CENG 471 Cryptography
Elliptic Curve Cryptosystems

Asst. Prof. Dr. Serap ŞAHİN

ELLIPTIC CURVE CRYPTOSYSTEMS

- For simplicity, we shall restrict our attention to elliptic curves over Zp, where p is a prime. The elliptic curves can more generally be defined over any finite field.

Elliptic Curves over F_q

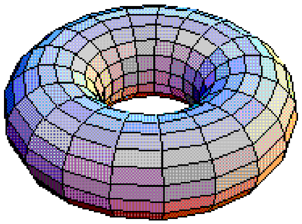
- Most standarts which specify the elliptic curve cryptographic techniques restrict the order of the underlying finite field !
- $q = p$ to be an odd prime number or a power of 2 ($q = 2^m$).
- Let $p > 3$ be an odd prime.
- Let elliptic curve E over F_p is defined by an equation of the form:

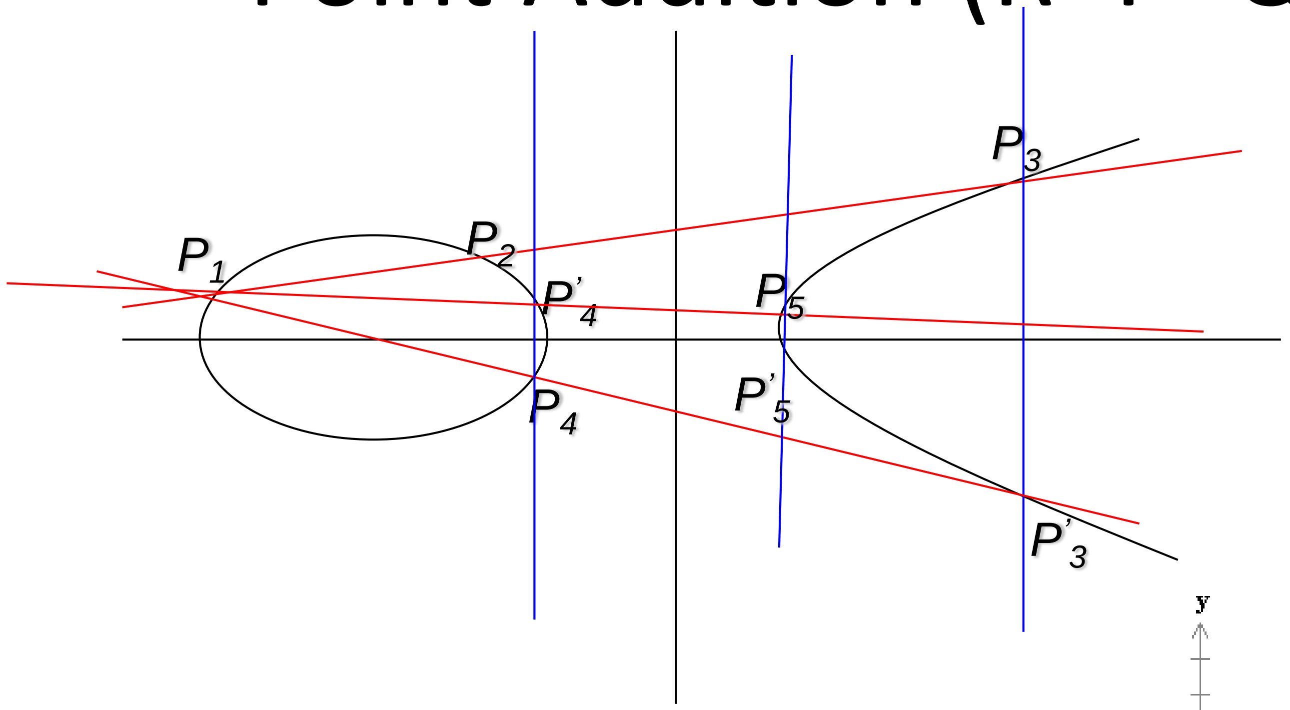
$$y^2 = x^3 + ax + b \text{ where } a, b \in F_p \text{ and}$$

