Chapter 5
Modification Check Values

- Cryptographic hash functions
- MDC, MAC
- MD5, SHA-1
- H-MAC, CBC-MAC

Motivation

- It is common practice in data communications to compute some kind of *error detection code* over messages, that enables the receiver to check if a message was ***accidentally altered*** during transmission
  - Examples: Parity, Bit-Interleaved Parity, Cyclic Redundancy Check (CRC)
- This leads to the wish of having a similar value that allows to check, if a message has been ***intentionally modified*** during transmission
  - If somebody wants to intentionally modify a message which is protected with a CRC value he can re-compute the CRC value after modification or modify the message in a way that it leads to the same CRC value
  - Therefore, a *modification check value* will have to fulfill additional properties that will make it impossible for attackers to forge it
- Two main categories of modification check values:
  - ***Modification Detection Code (MDC)***
  - ***Message Authentication Code (MAC)***

Cryptographic Hash Functions

- Definition: *hash function*
  - A *hash function* is a function h which has the following two properties:
    - *Compression:* h maps an input x of arbitrary finite bit length, to an output $h(x)$ of fixed bit length n
    - *Ease of computation:* Given h and x it is *easy* to compute $h(x)$
- Definition: *cryptographic hash function*
  - A *cryptographic hash function* h needs to satisfy the following properties:
    - *Pre-image resistance:* for essentially all pre-specified outputs y, it is computationally infeasible to find an x such that $h(x) = y$
    - 2^{nd} *pre-image resistance:* given x it is computationally infeasible to find any second input x' with $x \neq x'$ such that $h(x) = h(x')$
    - *Collision resistance:* it is computationally infeasible to find any pair (x, x') with $x \neq x'$ such that $h(x) = h(x')$
  - *Cryptographic hash functions* are used to compute *modification detection codes (MDC)*

Message Authentication Codes (MAC)

- Definition: *message authentication code*
  - A *message authentication code algorithm* is a family of functions h_k parameterized by a secret key k with the following properties:
    - *Compression:* h_k maps an input x of arbitrary finite bitlength to an output $h_k(x)$ of fixed bitlength, called the MAC
    - *Ease of computation:* given k, x and a known function family h_k the value $h_k(x)$ is easy to compute
    - *Computation-resistance:* for every fixed, allowed, but unknown value of k, given zero or more text-MAC pairs $(x_i, h_k(x_i))$ it is computationally infeasible to compute a text-MAC pair $(x, h_k(x))$ for any new input $x \neq x_i$
  - Please note that *computation-resistance* implies the property of *key non-recovery*, that is k can not be recovered from pairs $(x_i, h_k(x_i))$, but computation resistance can not be deduced from key non-recovery, as the key k need not always to be recovered to forge new MACs

A Simple Attack Against an Insecure MAC

- For illustrative purposes, consider the following MAC definition:
  - Input: message $m = (x_1, x_2, ..., x_n)$ with x_i being 64-bit values, and key k
  - Compute $\Delta(m) := x_1 \oplus x_2 \oplus ... \oplus x_n$ with $\oplus$ denoting bitwise exclusive-or
  - Output: MAC $C_k(m) := E_k(\Delta(m))$ with $E_k(x)$ denoting DES encryption
  - The key length is 56 bit and the MAC length is 64 bit, so we would expect an effort of about 2^{55} operations to obtain the key k and break the MAC (= being able to forge messages).

- Unfortunately the MAC definition is insecure:
  - Assume an attacker Eve who wants to forge messages exchanged between Alice and Bob obtains a message $(m, C_k(m))$ which has been "protected" by Alice using the secret key k shared with Bob
  - Eve can construct a message m' that yields the same MAC:
    - Let $y_1, y_2, ..., y_{n-1}$ be arbitrary 64-bit values
    - Define $y_n := y_1 \oplus y_2 \oplus ... \oplus y_{n-1} \oplus \Delta(m)$, and $m' := (y_1, y_2, ..., y_n)$
    - When Bob receives $(m', C_k(m))$ from Eve pretending to be Alice he will accept it as being originated by Alice as $C_k(m)$ is a valid MAC for m'

Applications to Cryptographic Hash Functions and MACs

- Principal application which led original design: ***message integrity***
  - An MDC represents a *digital fingerprint*, which can be signed with a private key, e.g. using the RSA or ElGamal algorithm, and it is not possible to construct two messages with the same fingerprint so that a given signed fingerprint can not be re-used by an attacker
  - A MAC over a message m directly certifies that the sender of the message possesses the secret key k and the message could not have been modified without knowledge of that key

