Terminology

- threat
  - potential violation of security
- vulnerability
  - concrete flaw in the implementation
- exploit/attack
  - concrete attempt to violate the security

Passwords

- they suck :)
- can be based on what someone knows/has/is
- password storage
  - plain passwords
    * weak to eavesdropping (e.g. replay attack)
    * vulnerable password table
  - hashed passwords
    * still allows mass dictionary attacks
  - hashed passwords with salt
    * no more parallel attacks
    * no longer leaks users with identical password
- weak authentication is susceptible to replay attacks
  - challenge-response
  - prove you know the secret without telling
  - e.g. TOTP

Authenticity using hash functions

3 Properties of Hash Functions:

1) Preimage restistance: T known, must be infeasible to find any message M that produces T (2^t)
2) Second preimage resistance: M known, must be infeasible to find M' with same hash (2^t)
3) Collision resistance: Must be infeasible to find two messages with the same hash ($2^{t/2}$)

-
- birthday paradox
  - $2^{t/2}$ messages => 2^{t-1} message pairs
  - collision probability for one pair is $\frac{1}{2^t} = 2^{-t}$
  - probability for at least one collision $\sim\frac{1}{2}$
- Compression function
  - hash function for fixed-size input
- MD-Hash
  - hash function for input of arbitrary size
    * iterates a compression function

    - padding always applied so the input is a multiple of the block size
    - hash=tag
- MAC
    - hash function which uses symmetric key K (k-bits) to compute tag t (t-bits) for authentication of message M
    - HMAC = hash based MAC
    - application
        * compute T for M
        * send M and T
        * receiver recomputes new tag T' on M
        * receiver verifies T=T'
    - unforgeability
        * infeasible for attacker to forge any new valid pair (M,T) even if they can query tags for any other messages
    - complexity
        * Exhaustive key search takes ~2^k offline trials
        * Guessing the tag takes ~2^t online trials
- Signatures
    - uses encryption instead of hashing
    - uses asymmetric private key K to encrypt message M to a signature S
    - send M and S
    - receiver decrypts S with public key to M' and verifies M=M'

Confidentiality using encryption

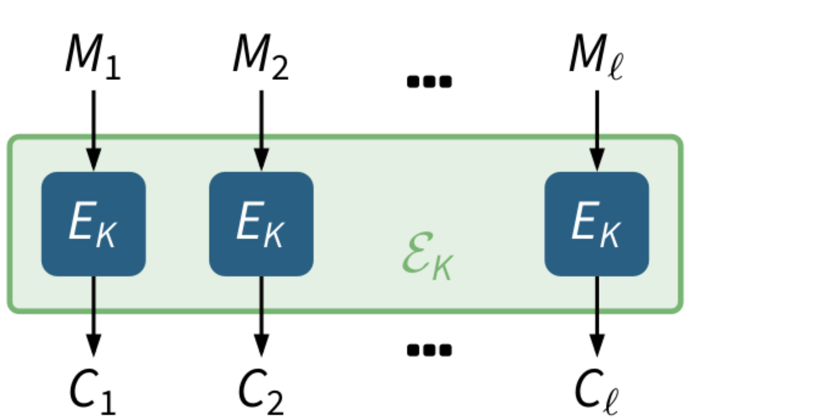
- block ciphers
    - bijective permutation E_K based on k-bit key K to encrypt n-bit message blocks M into n-bit cipher text blocks C
    - inverse permutation $D_K = E_K^{-1}$ for decryption
    - complexity
        * 2^k possible keys(mappings)
        * 2^n possible outputs for input
    - requirements
        * pseudorandomness
            - unable to learn M from C (or vice-versa)
        * key recovery security
            - unable to recover K given any arbitrary number of (M, C) pairs
- key-alternating using key schedule
    - each round/iteration depends on different round key which has been derived from K