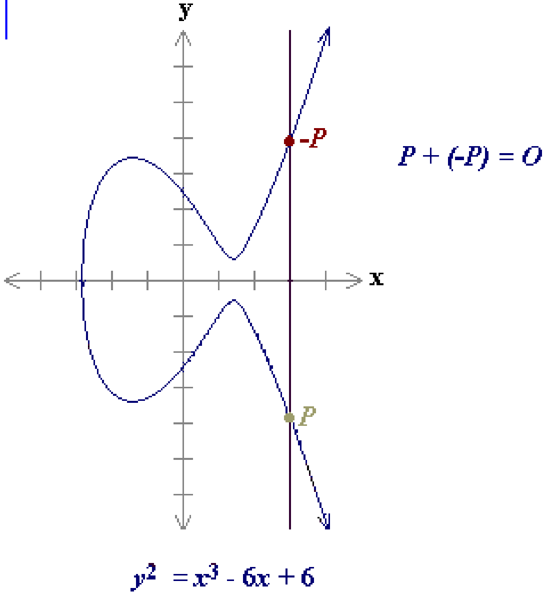
$$4a^3 + 27b^2 \neq 0 \pmod p$$

The set of $E(F_p)$ consists of all points (x, y), $x \in F_p$ and $y \in F_p$, which satisfy the $y^2 = x^3 + ax + b$ equation, together with a special point O, called the point at infinity.

Point Addition (R=P+Q







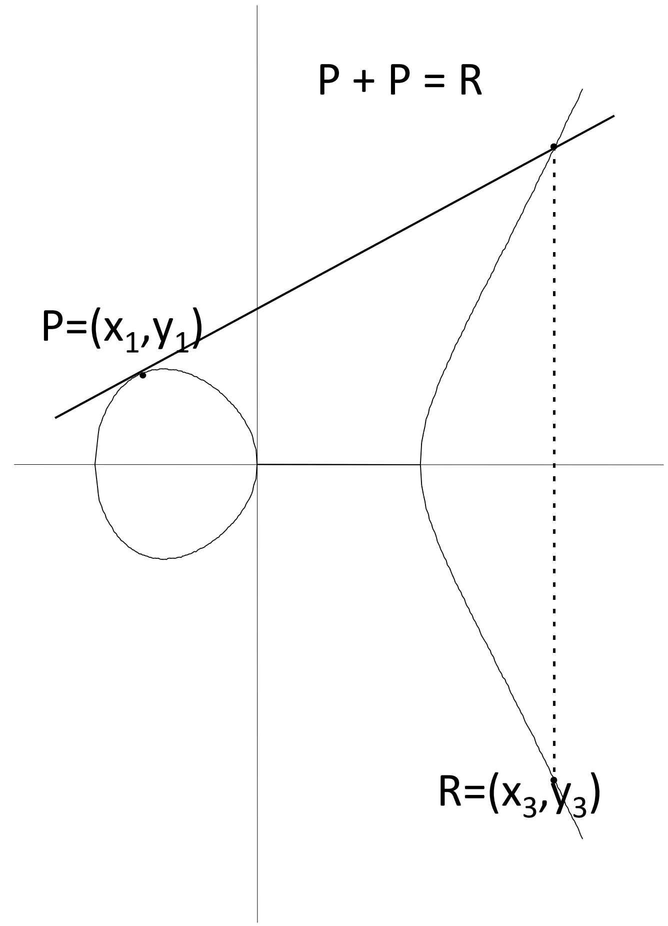
An Example for Elliptic Curves over F_p

- Let p = 23
- E: $y^2 = x^3 + x + 4$ defined over F_{23}
- $4a^3 + 27b^2 \neq 0 \pmod{p}$
- a = 1 and b = 4 then $4(1)^3 + 27(4)^2 \equiv 22 \pmod{23}$, so E is indeed an elliptic curve.
- The points of $E(F_{23})$ are O and the following:

