- Digital signatures
  - Motivation and principle of digital signatures
    - For a given message x , a digital signature is appended to the message
    - The signature is realized as a function with the message x and the private key as input.
    - The public key and the message x are the inputs to the verification function (hash).
  - Security services

The objectives of a security systems are called security services.

1. Confidentiality: Information is kept secret from all but authorized parties
2. Integrity: Ensures that a message has not been modified in transit.
3. Message Authentication: Ensures that the sender of a message is authentic. An alternative term is data origin authentication
4. Non-repudiation: Ensures that the sender of a message can not deny the creation of the message. (e.g., order of a pink car)

Identification/entity authentication: Establishing and verification of the identity of an entity, e.g. a person, a computer, or a credit card.
- **who are you?**

Access control/Authorization: Restricting access to the resources to privileged entities. (decide **who can do what?**)

Auditing: Provides evidences about security-relevant activities, e.g., by keeping logs about certain events. (provide a proof **who did what?**)

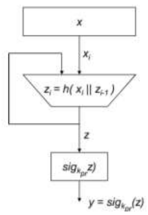
Availability: The electronic system is reliably available.

Auditing: Provides evidences about security-relevant activities, e.g., by keeping logs about certain events.

Physical security: Providing protection against physical tampering and/or responses to physical tampering attempts

Anonymity/privacy: Providing protection against discovery and misuse of identity. (what if we don't want to be identified?)

  - Digital signature schemes and attacks
    - Digital Signature Standard (DSA)
    - Elliptic Curve DSA
- Hash Functions
  - Motivation, security properties, attacks and algorithms
    - Naïve signing of long messages generates a signature of same length
    - Problems: Computational/double message overhead/security
    - Sign long messages with hash(x)
      - x has fixed length
      - z, y have fixed length
      - z, x do not have equal length in general
      - h(x) does not require a key.
      - h(x) is public.



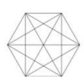
    - Properties:
      - Compression
      - Efficiency
      - Preimage resist.: for any msg=z impossible to find any input x such that h(x)=z , i.e., h(x) is one-way
      - 2nd preimage resist: computationally infeasible to find any x' s.t. h(x)= h(x')
      - Collision Resist (**<u>COLLISION ATTACK INFEASIBLE</u>**)
- Symmetric and asymmetric encryption schemes
  - Using RSA, El-Gamal, ... to encrypt/decrypt
    - ELGAMEL
      - BAD: ciphertext is twice as long as the plaintext.
      - GOOD:the same plaintext gives a different ciphertext each time it is encrypted.
  - Symmetric Cryptography: Shortcomings

Symmetric algorithms, e.g., AES or 3DES, are very secure, fast & widespread **but**:

Key distribution problem: The secret key must be transported securely
- i.e., when Alice and Bob communicate using a symmetric system, they need to securely exchange their shared key k_{ab}

Key management: In a network, each pair of users requires an individual key
- $\rightarrow$ n users in the network require $\frac{n\times(n-1)}{2}$ keys, each user stores $(n-1)$ keys
- If Alice wants to talk to Bob, Carol and Dave, she needs to exchange and maintain k_{ab}, k_{ac}, and k_{ad}

Example: 6 users (nodes)
$\frac{6\times5}{2} = 15$ keys (edges)



No Protection Against Cheating by Alice or Bob: Alice or Bob can **cheat each other**, because they have identical keys.

Who is the author of a message encrypted with k_{ab}, a key Alice and Bob share?
- Example: Alice can claim that she never ordered a TV on-line from Bob (he could have fabricated her order). To prevent this: "non-repudiation"

  - Practical Aspects, security mechanisms, and important Public-Key Algorithms

Here are main mechanisms that can be realized with asymmetric cryptography:
- Symmetric Key Distribution (e.g., Diffie-Hellman key exchange, RSA) without a pre-shared secret (key)
- Nonrepudiation and Digital Signatures (e.g., RSA, DSA or ECDSA) to provide message integrity
  - i.e., Integrity/authentication: encipher using private key, decipher using public one
- Encryption (e.g., RSA / Elgamal)
  - Confidentiality: encipher using public key, decipher using private key
  - Disadvantage: Computationally very intensive (1000 times slower than symmetric Algorithms!)

Message Authentication Codes (crypto checksums)

- Motivation and principle behind MACs
  - Message auth.:: Bob computes m = MACk(x) and sends (x,m) to Alice. Alice receives (x,m') and verifies that m'= m .
- Security properties and attacks
  - 1. Cryptographic checksum: A MAC generates a cryptographically secure authentication tag for a given message.
  - 2. Symmetric: MACs are based on secret symmetric keys. The signing and verifying parties must share a secret key.
  - 3. Arbitrary message size: MACs accept messages of arbitrary length.
  - 4. Fixed output length: MACs generate fixed-size authentication tags.
  - 5. Message integrity: MACs provide message integrity: Any manipulations of a message during transit will be detected by the receiver.
  - 6. Message authentication: The receiving party is assured of the origin of the message.
  - 7. No nonrepudiation: Since MACs are based on symmetric principles, they do not provide nonrepudiation

