L94 - Public-Key Cryptography

Public-Key Cryptography

Number Theory, RSA, Diffie-Hellmann

Agenda

- Introduction
- Number Theory
- RSA
- Diffie-Hellmann

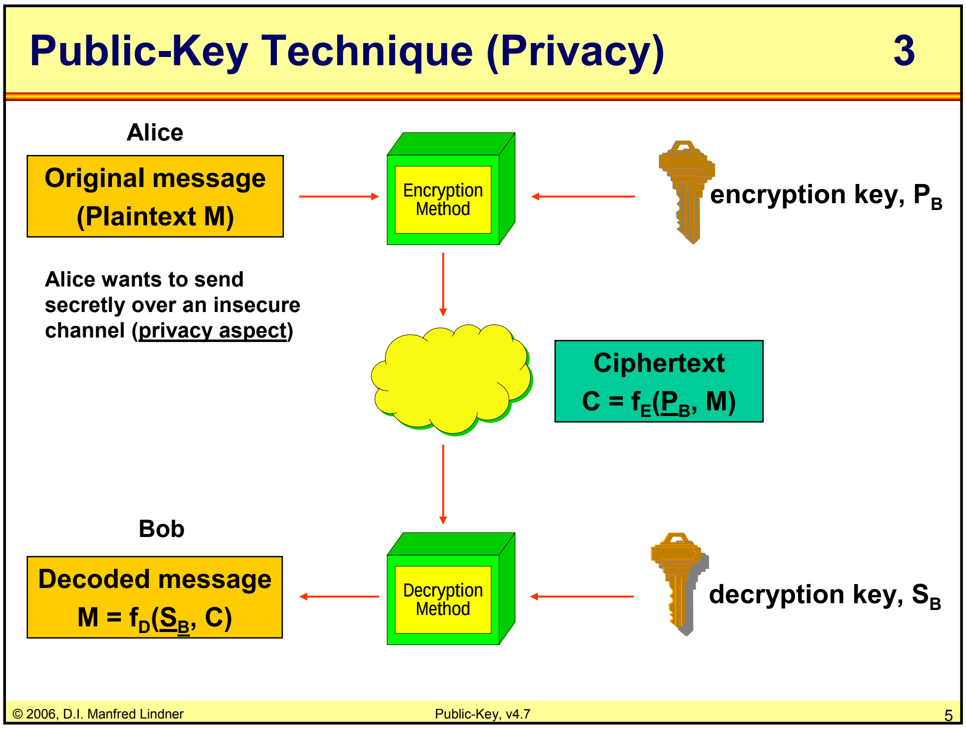
Public-Key Technique 1

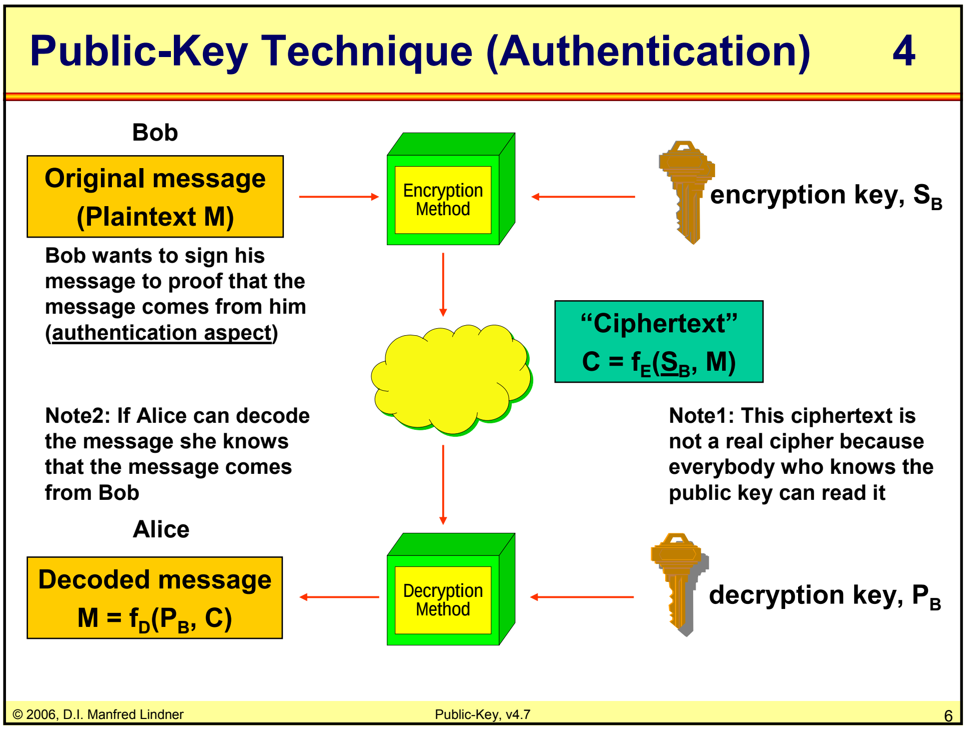
- **A pair of keys is used**
  - Private Key used by one party
    - key kept secret in one system
    - to sign messages to be sent to the other party for authentication
    - to decrypt messages received from the other party
  - Public Key used by the other party
    - key may widely be published to many systems
    - to encrypt messages to be sent to the other party for privacy
    - to verify messages received from the other party for authentication
- **Called asymmetric algorithms**