(0 , 1)	*(0 , 21)*	*(1, 11)*	*(1, 12)*	*(4, 7)*	*(4 , 16)*	*(7, 3)*	*(7, 20)*	*(8 , 8)*	*(8, 15)*
(9 , 11)	*(9 , 12)*	*(10, 5)*	*(10, 18)*	*(11, 9)*	*(11, 14)*	*(13, 11)*	*(13, 12)*	*(14, 15)*	*(14, 18)*
(15, 16)	*(15, 17)*	*(17, 9)*	*(17, 14)*	*(18, 9)*	*(18, 14)*	*(22, 5)*	*(22, 19)*		

Group Law Axioms

- Closure
- Identity:
  $P + O = O + P = P$ for all $P \in E(F_p)$
- Inverse:
  $(x, y) + (x, -y) = O$
- Associativity
- Commutativity

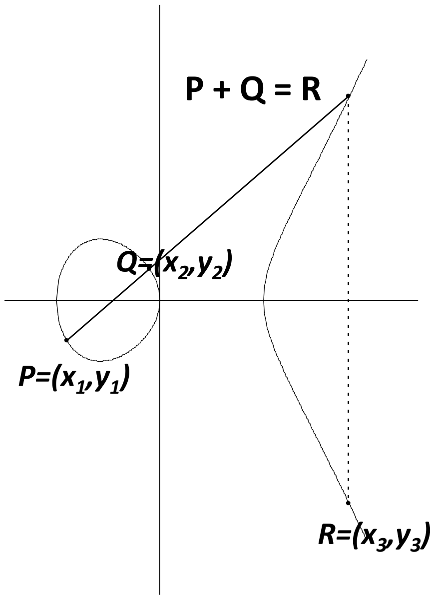


Addition Formulae

Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $P, Q \in E(F_p)$

where $P \neq \pm Q$ Then $P + Q = (x_3, y_3)$ or point doubling $2P = (x_3, y_3)$

<u>Then</u>;



λ is the slope of the line:

if $x_1 = x_2$ (point doubling)

$$\lambda = (3x_1^2+a)/2y_1$$

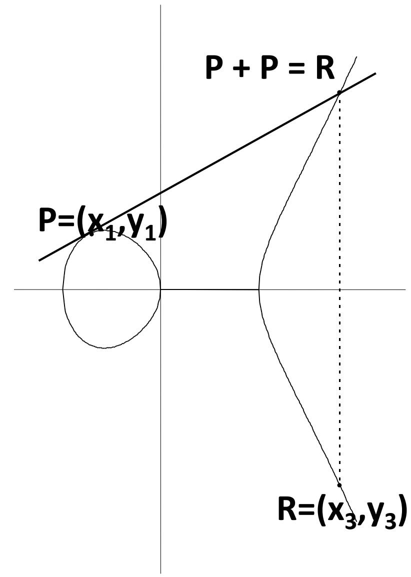
otherwise

$$\lambda = (y_2-y_1)/(x_2-x_1)$$

and

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda (x_1 - x_3) - y_1$$



Addition Formula

We will digress to modular division: 4/3 mod 11. We are looking for a number, say t, such that $3 * t \bmod 11 = 4$. We need to multiply the left and right sides by 3^{-1}

$$3^{-1} * 3 * t \bmod 11 = 3^{-1} * 4$$
$$t \bmod 11 = 3^{-1} * 4$$

Next we use the Extended Euclidean algorithm and get (inverse) 3^{-1} is 4 ($3 * 4 = 12 \bmod 11 = 1$).

$$4 * 4 \bmod 11 = 5$$

Hence,

$$4/3 \bmod 11 = 5$$

Example of Elliptic Curve Addition

1. Let $P = (3, 10)$ and $Q = (9, 7)$. Then $P + Q = (x3, y3)$ is computed as follows:

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in Z_{23}$$

$x_3 = 11^2 - 3 - 9 = 6 - 3 - 9 = -6 \equiv 17 \pmod{23}$, and $y_3 = 11(3 - (-6)) - 10 = 11(9) - 10 = 89 \equiv 20 \pmod{23}$.

Hence $P + Q = (17, 20)$.

