Lecture 6 - 09/24/2019

Mac vs Digital Signature

- Both serve authentication
  - With MAC you can send message and mac
  - Digital Signature you can send the signature and the message
- Both serve integrity
  - Any attack or fails in transmission will prevent the mac/signature from verifying

Digital Signature uses asymmetric technology
- Someone can sign and **<u>anyone</u>** can read it
- This is called **Nonrepudiation**
- **Nonrepudiation** is the assurance that someone cannot deny something.

MAC uses symmetric technology
- Someone can sign and only other people with the same key can verify

Cryptographic Algorithms

From the point of view of someone writing a system, it is not important to know the internal working of every single cryptographic algorithm. However they should know the properties of these algorithms.

Important notes on Cryptographic Algorithms

- Do not create your own cypher
- Do not purchase form vendor with proprietary algorithm
- Use standard algorithms. (e.g. by NIST, or CSE)
  - Crypto Agility: If your algo gets cracked, switch to another.

Keys

- It’s up to you to protect the keys as best as you possibly can.
- Keys should be randomly generated.
  - Don't use built in PRNGs (e.g. rand())
    - Not random enough
    - Not unpredictable
  - There are cryptographically secure PRNGs, but they’re slower
    - BBS - Blum-Blum-Shub (x2 mod n)
- Use a key for a single purpose (“Key Hygiene”)
  - Public key realm: One for enc/dec, and one to sign.

- Size (keyspace)
    - Symm: Length of bit string
        - If the length is n, then the keyspace is 2^n
        - 80 bits is considered the BARE minimum
        - 128 is typically used
    - aSymm: Algorithm to solve math problem
        - For RSA the hard problem is factoring
        - Minimum of 1024 bits
        - 3072 is needed to get the same amount of protection as symm-128

Example

"For Alice to 'protect' a file for bob"
What we mean by protection is Confidentiality, authenticity, and integrity.

- We need 2 asymmetric key pairs (One for alice, and one for bob)
- We need a symmetric algorithm for encryption
- We need an asymmetric algorithm for encryption
- We need a signature algorithm: SIG()
- We need a hash algorithm: H()

File: m

Alice	**Bob**
Compute H(m) = h	Compute Dec($Bob_{Private}$, Ck) = k
Compute SIG($Alice_{Private}$, h) = S	Compute Dec_K(Cm) = m
Generate symmetric key k	Compute H(m) = h_1
Compute Enc(k,m) = Cm	Compute verify($Alice_{Public}$,s) = h_2
Compute Enc(Bob_{Public}, k) = Ck	Check $h_1 = h_2$
send(Cm, Ck, s) to bob	

Its essential to ensure that all algorithms used have the same level of security

E.g. AES-256 key encrypted using RSA-1024
- AES-256 has 256 bits of security
- RSA-1024 has 80 bits of security

E.g. RSA-3072 signature hashed by SHA-1
- RSA-3072 has 256 bits of security
- SHA-1 has 80 bits of security

Key management

Alice is going to use symmetric key only. She wants to send that to bob.

We can use a trusted third party. This can be done through a key distribution center (KDC)

- Alice has a key K_{AC} that she shares with the KDC
- Bob has a key K_{BC} that he shares with the KDC