CS Revision Lecture 8

- ## Lecture 8 - Cryptography: cryptographic hash functions and MACs
  - ### Introduction
    - Encryption $\Longrightarrow$ confidentiality against eavesdropping
    - What about authenticity and integrity against an active attacker?
      - $\rightarrow$ cryptographic hash functions and Message authentication codes
  - ### One-way functions (OWFs)
    - A OWF is a function that is easy to compute but hard to invert
    - **Definition**
      - A function f is a one-way unction if **for all y there is no efficient algorithm which can compute x such that $f(x) = y$**
    - **Constant functions ARE NOT OWFs**
      - any x is such that $f(x) = c$
    - **The successor function in $\mathbb{N}$ IS NOT a OWF**

      S(n) = n+1

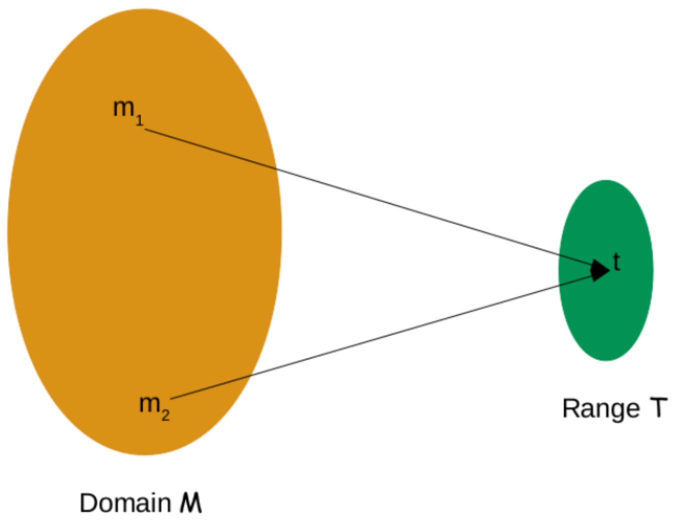
      n = S(n) - 1
      - given $succ(n)$ it is easy to retrieve $n = succ(n) - 1$
    - **Multiplication of large primes IS a OWF**
      - integer factorisation is a hard problem - given p x q (where p and q are primes) it is hard to retrieve p and q
  - ### Collision-resistant functions (CRFs)

- A function is a CRF if it is hard to find two messages that get mapped to the same value through this function
- **Definition**
  - A function f is collision resistant if there is no efficient algorithm that can find two messages m_1 and m_2 such that $f(m_1) = f(m_2)$
- **Constant functions ARE NOT CRFs**
  - for all m_1 and m_2, $f(m_1) = f(m_2)$
- **The successor function in $\mathbb{N}$ IS a CRF**
  - the predecessor of a positive integer is unique
- **Multiplication of large primes IS a CRF**
  - every positive integer has a unique prime factorisation

- Cryptographic hash functions
  - A cryptographic hash function **takes messages of arbitrary length** and **returns a fixed-size bit string** such that **any change to the data will** (with very high probability) **change the corresponding hash value.**
  - Definition
    - A cryptographic hash function $H: M \longrightarrow T$ is a function that satisfies the following 4 properties
      - $|M| >> |T|$

        length of message > length of the hash value
      - it is easy to compute the hash value for any given message
      - it is hard to retrieve a message from its hashed value (OWF)
      - it is hard to find two different messages with the same hash value (CRF)

- Examples: MD4, MD5, SHA-1, RIPEMD160, SHA-256, SHA-512, etc..
  - In new projects use SHA-256 or SHA-512
- Applications
  - **Commitments**
    - Allow a participant to commit to a value v by publishing the hash $H(v)$ of this value, but revealing v only later
    - Ex: electronic voting protocols, digital signatures
  - **File integrity**
    - Hashes are sometimes posted along with files on "read-only" spaces to allow verification of integrity of the files.
    - Ex:SHA-256 is used to authenticate Debian GNU/Linux software packages
  - **Password verification**
    - Instead of storing passwords in cleartext, only the hash digest of each password is stored. To authenticate a user, the password presented by the user is hashed and compared with the stored hash.
  - **Key derivation**
    - Derive new keys or passwords from a single, secure key or password
  - **Building block of other crypto primitives**
    - Used to build MACs, block ciphers, PRG, ...
- Collisions are unavoidable



