Information Security CENG418

week-4

Cryptography: DHKE, RSA

Terms

Cryptographic Algorithms

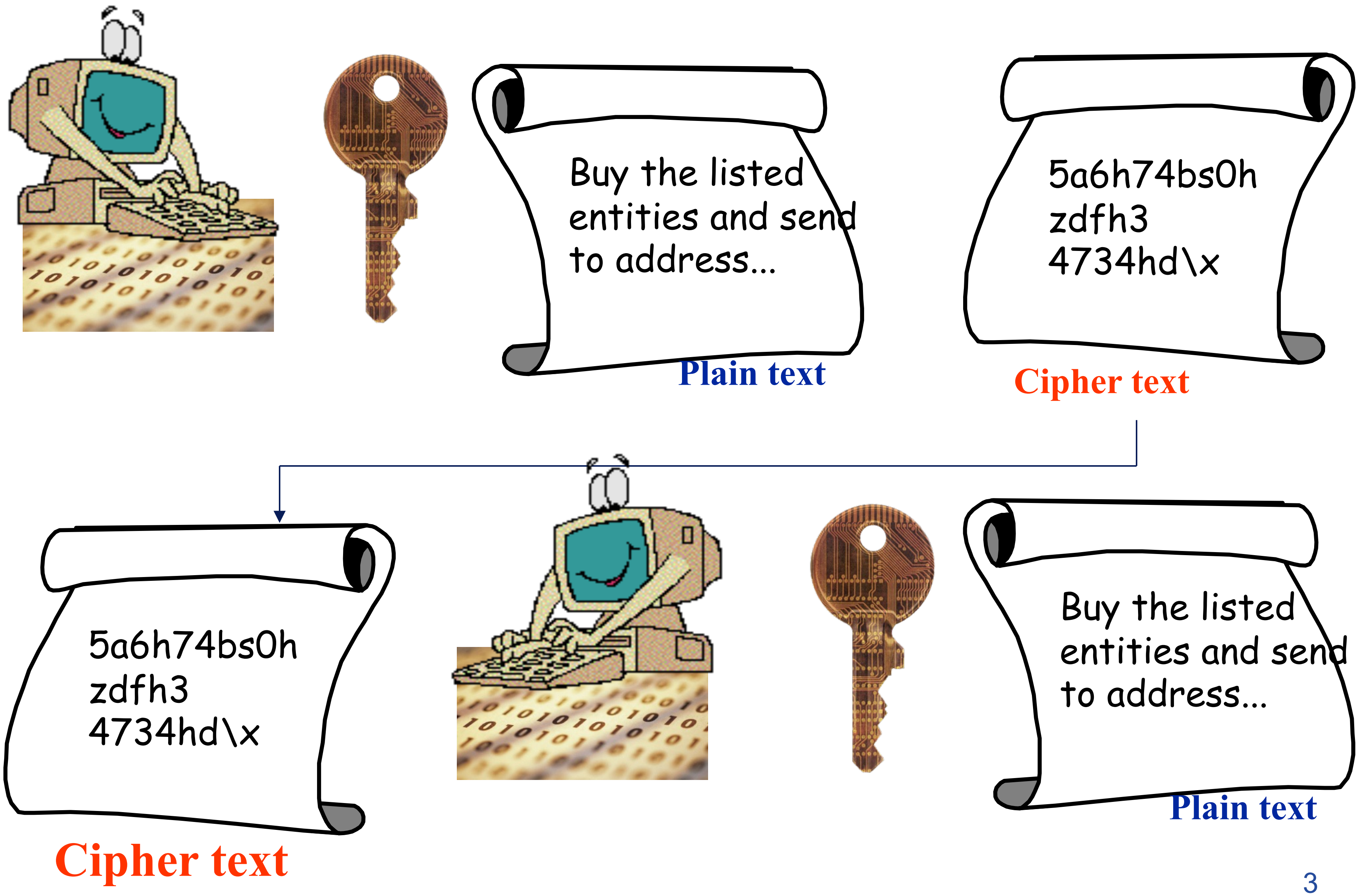
	Encryption	Decryption
	$E(P) = C$	$D(C) = P$
Key(k) $\Rightarrow$	$E_k(P) = C$	$D_k(C) = P$

$E_k = D_k \Rightarrow$ **Symmetrical**

$E_k \neq D_k \Rightarrow$ **Asymmetrical**

Ref: *Slides of Assoc. Prof. Dr. Ahmet Koltuksuz – slides from 2 to 8.*

Symmetrical=same key for E and D


Buy the listed entities and send to address...
Plain text
5a6h74bs0h zdfh3 4734hd\x
Cipher text
5a6h74bs0h zdfh3 4734hd\x
Cipher text
Buy the listed entities and send to address...
Plain text

Asymmetrical Cryptosystems

In 1976 invented by Diffie, Hellman and Merkle.

Main idea is using of two different keys for encryption and decryption (public and private keys).

There is no possibility to produce private key from known public key for any attack.