Public-Key Technique 2

- **Methodology**
  - S … Private Key, P … Public Key
  - Security association between Alice and Bob
    - Alice generates one key pair (P_A, S_A), keeps S_A secret in her system and give P_A to Bob
  - Security association between Bob and Alice
    - Bob generates one key pair (P_B, S_B), keeps S_B secret in his system and give P_B to Alice
  - Encrypted messages from Alice to Bob
    - $C = f_E(\underline{P_B}, M)$
    - $M = f_D(\underline{S_B}, C)$ done by Bob to decrypt
  - Encrypted messages from Bob to Alice
    - $C = f_E(\underline{P_A}, M)$
    - $M = f_D(\underline{S_A}, C)$ done by Alice to decrypt

L94 - Public-Key Cryptography





Public-Key Algorithms

- **RSA**
  - for encryption and digital signature
- **El-Gamal and DSS**
  - for digital signature
- **Diffie-Helman**
  - allows establishment of a shared secret
- **They are all very different from each other**
  - All hash algorithms do the same thing: they take a message and perform an irreversible transformation on it
  - All the secret-key algorithms do the same thing: they take a block and encrypt in a reversible way and allow message ciphers by chaining mechanism of blocks

Agenda

- **Introduction**
- **<u>Number Theory</u>**
- **RSA**
- **Diffie-Hellmann**

„Natürliche Zahlen“ 1

- **a teilt n -> b -> b * a = n**
- **b ist der komplementäre Teiler**
- **p = Primzahl („prime“)**
  - als Teiler nur 1 und p
  - Primzahlen sind ungerade Zahlen (Ausnahme p=2)
- **Untersuchung, ob eine Zahl Primzahl ist**
  - Überprüfen aller Zahlen bis Wurzel aus p
- **Primfaktorzerlegung („prime factorisation“)**
  - 240 = 24*10 = 3*8*2*5 = 2*2*2*2*3*5= 2^4*3* 5
  - 3750 = 25*15*10 = 5*5*3*5*2*5 = 2*3*5^4
  - Primfaktoren bleiben übrig

„Natürliche Zahlen“ 2

- **ggT (größter gemeinsamer Teiler)**
  - gcd („greatest common divisor)
  - bilde Produkt der Primfaktoren, die in beiden Zerlegungen gleichzeitig vorkommen
  - 240 = 24*10 = 3*8*2*5 = 2*2*2*2*3*5= $\underline{2}^4$*$\underline{3}$* $\underline{5}$
  - 3750 = 25*15*10 = 5*5*3*5*2*5 = $\underline{2}$*$\underline{3}$*$\underline{5}^4$
  - ggT (240, 3750) = 2*3*5 = 30
- **ggT (a, n) = 1 -> teilerfremd („relatively prime“)**
  - 54 = 2*3*3*3= 2*3^3
  - 65 = 5*13
  - ggT (54, 65) = 1

„Natürliche Zahlen“ 3

- **kgV (kleinstes gemeinsames Vielfaches)**
  - bilde Produkt der höchsten Potenzen aller überhaupt vorkommender Primfaktoren
  - 75 = 3*5*5= 3*$\underline{5^2}$
  - 189 = 3*3*3*7= $\underline{7}$*$\underline{3^3}$
  - kgV (75, 189) = $7*5^2*3^3$ = 4725
- **ggT (a, n) * kgV (a, n) = a * n**
  - 75 = 3*5*5= $\underline{3}$*5^2
  - 189 = 3*3*3*7= 7*$\underline{3}^3$
  - ggT (75, 189) = 3
  - kgV (75, 189) = $7*5^2*3^3$ = 4725
  - 3 * 4725 = 14175 = 75 * 189

ggT durch Euklid statt Faktorisierung

- **ggt lässt sich mit dem <u>Euklid-Algorithmus</u> leicht berechnen:**
  - Zwei Zahlen 792 (n) und 75 (a)
  - 792 = 10*75 + 42
    - (n = q*a + r) -> ggT(n,a) = ggT (a, r) mit 0 <= r < a
  - 75 = 1*42 + 33
  - 42 = 1*33 + 9
  - 33 = 3*9 + 6
  - 9 = 1*6 +3
  - 6 = 2*<u>**3**</u>
  - 3 ist ggT von 792 und 75