- The domain being much larger than the range, collisions necessarily exist

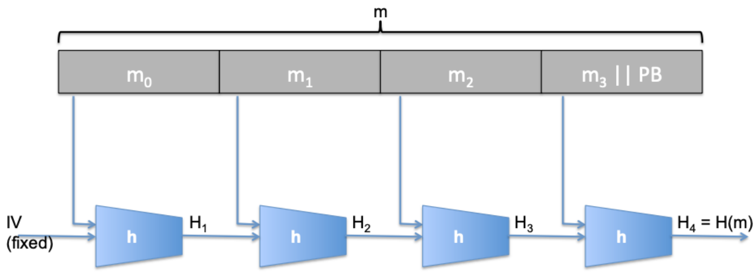
- The birthday attack -attack on all schemes
  - Theorem
    - Let *H*: $M \to \{0,1\}^n$ be a cryptographic hash function ($|M| >> 2^n$) Generic algorithm to find a collision in time $O(\sqrt{2^n}) = O(2^{n/2})$ hashes:
      - 1. Choose $2^{n/2}$ random messages in M : $m_1, ..., m_{2^{n/2}}$
      - 2. For $i = 1, ..., 2^{n/2}$ compute $t_i = H(m_i)$
      - 3. If there exist a collision($\exists i, j . t_i = t_j$)
        - then return (m_i, m_j)
        - else go back to 1
  - Birthday paradox
    - Let $r_1, ..., r_l \in \{1, ..., N\}$ be independent variables
    - For $l = 1.2 \times \sqrt{N}, Pr(\exists i \neq j . r_i = r_j) \geq \frac{1}{2}$

      For example, the probability that at least 2 out of 23 babies will have the same birthday is 0.5

      l = 1.2 *   = 23
      - $\Longrightarrow$ the expected number of iteration is 2
      - $\Longrightarrow$ running time $O(2^{n/2})$

- $\implies$ Cryptographic function used in new projects should have output length $n \geq 256$!

- The Merkle-Damgard construction



▶ Compression function: $h: \mathcal{T} \times \mathcal{X} \to \mathcal{T}$
▶ PB: $1000 \ldots 0||$mes-len (add extra block if needed)

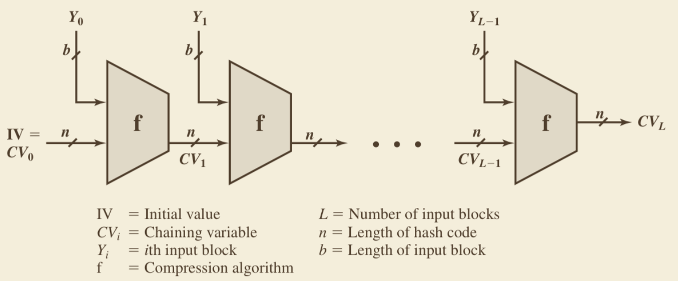
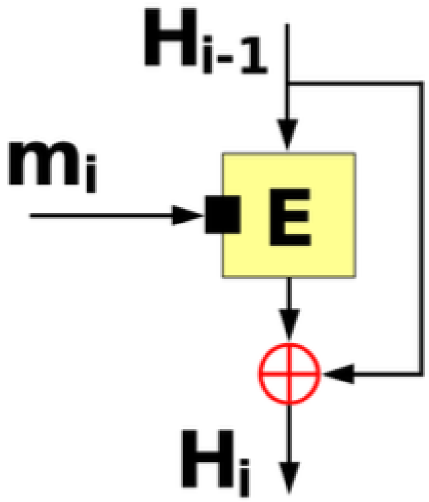


Figure 11.8 General Structure of Secure Hash Code

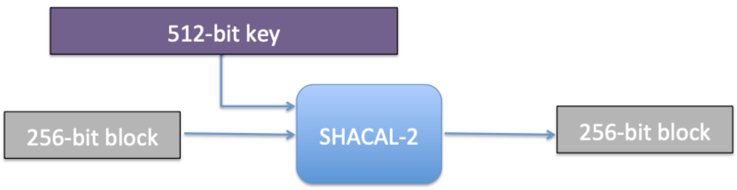
- How it works
  - The **compression function *h***, takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output.
  - At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, b > n; hence the term **compression**. The hash function can be summarized as
    - $CV_0 = IV$ = initial n-bit value
    - $CV_i = f(CV_{i-1}, Y_{i-1})\ 1 \leq i \leq L$
    - $H(M) = CV_L$
- Theorem
  - Let H be built using the MD construction to the compression function *h*. If H admits a collision, so does h.
- Example of MD constructions: MD5, SHA-1, SHA-2, ...