2. Let $P = (3,10)$. Then $2P = P + P = (x_3, y_3)$ is computed as follows:

$$\lambda = \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in Z_{23}$$

$x_3 = 6^2 - 6 = 30 \equiv 7 \pmod{23}$, and $y_3 = 6(3 - 7) - 10 = -24 - 10 = -11 \in 12 \pmod{23}$.
Hence $2P = (7, 12)$.
Consider the following elliptic curve with Z_p^*

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

Set $p = 11$ and $a = 1$ and $b = 2$. Take a point $P\,(4, 2)$ and multiply it by 3; the resulting point will be on the curve with $(4, 9)$.

EC Security

- Suppose Eve the middleman captures (p, a, b, Q_A, Q_B).
- Can Eve figure out the shared secret key without knowing either (d_B, d_A)?
- Eve could use $Q_A = P*d_A$ to compute the unknown d_A, which is known as the Elliptic Curve Discrete Logarithm problem.
- With appropriate cryptographic restrictions, this is believed to take exponential time.

Domain Parameters

- Common values shared by a group of users from which key pairs may be generated
- User or trusted party may generate domain parameters
- Anyone may validate domain parameters

EC Domain Parameters

- Finite field F_q
- E is an elliptic curve over F_q
- $\#E(F_q) = kr$
  - r is the prime divisor of $\#E(F_q)$
  - k is cofactor
  - GCD (k, r) = 1
- Base point $G \in E(F_q)$ of order r

Generating EC Domain Parameters

1. Select a prime power q
2. Select an elliptic curve E over F_q
   – order $\#E(F_q) = kr$
3. Generate a point G of order r
4. Output $D=(F_q, E(F_q), r, k, G)$ as domain parameters

Domain Parameters

$D=(F_q, E(F_q), r, k, G)$

- Instead of using E and G as system domain parameters, we could fix only the underlying finite field F_p for all users.
- And let each user select her own elliptic curve E and point $G \in E(F_p)$.

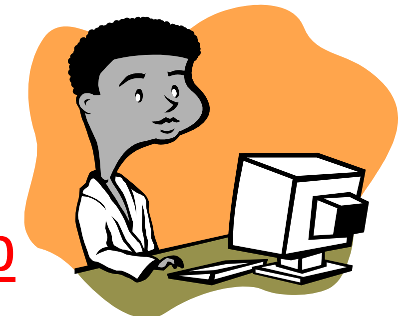
Generating an EC Key Pair

1. Randomly generate $s \in [1, r\text{-}1]$
2. Compute $W = sG$
3. Output $(KU, KR) = (W, s)$

Elliptic Curve Diffie-Hellman: Key Exchange

Alice

- Secretly select a random integer k_A
- Compute $k_A.Q$ and send it to Bob
- Receive $k_B.Q$ from Bob
- Common key $P = k_A.k_B.Q$



Bob

- Receive $k_A.Q$ from Alice
- Secretly select a random integer k_B
- Compute $k_B.Q$ and send it to Alice
- Common key $P = k_B.k_A.Q$

Elliptic Curve Diffie-Hellman: Key Exchange

Common key $P = k_A.k_B.Q$

An eavesdropper whould have to determine $P = k_A.k_B.Q$ knowing Q, $k_A.Q$ or $k_B.Q$ but not k_A or k_B .

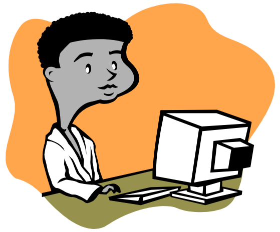
The eavesdropper's task called the

"Diffie-Hellman problem for elliptic curves".

Elliptic Curve Diffie-Hellman: Message Transfer

- Suppose that the set of message units has been imbedded in E in some agreed way to convert integer values.
- Bob wants to send Allice a message m ϵ E .
- Alice and Bob have already exchange $k_A.Q$ and $k_B.Q$ as in Diffie-Hellman key exchange protocol.

