosen uniformly at random, the above equality hods with probability $1/M$, which is the minimum possible

Polynomial evaluation MACs

- We need just a simple modification, at least to come up with a first MAC version
- The $\mathcal{H}$ family is defined through two parameters only (with fixed M), say K and R, belonging to $\mathbf{Z}_M$
- Now, if the message to be authenticated is made of L blocks: $m = m_1, m_2, \ldots, m_L$, we define:

$$h(K, R, m) = R + \sum_{i=1}^{L} m_i K^i \bmod M$$

- In real cases, the message blocks may be 128 bit long, hence M must satisfy $M > 2^{128}$
- The secret key is the pair (K, R), and given the key the hash function is uniquely determined

Polynomial evaluation MACs (cont.d)

- The proof that $h(K, R, m) = h(K, R, m')$, for $m \neq m'$, has negligible probability is similar as the one above (we assume, for simplicity, that the two messages have the same length)
- If K and R are randomly chosen then

$$R + \sum_{i=1}^{L} m_i K^i \equiv R + \sum_{i=1}^{L} m'_i K^i \mod M$$

implies

$$\sum_{i=1}^{L} (m_i - m'_i) K^i \bmod = 0$$

- The above equation is satisfied if K is a zero of the degree L polynomial $p(x) = \sum_{i=1}^{L} (m_i - m'_i) x^i$ over the field $\mathbf{Z}_M$ and this means that the equation has at most L solution
- Hence the probability of collision is at most $\frac{L}{M}$

Vulnerability

- The Polynomial MAC presented above must be used just once
- In fact, a CPA-able attacker could ask for the tag of just two messages and recover the key
- The two messages might be (among other possibilities):

$$
\begin{aligned}
m' &= \underbrace{00\ldots0}_{L \text{ blocks}} \\
m'' &= \underbrace{00\ldots0}_{L-1 \text{ blocks}}\ \underbrace{00\ldots01}_{128 \text{ bits}}
\end{aligned}
$$

- In this way, $h(K,R,m') = R$, so that the attacker could recover R, and $h(K,R,m'') = R + K$, and the attacker could recover K