C.S. 465 Midterm # 1 Notes

KNOW HEX!

Symmetric Encryption: Plain text, encrypt, cipher text, decrypt, plain text

* AES (Rijndael Cipher): State Array, 9X (Sub Bytes, Shift Rows, Mixed Columns (ffMultiply), Add Round Key), Sub Bytes, Shift Rows, Add Round Key, Expansion. Refer to Homework #2 and Lab #1.
  + How to implement the finite field add/multiply (Hex notation)

Addition: Turn the coefficients into bits. Do XOR. 57 + 83 = d4

Multiply:

FfMultiply (x,y) {

Byte Base = x;

Result = 0X00;

For all bits in y starting w/ least significant bit

Result = result XOR Base

Base = xtime (Base)

Return result }

Refer to H.W. #2

* Modes of execution – how they work, why we have them, pros and cons.: ECB, CBC, CTR, CFB, OFB. Some use an IV. Refer to Homework #3.
  + I will give you pictures of how the modes work - don't memorize them
* Padding – when and why

Block ciphers require that the plain text be a multiple of the block text. 512

One-way hash function

* 6 security properties:

1. H can be applied to a block of data of any size

2. H produces a fixed-length output

3. H(x) is relatively easy to compute for any given x

4. For any given value h, it is computationally infeasible to find x such that H(x) = h (one way)

5. For any given block x, it is computationally infeasible to find y ≠ x with H(y) = H(x)

(weak collision resistance)

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

(strong collision resistance)

* Pre-image attack vs. collision attack

First pre-image attack o Given a hash h, find a message m such that H (m) = h • Think property #4

Second pre-image attack o Given a message m1, find a different message m2 such that H (m2) = H (m1) • Cost: 2n where n = # of digest bits • Think property #5. Only save 1

Collision attack o Two messages both hash to the same value • Birthday Attack o Given n-bit digest, birthday attack says that we’ll find a match after 2n/2 attempts • Think property #6. Save all.

Refer to Lab #2

* I will give you SHA-1 diagram, understand how it works from the diagram

<https://wiki.cs.byu.edu/media/cs-465/hash.pdf>. SHA-0 broken, SHA-1 weak, SHA-2 ok, SHA-3 strong, doesn’t use Merkle-Damgard construction – uses a sponge construction.

Message authentication code

* Definition - abstract idea of a MAC: You can send an encrypted message along with an attachment that the receiver can look at to verify who sent it. CBC mode AES and the last block is the MAC.

Assure that the message has not been altered • Assure the source of the message is authentic

One must have a secret key K to get the correct decryption. (SENDER AND RECIEVER MUST KNOW)

Refer to Lab #3 and Homework #4

* Terminology and various meanings - MAC, HMAC:

MAC algorithm takes message and K to make a MAC, the appended bit to the message.

HMAC is a MAC generated using hashing.

* How the message extension attack works: The attacker intercepts and adds to a message as long as it fits in a 512 block. To do this, the attacker must know the original message, the message digest, and the length of the message. The attacker simply puts this in in the middle of the algorithm by putting their digest as out IV. Refer to H.W. # 4 and Lab #3.
* Design of HMAC specification to thwart extension attack

H (K || H (K || M)). A hash is there to verify message integrity. The strength of the HMAC depends on the underlying hash function, the size of its hash input, and on the size and quality of the key.

Public key cryptography

* Encryption vs. digital signatures: g^ab % p stuff

When encrypting, you use **their public key** to write message and they use **their private key** to read it.

When signing, you use **your private key** to write message's signature, and they use **your public key** to check if it's really yours.

