IK2206 – Internet Security and Privacy
Chapter 6 – Public Key Cryptography

6.1 Introduction (READ)

- Consider what the "motley crew" of public key algorithms have in common.

  The thing that all public key algorithms have in common is the concept of a pair of related quantities, one secret and one public key, associated with each principal.

6.2 Modular Arithmetic (READ)

6.2.1 Modular Addition (READ)

- What is an "additive inverse"?

  An additive inverse of x is the number you'd have to add to x to get 0. For example, 4's inverse will be 6, because in mod 10 arithmetic, 4+6=0. If the secret key were 4, then to encrypt we'd add 4 (mod 10), and to decrypt we'd add 6 (mod 10).

6.2.2 Modular Multiplication (READ)

- What's a "multiplicative inverse"?

  The multiplicative inverse of x (written: x^{-1}) is the number by which you'd multiply x to get 1. For example, 7 is the multiplicative inverse of 3.

- What does Euclid's algorithm do?
  - The important thing is to know what it can be used for!
  - Why isn't modular multiplication a good public key method?
- What does it mean than a value (X) is "relatively prime" to (Y)?

"X relatively prime to Y" means that the numbers X and Y do not share any common factors than 1.

- What is the "totient" function, Phi(n), i.e., how is it defined?

In number theory, the totient φ(n) of a positive integer n is defined to be the number of positive integers less than or equal to n that are coprime to n (i.e. having no common positive factors other than 1). For example, φ(9) = 6 since the six numbers 1, 2, 4, 5, 7 and 8 are coprime to 9.

- What is Phi(n) when "n" is product of two primes ("p" and "q")?

Maybe the number of "coprime numbers of n"???

6.2.3 Modular Exponentiation (READ)

- What's an "exponentiative inverse"?

exponentiative inverse y of x: $(a^x)^y = a$

OR???

*An "exponentiative inverse" of e is the number d such that e*d = 1 mod φ(n). For example, n= 10, φ(10)=4: e=3 and d=7 are "exponentiative inverses" since 3*7 = 21 = 1 mod 4*

6.3 RSA (READ)

- Can you state which steps are needed to create RSA keys?

Preparation: Choose two large primes, p and q (~256bits each) and multiply them together. The result is called n.

Create a Public Key: Choose a number e that is relatively prime to φ(n). (φ(n) = (p-1)(q-1)). The public key is (e, n).

Create a Private Key: Find the number d that is the multiplicative inverse of e mod φ(n). The private key is (d, n).

- How can you use RSA to encrypt a message?

$c = m^e \bmod n$ (e=part of the public key)

- How can you use RSA to sign a message?

$s = m^d \bmod n$ (d=part of the private key)

(Remember: In both cases the message length m must be smaller than n).

6.3.2 Why does RSA work? (READ)

6.3.3 Why is RSA secure? (READ)

- Why is it important to have a "random component" in the message being encrypted?

6.3.4 How efficient are the RSA operations? (READ)

6.3.4.1 Exponentiating with big numbers (READ)

- To do RSA encryption we compute "x^y mod n". It's good to get a rough understanding of how much work it means for a computer to do this.

6.3.4.2 Generating RSA keys (READ)

6.3.4.2.1 Find big primes "p" and "q" (READ BRIEFLY)

- Get the rough idea (pick and test)
- What would happen if your RSA implementation used a predictive algorithm to select the primes? Why is it important to have some source of "randomness" as input to your algorithm?

6.3.4.2.2 Finding "d" and "e" (READ)

- Once you have found your primes (p and q), can you explain how to find your keys {d, n} and {e, n}?

6.3.4.3 Having a small constant "e" (READ BRIEFLY)

- Why is it popular to pick a small "e", e.g., 3 or 65537 (0x8001)?

  Using a small number (like 3) as the public exponent maximizes the encryption performance.

- Why can't you pick a small value for "d"?

  If *d* were small, an attacker could search for small values to find *d*.

6.3.4.4 Optimizing RSA private key operations (SKIP)

6.3.5 Arcane RSA threats (READ BRIEFLY)

6.3.5.1 Smooth numbers (SKIP)

6.3.5.2 The cube root problem (SKIP)

6.3.6 Public-Key Cryptography Standard (PKCS) (READ BRIEFLY)

- Why is useful to have a standard for encoding information to be encrypted/signed through RSA?

  PKCS is a set of recommendations on how to properly use RSA without trapping into pitfalls of the algorithm. Implementers of RSA are no longer expected to know about all the attacks when implementing RSA. Instead, they just follow the PKCS encoding rules.

6.3.6.1 Encryption (READ BRIEFLY)

- What's the purpose of random number field in the packet format?

  RSA is typically used to encrypt symmetric keys. These keys are usually much smaller than the modulus. If it's a DES key, it's only 64 bits (128bit for 3DES). The random number is used as padding to make it harder for an attacker to guess the encrypted data. An attacker would have to guess the padding as well, and this is infeasible.