Asymmetrical Cryptosystems

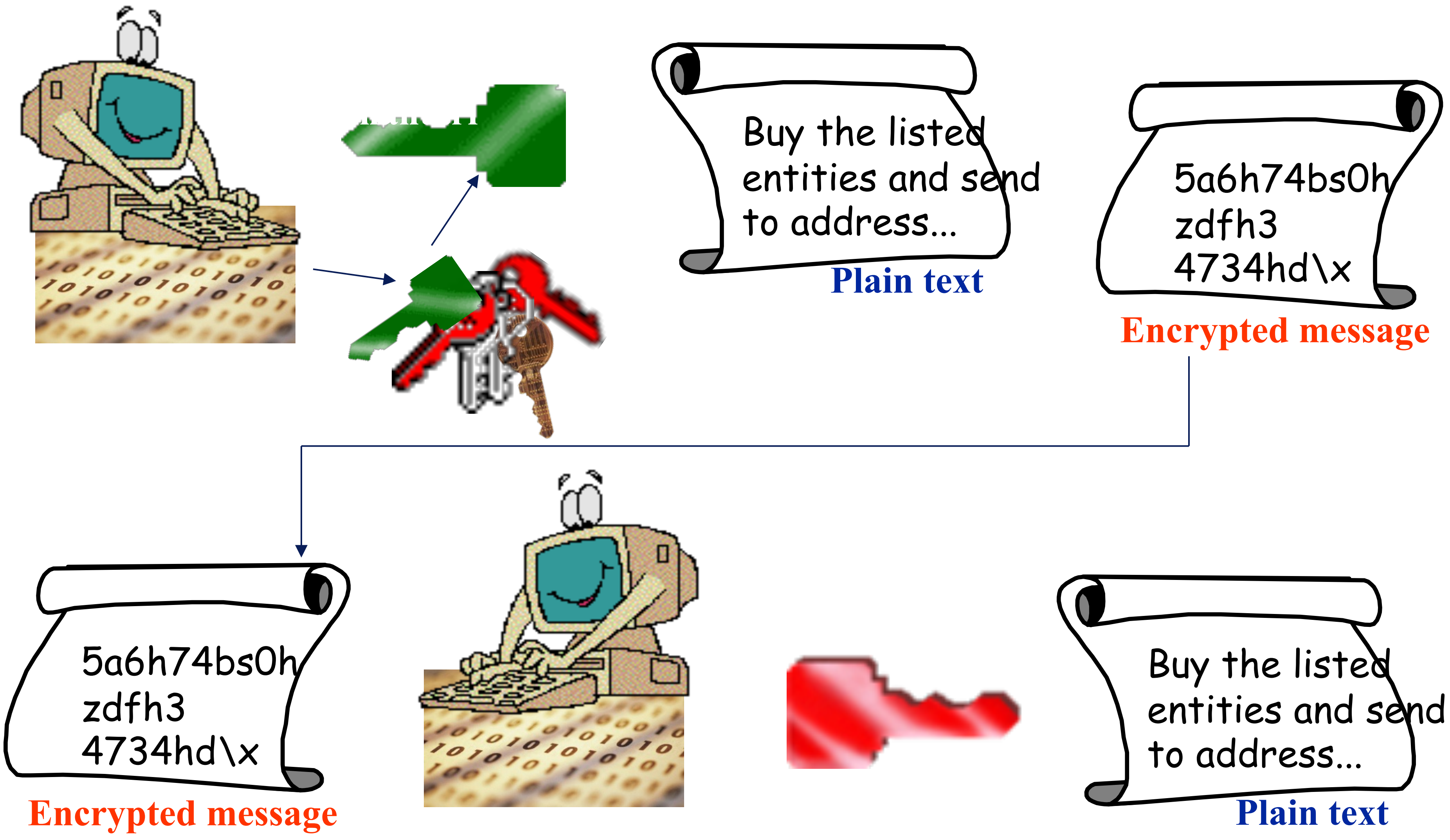
The receiver's public key is known by every one.

Any sender can use the receiver's public key to encrypt own message before to send it.