* Diffie Hellman and RSA: Both Allow 2 users to establish a secret key over insecure network w/out private keys. DH; uses 2 secret parameters p (large prime #, 2048 bits) & g (0<n<p, n = g^k %p). K = g^ab % p. K is then used for Symmetric encryption. Refer to H.W. #5 and Lab #4

RSA; Encryption: c = m^e % n • RSA Decryption: m = c^d % n

Refer to Lab #5

How to conduct a man-in-the-middle attack: Mallory could intercept and change some values that are exchanged between Alice and Bob. Mallory could replace with Eve’s public values so that they don’t have the same K, but she can intercept and interpret all messages the two will send to one another (g^ma and g^mb). Mallory could also simply stop both messages from moving from one person to the next and give 1 to both Alice and Bob so that no matter what the values of a and b are both will end up with 1 as k. Refer to H.W. #5

* Generating RSA parameters using the extended Euclidean algorithm:

Refer to H.W. #7 and extensive class notes.

* Why DH and RSA are secure: A passive attacker can’t get it. For Diffie-Hellman, it’s as safe as it is hard to break the discrete logarithm problem. For RSA, it’s as safe as it is hard to break the discrete d-th root problem.
* What is the public key and private key in RSA?

Public key is KU: {E, N} Private key is KR: {D, N}

You must concatenate them as bits for your final answer. Must be multiple of 32/64. Refer to H.W. #7

* How do we use RSA to encrypt and sign data?

RSA NEVER encrypts data directly, Symmetric encryption for data, RSA to encrypt symmetric key.

Do not sign whole document – too slow. Sign (“encrypt”) a hash of the document using the private key.

Digital certificates

Signatures:

(digest)^d %n = signature

They know if the message is legit if the resulting hashes are the same.

* Five steps a relying party uses to verify a certificate:

1. Integrity
2. Expiration
3. Revocation
4. Usage constraints
   1. Basic Constraints
      1. Can the subject act as a CA?
      2. Is there a limit to the length of the certificate chain?
      3. Limitation on key use
5. Ownership
   1. Does the entity presenting the certificate have access to the associated private key?

* PKI (Public Key Infrastructure):

Infrastructure necessary to deploy and use public key technology.

The infrastructure needed to recognize which public key belongs to whom

* Certificate chains:

Refer to the BYU CA class notes pictures.

* How to repair a certificate hierarchy when a key is compromised:

Refer to the BYU CA class notes example.

TLS

* How the RSA and DHE handshakes work
* I will give you diagrams from the slides:

See Lecture Slide picture.

* How is the key block material generated?
* How many keys or values does each side generate in the key block material?
* RSA

› Client generates pre-master secret

› Sends to server encrypted with server’s public key 

› Client doesn’t have proof its valid will they get the message.

DHE

› DH shared key is the pre-master secret

Pre-master secret and random values used to compute master secret

Master secret and random values used to compute key block material

› Key block contains 4 or 6 keys

› 2 keys for AES, 2 keys for MAC, 2 keys (values) for block cipher mode if needed (IVs)

Has Perfect Forward Security (An attacker couldn’t get any past and future session even if they steal the private key. Not the same for vanilla RSA and the majority of TLS uses vanilla RSA. That’ why most sites use DHE-RSA or ECDHE).

Other helpful notes:

* Some examples of MAC algorithms: CMAC, SHA1-HMAC, MD5-HMAC, UMAC, Poly1305-AES.
* Some examples of cryptographic hash functions: SHA256, SHA1, MD5.

**Integrity:** Can the recipient be confident that the message has not been accidentally modified?

**Authentication:** Can the recipient be confident that the message originates from the sender?

**Non-repudiation:** If the recipient passes the message and the proof to a third party, can the third party be confident that the message originated from the sender? (Please note that I am talking about non-repudiation in the cryptographic sense, not in the legal sense.) Also important is this question:

**Keys:** Does the primitive require a shared secret key, or public-private key pairs? I think the short answer is best explained with a table:

Cryptographic primitive | Hash | MAC | Digital

Security Goal | | | signature

------------------------+------+-----------+-------------

Integrity | Yes | Yes | Yes

Authentication | No | Yes | Yes

Non-repudiation | No | No | Yes

------------------------+------+-----------+-------------

Kind of keys | none | symmetric | asymmetric

| | keys | keys

Please remember that authentication without confidence in the keys used is useless. For digital signatures, a recipient must be confident that the verification key actually belongs to the sender. For MACs, a recipient must be confident that the shared symmetric key has only been shared with the sender.