Vielfachsummendarstellung des ggT

- **Vielfachsummendarstellung:**
  - ist g = ggT (n, a)
  - dann gibt es ganze Zahlen s und t so dass
  - g = t*a + s*n
  - "Lemma von Bezout"
  - die Werte t und s lassen sich mit dem **erweiterten Euklid-Algorithmus** leicht berechen

Restklassenarithmetik – Modulo (mod)

- **Restklassenarithmetik verwendet natürliche Zahlen kleiner n**
  - 0, 1, 2, … n-1
- **Auf diese Zahlen lassen sich alle Operationen wie Addition, Multiplikation, Subtraktion anwenden**
- **Das Ergebnis einer solchen Operation wird durch n geteilt und der dabei entstandene Rest als modulo von n (mod n) bezeichnet**

a mod n = r

ist gleichbedeutend mit der Aussage

es gibt eine ganze Zahl k gibt für die gilt: a = k * n + r

ggT und modulo

- **Wenn ggT (n, a) = 1 gilt laut Vielfachsummendarstellung**
  - 1 = t*a + s*n
- **Betrachtet man diese Gleichung bezüglich mod n**
  - 1 = t*a

    oder auch
  - (t*a) mod n = 1
    - t*a geteilt durch n ergibt Rest 1
    - Man schreibt auch anstelle von t ≡ a^{-1} (mod n)
- **Dann ist a ist bezüglich modulo n <u>invertierbar</u>**
  - Es existiert daher ein komplementärer Teiler t der obige Gleichung erfüllt
  - Anmerkung:
    - in der normalen Arithmetik wäre t = 1/a = a^{-1} (der Kehrwert von a) und daher ein Bruch (rationale Zahl)
    - In der Restklassenarithmetik gibt es aber keine Brüche sondern nur natürliche Zahlen
    - dennoch lässt sich hier (wenn a und n teilerfremd sind) mittels "**<u>erweiterten Euklid</u>**" eine natürliche Zahl t finden, die obige Gleichung erfüllt

Introduction

- **Most of the public-key algorithms**
  - are based on modular arithmetic
- **Modular arithmetic**
  - uses non-negative integer numbers less than some positive integer ***n***
  - performs ordinary arithmetic operations like addition or multiplication on such numbers but replaces the ordinary arithmetic result ***a*** with its remainder ***r*** when divided by ***n***
  - the result ***r*** is expressed by "***a*** mod ***n***"
    - ***r* = *a*** mod ***n***
  - that means we can find an integer "***k***" such that
    - ***a* = *k* * *n* + *r***

Modular Addition (mod 10)

K

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

M

C

Encryption / Decryption (add mod 10)

- **Addition of constant K mod 10**
  - can be used for encryption of digits
  - each decimal digit maps to a different decimal digit
  - reversible way
  - constant K is the secret key
  - cipher like “Caesar Cipher”
- **Mono alphabetic substitution**
  - of course not a good cipher
- **Decryption**
  - subtracting the secret constant
  - if less than 0 then add 10

Encryption / Decryption (add mod 10)

- **Decryption (cont.)**
  - instead of subtracting you can add the "additive inverse"
  - additive inverse of K is the number you have to add in order to get 0
    - e.g. 7 is the inverse of 3 because (3 + 7) mod 10 = 0

- **Encryption with K = 3**
  - (M + 3) mod 10 = C
- **Decryption with add-inv(K) = 7**
  - (C + 7) mod 10 = M

Ciphertext
$C = f_E(\underline{K}, M)$

Plaintext
$M = f_D(\underline{K}, C)$

Modular Multiplication (mod 10)

K

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

M C

Encryption / Decryption (mult mod 10)

- **Encryption with K = 3**
  - (M * 3) mod 10 = C

Ciphertext
$C = f_E(\underline{K}, M)$

- **Decryption**
  - multiplying C with the "multiplicative inverse"
  - multiplicative inverse of K is the number you have to multiply to get 1
    - (K * mult-inv(K)) mod n = 1
  - in ordinary arithmetic
    - mult-inv(K) = 1/K because K * 1/K = 1
    - 1/K is a fraction
  - but in modulo arithmetic we have only integers
    - so we cannot find mult-inv(K) for all possible values of K