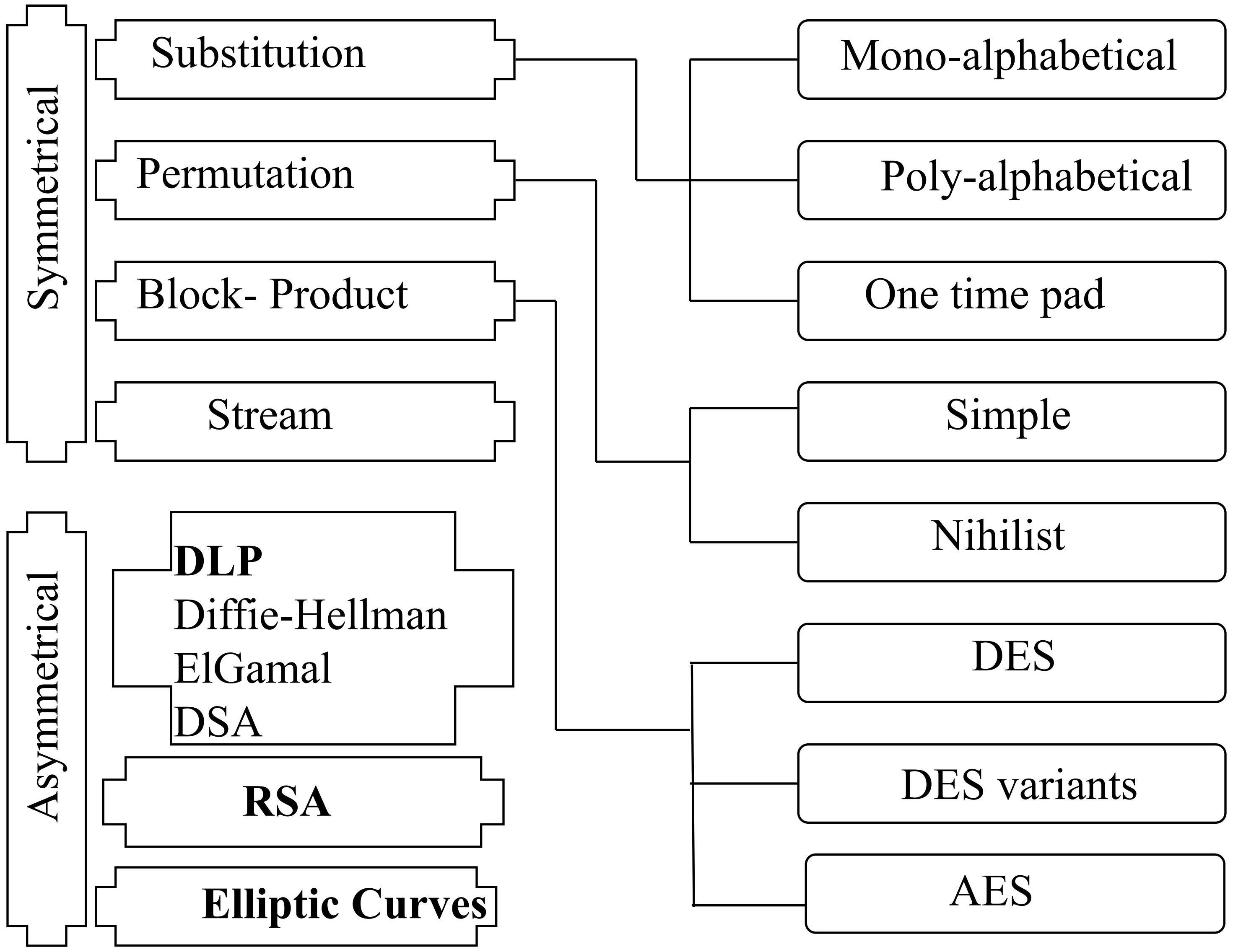
Private key is only known by the receiver and receiver use it to decrypt the received cipher text.

So, everyone should have (public key, private key) pair. Public key is announced to every one. But the secrecy of private key is crucial.

Asymmetrical= two keys


Buy the listed entities and send to address...
Plain text
5a6h74bs0h zdfh3 4734hd\x
Encrypted message
5a6h74bs0h zdfh3 4734hd\x
Encrypted message
Buy the listed entities and send to address...
Plain text

Terms


Symmetrical
Substitution
Permutation
Block- Product
Stream
Asymmetrical
DLP
Diffie-Hellman
ElGamal
DSA
RSA
Elliptic Curves
Mono-alphabetical
Poly-alphabetical
One time pad
Simple
Nihilist
DES
DES variants
AES

Terms

Discrete logarithm problem

Diffie-Hellman, 1976

$y = g^x \pmod p$

Diffie-Hellman key exchange

ElGamal cryptosystem

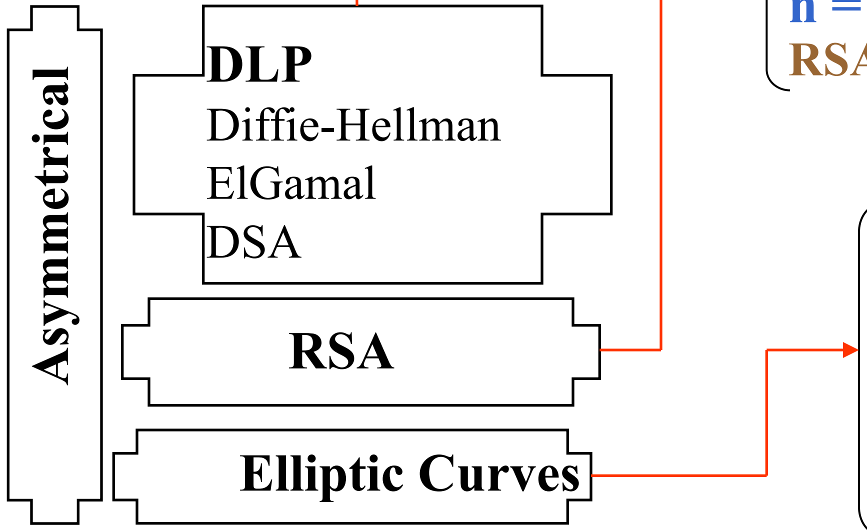
Digital Signature Algoritm (DSA)

Factorization

Rivest-Shamir-Adleman, 1978

$n = pq$

RSA, PGP



Elliptic Curves

EC Discrete logarithm problem

Miller-Koblitz, 1985-87

$y^2 = x^3+ax+b \pmod p$

ECC, ECDSA

Asymmetrical Cryptosystems

Rivest-Shamir-Adleman (RSA) Cryptosystem

References: *Complexity and Cryptography, An Introduction, J.Talbot, D.Welsh, Cambridge*
Stinson_Cryptography_Theory_and_Practice, ch04
Introduction to Cryptography and Security Mechanisms 2005, Lecture Notes of Dr Keith Martin, keith.martin@rhul.ac.uk

RSA Cryptosystem

- The idea of a public-key system was due to Diffie and Hellman in 1976.
- The first realization of a public-key system came in 1977 by Rivest, Shamir, and Adleman, who invented the well-known **RSA Cryptosystem**.
- The security of **RSA** is based on the difficulty of factoring large integers.

