Security Issues in Cloud Computing

Modern Cryptography I – Symmetric Cryptography (ctd.)

Peter Schwabe

October 21, 2011

Hash functions

Definition:
A *hash function* is a function that maps bit strings of arbitrary length to bit string of a fixed length.
A *cryptographic hash function* must at least fulfill the following three properties:

Collision resistance: it must be infeasible to find two different messages mapping to the same value;

preimage resistance: given the output of a hash function, it must be infeasible to find a corresponding input;

second-image resistance: given a message, it must be infeasible to find another message, mapping to the same value.

These properties yield a "digital fingerprint".

Attacks

- For any hash function with n-bit output collisions can be found with $2^{n/2}$ operations (for comparison: trying all keys of a stream cipher with n-bit keys takes 2^n operations)
- For any hash function with n-bit output we can find preimages (or second preimages) using 2^n operations, preimages are (with generic attacks) much harder to find than collisions.

Example: If 1,000,000 computers are needed to find a preimage in one year (using a generic attack), collisions can be found in one year using only about 1000 computer (again using a generic attack).

Popular hash functions

Name	Output length	Security/Remarks
SHA(SHA-0)	160 bits	broken, withdrawn
MD5	160 bits	broken, (collisions)
SHA-1	160 bits	weakened, (no collisions yet, but does not provide the targeted security)
RIPEMD160	160 bits	not broken, (but relatively short hashes, security against collisions comparable to a block cipher with 80-bit key)
SHA-2	256, 384 or 512 bits (SHA-256, SHA-384, and SHA-512)	not broken, (construction similar to SHA-1)

The SHA-3 competition

- In Nov. 2007, NIST issued a public call for submissions to a hash-function competition. (SHA-3 competition)
- Algorithms are required to support hash lengths of 224, 256, 384 and 512 bits.
- NIST received 64 submissions (56 are publicly known) by Oct. 2008.
- 51 were selected as round-1 candidates
- 14 candidates were selected as round-2 candidates on July 24, 2009
- 5 finalists selected, announced on Dec 9, 2010:
  - Blake (Aumasson, Henzen, Meier, Phan)
  - Grøstl (Gauravaram, Knudsen, Matusiewicz, Mendel, Rechberger, Schläffer, Thomsen)
  - JH (Wu)
  - Keccak (Bertoni, Daemen, Peeters, Van Assche)
  - Skein (Ferguson, Lucks, Schneier, Whiting, Bellare, Kohno, Callas, Walker)
- Similar selection criteria as for AES: security, software performance, hardware performance
- Final decision is expected in 2012
- Side effect of the SHA-3 competition: Nobody takes the time to break SHA-1

An application of hash functions: password files

Storing plaintext password on a computer is very risky and insecure. Instead hash the password and store the hash. Every time the user enters his or her password, hash the input and compare to the stored hash value.

Attacks and countermeasures

Rainbow tables: Huge precomputed table of frequently used passwords with corresponding hash value, sorted by hash value. For example, 30 bytes per table entry: 1 TB harddisk can store > 33 billion password-hash pairs.

Countermeasure: Salted hashes: When setting the password, generate an additional random salt value s (e.g., 4 bytes), store $s, h(\text{password}, s)$ and thus obtain different possible entries for the same password (depending on the salt value). Now precomputing all possible entries becomes infeasible: A 1 TB harddisk can store all possible entries of about 33 passwords (with 4-byte salt).

Online attack: Read the salts from the password file, try with many frequently used passwords until the hash value matches. Various tools and password lists available, for example: "John the ripper", wordlist from openwall.org.

Countermeasure: Use good passwords, i.e., passwords that won't be found by this attack:

- long (≥ 8 characters)
- not from any language
- involve capital letters, small letters, numbers, special characters
- For example: On a Unix system use `pwgen -s -n 10 -y`
- keep in mind: Breaking many passwords costs essentially as much as breaking one.

Message-authentication codes

Assume that Alice sends a message M to Bob. Alice and Bob (and nobody else) have a key K. How can they make sure that the message cannot be modified by Oscar on the way to Bob without Bob noticing this.
Definition: A message-authentication code (MAC) algorithm takes as input an (arbitrary length) message and a key and computes a tag called MAC with the following properties:

- It is impossible to compute a valid MAC without knowledge of the key.
- Even if an attacker can obtain valid message tags for adaptively chosen messages from an oracle, he is afterwards not able to generate one valid message-tag pair that he has not seen before.

A first simple construction: CBC-MAC

Idea: If a block cipher is used in CBC mode, the last ciphertext block depends on all plaintext blocks, use it as a MAC.

Example: Message M consisting of three blocks M_1, M_2, M_3, Key K, $IV = 0, \ldots, 0$, compute MAC T as

$$T = E_K(E_K(E_K(M_1 \oplus IV) \oplus M_2) \oplus M_3)$$

Problem: This is only secure for fixed length messages! Consider for example $M = (M_1, M_2)$ with valid MAC T and $M' = (M_1', M_2')$ with valid MAC T' Then the message $M'' = (M_1, M_2, (M_1' \oplus T), M_2')$ also has the valid tag T'.
Why is this to the case? Compute the MAC of M:

$$T' = E_K(\underbrace{E_K(\underbrace{E_K(E_K(M_1 \oplus IV) \oplus M_2)}_{=T} \oplus (M_1' \oplus T))}_{=M_1'=M_1' \oplus IV} \oplus M_2')$$

A better, more general, approach: HMAC

The basic idea is to combine the message M and the key K and use this combination as input to a hash function h. This should be secure without requiring collision resistance of the hash function. Simple constructions, such as $h(K|M)$, $h(M|K)$ or $h(K_1|M|K_2)$ have weaknesses, (| means concatenation).
The specification of HMAC in RFC2104 from Feb. 1997 avoids these weaknesses:
Compute $T = \text{HMAC} - h(M, K)$ as

$$T = h((K \oplus opad)|H((K \oplus ipad)|M)),$$

with $opad = Ox5c5c \ldots 5c = 0101110001011100 \ldots 01011100$,
and $ipad = Ox3636 \ldots 36 = 0011011000110110 \ldots 00110110$, both of the length of one block used by the hash function h, for example 64 bits in HMAC-SHA1. Also K needs to have the length of one block used by the hash function.

Some general remarks about MACs

- Bob can be sure that the message comes from Alice (assuming he is sure that only he and Alice know the key),but he cannot prove this to anyone else. MACs do not offer non-repudiability.
- Authenticating a ciphertext with a MAC can protect Bob from deciphering malicious ciphertexts. (First validate the MAC, then decrypt).