- e.g. AES
  - Block size $n = 128$ bits
  - Key size $k \in \{128, 192, 256\}$ bits $\rightarrow$ ciphers AES-128, AES-192, AES-256
  - The 16-byte input block $M = s_{00} \| s_{10} \| s_{20} \| s_{30} \| s_{01} \| \ldots \| s_{33}$ is written as a 4×4 matrix of bytes, the $\{16, 24, 32\}$-byte key K as a $4 \times \{4, 6, 8\}$ matrix:

    $$M = \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix}, \qquad K = \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} & k_{04} & k_{05} & k_{06} & k_{07} \\ k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} \\ k_{20} & k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} \\ k_{30} & k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} \end{bmatrix}$$

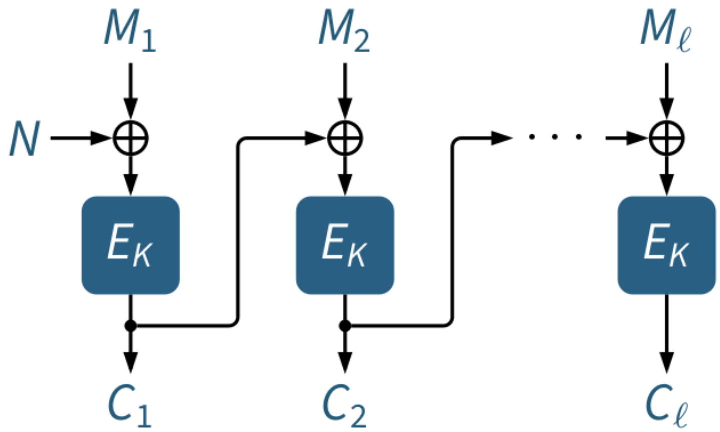
  - The state is initialized to M and updated in 10 rounds (for AES-128) or 12 rounds (AES-192) or 14 rounds (AES-256).

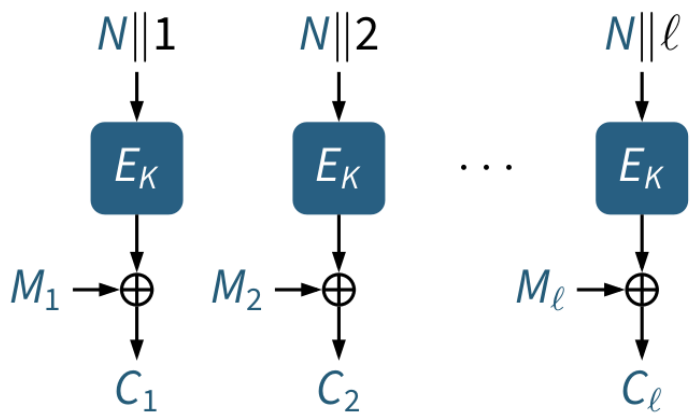
  *
  * SubBytes
    - substitute using lookup table S-box with original byte as key
    - $b_{ij} = S[a_{ij}]$
  * ShiftRows
    - shift row i by i bytes to the left
    - $b_{ij} = a_{i(j+i\%4)}$
  * MixColumns
    - multiplication of each column with constant matrix M
    - $$(b_{0j}, b_{1j}, b_{2j}, b_{3j}) = M \cdot (a_{0j}, a_{1j}, a_{2j}, a_{3j})$$
  * AddRoundKey
    - XOR with $k^{(r)}$
    - $b_{ij} = a_{ij} \oplus k^{(r)}$