RSA

- Step 1: $N = p \times q$, p and q very large prime
- Step 2: $T = \Phi(N) = (p-1)(q-1)$
- Step 3: Choose e s.t. $\gcd(e, T) = 1$
- Step 4: Find $de \equiv 1 \pmod{T}$
- Public key: (N, e)
- Private key: (N, d)

Encryption ➔ $C = m^e \mod N$

Decryption ➔ $m = C^d \mod N$

Rivest

Shamir

Adleman

Euler

Fermat

Fast Modular Exponentiation

Q: How is it even possible to compute 2853^{3397} **mod** 4559 ? After all, 2853^{3397} has approximately 3397·4 digits!

A: By taking the **mod** after each multiplication:

$$
\begin{aligned}
23^3 \textbf{ mod } 30 &\equiv -7^3 \pmod{30} \equiv (-7)^2 \cdot (-7) \pmod{30} \\
&\equiv 49 \cdot (-7) \pmod{30} \equiv 19 \cdot (-7) \pmod{30} \\
&\equiv -133 \pmod{30} \equiv 17 \pmod{30}
\end{aligned}
$$

Therefore, 23^3 **mod** 30 = 17.

Q: What if had to figure out 23^{16} **mod** 30. Same way tedious: need to multiply 15 times. Is there a better way?

Fast Modular Exponentiation

A: Notice that $16 = 2\cdot2\cdot2\cdot2$ so that

$23^{16} = 23^{2\cdot2\cdot2\cdot2} = (((23^2)^2)^2)^2$

Therefore:

23^{16} **mod** $30 \equiv (((-7^2)^2)^2)^2 \pmod{30}$

$\equiv (((49)^2)^2)^2 \pmod{30} \equiv (((-11)^2)^2)^2 \pmod{30}$

$\equiv ((121)^2)^2 \pmod{30} \equiv ((1)^2)^2 \pmod{30}$

$\equiv (1)^2 \pmod{30} \equiv 1 \pmod{30}$

Which implies that 23^{16} **mod** $30 = 1$.

Q: How about 23^{25} **mod** 30 ?

Fast Modular Exponentiation

A: The previous method of ***repeated squaring*** works for any exponent that's a power of 2. 25 isn't. However, we can break 25 down as a sum of such powers: 25 = 16 + 8 + 1. Apply repeated squaring to each part, and multiply the results together. Previous calculation:

23^8 **mod** 30 = 23^{16} **mod** 30 = 1

Thus: 23^{25} **mod** 30 $\equiv 23^{16+8+1}$ (mod 30) $\equiv$

$23^{16}\cdot 23^{8}\cdot 23^{1}$ (mod 30) $\equiv 1\cdot 1\cdot 23$ (mod 30)

Final answer: 23^{25} **mod** 30 = 23

Fast Modular Exponentiation

Q: How could we have figured out the decomposition 25 = 16 + 8 + 1 from the binary (unsigned) representation of 25?

A: $25 = (11001)_2$ This means that

$25 = 1\cdot16+1\cdot8+0\cdot4+0\cdot2+1\cdot1 = 16+8+1$

Can tell which powers of 2 appear by where the 1's are. This follows from the definition of binary representation.

Greatest Common Divisor

- The "greatest common divisor" (GCD or gcd), of a and b is the largest positive integer dividing both a and b and is denoted by either gcd(a,b) or by (a,b).

  Examples: gcd(6,4)=2, gcd(5,7)=1, gcd(24,60)=12.
- If gcd(a,b)=1 then a and b are relatively prime.
- There are two standard ways to find the gcd:

1. If you can factor a and b into primes; for each prime number, look at the powers that it appears in the factorization of a and b, take the smaller of the two. Put these prime powers together to get the gcd.

   $576=2^6 . 3^2$, $135=3^3.5$, $gcd(576,135)=3^2=9$

   $gcd(2^5 . 3^4 . 7^2, 2^2 . 5^3 . 7)= 2^2 . 3^0 . 5^0 . 7^1= 2^2 . 7=28$.

Greatest Common Divisor

2. Suppose a and b are large numbers. The gcd can be calculated by using Euclidean Algorithm.

Example: gcd(482, 1180)=?

1180=2.482+216

482=2.216+50

216=4.50+16

50=3.16+**2**

16=8.2+0

Notice that how the numbers are shifted?

The last non-zero remainder is the GCD. gcd(482, 1180)=2

Greatest Common Divisor

So, the formal description of the Euclidean Algorithm:

Suppose that a>b , if not; switch a and b.

Step 1. divide a by b and represent in the form: $a=q_1b+r_1$

Step 2. If $r_1=0$ then b divides a and gcd is b.

If $r_1\neq 0$ then continue by representing b in the form $b=q_2r_1+r_2$

Continue in this way until remainder is zero, giving the following sequence steps:

$$\begin{aligned} a &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{k-2} &= q_k r_{k-1} + r_k \\ r_{k-1} &= q_{k+1} r_k \end{aligned}$$

The conclusion is $\gcd(a,b)=r_k$.
This algorithm does not require factorization of numbers and it is fast.

