Network Security and Resilience

Cryptography – Asymmetric Key

Lecture twenty-one

Outline of Lecture

- Diffie-Hellman Hybrid Key Exchange
- Public key cryptography
  - Introduction
  - RSA algorithm
- Tests for primality

Learning objectives

- At the end of this lecture, students should be able to:
  - Understand the difference between symmetric and asymmetric encryption
  - Use Diffie-Hellman hybrid key exchange to construct a shared secret key across an insecure channel
  - Encode and decode a short plaintext message using RSA
  - Be able to test whether a number is prime
  - Have an appreciation of Elliptic Curve Cryptography

Modulo Arithmetic

Asymmetric key cryptography can generate very large numbers that require their modulus to be calculated. Fortunately, modulo arithmetic is associative and commutative. That is:

$a^{p+q+r} \bmod N = (a^p \bmod N)\ (a^q \bmod N)\ (a^r \bmod N) \bmod N$

For example

$3^6 \bmod 5 = (3^2 \bmod 5)\ (3^2 \bmod 5)\ (3^2 \bmod 5) \bmod 5$

$= (9 \bmod 5)(9 \bmod 5)(9 \bmod 5) \bmod 5$

$= 4^3 \bmod 5 = 64 \bmod 5$

$= 4$

Try this approach with $4^4 \bmod 9$

Diffie-Hellman Hybrid Key Exchange

- Invented by Diffie and Hellman before any viable public key algorithms were known
- Enables secure agreement of a symmetric key across an insecure network
- Symmetric key is used only for this session. Deleted when session is completed.

Diffie-Hellman Hybrid Key Exchange

- Enables two parties (Alice and Bob) to agree on a secret key without exchange of the key in plain-text
- Makes use of the difficulty in reversing modulo-arithmetic
- Both parties use a prime number p and a base g
  - It is assumed that any eavesdropper will know these values
- Alice chooses a secret number a
  - She then calculates $A = g^a \bmod p$ and transmits it to Bob
- In the meantime Bob chooses a secret number b and calculates $B = g^b \bmod p$
- Alice then computes $s = (B)^a \bmod p$
- Bob then computes $s = (A)^b \bmod p$
- Both computations $(B)^a \bmod p$ and $(A)^b \bmod p$ result in the same value s

Diffie-Hellman Hybrid Key Exchange Example

- Alice and Bob agree to use a prime number p=23 and base g=5.
- Alice chooses a secret integer a=6, then sends Bob A=(g^a mod p)
  - A = 15625 mod 23 = 8.
- Bob chooses a secret integer b=15, then sends Alice B=(g^b mod p)
  - B = 5^{15} mod 23 = 19.
- Alice computes s = $(B)^a$ mod p
  - 19^6 mod 23 = 2.
- Bob computes s = $(A)^b$ mod p
  - 8^{15} mod 23 = 2.

Diffie-Hellman Hybrid Key Exchange Example

- Diffie-Hellman makes use of the useful fact that

  $$A^b \; mod \; p = (g^a)^b \; mod \; p = g^{ab} \text{ mod p}$$

  and $B^a \; mod \; p = (g^b)^a \; mod \; p = g^{ba} \; mod \; p = g^{ab} \; mod \; p$
- Its security comes from the difficulty in reversing modulo arithmetic
  - A man-in-the-middle (traditionally Eve) will know *g* and *p* but calculating *a* from g^a mod *p* is computationally intense, provided *p* and *a* are large and *p* is prime. Typically, p will be around 1000 bits long while *a* and *b* will be around 250 bits
  - Interestingly *g* can be quite small and is usually *2* or *5*
  - Also worth noting that neither party to the exchange (Alice or Bob) can determine the other party's secret (a and b).
- A way for two parties to agree on a shared secret by using but not disclosing, their own secret key

Public key cryptography

- Asymmetric cryptography
- Sender and receiver have a public and private key for encryption and decryption
  - a key pair
  - either key can be used for encryption or decryption
- Key pairs are mathematically dependent
  - message encrypted by one key can only be decrypted by the other key of the key pair
- Anyone can encrypt with the public key but only the holder of the private key can decrypt it
- Main use is in digital signatures and in exchange of symmetric keys