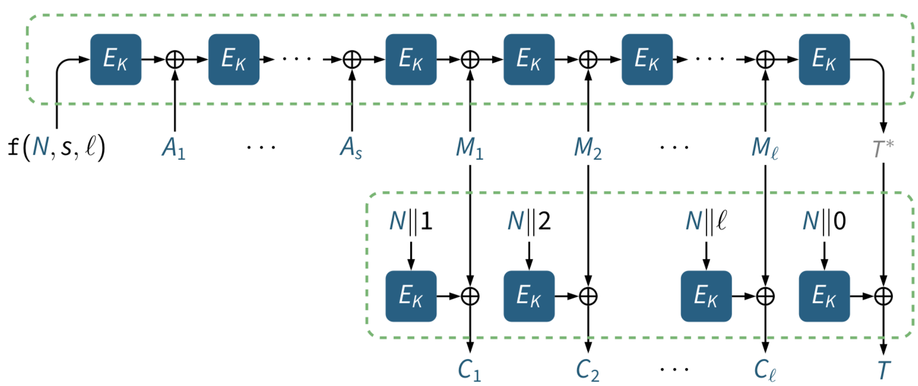
- regular encryption
  - does not provide authentication
  - ECB
    * 

      ⚠ Patterns: Two identical blocks M_i, M_j get encrypted to the same C_i, C_j

    * ⚠ Context: Two identical messages M, M' get encrypted to the same C, C'

- CBC
  - 

  - C also depends on nonce and previous blocks
- CTR
  - 

  - C also depends on nonce and block index
- Authenticated Encryption (with Associated Data)
  - produces cipher text C and tag T for message M using symmetric key K, nonce N and associated data A (e.g. metadata or system parameters)
  - some TLS 1.3 authenticated ciphers
    - AES-CCM (CTR using AES encryption with CBC-MAC authentication)
      - 

    - AES-GCM (default)
- Asymmetric encryption schemes

- Preliminary maths
  - Euler function for product n of 2 primes p, q
    - $\varphi(n) = \varphi(pq) = (p-1)(q-1)$
  - Euler theorem
    - a,n are coprime $\Leftrightarrow a^{\varphi(n)} \equiv 1 \pmod{n}$
  - **Discrete Logarithm Problem**

    Given a prime number p, a generator $g \in \mathbb{Z}_p^*$, and an element $y \in \mathbb{Z}_p^*$, find the integer $x \in \{0, \ldots, p-2\}$ such that $\underbrace{g \cdot g \cdots g}_{x \text{ times}} = g^x \equiv y \pmod{p}$.
  - **Integer Factorization Problem**

    Given $n \in \mathbb{N}$, find primes p_i and exponents $e_i \in \mathbb{N}$ such that $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$
  - **Diffie–Hellman Problem (DHP)**

    Given generator $\alpha \in \mathbb{Z}_p^*$ and $\alpha^a \pmod{p}$, $\alpha^b \pmod{p}$, find $K_{AB} = \alpha^{a \cdot b}$.
  - **RSA Problem (RSAP)**

    Given modulus n, exponent e, ciphertext C: find M such that $M^e \equiv C \pmod{n}$.
- Key exchange
  - agree on shared symmetric key while communicating over insecure channel
  - Diffie-Hellman
    - public: large prime p and generator α
    - Alice chooses private key $a \in 2, ..., p-2$ and sends public key α^a to Bob
    - Bob chooses private key $b \in 2, ..., p-2$ and sends public key α^b to Alice
    - $K_{AB} \equiv (\alpha^b)^a \pmod{p} \equiv (\alpha^a)^b \pmod{p}$
- Asymmetric encryption
  - uses private and public key
  - RSA
    - **Key Generation**

      Choose 2 large, random primes p, q

      Compute modulus $n = p \cdot q$

      Choose public exponent e co-prime to $\varphi(n)$

      Compute private exponent $d \equiv e^{-1} \pmod{\varphi(n)}$

      public key $= (e, n)$ private key $= (d, n)$

      Euler function:
      $\varphi(pq) = (p-1)(q-1)$

      Euler theorem:
      if a, n are coprime, then
      $a^{\varphi(n)} \equiv 1 \pmod{n}$

      Encrypt $\mathcal{E}(M)$

      Encrypt message M:

      $C \equiv M^e \pmod{n}$

      Decrypt $\mathcal{D}(C)$

      Decrypt ciphertext C:

      $M \equiv C^d \pmod{n} \equiv M^{e \cdot d} \equiv M^{1+k\varphi(n)} \equiv M$
    - Square-and-Multiply b^e
      - $result := 1$
      - for each bit in e