Solving ax+by=d

- We did not use the quotients in the Euclidean Algorithm.
  ax+by=gcd(a,b) ➔ How we find x and y?

$$\gcd(482,1180)=2$$
$$1180 = 2.482 + 216$$
$$482 = 2.216 + 50$$
$$216 = 4.50 + 16$$
$$50 = 3.16 + 2$$
$$16 = 8.2 + 0$$

The successive quotients be $q_1=2$, $q_2=2$, $q_3=4$, $q_4=3$ and $q_5=8$.
From the following sequences:
$x_0=0$, $x_1=1$, $x_j=-q_{j-1}.x_{j-1}+x_{j-2}$
$y_0=1$, $y_1=0$, $y_j=-g_{j-1}.y_{j-1}+y_{j-2}$
Then $ax_n+by_n=\gcd(a,b)$

⇩

$$x_0 = 0; x_1 = 1$$
$$x_2 = -2x_1 + x_0 = -2$$
$$x_3 = -2x_2 + x_1 = 5$$
$$x_4 = -4x_3 + x_2 = -22$$
$$x_5 = -3x_4 + x_3 = 71$$

Similarly we calculate $y_5=-29$.
An easy calculation shows that 482.71+1180.(-29)=2
gcd(482, 1180)=2
Notice that we did not use the final quotient. If we had used it, we would have calculated $x_{n+1}=590$, which is the 1180/2 and similarly $y_{n+1}=241$ is 482/2.

This method is called Extended Euclidean Algorithm and it will use for solving congruencies!

Solving ax+by=d

- Example: 22x + 60y = gcd(60,22) find the gcd(60,22) by Euclidean Algorithm.

$$60 = 2.22 + 16$$

gcd(60,22)=2

$$22 = 1.16 + 6$$

$$16 = 2.6 + 4$$

$$6 = 1.4 + 2$$

$$4 = 2.2 + 0$$

$$a = 2.b + 16 \Rightarrow 16 = a - 2b$$

$$b = 1.16 + 6 \Rightarrow 6 = b - 1.16 = b - (a - 2b) = -a + 3b$$

$$16 = 2.6 + 4 \Rightarrow 4 = 16 - 2.6 = (a - 2b) - 2.(-a + 3b) = 3a - 8b$$

$$6 = 1.4 + 2 \Rightarrow 2 = 6 - 4 = (-a + 3b) - (3a - 8b) = -4a + 11b$$

$$-4a + 11b = \gcd(a, b) = 2 = -4.60 + 11.22$$

$$= -240 + 242 = 2$$

Congruences

- Definition
  - Let a,b,n Є Z with n≠0, we say that a≡b (mod n), or a is congruent to b mod n.
  - If (a – b) is a multiple (positive or negative) of n.
  - This can be rewritten as a=b+n.k for some integer k.

Examples: 16 ≡ 1 (mod 5)

-3 ≡ 6 (mod 9)

-12 ≡ 2 (mod 7)

Congruences

- Division: The general rule is that you can divide by a (mod n) when gcd(a,n)=1.
- Proposition: Let a,b,c,n Є Z with n≠0 and with gcd(a,n)=1.
  - If a.b ≡ a.c (mod n) then b ≡ c (mod n). In other words, if a and n are relatively prime, we can divide both sides of the congruence by a.
- Proof: Since gcd(a,n)=1, there exist integers x, y such that ax+ny=1. Multiply by (b – c) to obtain

  (ab – ac)x + n(b – c)y = b – c

  Since a.b – a.c is a multiple of n, by assumption and n(b – c)y is also a multiple of n, we find that b – c is a multiple of n. This means than b ≡ c (mod n)

Congruences

- Example: Solve $2x+7 \equiv 3 \pmod{17}$

  $2x \equiv 3 - 7 \equiv -4$ so $x \equiv -2 \equiv 15 \pmod{17}$ The division by 2 is allowed since $\gcd(2, 17)=1$.
- Example: Solve $5x + 6 \equiv 13 \pmod{11}$

  $5x \equiv 7 \pmod{11} \rightarrow$ Note that $7 \equiv 18 \equiv 29 \equiv 40 \equiv \ldots \pmod{11}$

So; $5x \equiv 7 \pmod{11}$ is the same as $5x \equiv 40 \pmod{11}$. Now we can divide by 5 and obtain $x \equiv 8 \pmod{11}$.

Note that $7 \equiv 8.5 \pmod{11}$, so 8 acts like 7/5.

Another solution is; since $5.9 \equiv 1 \pmod{11}$. We see that 9 is the multiplicative inverse of 5 (mod 11). Therefore dividing 5 can be accomplished by multiplying by 9.

$5x \equiv 7 \pmod{11} \rightarrow x \equiv 7/5 \equiv 7.9 \equiv 63 \equiv 8 \pmod{11}$

Fermat's Little Theorem

- $x^{p-1} \equiv 1\ (mod\ p)$ is FLT
- We can use FLT to simplify computations for large numbers;

$$2^{35} \equiv ?(\text{mod}\,7) \Rightarrow 35 \equiv 6.5 + 5 \qquad and$$

$$2^{35} = (2^6)^5.2^5 \equiv 1^5.2^5 \equiv 32 \equiv 4(\text{mod}\,7)$$

EULER's Phi Function

- $\Phi(m) = \textit{the order of the relatively prime numbers with } m.$
- Euler's formula is; if a,n are rel. prime: $a^{\Phi(m)} \equiv 1(\bmod m)$

1. If m=p is prime then every integer 1≤a≤p-1 is relatively prime to m, thus $\Phi(p) = p-1$
2. If $m = p^k \Rightarrow \Phi(p^k) = p^k - p^{k-1}$
3. If $m = p^j.q^k \Rightarrow \Phi(p^j.q^k) = \Phi(p^j).\Phi(q^k)$
4. If $\gcd(m,n) = 1 \Rightarrow \Phi(m.n) = \Phi(m).\Phi(n)$

This is important for composite numbers and simplifying computation for large composite numbers;

If $\gcd(a,m) = 1 \Rightarrow a^{\Phi(m)} \equiv 1(\bmod m)$

Example: a=7 and m=9. Phi(9)=6 (1,2,4,5,7,8)

7^6 = 1 (mod 6)

RSA Cryptosystem

- The idea of a public-key system was due to Diffie and Hellman in 1976.
- The first realization of a public-key system came in 1977 by Rivest, Shamir, and Adleman, who invented the well-known **RSA Cryptosystem**.
- The security of **RSA** is based on the difficulty of factoring large integers.