- Other applications, which require some caution:
  - Confirmation of knowledge
  - Key derivation
  - Pseudo-random number generation

Attacks Based on the Birthday Phenomenon

- The Birthday Phenomenon:
  - How many people need to be in a room such that the possibility that there are at least two people with the same birthday is greater than 0.5?
  - For simplicity, we don't care about February, 29, and assume that each birthday is equally likely
- Define $P(n, k)$:= Pr[at least one duplicate in k items, with each item able to take one of n equally likely values between 1 and n]
- Define $Q(n, k)$:= Pr[no duplicate in k items, each between 1 and n]
  - We are able to choose the first item from n possible values, the second item from n - 1 possible values, etc.
  - Hence, the number of different ways to choose k items out of n values with no duplicates is: $N = n \times (n - 1) \times ... \times (n - k + 1) = n! / (n - k)!$
  - The number of different ways to choose k items out of n values, with or without duplicates is: n^k
  - So, $Q(n, k) = N / n^k = n! / ((n - k)! \times n^k)$

Attacks Based on the Birthday Phenomenon

- We have:
$$\begin{aligned}P(n,k) &= 1 - Q(n,k) = 1 - \frac{n!}{(n-k)! \times n^k} \\ &= 1 - \frac{n \times (n-1) \times ... \times (n-k+1)}{n^k} \\ &= 1 - \left[\frac{n-1}{n} \times \frac{n-2}{n} \times ... \times \frac{n-k+1}{n}\right] \\ &= 1 - \left[\left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times ... \times \left(1 - \frac{k-1}{n}\right)\right]\end{aligned}$$
- We will use the following inequality: $(1 - x) \leq e^{-x}$ for all $x \geq 0$
- So:
$$\begin{aligned}P(n,k) &> 1 - \left[\left(e^{-1/n}\right) \times \left(e^{-2/n}\right) \times ... \times \left(e^{-(k-1)/n}\right)\right] \\ &= 1 - e^{-\left[\left(1/n\right) + \left(2/n\right) + ... + \left(k-1/n\right)\right]} \\ &= 1 - e^{-k \times (k-1)/2n}\end{aligned}$$
- In the last step, we used the equality: $1 + 2 + ... + (k - 1) = (k^2 - k) / 2$
  - Exercise: proof the above equality by induction

Attacks Based on the Birthday Phenomenon

- Let's go back to our original question: how many people k have to be in one room such that there are at least two people with the same birthday (out of n = 365 possible) with probability $\geq$ 0,5?
  - So, we want to solve:

$$\tfrac{1}{2} = 1 - e^{-k\times(k-1)/2n}$$

$$\Leftrightarrow \quad 2 = e^{k\times(k-1)/2n}$$

$$\Leftrightarrow \ln(2) = \frac{k\times(k-1)}{2n}$$

  - For large k we can approximate $k \times (k - 1)$ by k^2, and we get:

$$k = \sqrt{2\ln(2)n} \approx 1.18\sqrt{n}$$

  - For n = 365, we get k = 22.54 which is quite close to the correct answer 23

Attacks Based on the Birthday Phenomenon

- What does this have to do with MDCs?
- We have shown, that if there are n possible different values, the number k of values one needs to randomly choose in order to obtain at least one pair of identical values, is in the order of $\sqrt{n}$
- Now, consider the following attack [Yuv79a]:
  - Eve wants Alice to sign a message *m1*, Alice normally never would sign. Eve knows that Alice uses the function MDC1(m) to compute an MDC of m which has length r bit before she signs this MDC with her private key yielding her digital signature.
  - First, Eve produces her message *m1*. If she would now compute MDC1(*m*1) and then try to find a second harmless message *m2* which leads to the same MDC her search effort in the average case would be on the order of $2^{(r-1)}$.
  - Instead she takes any harmless message *m2* and starts producing variations *m1'* and *m2'* of the two messages, e.g. by adding <space> <backspace> combinations or varying with semantically identical words.