Secret vs. Public Key Encryption

- Secret Key Encryption
  - Same key used to both encrypt and decrypt data
  - Key must be known only by communicating parties
- Public Key Encryption
  - Pair of keys: Ka and Kb
    - Encrypt with K_a, must decrypt with K_b
    - Encrypt with K_b, must decrypt with K_a
  - One key is made public whilst the other is private
  - Most commonly known cipher is RSA

Public Key – Signing & Authentication

- Allows us to consider Signing and Non repudiation
  - A very important application of public key cryptography
  - An electronic document can be signed
    - A hash of the document is calculated
    - The hash is encrypted with a private key
    - A third party can later recompute the hash, decrypt the encoded hash using a public key, and compare
    - Document is protected by hash within signature
  - Better than written – private key cannot be forged
  - If the document is private, it can also be encrypted
  - Can use known public keys to authenticate signatures

Public Key – Non-repudiation

- Non-repudiation
  - Ensures that a party to a digital transaction cannot repudiate the transaction (claim that it did not take place)
  - Protects other parties to contracts
  - Protects commercial interests

Key Lengths – Public Key Ciphers

- RSA based on prime factors of large numbers
- Brute force approach on keys > 512 bits not feasible
- Cryptanalytic approach is:
  - Factor n (using a brute force approach)
  - Select likely candidates for p and q
  - Use e to find potential values for d
- Required Key Lengths
  - 1024 bits recommended, 2048 for 20 years protection
  - Moore's Law requires key lengths upgraded periodically

Public key cryptography

- Makes use of one way functions
- Functions that are easy to calculate in one direction but impossible (or difficult) to find the inverse
  - 52,396 x 842,412 = 44,139,019,152 – Easy ☺
  - Finding factors of 44,139,019, 152 – Hard ☹
- Do not provide encryption – used as parts of ciphers
- Example – Breaking a plate
  - Easy to break. Hard to fix.