RSA

- Step 1: $N = p \times q$, p and q very large prime
- Step 2: $T = \Phi(N) = (p-1)(q-1)$
- Step 3: Choose e s.t. $\gcd(e, T) = 1$
- Step 4: Find $de \equiv 1 \pmod{T}$
- Public key: (N, e)
- Private key: (N, d)

Encryption ➔ $C = m^e \; mod \; N$

Decryption ➔ $m = C^d \; mod \; N$

RSA

- Example 1
  - Let $p = 7, q = 11, e = 13$
  - Give the public and private keys in RSA cryptosystem

RSA

- Step 1: $N = 7 \times 11 = 77$
- Step 2: $T = 6 \times 10 = 60$
- Step 3: $\gcd(13, 60) = 1$, the choice is ok
- Step 4:

$$\begin{aligned} 60 &= 4 \times 13 + 8 \\ 13 &= 1 \times 8 + 5 \\ 8 &= 1 \times 5 + 3 \\ 5 &= 1 \times 3 + 2 \\ 3 &= 1 \times 2 + 1 \end{aligned} \quad \Longleftrightarrow \quad d = 37$$

RSA

- Public key: $(77, 13)$
- Private key: $(77, 37)$
- Example 2: Encrypt 5

$$C = 5^{13} \bmod 77 = 26$$

- Example 3: Decrypt

$$M = 26^{37} \bmod 77$$

RSA

$60^2 \, mod \, 77 \equiv 58$

$58^2 \, mod \, 77 \equiv 53$

$53^2 \, mod \, 77 \equiv 37$

- Example 3

$$37 = 32 + 4 + 1$$

$$\begin{aligned}
26^2 \bmod 77 &= 60 \\
26^4 \bmod 77 &= 58 && \Longrightarrow \quad 26^5 \bmod 77 = 45 \\
26^8 \bmod 77 &= 53 && \qquad\qquad \Downarrow \\
26^{16} \bmod 77 &= 37 \\
26^{32} \bmod 77 &= 60 && \Longrightarrow \quad 26^{37} \bmod 77 = 5
\end{aligned}$$

$37^2 \, mod \, 77 \equiv 60$

One way functions

- They play a central role in cryptography; they are important for constructing public-key cryptosystems.

- Example; Suppose ***n*** is the product of two large primes ***p*** and ***q***, and let ***b*** be a positive integer. Then define $f : Z_n \to Z_n$ to be $f(x) = x^b \bmod n$

- For a suitable choice of *b* and *n*, this is in fact the **RSA** encryption function.

Trapdoor One way functions

- If we construct a public-key cryptosystem, We do not want $\mathbf{e_K}$ to be a one-way function from Bob's (receiver's) point of view, since he wants to be able to decrypt messages that he receives in an efficient way.
- It is necessary that Bob possesses a trapdoor, which consists of secret information that permits easy inversion of $\mathbf{e_K}$.
  - That is, Bob can decrypt efficiently because he has some extra secret knowledge about K.
- So, we say that a function is a trapdoor one-way function if it is a one-way function, but it becomes easy to invert with the knowledge of a certain trapdoor.

The RSA trapdoor 1-to-1 function

- Parameters: $\left\{ \begin{array}{l} N=pq. \quad N \approx 1024 \text{ bits.} \quad p,q \approx 512 \text{ bits.} \\ e – \text{encryption exponent.} \quad \gcd(e, \varphi(N)) = 1 . \end{array} \right.$

- 1-to-1 function: $\mathbf{RSA(M) = M^e}$ (mod N) where $M \in Z_N^*$

- Trapdoor: d - decryption exponent.

  Where $e \cdot d = 1 \pmod{\varphi(N)}$

- Inversion: $RSA(M)^d = M^{ed} = M^{k\varphi(N)+1} = M \pmod{N}$

- (n,e,t,ε)-RSA Assumption: For any t-time alg. A:

$$\Pr\left[A(N,e,x) = x^{1/e} (N) \;:\; \begin{array}{l} p,q \xleftarrow{R} \text{n-bit primes,} \\ N \leftarrow pq, \quad x \xleftarrow{R} Z_N^* \end{array} \right] < \varepsilon$$

Security of RSA Cryptosystem

- The security of **RSA** is based on the hope that the encryption function $e_K(x) = x^b \bmod n$ is one-way, so it will be computationally infeasible for an opponent to decrypt a ciphertext.
- The trapdoor that allows Bob to decrypt is the knowledge of the factorization $n = pq$.
- Since Bob knows this factorization, he can compute $\varphi(n) = (p - 1)(q - 1)$ and then compute the decryption exponent a using the **Extended Euclidean algorithm**.

