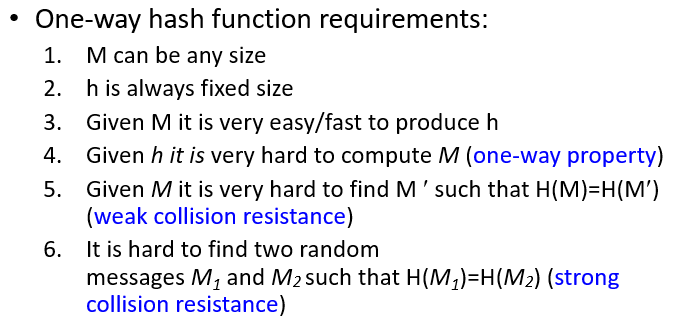
**2 Cryptography**

Cryptography Primitives

* Message Properties
  + Integrity: Message has not been changed in transit
  + Authenticity: Message has been sent by the alleged sender
  + Non-repudiation: Prove the message was sent by the alleged sender
* Cryptographic Functions
  + One-way hash functions: No key
    - For integrity, plaintext → ciphertext
  + Symmetric crypto: One key
    - For confidentiality
    - Both parties know the same *shared key*, plaintext → ciphertext
  + Asymmetric crypto: Two keys
    - For confidentiality and authenticity
    - Public and private keys, plaintext → ciphertext

Attacks on Cryptography

* Ciphertext-Only: Analyze enough ciphertext to exploit regularity and learn key
* Known-Plaintext: Learn decryption key by observing ciphertext with plaintext
* Chosen-Plaintext: Learn key by feeding messages into algorithm to see ciphertext
* Man-In-The-Middle: Substitution, modification, dropping messages, replaying messages
* Brute-Force: Try every possible input to produce valid output
* One-Way Hash: Public algorithm takes variable input M and produces fixed length H(M)
  + Takes 2m/2 trials to find any two messages that hash to the same value
  + Standard algorithms: MD5, SHA1, SHA2 (256, 512), SHA3
* 

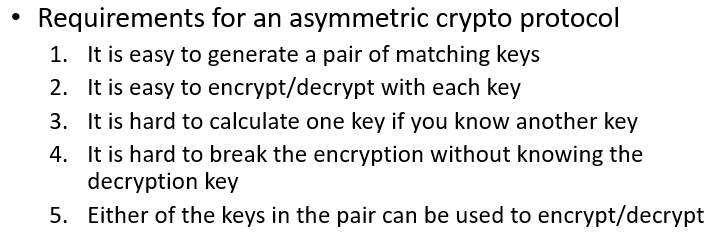
Symmetric Encryption

* Requirements
  + Knows the algorithm and has ciphertexts, but cannot decipher or retrieve key
  + Knows the algorithm and has ciphertexts and plaintexts, but cannot retrieve key
* Substitution: Obscures relationship between plaintext and ciphertext
  + Substitute parts of plaintext with parts of ciphertext e.g. Caesar
  + Monoalphabetic: Each character is replaced with another character
    - Can be broken by frequency analysis or brute force
  + Homophonic: Each character is replaced with a character chosen randomly
    - Has multiple symbols for common letters to minimize frequency analysis
  + Polygram: Each sequence of characters of length n is replaced with another sequence of characters of length n - like monoalphabetic but on n-grams
    - Block ciphers: Work on message block by block
      * Data Encryption Standard, 3-DES, Advanced Encryp. Stand.
  + Polyalphabetic: Many monoalphabetic ciphers are used sequentially
    - XOR is a polyalphabetic cipher in binary domain
    - Stream ciphers: Uses key to encrypt plaintext one bit at a time e.g. Rc4
    - One-Time Pad: Unbreakable cipher with infinite key, never reused
* Transposition: Dissipate redundancy of plaintext by spreading over ciphertext
  + Changing one bit of plaintext affects many bits of ciphertext e.g. route cipher

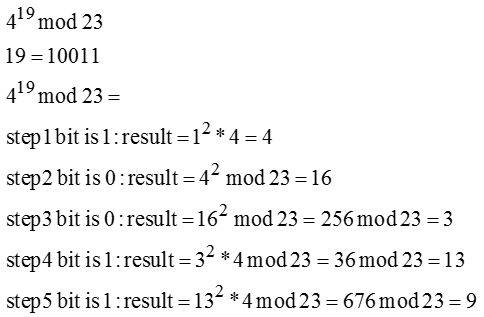
Large Message Encryption

* Electronic Code Book (ECB): Store mapping for every possible block
  + Fast encryption/decryption - just a table lookup
  + Ability to process text in and order and in parallel
  + Depends on key to translate encryption and decryption
  + Malory can detect mapping → can replay and fabricate new messages
* Replay attack: capture a message, copy it, let it go, and later repeat the same message
* Message Fabrication: Learn structure and modify necessary parts to make success
* Cipher Block Chaining (CBC): Dependency on earlier pieces, provides feedback
  + Some plaintext blocks will encrypt to different ciphertext blocks thus obscuring patterns in plaintext → encryption/decryption cannot be parallelized
  + Uses initialization vector (IV), a block of random numbers, to ensure uniqueness
    - Sender and receiver must know IV