Decryption (mult mod 10)

- **Decryption (cont.)**
  - for our example we can find multiplicative inverse mod 10
    - just by trying if (K * mult-inv(K)) mod 10 = 1
  - so we find:
    - (1 * mult-inv(1)) mod 10 = 1 -> mult-inv(1) = 1
    - (3 * mult-inv(3)) mod 10 = 1 -> mult-inv(3) = 7
    - (7 * mult-inv(7)) mod 10 = 1 -> mult-inv(7) = 3
    - (9 * mult-inv(9)) mod 10 = 1 -> mult-inv(9) = 9
- **Therefore for K = 3 we can decrypt by multiplying with mult-inv(K) = 7**
  - (C * 7) mod 10 = M

Plaintext
$M = f_D(\underline{K}, C)$

- **Again that is not a good cipher**

How to Find the Multiplicative Inverse

- **Finding multiplicative inverse t in mod n arithmetic**
  - is not obvious especially for a large number n
    - for a given a and n -> find number t such that (a*t) mod n = 1
  - brute-force search with n = 100 digit number will not work
- **But if the two numbers a and n are relatively prime**
  - then because of the general rule of "Bezout"
    - 1 = t*a + s*n
  - and finally if the rule of "Bezout" is performed modulo n
    - (t*a) mod n = 1
  - 1.) there must exists such inverse number t
  - 2.) the number t can be found by the Extended Euclid's algorithm
    - note to Euclid:
      - Basic Euclid's algorithm finds gcd (greatest common divisor) of two numbers (a, n)
    - alternatively t can be found by using the Euler formula $a^{\varphi(n)}$ mod n = 1
      - (t*a) mod n = $a^{\varphi(n)}$ mod n -> t = $a^{\varphi(n)-1}$ mod n
      - useful only if value for φ(n) is known

Relatively Prime

- **Why are the numbers 1, 3, 7 and 9**
  - the only ones with multiplicative inverse mod 10?
- **They are the only ones which are relatively prime to 10**
  - each of these numbers does not share any common factors with 10 other than 1 (gcd is 1)
    - the largest integer that divides 9 and 10 is 1
    - the largest integer that divides 7 and 10 is 1
    - the largest integer that divides 3 and 10 is 1
    - but 6 and 10 have two factors in common -> 1 and 2

Euler Function

- **In turns out that in general**
  - when working with mod n all numbers which are relatively prime to n have a multiplicative inverse but none of the other numbers
- **How many numbers less than n are relatively prime to n?**
  - notation for that amount is **φ(n)**
  - **φ(n) …. Euler function**
  - there is no simple formula but in special cases of n we can determine φ(n)
    - -> see next slides

φ(n) for Prime Numbers

- **If n = prime then φ(n) = n-1**
  - all the integers 1, 2, … n-1 are relatively prime to n
    - greatest common divisor with n is 1 for all of them
  - note:
    - a number n is prime means that the only divisors are 1 and n itself
    - it cannot written as a product of other numbers
- **If n = product of two distinct prime numbers p and q then**

$$\varphi(n) = (p-1)*(q-1)$$

Explanation φ(n)

- **Why?**
  - p and q are prime numbers and n = p*q
    - n could be divided only by p or q
    - n could not divided by other numbers because p and q are prime (only 1, p or q are possible as factors)
  - there are n = (p*q-1) numbers -> (1,2, … n-1)
    - we want to exclude all numbers which are not relatively prime
    - those are the numbers that are either multiples of p or q
      - we have the numbers 1q, 2q, … (p-1)q which are in sum p-1 multiples of q which are less than p*q
      - we have the numbers 1p, 2p, … (q-1)p which are in sum q-1 multiples of p which are less than p*q
    - the rest must be relatively prime
  - φ(n) = (p*q -1) - (p-1) - (q-1) = p*q - 1 - p + 1 - q + 1 = p*q - p - q + 1 = (p - 1) * (q - 1)

Example

- **p = 5, q = 7**
- **p*q = 35 = n**
- **1*7, 2*7, 3*7, 4*7**
  - are not relatively prime to 35 because greatest common divisor is 7 instead of 1
  - that is p-1 times -> 4 times
- **1*5, 2*5, 3*5, 4*5, 5*5, 6*5**
  - are not relatively prime to 35 because greatest common divisor is 5 instead of 1
  - that is q-1 times -> 6 times
- **φ(35) = (p-1)*(q-1) = 24**
  - all other numbers are relatively prime to 35

Example

$p = 5, q = 7$

$p*q = 35 = n$

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35

Euler

- **Euler's theorem**

  if two numbers a and n are relatively prime then

  $a^{\varphi(n)} \bmod n = 1$ (Euler 1.)

  and for any integer k

  $a^{1+k*\varphi(n)} \bmod n = a$ (Euler 2.)

  that is because

  $(a^1 * a^{k*(\varphi(n))}) \bmod n = (a^1 * a^{(\varphi(n))k}) \bmod n = a * 1^k = a$

- **It turns out that**

  if n = product of two distinct prime numbers p and q then the formula's of Euler are valid even if the number a is not relatively prime to n (so for all x with x <= n)

  $\varphi(n) = (p-1)*(q-1)$

  $x^{(p-1)*(q-1)} \bmod n = 1$ (Euler 3.)