- What if the field was shorter, e.g., two bytes?

The less (random) information, the easier for an attacker to guess the encrypted information by doing a brute force attack.

6.3.6.2 Encryption - Take 2 (SKIP)

6.3.6.3 Signing (SKIP)

6.4 Diffie-Hellman (READ)

- What can you use the Diffie-Hellman exchange for? Encryption?
  - Integrity protection?

Diffie-Hellman can only be used to exchange keys.

- Can you show the Diffie-Hellman message exchange?
  - What values are exchanged?

    Two numbers, p and g, must be generated by either of the participants (Alice/Bob).
    p = a large prime
    g = a number less than p (+ some minor restrictions)

  - What is public? What is private?

    The numbers p and g are public.
    The numbers S_A (Alice's Secret) and S_B (Bob's Secret) are private.
    Based on S_A and S_B , they compute the numbers T_A ($=g^{SA}$ mod p) and T_B ($=g^{SB}$ mod p) which are exchange over the network and therefore known in public.

6.4.1 The Bucket Brigade/Man-in-the-middle Attack (READ)

- Does Diffie-Hellman protect you against passive attacks such as eavesdropping?

  Yes.

- Does Diffie-Hellman protect you against active attacks, such as "man-in-the-middle" attacks? If not, how can such an attack be launched?

  No. A bad guy (X) generates his own key pair and tries to catch & overwrite transmitted public keys. Communication partners cannot figure out that they're not communicating with the right opponent, since the authenticity is not checked with the Diffie-Hellman algorithm. (See page 168).

6.4.2 Defenses against man-in-the-middle attack (READ)

- Can you suggest a method how to defend yourself against man-in-the-middle attacks when using Diffie-Hellman?

  Each person gets a permanent key pair instead of inventing a new one every time. The public key could be published in a public register (similar to PKI).

6.4.2.1 Published Diffie-Hellman numbers (READ BRIEFLY)

6.4.2.2 Authenticated Diffie-Hellman (READ)

- With Diffie-Hellman you can establish a secure connection to someone, but you do not know to whom. Can you suggest a method to authenticate your Diffie-Hellman exchange, thereby defeating man-in-the-middle attacks?

  Another form of authentication could be done using pre-shared secrets. Each of the communication partners knows a secret. **????**

6.4.3 Encryption with Diffie-Hellman (READ BRIEFLY)

- If Alice likes to send an encrypted message to Bob using a key derived by a Diffie-Hellman exchange, they are first required to conduct the exchange.
  - How well does Diffie-Hellman suit asynchronous communication, e.g., if Alice likes to send a message to Bob at a time when Bob is off-line?
  - Would this be possible with RSA?
  - Is there a way to fix it for Diffie-Hellman? What does the book suggest?

6.4.4 ElGamal Signatures (READ BRIEFLY)

- What can you use "ElGamal" for? Encryption? Digital Signatures? (You only need to know what you can use it for; knowing the ElGamal message exchange is not important in this course)

  Thomas: As far as I know there are two different ElGamal cryptosystems: one is used for encryption and one for signing.

- If RSA can do all ElGamal can plus more, why could still be of interest to use an algorithm such as ElGamal in a security protocol?

  ElGamal uses keys of a much shorter length than RSA. To achieve approx. the same security with both algorithms, ElGamal could be used with a 160 bit key while RSA should have a 512 bit key. That makes ElGamal much faster than RSA (with comparable security).

  See http://stackoverflow.com/questions/6035200/el-gamal-faster-than-rsa-with-the-same-modulus-length

6.4.5 Diffie-Hellman Details - Safe Primes (SKIP)

6.5 Digital Signature Standard (DSS) (SKIP)

- Outside the scope of the course

6.6 How Secure are RSA and Diffie-Hellman? (READ BRIEFLY)

- This section contains some terminology we don't require you to know the details of, however, it's good to know the relative strength of different protocols.
  - Are RSA (factoring a large number) easier, more difficult or of equal difficulty to break than Diffie-Hellman (discrete log problem)?

    Both mathematical problems have been proven to be equivalently difficult.

- Does the difficulty to break RSA increase exponentially (something raised to "x") or "as a polynom" ("x" raised to some fix value) with/of the RSA key size ("x" is here the key length)? Or something in between?

  The best known algorithm for solving these mathematical problems are subexponential (less than exponential) but superpolynomial (more than any fixed degree polynomial).

6.7 Elliptic Curve Cryptography (ECC) (SKIP)

- Outside the scope of the course

6.8 Zero Knowledge Proof Systems (SKIP)

- Outside the scope of the course

More Information
http://www.cs.odu.edu/~cs772/fall06/lectures/public_key_cryptography.html