Alice **Oscar** **Bob**

Bob: $x = (x_1, \ldots, x_n)$, $m = h(k\|x_1, \ldots, x_n)$ $\xleftarrow{(x,m)}$

Oscar: ↯ intercept; $x_O = (x_1, \ldots, x_n, x_{n+1})$; $m_O = h(m\|x_{n+1})$ $\xleftarrow{(x_O,m_O)}$

Alice: $m' = h(k\|x_1, \ldots, x_n, x_{n+1})$; since $m' = m_O$

  - $\Rightarrow$ valid signature!
  - Sec.suffix attackable by b-day attack:

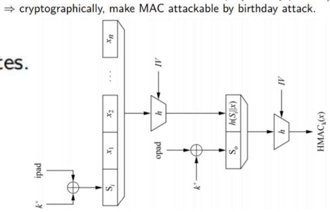
- Assume $x = (x_1, x_2, \ldots, x_n)$
- $m = MAC_k(x) = h(x\|h) = h(x_1\|x_2\|\ldots\|x_n\|k)$
- **Attack**:
  - Assume Oscar can find collisions, x and x_o such that $h(x) = h(x_o)$, then $m = h(x\|k) = h(x_o\|k)$
  - Can replace x with x_o
- Q: Is this a problem, i.e., does Oscar gain anything?
  - $\Rightarrow$ Compare brute-force effort with collision-finding effort:
  - Example: $h() \rightarrow$ SHA-1 (160 bit output) $|K| = 128\text{-}bit \rightarrow$ we expect attacker complexity of 2^{128}
  - but collision search takes $\approx \sqrt{2^{160}} = 2^{80}$ steps (birthday paradox)
  - $\Rightarrow$ cryptographically, make MAC attackable by birthday attack.

- MAC with hash functions and with block ciphers
  - HMAC in SSL/TLS

Let B be the block length of hash in bytes.

$0x36$ repeated B times for $ipad$

$0x5c$ repeated B times for $opad$



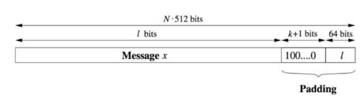
    - e.g., $B = 64$ for MD5 and SHA-1
    - Chained Block Cipher MACs
      - MAC Generation
        - Divide the message x into blocks x_i
        - Compute first iteration $y_1 = e_k(x_1 \oplus IV)$
        - Compute $y_i = e_k(x_i \oplus y_{i-1})$ for the next blocks
        - Final block is the MAC value: $m = MAC_k(x) = y_n$
      - MAC Verification
        - Repeat MAC computation (m')
        - Compare results:
          - If $m' = m$, the message is verified as correct
          - If $m' \neq m$, the message and/or the MAC value m have been altered during transmission
    - Signatures verified by public key of asymmetric key pair
    - MACs verified by recipient via hash with shared secret key
  - RSA Encryption

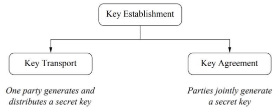
Perform encryption using the RSA algorithm, for the following: 1. p = 3, q = 11, e = 7, M = 5	**In a public-key system using RSA, you intercept the ciphertext C =10 public key is e=5 , n=35 . What is the plaintext M ?**
$n = 3\times 11 = 33$ $\varphi(pq) = (3-1)(11-1) = 20$ $(de) mod\ 20 = 1$ de = 20+1= 21 such that e is a number where gcd(20,e)=1 e=7, d = 3 (trial and error) $C = M^e mod\ n = 5^7 mod\ 33 = 14$	$M = C^d\ mod\ n = M = 10^d mod(35)$ p = 5, q = 7 $de\ mod\varphi(pq) = 1 = 5(d) mod 24$ d = 5 $M = 10^5 mod(35)$ M = 5

RSA IS A PROBLEM OF FACTORING

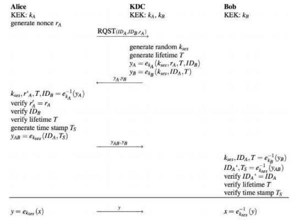
DHKE IS A PROBLEM OF COMPUTATIONAL REDUCTION BY LOSS IN SECURITY (See formula diffs)

SHA-1: Message x has to be padded to fit a size of a multiple of 512 bit.
- Let x with a length of l bit
- To obtain an overall message size of a multiple of 512 bits
  - Append a single 1 followed by k zero bits and the binary 64-bit representation of l.
  - Consequently, the number of required zeros k is given by $k \equiv 512 - 64 - 1 - l = 448 - (l + 1)\ mod\ 512$



- Key Management & Identity
  - Key Distribution Problem
    - Each pair of users has a key (n-1 keys per user so n(n-1)~n^2 keys)
  - Classification of Key Establishment Methods.
    - 

  - Key Establishment with Key Distribution Center, Kerberos, …
    - Kerberos protects against: Replay atk/key confirmation atk (intercept)
    - **Client create authenticator**
    - **Authenticator creates "tickets" for KDC interaction**