  $x^{1+k*(p-1)*(q-1)} \bmod n = x$ (Euler 4.)

Modular Exponentiation (mod 10)

K

X^Y	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
1	1	1	1	**1**	1	1	1	1	1	1	1	1	1
2	1	2	4	**8**	6	2	4	8	6	2	4	8	6
3	1	3	9	**7**	1	3	9	7	1	3	9	7	1
4	1	4	6	**4**	6	4	6	4	6	4	6	4	6
5	1	5	5	**5**	5	5	5	5	5	5	5	5	5
6	1	6	6	**6**	6	6	6	6	6	6	6	6	6
7	1	7	9	**3**	1	7	9	3	1	7	9	3	1
8	1	8	4	**2**	6	8	4	2	6	8	4	2	6
9	1	9	1	**9**	1	9	1	9	1	9	1	9	1

M C

Encryption (exp mod 10)

Ciphertext
C = f_E(<u>K</u>, M)

- **Encryption with K = 3**
  - (M^3) mod 10 = C
  - K = 3 or K = 7 can be used
    - because all input digits are mapped to a unique output digit
  - K = 2 or 4 or 6 or 8 cannot be used
    - because some input digits are mapped to the same output digit
  - K = 1 or K = 5 or K = 9 makes no sense
    - because input and output digits are the same
- **Exists exp-inverse of K for decryption?**
  - exp-inverse is the number so that $(M^K)^{\text{exp-inv(K)}}$ mod 10 = M
    - note: in ordinary arithmetic exp-inv(K) = 1/K = K^{-1}
  - Yes sometimes!

Decryption Aspects (exp mod 10)

- **Example mod 10:**
  - only numbers 1, 3, 7 and 9 are relative prime to 10
  - φ(10)= 4
  - for K = 3

    $K^{\varphi(n)}$ mod n = 1 is valid because of Euler
    Proof : 3^4 mod 10 = 81 mod 10 = 1
    (t*K) mod n = 1 because of lemma of bezout performed modulo n
  - exp-inv(K) can be found by using

    (t*K) mod n = 1 = $K^{\varphi(n)}$ mod n -> t = $K^{\varphi(n)-1}$ mod n

    (exp-inv(3)*3) mod 10 = 3^4 mod 10 -> exp-inv(3)* = 3^{4-1} mod 10 = 27 mod 10 = 7
- **Therefore for K = 3 decryption with exp-inv(K) = 7**
  - (C^7) mod 10 = M
  - examples
    - 2^3 mod 10 = 8 , 8^7 mod 10 = 2097152 mod 10 = 2
    - 3^3 mod 10 = 7 , 7^7 mod 10 = 823543 mod 10 = 3
    - 4^3 mod 10 = 4 , 4^7 mod 10 = 16384 mod 10 = 4

Plaintext
M = f_D(<u>K</u>, M)

Important Rule for Exponentiation

- **Looking to the table**
  - columns 1 and 5 are the same, 2 and 6 are the same, 3 and 7 are the same, 4 and 8 are the same, …
- **Theory proofs**
  - x^y mod n = $x^{y \bmod \varphi(n)}$ mod n
    - this formula is not true for all values of n but it is true if n is prime or the product of two primes for any x with x < n
  - in our case n = 10 (prime factors are p=2 , q=5)
    - numbers 1, 3, 7 and 9 are relative prime to 10
    - φ(10) = 4 = (p-1)*(q-1) = 1 * 4
    - x^y mod 10 = $x^{(y \bmod 4)}$ mod 10
    - so column i+4th and ith are the same
- **Special important case for public-key algorithm (RSA)**
  - if <u>y mod φ(n) = 1</u> then for any number x

    x^y mod n = x mod n or (x^y = x) mod n
  - we need this trick for later for to perform RSA decryption
    - modulo function is a one-way function, in order to reverse we need a trapdoor, this trick shows the possible trapdoor

Agenda

- Introduction
- Number Theory
- <u>RSA</u>
- Diffie-Hellmann

RSA algorithm

- **generate public and private keys**
  - two large primes: p and q
  - build n = p*q
  - φ(n) = (p-1)*(q-1)
  - p and q remains secret
  - select **<u>e</u>** that is relatively prime to φ(n) =(p-1)*(q-1)
    - use Basic Euclid to proof you have found such an e
      - gcd (φ(n) , e) = 1 with 1 < e < φ(n)
    - therefore it must also exists a multiplicative inverse d of e
      - d such that (e*d) mod φ(n) = 1
    - note: because of (Euler 3.)