Attacks Based on the Birthday Phenomenon

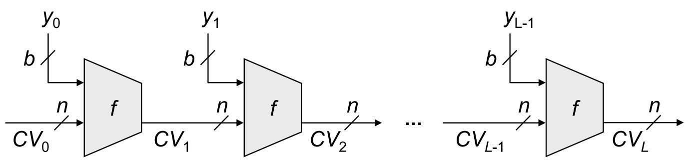
- As we learned from the birthday phenomenon, she will just have to produce about $\sqrt{2^r} = 2^{r/2}$ variations of each of the two messages such that the probability that she obtains two messages *m1'* and *m2'* with the same MDC is at least 0.5
- As she has to store the messages together with their MDCs in order to find a match, the memory requirement of her attack is on the order of $2^{r/2}$ and its computation time requirement is on the same order
- After she has found *m1'* and *m2'* with *MDC1(m1') = MDC1(m2')* she asks Alice to sign *m2'*. Eve can then take this signature and claim that Alice signed *m1'*.
- Attacks following this method are called *birthday attacks*
- Consider now, that Alice uses RSA with keys of length 2048 bit and a cryptographic hash function which produces MDCs of length 96 bit.
  - Eves average effort to produce two messages *m1'* and *m2'* as described above is on the order of 2^{48}, which is feasible today. Breaking RSA keys of length 2048 bit is far out of reach with today's algorithms and technology.

Overview of Commonly Used MDCs and MACs

- Cryptographic Hash Functions for creating MDCs:
  - Message Digest 5 (MD5):
    - Invented by R. Rivest
    - Successor to MD4
  - Secure Hash Algorithm 1 (SHA-1):
    - Invented by the National Security Agency (NSA)
    - The design was inspired by MD4
- Message Authentication Codes:
  - DES-CBC-MAC:
    - Uses the Data Encryption Standard in Cipher Block Chaining mode
    - In general, the CBC-MAC construction can be used with any block cipher
  - MACs constructed from MDCs:
    - This very common approach raises some cryptographic concern as it makes some implicit but unverified assumptions about the properties of the MDC

Common Structure of Cryptographic Hash Functions

- Like most of today's block ciphers follow the general structure of a Feistel network, most cryptographic hash functions in use today follow a common structure:
  - Let y be an arbitrary message. Usually, the length of the message is appended to the message and it is padded to a multiple of some block size b. Let $(y_0, y_1, \ldots, y_{L-1})$ denote the resulting message consisting of L blocks of size b
  - The general structure is as depicted below:



  - CV is a *chaining value*, with $CV_0 := IV$ and $MDC(y) := CV_L$
  - f is a specific compression function which compresses $(n + b)$ bit to n bit

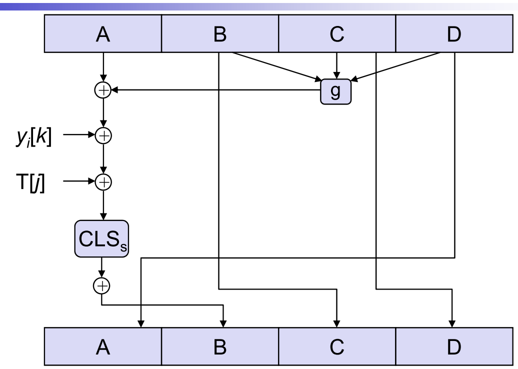
Common Structure of Cryptographic Hash Functions

- The hash function H can be summarized as follows:
  - CV_0 = IV = initial n-bit value
  - $CV_i = f(CV_{i-1}, y_{i-1})$ $\quad 1 \le i \le L$
  - $H(y) = CV_L$
- It has been shown [Mer89a] that if the compression function f is collision resistant, then the resulting iterated hash function H is also collision resistant.
- Cryptanalysis of cryptographic hash functions thus concentrates on the internal structure of the function f and finding efficient techniques to produce collisions for a single execution of f
- Primarily motivated by birthday attacks, a common minimum suggestion for n, the bit length of the hash value, is 160 bit, as this implies an effort of order 2^{80} to attack which is considered infeasible today