- Compression functions from block ciphers
    - Let $E : K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher



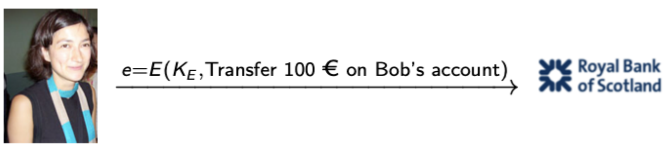
Davies-Meyer

- Example of cryptographic hash function: SHA-256

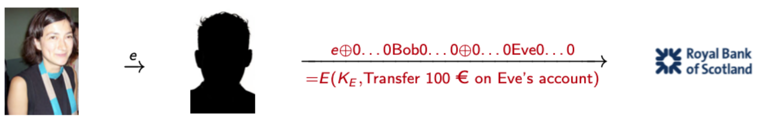


    - Structure: Merkle-Damgard
    - Compression function: Davies-Meyer
    - Block cipher: SHACAL-2

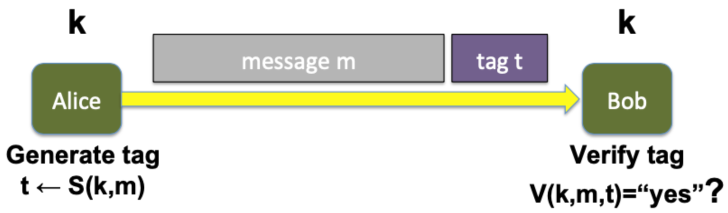
- Message Authentication Codes (MACs)
    - Encryption is not always enough



    - What if the encryption scheme *E* is the OTP
        - e = K_E ⊕ Transfer 100 AC on Bob’s account?
    - Using XOR to modify the original message

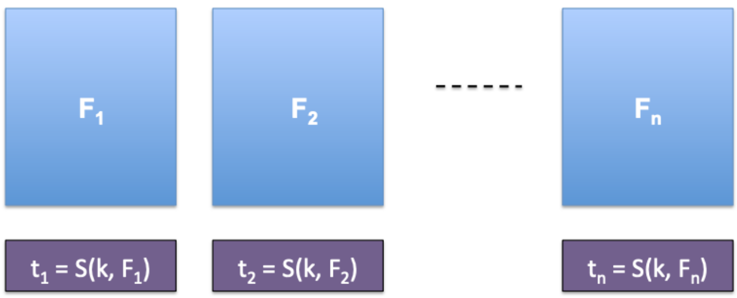


- Goal: message integrity



  - A MAC is a pair of algorithms (*S, V*) defined over (*K, M, T*):
    - $S : K \times M \to T$
    - $V : K \times M \times T \to \{\top, \bot\}$

      Return True or False
    - Consistency: $V(k, m, S(k, m)) = T$
  - **Unforgeability**
    - It is hard to compute a valid pair $(m, S(k, m))$ without knowing k
- File system protection
  - At installation time

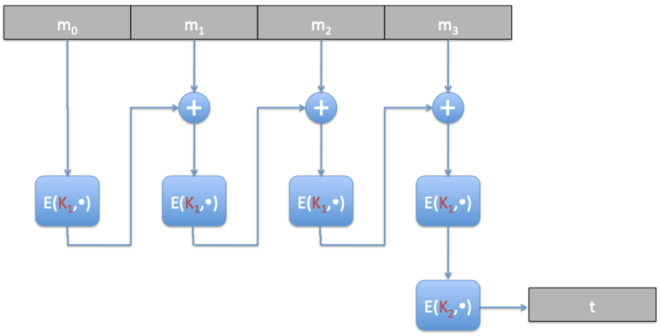
    

    k derived from user password
  - To check for virus file tampering/alternation
    - reboot to clean OS
    - supply password
    - any file modification will be detected
- Block ciphers and message integrity
  - Let (E, D) be a block cipher. We build a MAC (S, V) using (E, D) as follows:

- $S(k,m) = E(k,m) = t$
- $V(k,m,t)$ = if m = $D(k,t)$ then return $\top$ else return $\perp$

- But: block ciphers can usually process only 128 or 256 bits
- Our goal now: construct MACs for long messages

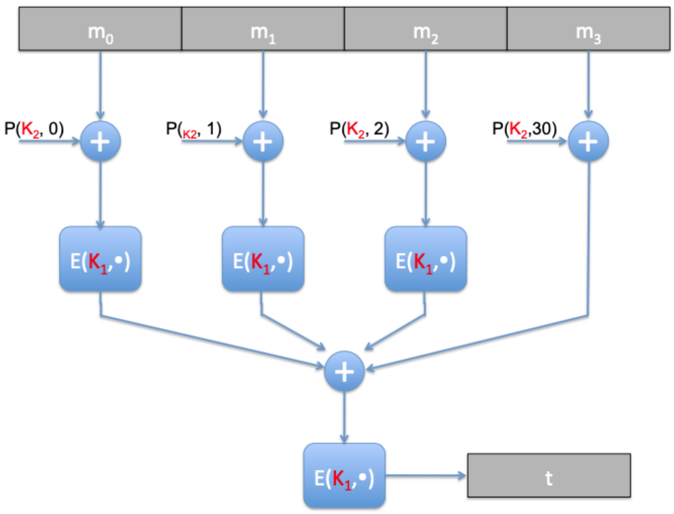
- ECBC-MAC



- Encrypt the message block, then next message block XOR with the encrypted message. Keep doing until the end
- $E : K \times \{0,1\}^n \to \{0,1\}^n$ a block cipher
- ECBC-MAC : $K^2 \times \{0,1\}^* \to \{0,1\}^n$
- **the last encryption is crucial to avoid forgeries!!**

  Use different key to encrypt it again
- Example: 802.11i (WPA for Wi-Fi) uses AES based ECBC-MAC

- PMAC (parallel MAC)



- $E : K \times \{0,1\}^n \to \{0,1\}^n$ a block cipher

        - $P : K \times \mathbb{N} \to \{0,1\}^n$ any easy to compute function
        - PMAC : $K^2 \times \{0,1\}^* \to \{0,1\}^n$
    - HMAC
        - MAC build from cryptographic hash functions
            - *HMAC* $(k, m) = H(k \oplus OP || H(k \oplus IP || m))$
        - IP, OP: publicly know padding constants
        - Example: SSL, IPsec, SSH, ...
- Authenticated Encryption
    - Plain encryption is malleable
        - The decryption algorithm never fails
        - Changing one bit of the i^{th} block of the ciphertext
            - CBC decryption: will affect i^{th} and i + 1^{th} block of the plaintext
            - ECB decryption: will only affect the i^{th} block of the plaintext
            - CTR decryption: will only affect one bit of the i^{th} block of the plaintext
        - Decryption should fail if a ciphertext was not computed using the key
        - Goal
            - Simultaneously provide data confidentiality, integrity and authenticity
                - $\rightarrow$ decryption combined with integrity verification in one step
- Encrypt-then-MAC
    - 1. Always compute the MACs on the ciphertext, never on the plaintext
    - 2. Use two different keys, one for encryption (K_E) and one for the MAC (K_M)

Encryption

1. $C \leftarrow E_{AES}(K_E, M)$
2. $T \leftarrow HMAC\text{-}SHA(K_M, C)$
3. return $C||T$

Decryption

1. if $T = HMAC\text{-}SHA(K_M, C)$
2. then return $D_{AES}(K_E, C)$
3. else return $\perp$

    - **Do Not**
        - Encrypt-and-MAC: $E_{AES}(K_E, M)$ || HMAC-SHA (K_M, M)
        - MAC-then-Encrypt: $E_{AES}(K_E, M$ || HMAC_SHA(K_M, M))
- AES-GCM
    - Galois Counter Mode
        - Combines
            - 1. **Galois field** based One-time MAC for authentication
            - 2. **AES** based Counter Mode for encryption
    - Trick: One-time MAC is encrypted too
        - $\implies$ secure for many messages
    - Widely adopted for its performance
    - Many good implementations of this mode