Asymmetric Crypto

* Two keys: public and private → scrambles messages between Alice and Bob
  + Either: others can’t make sense (confidentiality) - Bob’s public key
  + Or: can verify sender (authenticity) - Alice’s private key
* Algorithm exists such that: DK2( EK1(M) ) = M
* 
* Functionality of asymmetric crypto is greater than symmetric
  + Asymmetric operations are ~1500 times slower

RSA Asymmetric Algorithm

* Modular exponentiation in Galois Field GF(n)
  + (a+b) mod n = ( (a mod n) + (b mod n) ) mod n
  + 
* Two keys x and y such that:
  + (Mx mod n)y = Mx\*y mod n = M
  + X and y need to be multiplicative inverses
    - x\*y mod n = 1
* 7 and 15 are relatively prime
* Algorithm:

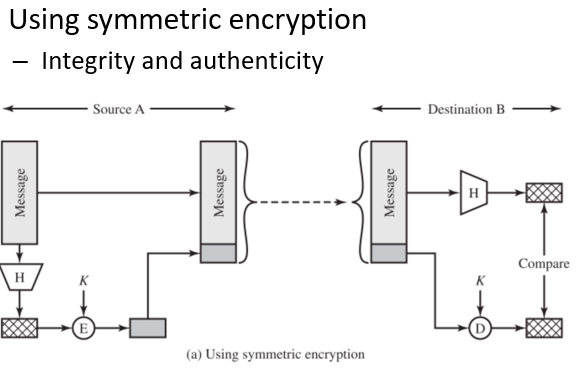
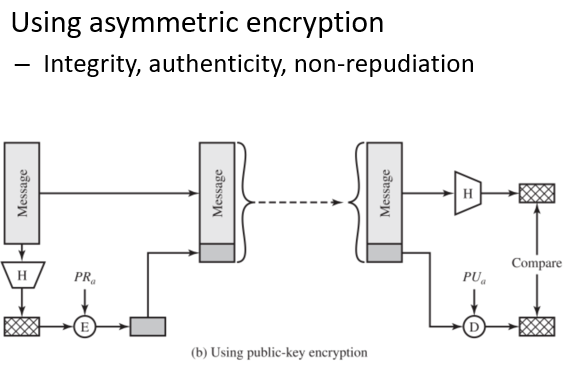
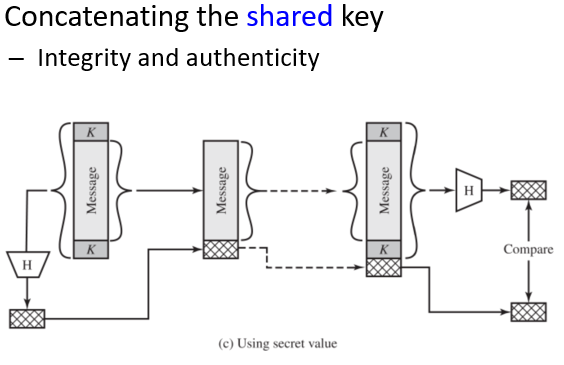
1. Choose two prime numbers *p* and *q*  of equal length
2. Compute n=p\*q, and Euler Totient function ф(n)=(p-1)\*(q-1)
3. Choose private key x relatively prime to ф(n)
4. Using Euclidean algorithm calculate y, which is inverse of x mod ф(n)

* x\*y = 1 mod ф(n)

1. Encryption: Ex(M) = Mx mod n
2. Decryption: Dy(C) = Dy( Ex(M) ) = (Mx mod n)y mod n = M1+k(p-1)(q-1) mod pq = M

* Chinese Remainder Theorem proves M1+k(p-q)(q-1) mod pq = M
* Factoring time of a large number n increases exponentially with each binary digit

Digital Signatures

* Signature: encryption of messages or its one-way hash with the sender’s private key
  + \*\*\* Provides non-repudiation (encrypts with private key, verification by public key)
  + Calculate the hash of original message and compare to decrypted hash
    - Message authenticity (sent by alleged sender)
    - Message integrity (has not been changed in transit)
  + Symmetric systems *CANNOT* provide non-repudiation
* MAC (message authentication code): appended hash to verify integrity/authenticity
*  
* 

Class Example

* M,E(M), H(M), E(H(M)), H(E(M))
* M + H(M) —> No confidentiality, no integrity
* M + EKAB(H(M)) —> No conf., yes int., yes auth., no non-rep.
* M + EPrivA(H(M)) —> No conf., yes int., yes auth., yes non-rep.
* M + EPubB(H(M)) —> No conf., anyone can encrypt with PubB
* M + H(EKAB(M)) —> No conf., yes int., yes auth., no non-rep., costly
* M + H(EPrivA(M)) —> No conf., useless
* EKAB(M) + H(M) —> Yes conf., yes int., yes auth., no non-rep.
* EPubB(M) + H(M) —> Yes conf., no int., no auth., no non-rep.
* EPrivA(M) + H(M) —> No conf., yes int., yes auth., yes non-rep.
* EKAB(M) + EKAB(H(M)) —> Yes conf., yes int., yes auth., no non-rep.
* EKAB(M) + EPrivA(H(M)) —> Yes conf., yes int., yes auth., yes non-rep.
* EPubB(M) + EPrivA(H(M)) —>Yes conf., yes int., yes auth., yes non-rep., costly