The Message Digest 5

- MD5 follows the common structure outlined before [Riv92a]:
  - The message y is padded by a "1" followed by 0 to 511 "0" bits such that the length of the resulting message is congruent 448 modulo 512
  - The length of the original message is added as a 64-bit value resulting in a message that has length which is an integer multiple of 512 bit
  - This new message is divided into blocks of length b = 512 bit
  - The length of the chaining value is n = 128 bit
    - The chaining value is "structured" as four 32-bit registers A, B, C, D
    - Initialization: `A := 0x 01 23 45 67` `B := 0x 89 AB CD EF`
      `C := 0x FE DC BA 98` `D := 0x 76 54 32 10`
    - This initialization vector is in little-endian format
  - Each block of the message y_i is processed with the chaining value CV_i with the function f which is internally realized by 4 rounds of 16 steps each
    - Each round uses a similar structure and makes use of a table T containing 64 constant values of 32-bit each,
    - Each of the four rounds uses a specific logical function g

The Message Digest 5 – Structure of One Step



- The function g is one of four different logical functions
- $y_i[k]$ denotes the k^{th} 32-bit word of message block i
- $T[j]$ is the j^{th} entry of table t with j incremented modulo 64 every step
- CLS_s denotes cyclical left shift by s bits with s following some schedule

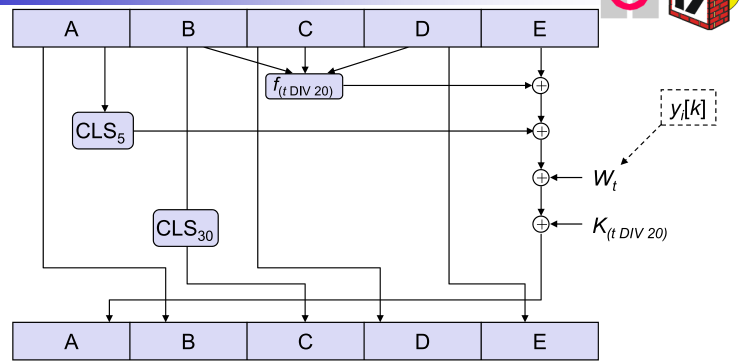
The Message Digest 5

- The MD5-MDC over a message is the content of the chaining value CV after processing the final message block
- Security of MD5:
  - Every bit of the 128-bit hash code is a function of every input bit
  - Between 1992 and 1996 significant progress in cryptanalyzing MD5 has been published:
    - In 1996 H. Dobbertin published an attack that allows to generate a collision for the function f (realized by the 64 steps described above).
    - While this attack has not yet been extended to a full collision for MD5 with its initialization vector, it raises nevertheless serious concern.
  - In reaction to this RSA Laboratories publish in 1996 [Rob96a]:
    - *"Existing signatures formed using MD5 are not at risk and while MD5 is still suitable for a variety of applications (namely those which rely on the one-way property of MD5 and on the random appearance of the output) as a precaution it should not be used for future applications that require the hash function to be collision-resistant."*

The Secure Hash Algorithm SHA-1

- Also SHA-1 follows the common structure as described above:
  - SHA-1 works on 512-bit blocks and produces a 160-bit hash value
  - As it design was also inspired by the MD4 algorithm, its initialization is basically the same like that of MD5:
    - The data is padded, a length field is added and the resulting message is processed as blocks of length 512 bit
    - The chaining value is structured as five 32-bit registers A, B, C, D, E
    - Initialization: `A = 0x 67 45 23 01` `B = 0x EF CD AB 89` `C = 0x 98 BA DC FE` `D = 0x 10 32 54 76` `E = 0x C3 D2 E1 F0`
    - The values are stored in big-endian format
  - Each block y_i of the message is processed together with CV_i in a module realizing the compression function f in four rounds of 20 steps each.
    - The rounds have a similar structure but each round uses a different primitive logical function f_1, f_2, f_3, f_4
    - Each step makes use of a fixed additive constant K_t, which remains unchanged during one round

The Secure Hash Algorithm SHA-1 – One Step



- $t \in \{0, ..., 15\} \Rightarrow W_t := y_i[t]$
  $t \in \{16, ..., 79\} \Rightarrow W_t := CLS_1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$
- After step 79 each register A, B, C, D, E is added modulo 2^{32} with the value of the corresponding register before step 0 to compute CV_{i+1}