Elliptic Curve Diffie-Hellman: Message Transfer



Ciphering the message:

- Bob, chooses another secret random integer l and sends Alice the pair of points

$$(l.Q\ ,\ M+l.(k_A.Q))$$

Deciphering the message:

- Alice, multiplies the first point in the pair by her secret key k_A

$$(k_A\ l.Q)$$

and then subtracts the result from the second point in the pair

$$M + l.(k_A.Q) - (k_A\ l.Q) = M$$

The Diffie-Hellman system can be broken if someone can solve the “discrete logarithm problem” in the group E.

Elliptic Curve Encryption System

System Entities

- Finite field F_q
- E is an elliptic curve over F_q with a point P lying in $E(F_q)$
- $\#E(F_q) = kr$
- $D=(F_q , E(x), P, [1, r-1])$

Digital Signature : The Motivation

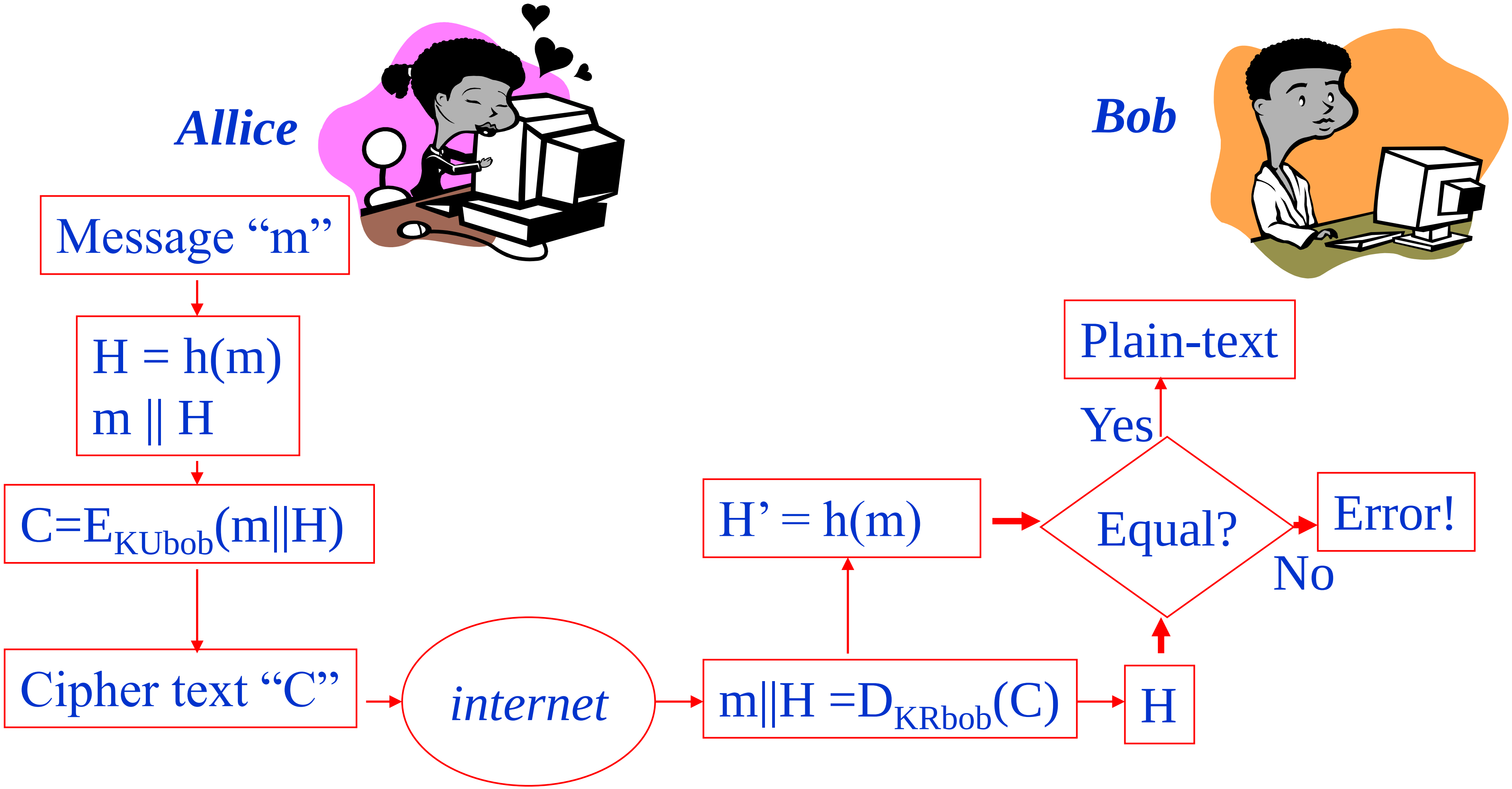
- How Bob can be sure the message come from Allice !!