Public key cryptography

$E_{k1}(M) = C$

$D_{k2}(C) = M$

K1 = encryption key

K2 = decryption key

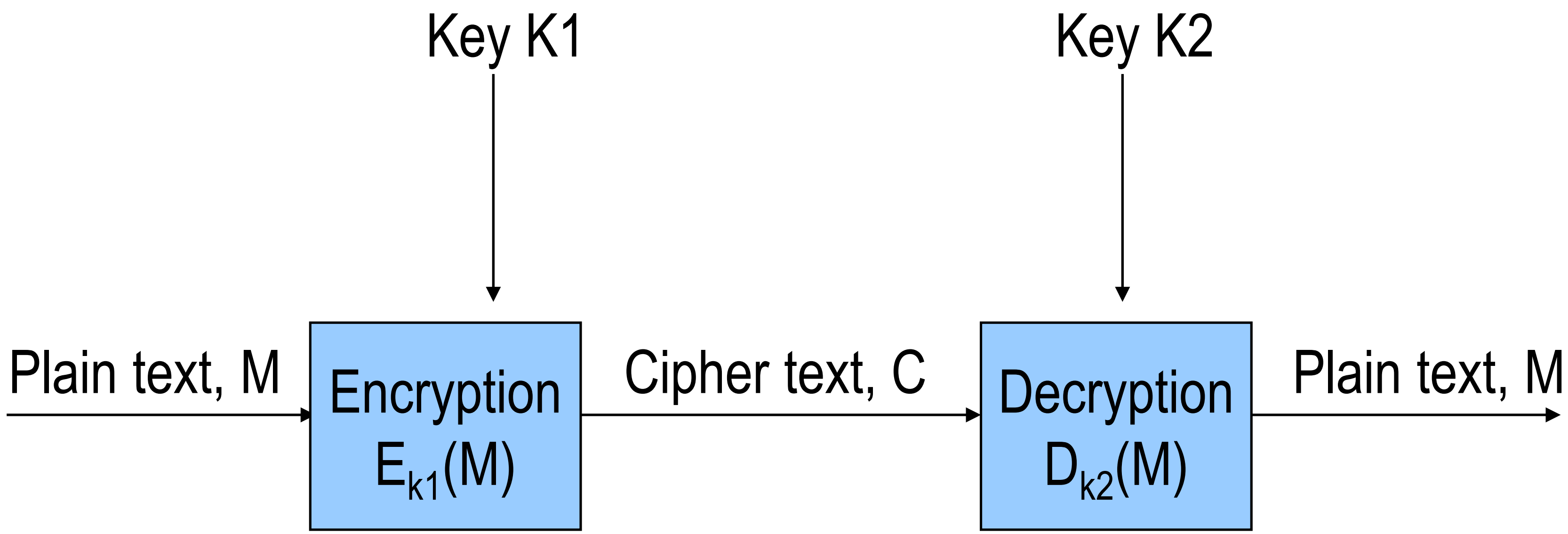
M plaintext

C cipher text

$D_{k2}(E_{k1}(M)) = M$

Converse is usually true

Public key cryptography



Public key cryptography

- Rivest – Shamir – Adleman (RSA)
- Based on factoring of 100 to 200 digit prime numbers
  - Easy to multiply and calculate products of large numbers
  - very difficult to factor a large number you know to be the product of two prime numbers
- Advantage of public key cryptography compared with symmetric key cryptography is that no confidential information need be exchanged before communication takes place
- Can provide a secure but open communication channel
- Public key can be very public
  - attached to emails
  - located in register
  - on web pages

Public and private keys

- Asymmetric keys need to be much larger than symmetric keys for the same level of security
- 128 bit symmetric keys should be used in associated with 1024 bit asymmetric keys
- Public key cryptography much slower than private key cryptography

Public key algorithms

- Public key algorithms
  - Goal is to enable some specific message to be exchanged
  - Some similarities to Diffie-Hellman
    - Makes use of computations that are easy to carry out but difficult to reverse
    - Most common is RSA
- Some preliminaries
  - Prime numbers are numbers which have no factors other than themselves and 1
  - Relatively prime numbers are two numbers which have no common factors other than 1
  - modulo arithmetic (sometimes clock arithmetic)
    - x mod y is the remainder when x is divided by y

Questions

- Which of the following are prime numbers?
  - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
- Which of the following pairs are relatively prime?
  - (4, 5), (4, 6), (8, 9), (9, 10), (11, any number less than 11), (3, 12),
- Calculate the following
  - 5 mod 6, 12 mod 6, 3 mod 2, 15 mod 5

Rivest – Shamir – Adleman (RSA)

To create the public key select two large positive prime numbers p and q	p = 7, q=17
Compute n = p*q	n = 119
Compute x = (p-1)*(q-1)	x = 96
Choose an integer e which is relatively prime to x	e = 5
Public key is then [n, e]	[119, 5]
To create the private key compute d with (d*e) mod x = 1	d = 77
Private key is then [n, d]	[119, 77]
Data to encrypt is m	m=19
To encrypt m, compute c =(m^e)mod n	c = 66
To decrypt c, compute m=(c^d)mod n	m = 19

Public Key Encryption (RSA)

- Given p*q = n then the public key is [n, e] and the private key is [n, d]
- Essentially, to calculate [n,d] we calculate the inverse of e in the finite field of integers mod n.
  - Without knowing p and q it is very difficult to calculate x and hence to calculate d.
- Consequently p and q should be discarded once the keys are generated
- For large n it is very difficult to determine p and q
- The strength of this algorithm is the difficulty in factoring large numbers (100 to 200 digits) which are the product of two primes
  - Primes guarantee existence of single solutions

Questions

- Use the public key [119, 5] to encrypt the value 5
- Use the private key [119, 77] to decrypt the value calculated above
- 221 is the product of two primes. What are they?