      $e^{\varphi(n)}$ mod n = 1 is valid

      from (Euler 1.): $a^{\varphi(n)}$ mod n = 1 is valid if a and n are relatively prime to each other but in our case e is only relatively prime to φ(n);

      but if n = p*q (product of two primes) then (Euler 1.) is valid for all numbers x with x less n (Euler 3.) -> hence also for number e

RSA algorithm

- **generate public and private keys (cont.)**
  - find **d** that is the multiplicative inverse of e
    - that is (e*d) mod φ(n) = 1
  - d can be found using the **Extended Euclid** algorithm
- **now the public key is <e, n>**
- **and the private key is <d, n>**
- **it is not feasible**
  - to determine the private key from the public key
    - you need to know p and q in order to build φ(n)
  - but factoring is a hard problem
    - finding p and q based on n

RSA Encryption / Decryption

- **encryption with public key**
  - divide the plaintext message (regarded as bit string) into blocks of M's where every M falls in the interval 0 < M < n
  - compute $C = M^e$ mod n (with public key)
  - decryption $M = C^d$ mod n (with private key)
  - privacy aspect
- **sign a message using private key**
  - compute $S = M^d$ mod n (with private key)
  - verification $M = S^e$ mod n (with public key)
  - authentication and integrity aspect

RSA Algorithm Proof

- **Proof of correctness**
  - we have chosen (e*d) mod φ(n) = 1
    - that means that we can write e*d = 1 + k*φ(n) = 1 + k*(p-1)*(q-1)
  - because of (Euler 4.) the following rule is valid

    $x^{1+k*(p-1)*(q-1)}$ mod n = x for all x less then n
  - therefore for any x if we build x^{ed} mod n
    - then x^{ed} mod n = $x^{(1 + k*(p-1)*(q-1))}$ mod n = x mod n
  - if we encrypt and decrypt:
    - Encrypt: C = M^e mod n
    - Decrypt: M = C^d mod n

      $C^d = (M^e)^d = (M^{ed}) = M$ mod n
  - If we sign and verify:
    - Sign: S = M^d mod n
    - Verify: M = S^e mod n

      $S^e = (M^d)^e = M^{de} = M$ mod n
  - q.e.d

 Public-Key, v4.7 39

RSA Example

- **p = 3, q = 11**
- **hence n = 33, φ(n) = 20**
- **choose e = 3 relatively prime to 20**
  - take 3 (gcd = 1)
- **compute d = 7**
  - 3d = 1 (mod 20)
- **encryption C = M^3 mod 33**
- **decryption M = C^7 mod 33**
- **M < 33**
  - therefore encode every letter of the message as single block; numbers 1 … 26 represent A … Z

 Public-Key, v4.7 40

RSA Example (cont.)

Plaintext Sender		M^3	M^3 (mod 33)	C^7	C^7 (mod 33)	
S	19	6859	28	13492928512	19	S
U	21	9621	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

Ciphertext

Plaintext Receiver

RSA Security 1

- **Security depends on difficulty of factoring**
  - find p and q from n
- **If you can factor quickly you can break RSA**
  - factoring the public value n into p and q, building φ(n) and knowing public value e allows you to find d by performing the same computations (Euclid's algorithm) as done by the key generation
- **Fortunately factoring is a hard problem**
  - Currently 500-bit numbers are the largest which can be factorized
- **The minimum length proposed for n is 1024 bits and for p and q 512 bits each**

RSA Security 2

- **There is a possible misuse**
  - an eavesdropper can guess what is going to be transmitted although he is not able to decrypt
  - he can encrypt all expected messages with the public key and compares the result with the ciphertext he has eavesdropped
    - especially a problem with short messages
  - if there is a match then he knows what was transmitted
- **Therefore is necessary to use special guidelines of how to format RSA message**
  - e.g. short messages should be concatenated with a large random number (e.g. 64 bits)
  - PKCS … Public-Key Cryptography Standard