“Authentication”

- How Bob can be sure the message did not changed !!

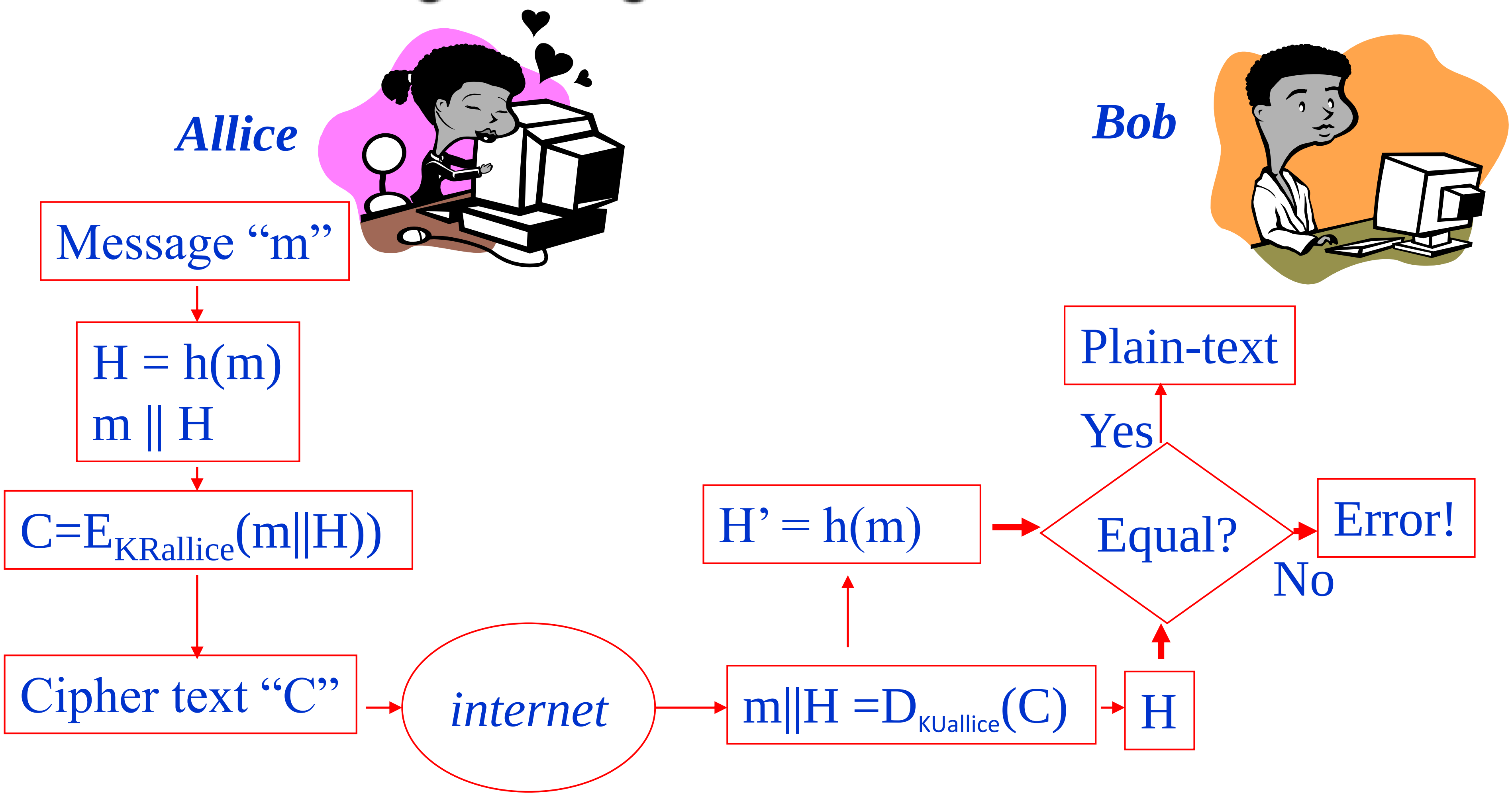
“Integrity”

Digital Signature: Secrecy



Integrity !!??