                - $result := result^2$
                - if bit is set
                    - $result := result * b$
        - textbook RSA is deterministic
            - use padding scheme
            - **Indistinguishability (under Adaptive Chosen-Ciphertext Attack)**

              An attacker who knows the public key, chooses 2 messages M_0, M_1, and gets ciphertext C **can not distinguish** whether $C = E(M_0)$ or $C = E(M_1)$,

              even if they can ask for decryption of any $C^* \neq C$.
            - e.g. RSAES-OAEP

Protocols

- problem with static asymmetric crypto
- no forward secrecy
    - if private key is leaked $\Rightarrow$ all past communications compromised
- no authenticity
    - no assurance with whom the key is exchanged
- Ephemeral Diffie-Helman DHE
    - Alice and Bob both have long term private/public key pair
    - execute regular DH over insecure channel
        - both compute the same K_{AB}
    - send each other the signed transcript (all previous message) of the exchange
        - signed with long term private keys
    - send each other MAC-tag of transcript
        - use K_{AB} to create tag
    - throw away public/private keys a, b, α^a, α^b from DH
- Transport Layer Security TLS
    - Key exchange using DHE
        - exchange ephemeral public DH key, randomness and list of preferred symmetric ciphers
        - 

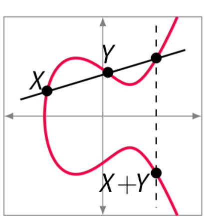
    - Authentication
        - server sends certificate, signature over transcript and HMAC of transcript

      - ◆ signature using long term private key
      - ◆ HMAC using K_AB
    - ∗ client sends HMAC of transcript back

    - ∗
  - Sending application data
    - ∗ send messages encrypted with new symmetric keys derived from K_{AB} with HKDF
      - ◆ HMAC-based key derivation function

Certificates + ties public key to an identity + X.509 standard contains + public key + identity information (e.g. name) + validity period + signature from a certificate authority CA + which issued the certificate

Miscellaneous

- Kerckhoffs' Principle
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
  - aka Shannon's Maxim: "The enemy knows the system"
    Opposite of "Security by obscurity"
- Elliptic Curve Cryptography ECC
  - ■ An attractive alternative is the Elliptic Curve group, where each element is not an integer but a 2-dimensional point with two integer coordinates. The group operation is addition with special point addition formulas.
  - 

    EC Discrete Logarithm Problem (ECDLP)

    Given points P, Q on an elliptic curve with

    $$Q = k \cdot P = \underbrace{P + P + \ldots + P}_{k \text{ times}},$$

    Find k.
- End-to-End Encryption
  - may require more security properties

        * Security properties: confidentiality, integrity, authentication, forward secrecy, post-compromise security, participant consistency, destination validation, causality preservation, message unlinkability, message repudiation, participation repudiation, asynchronicity, …

- Secure Multiparty Computation
    - multiple parties compute a result together without sharing their inputs
    - e.g. compute sum of consumed electricity without exposing each household's individual consumption
- Private Set Intersection
    - find intersection of two sets without sharing their content
    - e.g. tell new user which of their contacts also use Whatsapp without exposing all contacts to Whatsapp or all Whatsapp users to the new user
- RNG
    - nondeterministic hardware source
        * generate random number from physical process
    - deterministic pseudorandomness
        * PRNG generates random number (sequence) based on initial value
- Quantum Computing
    - new means of solving algorithms
    - Shor's algorithm solves IFP and DLP in polynomial time
        * breaks signatures (RSA) and key exchange (DH, ECC)
    - symmetric encryption is now slightly weaker
- Common crypto failures
    - using no/obsolete/backdoored/insufficient crypto
    - homebrew protocols
        * combining secure primitives in an insecure manner
    - improper key usage
    - improper password storage
    - bad RNG, low entropy
    - reusing nonces

[[Kryptographie]]