 Public-Key, v4.7 43

RSA Performance Aspects 1

- **Finding large prime numbers**
  - typically greater than 100^{100}
  - pseudo prime numbers (random numbers) are used and checked with Fermat´s theorem
    - If n is prime then for 0 < a < n -> a^{n-1} mod n = 1
      - taken from (Euler 1.): a $^{\varphi(n)}$ mod n = 1 if a and n are relatively prime
      - if n is prime then φ(n) =n-1
      - If n is really a prime number all numbers a are relatively prime and Fermat's rule is valid
  - primality test:
    - For a number n to check pick a number a < n
    - If a^{n-1} mod n = 1 is not fulfilled then n is certainly not prime
    - If a^{n-1} mod n = 1 is fulfilled then n may or may not be prime (risk for failure is 1 to 10^{13} in case of randomly generated number of about hundred digits)
    - Try multiple values of number a to make the test more reliable
    - Special attention for Carmichael numbers

 Public-Key, v4.7 44

RSA Performance Aspects 2

- **Finding e and d**
  - Euclid´s algorithm
    - normal -> gcd of e and φ(n) = 1
    - extended -> multiplicative-inverse
- **Exponentiation with big numbers**
  - square and multiply algorithm and do modular reduction after each multiply
    - e.g. result of 123^{32} mod 678
      - 123^2 = 123*123 = 15129 = 213 mod 678
      - 123^4 = 213*213 = 45369 = 621 mod 678
      - 123^8 = 621*621 = 385641= 537 mod 678
      - 123^{16} = 537*537 = 288369 = 219 mod 678
      - 123^{32} = 219*219 = 47961 = 501 mod 678

RSA Facts

- **RSA is special type of block cipher**
- **Variable key-length**
  - usually 512 - 2048 bits
  - compromise between enhanced security and efficiency
  - plaintext block need to be smaller than the key length
- **Ciphertext block will be the length of the key**
- **Typically much slower to implement than conventional block ciphers like DES or IDEA**
  - unsuitable for encrypting large messages
  - 1000 times (HW) to 100 times (SW) slower
  - mostly used to encrypt a session key for performing a secret-key algorithm

Agenda

- Introduction
- Number Theory
- RSA
- Diffie-Hellmann

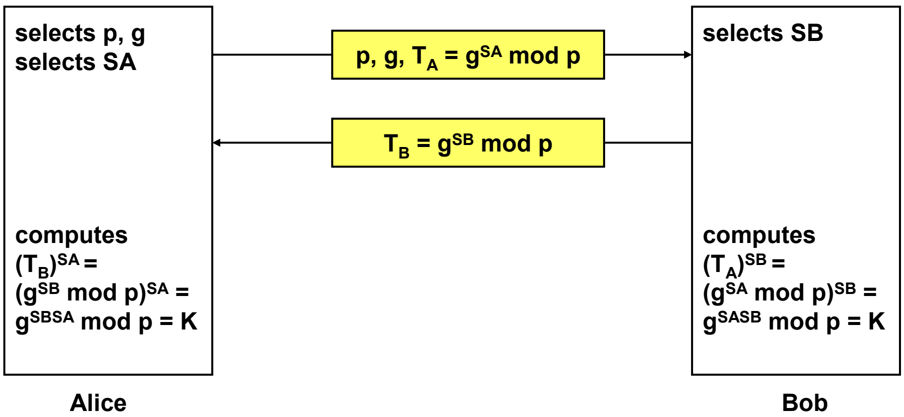
Diffie-Hellman (DH)

- **Oldest public-key system still in use**
- **Compared to RSA**
  - it does neither encryption nor signatures
- **Allows two individuals to agree on a shared key**
  - even though they can only exchange information in public
  - this secret is used for symmetric encryption
  - better performance than doing this with RSA
- **DH used for key establishment**

Diffie-Hellman Algorithm 1

- **Select p prime and g < p**
  - g and p might be public, strong primes are used for p
    - -> more secure if (p-1)/2 is also a prime
- **Select random secret numbers SA, SB**
  - SA,SB are the private DH keys
  - Alice: SA -> $T_A = g^{SA}$ mod p
  - Bob: SB -> $T_B = g^{SB}$ mod p
- **Exchange T_A and T_B**
  - these are the public DH keys
- **Produce shared key K**
  - Alice: $K = T_B^{SA} = (g^{SB})^{SA} = g^{SBSA} = g^{SASB}$ mod p
  - Bob: $K = T_A^{SB} = (g^{SA})^{SB} = g^{SASB} = g^{SBSA}$ mod p