Digital Signature: Authentication



Integrity= Authentication !!

With PKI infrastructure= Ensure of ID’s of sides

Elliptic **C**urve **D**igital **S**ignature **A**lgorithm: Key Generation

- Let E be an elliptic curve over F_p
- Let P be a point of prime order q in $E(F_p)$
  - These are the system domain parameters.
  - In the ECDSA, q is about the same size as p.
- Each user, selects a random integer x in the interval $1 < x < q-1$ and computes $Q = x.P$
- Q is the public key “KU” and x is the private key “KR”.

The key pair (KU,KR) = (Q, x)

Elliptic **C**urve **D**igital **S**ignature **A**lgorithm: Key Generation

- Define equation for E, the coordinates of the point P, and the order q of P must be included in the user's public key.

Public Key (F_p, E, P, q, Q)

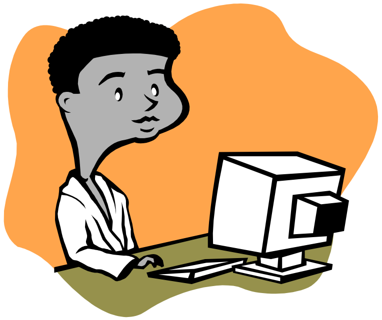
- At the same time, there are an enormous number of choices of elliptic curve E over the finite field F_p.

Elliptic Curve Digital Signature Algorithm: Signature Generation

To sign the message m, Alice does the following.

1. Select a random integer k in interval $1 < k < q - 1$
2. Computes $k.P = (x_1, y_1)$ and $r = x_1 \bmod q$
   - $0 < x_1 < p - 1$
   - r is taken to be its least non-negative residue modulo q
   - If $r = 0$ then she return to step 1.
3. Computes $k^{-1} \bmod q$
4. Computes $s = k^{-1}(H(m) + x.r) \bmod q$
   - H(m) is the hash value of the message.
   - If $s = 0$ then she return to step 1.
5. The signature for the message m is the pair of integer (r,s)

Elliptic Curve Digital Signature Algorithm: Signature Verification



Bob should do the following, to verify Alice's signature (r,s);

1. Obtain an authenticated copy of Alice's public key Q.
2. Verify that r and s are integers in the interval [1, q-1].
3. Computes $w = s^{-1} \bmod q$ and H(m)
4. Compute $u_1 = H(m).w \bmod q$ and $u_2 = r.w \bmod q$
5. Compute $u_1.P + u_2.Q = (x_0, y_0)$ and $v = x_0 \bmod q$
6. Accept the signature if and only if $v = r$

Proof that signature verification works!

If a signature (r,s) on a message m was indeed generated by Alice then

- $s = k^{-1}.(\ H(m) + x.r\ ) \bmod q$
- ($w = s^{-1} \bmod q$, $u_1 = H(m).w \bmod q$ and $u_2 = r.w \bmod q$)
- $k \equiv s^{-1}.(\ H(m) + x.r) \equiv s^{-1}.H(m) + s^{-1}.x.r \equiv w.H(m) + w.r.x$

  $\equiv u_1 + u_2.x \bmod q$
- ($Q = x.P$, $KU = Q$ and $KR = x$ and $u_1.P + u_2.Q = (\ x_0, y_0)$ and $v = x_0 \bmod q$)
- Thus $u_1.P + u_2.Q = (\ u_1 + u_2.x).P = k.P$,

  $k.P = (x_1, y_1)$ and $r = x_1 \bmod q$

  and so $v = r$ as required.