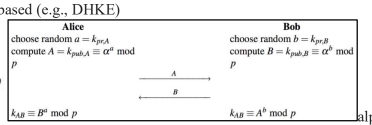


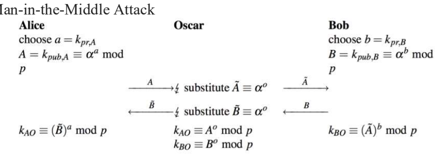
Protects against both attacks

Adds **Lifetime** T, **Time stamp** T_S and **Nonce** (only KDC can send new keys)

  - Short coming of key establishment-symmetric key based
    - No Perfect Forward Secrecy: If the KEK s are compromised, an attacker can decrypt past messages if he stored the corresponding ciphertext
    - Single point of failure: The KDC stores all KEK s. If an attacker gets access to this database, all past traffic can be decrypted.
    - Communication bottleneck: The KDC is involved in every communication in the entire network (can be countered by giving the session keys a long lifetime)
    - Secure channel during initialization: when a new user joins - public key cipher for new key transport
    - A cryptographic protocol has perfect forward secrecy (PFS) if the compromise of long-term keys does not allow an attacker to obtain past session keys. The main mechanism to assure PFS is to employ public-key techniques.
  - key establishment-Asymmetric key based (e.g., DHKE)
    - $K_{pub,A} = \propto^{K_{pr,A}} \, mod\ p$

      $SharedKey(A) = K_{pub,B}{}^{K_{pr,A}} mod\ p$

      $SharedKey(B) = K_{pub,A}{}^{K_{pr,B}} mod p$  alpha primitive root
  - Man-in-the-Middle Attack



Oscar computes a session key k_{AO} with Alice, and k_{BO} with Bob

However, Alice and Bob think they are communicating with each other!

The attack efficiently performs 2 DH key-exchanges: Oscar-Alice and Oscar-Bob

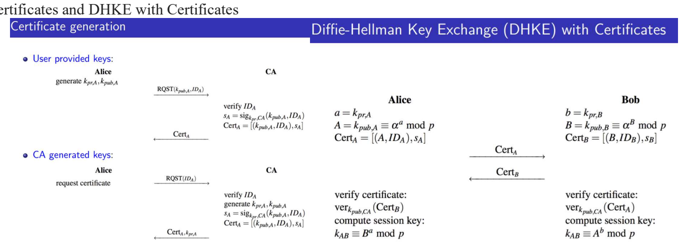
Here is why the attack works:

Alice computes: $k_{AO} = (B')^a = (\alpha^o)^a$ Bob computes: $k_{BO} = (A')^b = (\alpha^o)^b$

Oscar computes: $k_{AO} = A^o = (\alpha^a)^o$ Oscar computes: $k_{BO} = B^o = (\alpha^b)^o$

Oscar has now complete control over the channel, e.g., if Alice wants to send an encrypted message x to Bob, Oscar can read the message

  - Certificates and DHKE with Certificates



Authentication

  - User Authentication Factors
    - Knowledge-based
    - Possession-based
    - Static biometrics
    - Dynamic biometrics
  - Password Vulnerabilities
    - Offline dictionary attack
      - The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination.
    - Specific account attack
      - The attacker targets a specific account and submits password guesses until the correct password is discovered.
    - Popular password attack
      - A variation of the preceding attack is to use a popular password and try it against a wide range of user IDs.
    - Password guessing against single user
      - The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password
    - Workstation hijacking
      - The attacker waits until a logged-in workstation is unattended
    - User mistakes/multiple password use/electronic monitoring or sniffing
  - Storing passwords
    - Hashing passwords (salting breaks brute force and forward search (RAINBOW ATTACK precomputed hash value table)
  - Password Selection Strategies
    - User education
      - Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords
    - Computer generated passwords
      - Users have trouble remembering them
    - Reactive password checking
      - System periodically runs its own password cracker to find guessable passwords
    - Complex password policy/proactive password checker
      - A promising approach to improved password security
      - User is allowed to select their own password, however the system checks to see if the password is allowable, and if not, rejects it
      - i.e., don't let the user pick a "bad" password in the first place
      - Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable
  - Remote user authentication protocols
    - Static: user authenticates herself to the token and then the token authenticates the user to the computer
    - Dynamic password generator: the token generates a unique password periodically (e.g., every minute).
    - Challenge response: Computer system generates a challenge, such as a random string of numbers. The smart token generates a response based on the challenge.

- **Man-in-the-middle** is an *active attack to a cryptographic protocol*, where the attacker is, effectively, in between the communications of two users, and is capable of intercepting, relying, and (possibly) altering messages. In this case, the meaning of "in the middle" is direct: the attacker is in the middle of two communicating users.
- **Meet-in-the-middle** is a type of *cryptanalytic attack* that uses some sort of time-space trade-off to drastically reduce the effort to perform a brute-force attack (e.g., transforming an attack that requires 21282128 time into one that takes 264264 time and 264264 space). In this case, the name of the attack comes from the expression "let's meet in the middle", which means "to make a compromise". It may also refer to a type of attack over certain block ciphers, where the attacker decompose the problem in two halves and proceeds on each part separately.