The Secure Hash Algorithm SHA-1

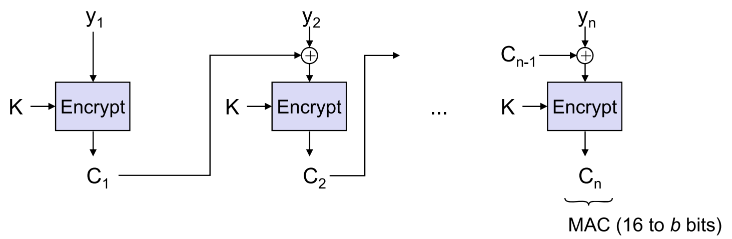
- The SHA-1-MDC over a message is the content of the chaining value CV after processing the final message block
- Security of SHA-1:
  - As SHA-1 produces MDCs of length 160 bit, it offers better security against brute-force and birthday attacks than MD5
  - Up to now, no cryptanalytic results against the compression function of SHA-1 have been published
    - However, it has to be stated, that the design criteria of SHA-1 are not known, which makes cryptanalysis more difficult
- Further comparison between SHA-1 and MD5:
  - Speed: SHA-1 is about 25% slower than MD5 (CV is about 25% bigger)
  - Simplicity and compactness: both algorithms are simple to describe and implement and do not require large programs or substitution tables
  - Little-endian vs. big-endian architecture: no advantage of either approach
  - RSA Laboratories (who invented MD5) recommend SHA-1 or RipeMD-160 for applications that require collision resistance [Rob96a]

Constructing a MAC from a MDC

- Reasons for constructing MACs from MDCs:
  - Cryptographic hash functions generally execute faster than symmetric block ciphers
  - There were no export restrictions to cryptographic hash functions
- Basic idea: “mix” a secret key *K* with the input and compute an MDC
  - The assumption that an attacker needs to know *K* to produce a valid MAC nevertheless raises some cryptographic concern:
    - The construction *H(K, m)* is not secure (see note 9.64 in [Men97a])
    - The construction *H(m, K)* is not secure (see note 9.65 in [Men97a])
    - The construction *H(K, p, m, K)* with *p* denoting an additional padding field does not offer sufficient security (see note 9.66 in [Men97a])
  - The most used construction is: $\boldsymbol{H(K, p_1, H(K, p_2, m))}$
    - Two different padding patterns p_1 and p_2 are used to fill up the key to one input block of the cryptographic hash function
    - This scheme seems to be secure (see note 9.67 in [Men97a])
    - It has been standardized in RFC 2104 [Kra97a] and is called *HMAC*

Cipher Block Chaining Message Authentication Codes

- A CBC-MAC is computed by encrypting a message in CBC Mode and taking the last ciphertext block or a part of it as the MAC:



  - This MAC needs not to be signed any further, as it has already been produced using a shared secret K
    - However, it is not possible to say who exactly has created a MAC, as everybody (sender, receiver) who knows the secret key K can do so
  - This scheme works with any block cipher (DES, IDEA, ...)

Cipher Block Chaining Message Authentication Codes

- Security of CBC-MAC:
  - As an attacker does not know *K*, a birthday attack is much more difficult to launch (if not impossible)
  - Attacking a CBC-MAC requires known (message, MAC) pairs
  - This allows for shorter MACs
  - A CBC-MAC can optionally be strengthened by agreeing upon a second key K’ ≠ K and performing a triple encryption on the *last* block:

$$MAC = E(K, D(K', E(K, C_{n-1})))$$

  - This doubles the key space while adding only little computing effort
- There have also been some proposals to create MDCs from symmetric block ciphers with setting the key to a fixed (known) value:
  - Because of the relatively small block size of 64 bit of most common block ciphers, these schemes offer insufficient security against birthday attacks
  - As symmetric block ciphers require more computing effort than dedicated cryptographic hash functions, these schemes are relatively slow

Summary (what do I need to know)

- Principles of cryptographic hash functions
  - Modification detection code (MDC)
  - Message authentication code (MAC)
- MD5
  - Operation principles
  - Security
- MAC
  - H-MAC – using a cryptographic hash function
  - CBC-MAC – using a symmetric block cipher in CBC mode

Additional References

[Men97a] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1997.

[Kra97a] H. Krawczyk, M. Bellare, R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* Internet RFC 2104, February 1997.

[Mer89a] R. Merkle. *One Way Hash Functions and DES.* Proceedings of Crypto '89, Springer, 1989.

[Riv92a] R. L. Rivest. *The MD5 Message Digest Algorithm.* Internet RFC 1321, April 1992.

[Rob96a] M. Robshaw. *On Recent Results for MD2, MD4 and MD5.* RSA Laboratories' Bulletin, No. 4, November 1996.

[Yuv79a] G. Yuval. *How to Swindle Rabin.* Cryptologia, July 1979.