Diffie-Hellman Algorithm 2



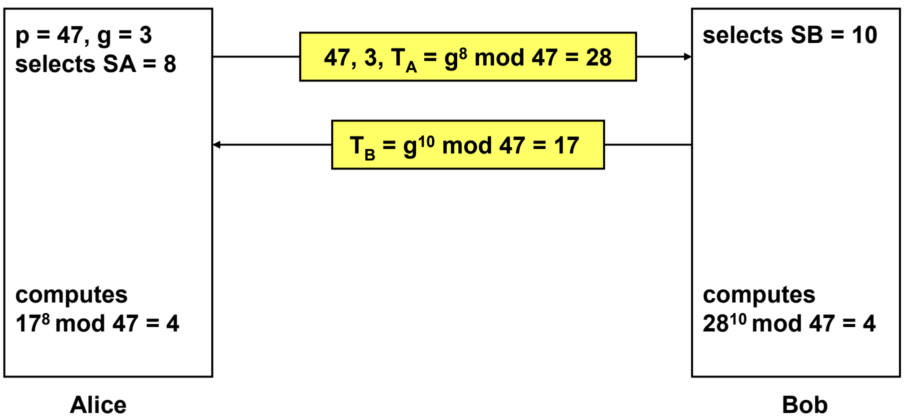
Diffie-Hellman Algorithm 3

- **Nobody else can compute**
  - SA or SB and hence g^{SASB} in reasonable time
  - even though he knows $T_A = g^{SA} \bmod p$
  - even though he knows $T_B = g^{SB} \bmod p$

- **Based on the fact**
  - that discrete logarithm modulo a very large prime number is hard to compute
  - no one has found any efficient method to do this

 Public-Key, v4.7 51

Diffie-Hellman Example

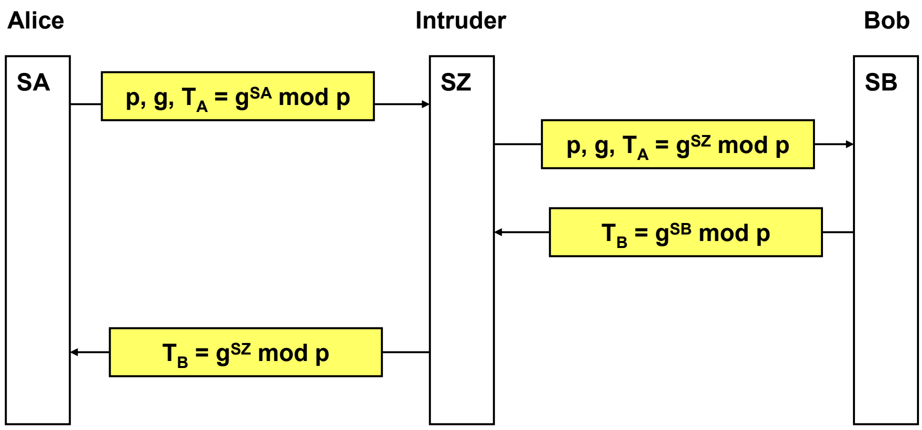


 Public-Key, v4.7 52

Man-in-the-middle Attack 1

- **Problem:**
  - when T_A and T_B are sent over a public network and an active intruder is present
  - Alice will establish a secret key with whoever transmitted T_B but it might not be Bob
  - Bob will establish a secret key with whoever transmitted T_A but it might not be Alice
- **There is no authentication (!) in the key exchange process of DH**
  - vulnerable against man-in-the-middle attack
  - also known as bucket brigade attack
- **DH is only secure against passive attacks**

Man-in-the-middle Attack 2



computes
$(g^{SZ}$ mod p$)^{SA}$ =
g^{SZSA} mod p

uses for messages to Alice
$(g^{SA}$ mod p$)^{SZ}$ = g^{SASZ} mod p
uses for messages to Bob
$(g^{SB}$ mod p$)^{SZ}$ = g^{SBSZ} mod p

computes
$(g^{SZ}$ mod p$)^{SB}$ =
g^{SZSB} mod p

Man-in-the-middle Attack 3

- **Man-in-the-middle Attack**
  - Intruder intercepts communication, impersonates in both directions
  - intruder handles a different secret key with every party
- **Solutions to overcome this attack**
  - published public numbers
    - e.g. in the newspapers or PKI techniques
    - not possible to make active attacks, because the advertised values cannot be changed or cannot at least be easily changed
  - authentication via out-band channel
    - T_A and T_B are transmitted by voice communication over telephone network
    - authentication based on voice recognition of other party

Man-in-the-middle Attack 3

- **Solutions to overcome this attack (cont.)**
  - authentication via encryption
    - Alice and Bob know some sort of secret e.g. each other's public-key
    - encrypt DH value with other side's public-key
    - receiver can decrypt the message by using its own corresponding private-key