Some obvious issues…

- Are there enough primes?
  - public key algorithms need lots of prime numbers. Is there a risk of running out of them?
- What if two people accidentally choose the same prime number for their algorithm?
- Couldn’t someone create a database (similar to a rainbow table) of prime numbers and use that to break public key algorithms?
- If factoring prime numbers is hard how can generating prime numbers be easy?

Some obvious issues…

- Are there enough primes?
  - Yes.
  - Assume we are dealing with 512 bit prime numbers. From number theory it is known that there are approximately 10^{151} prime numbers of this length or less. For comparison there are less than 10^{10} people in the world and only 10^{77} atoms in the universe. There are plenty of primes.
- What if two people accidentally choose the same primes
  - It won't happen.
  - There are 10^{151} primes of 512 bits in length. There are 10^{10} people. Provided they are chosen randomly, the probability of two people choosing the same primes are approximately 1 in 10^{141}

Some obvious issues…

- A database of primes?
  - Again too many to record on a database
    - Many more primes of this length than there are atoms in the universe
- If factoring large numbers is hard how can generating large prime numbers be easy?
  - Solution is to ask the question:
    - "Is *n* prime?" rather than "Are the only factors of *n* itself and 1?"
  - There are fast algorithms to determine whether or not a number is prime without having to calculate the factors of the number

Tests for primality

- Tests for primality are probabilistic
- Lehmann
  - Tests if p is prime
  - Choose a random integer a less than p
  - Calculate $a^{(p-1)/2} \mod p$
  - If not equal to 1 or p-1 then p is definitely not prime
  - If equal to 1 or p-1 then p is prime with probability greater than 0.5
- Rabin-Miller algorithm
  - Similar to Lehmann
  - Easy to implement in hardware

Tests for primality

- Use Rabin-Miller or Lehmann repeatedly but with different values of *a.*
- Each time the test is carried out and the result is 1 or p-1 we have an independent test that p is prime to a probability of at least 0.5
- If test is repeated t times with different values of a and the calculation equals 1 or p-1 each time (but not always 1) then the probability of p not being prime is 0.5^t
- Consequently the probability of p being prime is $1 - 0.5^t$

Tests for primality

- Example: Test for 17 being prime
  - Use *a* equal to 3
    - $a^{(17-1)/2} = 3^8 \bmod 17 = 16$ (=p-1),
    - so probability of 17 not being prime is less than 0.5
  - Use *a* equal to 4
    - $a^{(17-1)/2} = 4^8 \bmod 17 = 1$
    - So probability 17 not being prime is now less than 0.25 and the probability of it being prime is at least 0.75
  - Use a equal to 5
    - $a^{(17-1)/2} = 5^8 \bmod 17 = 16$ (=p-1)
    - So probability 17 not being prime is now less than 0.125 and the probability of it being prime is at least 0.875

Question

- Use Lehmann's test to show that the probability of 7 being prime is at least 0.75
- Solution:

  $2^{(7-1)/2} \bmod 7 = 2^3 \bmod 7 = 8 \bmod 7 = 1$

  So probability 7 is prime is at least 0.5

  $3^{(7-1)/2} \bmod 7 = 3^3 \bmod 7 = 27 \bmod 7 = 6$

  So probability 7 is prime is at least 0.75

RSA Public Key Encryption

- Recommendations for implementations include:
  - A common modulus ***n*** should not be shared in a community of users
  - Plaintext should be padded with random values
  - ***d*** should be large
- Very secure if these recommendations are followed and appropriate key lengths are used
- Speed is very slow due to difficulty of performing exponential operations on large numbers
  - Similar difficulties faced with other Public Key Ciphers
  - Generally much slower than symmetric key encryption

Elliptic curve cryptography

- Much the same functionality as RSA but much more computationally efficient
  - Important for battery powered devices
- Based on discrete logarithms on an elliptic curve (not an ellipse)
  - Elliptic curve is of the form $y^2 = x^3 + ax + b$
- Able to provide same level of security with shorter keys than RSA
  - Popular on battery and other power constrained devices
- Also Diffie-Hellman over elliptic curves
- (Nice explanation at http://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography

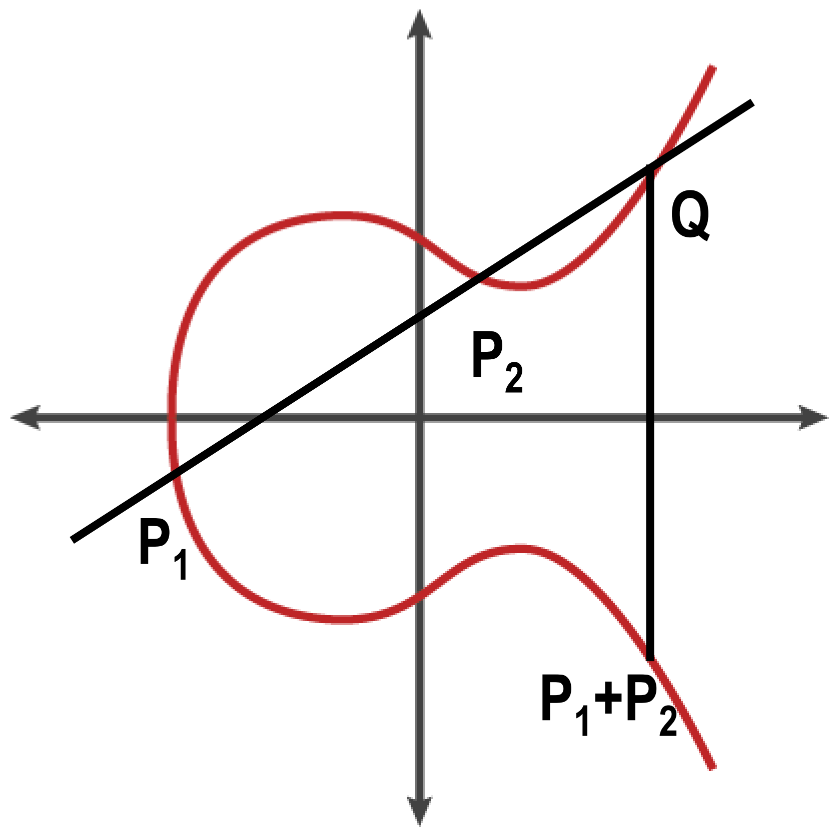
Elliptic curves cryptography

- Elliptic curves are not ellipses
- Elliptic curve cryptography defines operations for addition of points on the elliptic curve, modulo some large prime p.
- If we have two points on the curve P1 and P2 their sum is defined as the reflection of the intersection with the curve of a straight line between them

  Q = intersection (P1,P2)

  P1 + P2 = reflection (Q)

Elliptic curve cryptography



Elliptic curve cryptography

- If we define addition in this way, we can define multiplication by a scalar as repeated addition

  kA = (A+A+A+A…)

- In Elliptic curve arithmetic there is no concept of proximity
  - The sum of two numbers (points on the curve) even if very close in Euclidean distance may be widely separated
- Elliptic curve cryptography relies on there being no approach other than brute force to solving the equation

  B = kA = (A+A+A+A…)

  - The “discrete logarithm” problem
  - Construct keys in similar way to RSA but using multiplication as defined for the elliptic curve

Public Key Cryptography Standards (PKCS)

- PKCS was developed by RSA to describe data-structures and syntax for programming public key cryptography systems.
- Not formal (ANSI) standards

PKCS#1	RSA encryption and decryption
PKCS#3	Diffie-Hellman key exchange
PKCS#5	Encrypt messages with a secret key derived from a password
PKCS#6	Public key certificates (superset of X.509)
PKCS#7	General syntax for encrypting or signing data (recursive)
PKCS#8	Private key syntax
PKCS#9	Extensions to PKCS#7, 8, 9
PKCS#10	Certification requests
PKCS#11	Cryptographic tokens
PKCS#12	Storing in software, a user's public keys

Conclusion

- Diffie-Hellman hybrid key exchange
- Public key cryptography
  - Introduction
  - RSA algorithm
  - Elliptic curve
- Tests for primality