Security of RSA Cryptosystem

- One obvious attack on the cryptosystem is for a cryptanalyst to attempt to factor n.
- If this can be done, it is a simple manner to compute $\varphi(n)=(p - 1)(q - 1)$ and then compute the decryption exponent a from b exactly as Bob did.
- if the **RSA Cryptosystem** is to be secure, it is certainly necessary that $n = pq$ must be large enough that factoring it will be computationally infeasible.
  - Current factoring algorithms are able to factor numbers having up to **130 decimal digits.**

Security of RSA Cryptosystem

- For example; It is recommended that, to be on the safe side;
  - One should choose *p* and *q* to each be primes having about 100 digits; then *n* will have 200 digits.
  - Several hardware implementations of **RSA** use a modulus which is 512 bits in length.
  - However, a **512-bit modulus corresponds to about 154 decimal digits** and hence it does not offer good long-term security.
- An encryption (or decryption) involves performing one exponentiation modulo *n*. Since *n* is very large, we must use multiprecision arithmetic to perform computations in Z_n and the time required will depend on the number of bits in the binary representation of *n.*

Security of RSA

- Possible security holes:
  - Need to use “safe” primes p and q. In particular p-1 and q-1 should have large prime factors …
  - p and q should not have the same number of digits. Can use a middle attack starting at sqrt(n).
  - e (encryption key) cannot be too small
  - Don’t use same n for different e’s.

Decrypting ciphertext without the key

The encryption process in RSA involves computing the function $C=M^e$ mod n, which is regarded as being easy.

An attacker who observes this ciphertext, and has knowledge of e and n, needs to try to work out what M is.

Computing M from C, e and n is regarded as a hard problem.

Factoring in the Real World

- **<u>Quadratic Sieve (QS):</u>**

$$T(n) = e^{(1+o(n))(\ln n)^{1/2}(\ln(\ln n))^{1/2}}$$

  – Used in 1994 to factor a 129 digit (428-bit) number. 1600 Machines, 8 months.

- **<u>Number field Sieve (NFS):</u>**

$$T(n) = e^{(1.923+o(1))(\ln n)^{1/3}(\ln(\ln n))^{2/3}}$$

  – Used in 1999 to factor 155 digit (512-bit) number. 35 CPU years. At least 4x faster than QS

RSA in the “Real World”

- **Part of many standards**: PKCS, ITU X.509, ANSI X9.31, IEEE P1363
- **Used by**: SSL, PEM, PGP, Entrust, …

- The standards specify many details on the implementation, e.g.
  - e should be selected to be small, but not too small
  - “multi prime” versions make use of n = pqr…
    this makes it cheaper to decode especially in parallel (uses Chinese remainder theorem).

RSA security summary

There are two one-way functions involved in the security of RSA.

One-way function	Description
Encryption function	The encryption function is a trapdoor one-way function, whose trapdoor is the private key. The difficulty of reversing this function without the trapdoor knowledge is **believed** (but not known) to be as difficult as factoring.
Multiplication of two primes	The difficulty of determining an RSA private key from an RSA public key is **known** to be equivalent to factoring n. **An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless they can factor n.** Because multiplication of two primes is believed to be a one-way function, determining an RSA private key from an RSA public key is believed to be very difficult.

Rivest-Shamir-Adleman (RSA) Cryptosystem

- RSA seems to be a secure public-key cryptosystem, despite our best efforts.
- Most attacks on RSA come from poor configuration or bad implementations.
- Post-Quantum Cryptography address the RSA as insecure cryptosystem due to parallelization potential of quantum computers.

Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman

The **Diffie–Hellman (DH) key exchange** technique was first defined in their seminal paper in 1976.

DH key exchange is a method of exchanging public (i.e. non-secret) information to obtain a shared secret.

DH is not an encryption algorithm.

DH key exchange has the following important properties:

1. The resulting shared secret cannot be computed by either of the parties without the cooperation of the other.
2. A third party observing all the messages transmitted during DH key exchange cannot deduce the resulting shared secret at the end of the protocol.

Principle behind DH

DH key exchange was first proposed before there were any known public key algorithms, but the idea behind it motivated the hunt for practical public key algorithms.

DH key exchange is not only a useful and practical key establishment technique, but also a significant milestone in the history of modern cryptography.

DH key exchange assumes first that there exists:

1. A public key cipher system that has a special property (we come to this shortly).

2. A carefully chosen, publicly known function F that takes two numbers x and y as input, and outputs a third number F(x,y) (for example, multiplication is such a function).

Principle behind DH

Assume that Alice and Bob are the parties who wish to establish a shared secret, and let their public and private keys in the public key cipher system be denoted by (PA , SA) and (PB , SB) respectively.

The basic principle behind Diffie–Hellman key exchange is as follows:

1. Alice and Bob exchange their public keys PA and PB.
2. Alice computes F(SA , PB)
3. Bob computes F(SB, PA)
4. The special property of the public key cipher system, and the choice of the function F, are such that F(SA , PB) = F(SB, PA). If this is the case then Alice and Bob now share a secret.
5. This shared secret can easily be converted by some public means into a bitstring suitable for use as, for example, a DES key.

Diffie-Hellman key exchange

The most commonly described implementation of DH key exchange uses the keys of the ElGamal cipher system and a very simple function F.

The system parameters (which are public) are:

- **a large prime number p – typically 1024 bits in length**
- **a primitive element g**

1. Alice generates a private random value **a**, calculates $\mathbf{g}^{\mathbf{a}}$ (mod **p**) and sends it to Bob. Meanwhile Bob generates a private random value **b**, calculates $\mathbf{g}^{\mathbf{b}}$ (mod **p**) and sends it to Alice.
2. Alice takes $\mathbf{g}^{\mathbf{b}}$ and her private random value **a** to compute $(\mathbf{g}^{\mathbf{b}})^{\mathbf{a}} = \mathbf{g}^{\mathbf{ab}}$ (mod **p**).
3. Bob takes $\mathbf{g}^{\mathbf{a}}$ and his private random value **b** to compute $(\mathbf{g}^{\mathbf{a}})^{\mathbf{b}} = \mathbf{g}^{\mathbf{ab}}$ (mod **p**).
4. Alice and Bob adopt $\mathbf{g}^{\mathbf{ab}}$ (mod **p**) as the shared secret.

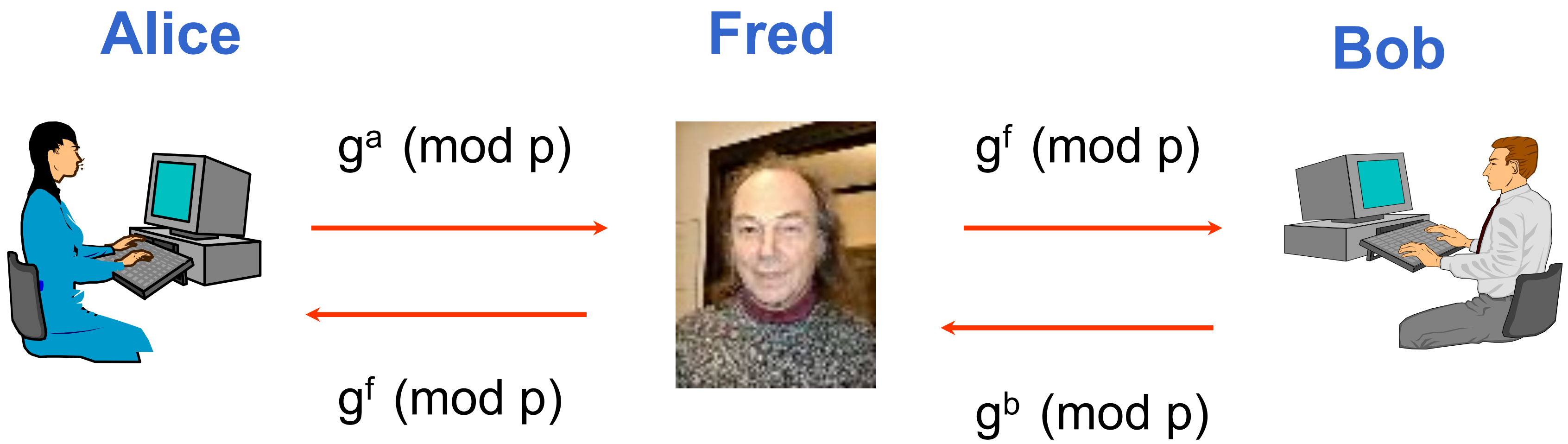
DH questions

1. What is the hard problem on which the DH key exchange algorithm is based?
2. The example of DH key exchange that we described is based on ElGamal keys. Can you use the public and private keys of any established public key encryption algorithm to implement DH key exchange?

$$b = a^x \bmod p \longrightarrow \textbf{Easy}$$

$$x = \log_a b \bmod \mathrm{p} \longrightarrow \textbf{Hard problem !}$$

Man-in-the-middle attack



1. What will happen when Alice tries to send a message to Bob, encrypted with a key based on her DH shared secret?
2. Can Fred obtain the correct DH shared secret that would have been established had he not interfered?

Symmetrical vs Asymmetrical Cryptosystems

Symmetrical vs Asymmetrical Cryptosystems

Symmetrical

☺Fast encryption, decryption

☺Proved security

(one time pad)

☹Key Distribution problem

☹Open for cryptanalysis

Information Theory

ASİMETRİK

☹Slow encryption, decryption

☹ Security has not proven

☺ No key distribution problem

☺ Resilient for cryptanalysis

Complexity Theory

Symmetrical vs Asymmetrical Cryptosystems

- Monitoring
- Frustration
- Duplication

$\Rightarrow$ Secrecy $\Rightarrow$ Symmetrical Cryptosystems

- Modification
- Fabrication

$\Rightarrow$ Integrity control

- Denying

$\Rightarrow$ Identification

$\left.\begin{array}{l}\text{Integrity control}\\ \text{Identification}\end{array}\right\} \Rightarrow$ Asymmetrical Cryptosystems

Q. Which cryptosystem is more secure?

Security is depend on key length and secrecy of private key.

For instance; DES is approximately 100 times faster than RSA but it is breakable...

Suggestions

To improve security:

Use symmetrical and asymmetrical cryptosystems combinations.

For instance: Use Triple DES for secrecy of your communication, and use Asymmetrical cryptosystems to exchange the keys and for their secrecy. (such that use PGP). Periodicaly generate and share new keys for Triple-DES.

Try out this simple implementation of RSA:

- https://gist.github.com/marnix135/582c78891b29186ba4c6882a4bc62822

Next Week

- Identification and Non-Repudiation
  - PKI and Digital Signature (DSA – Digital Signature Algorithm examples)