Cryptography and its Applications Part II

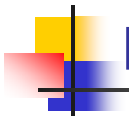
Recall Symmetric Crypto

- When two parties want to communicate securely using symmetric cryptosystems:
  - (1) Alice and Bob agree on a cryptosystem and on a key
  - (2) Alice encodes plaintext using this key
  - (3) Alice sends resultant ciphertext to Bob
  - (4) Bob receives and decrypts ciphertext using key
- The key distribution problem of secret key systems
  - Must share the secret key with other party before initiating communication. Key distribution done secretly (difficult when parties are geographically distant, or don't know each other)
  - If you want to communicate with n parties, you require n different keys
  - Need a key for each pair of users
    - 10 users need 10*(10-1)/2=45 keys,
    - 18 users need 153 keys.

A revolution of sorts

- Diffie & Hellman sought to solve 2 problems:
  1. Find a better way to *distribute keys*
  2. provide for a *digital* document *signature*
- public key encryption is based on mathematical functions, not on substitution and permutation
- asymmetric -- two different keys, one kept private, one made public, generated by the principal
  - Things encrypted with the private key may only be decrypted with the corresponding public key
  - Things encrypted with the public key may only be decrypted with the corresponding private key
- Succeeded only partially (1)

3



Diffie-Hellman Key Exchange

- Proposed in 1976; first public key algorithm (predates RSA)
- Allows a group of users to agree on a secret (session) key over an insecure channel. Not used to encrypt messages
- Suppose Alice and Bob want to agree on a shared key
- They agree on two large integers n and g such that $1<g<n$ (these will be shared by every member of a group)
- No prior communication between Alice and Bob needed
- Security depends on the difficulty of computing the private key privA given the public key pubA = g^{privA} mod n (discrete logarithm problem, similar, but not identical, to factorization)
- Choices for g and n are critical:
  - both n and (n–1)/2 should be prime,
  - n large (at least 512 bits, possibly 1024 bits),
  - g is a primitive root mod n (i.e., $x^{\phi(n)} = 1 \bmod n$)

4

Diffie-Hellman Key Exchange

- Alice chooses a random privA (private key), computes (the public key) pubA = g^{privA} mod n, and sends it to Bob
- Bob chooses a random privB (private key), computes (the public key) pubB = g^{privB} mod n, and sends it to Alice
- Alice computes k = pubB ^{privA} mod n
- Bob computes k' = pubA ^{privB} mod n
- Note that

k= pubB ^{privA} mod n = g $^{privA\ privB}$ mod n= pubA ^{privB} mod n =k' and thus Alice and Bob now shared a session key

- If someone is listening, he/she knows n, g, pubA, and pubB, but not privA and privB

5

Diffie-Hellman Key Exchange

- Susceptible to intruder-in-the-middle attack:
  - Mal notices Alice sending Bob her public key K_{PubA} , and intercepts it.
  - Mal then sends Bob his public key: K_{PubM} , claiming it is Alice's
  - Bob now sends Alice his public key. Again, this is intercepted by Mal, who substitutes K_{PubM}
  - When Alice sends a message to Bob, she will use K_{PubM}
  - Mal can intercept this and read it (with his own private key)
  - Mal can generate an 'appropriate' substitute message, encode it with Bob's public key K_{PubM} , and send the new message to Bob

active malicious intruder

6

Public Key Cryptosystems

- Public key cryptosystems solve the key distribution problem for secret key systems (if a reliable channel for communication of public keys can be implemented)
- Requires the **reliable** (not secret) dissemination of one public key per party and thus scales well for large systems
- Concept conceived by Diffie and Hellman in 1976 (discovered by J.Ellis (UK CESG) in 1970 but classified)
- Rivest, Shamir and Adleman (RSA) were first to describe a public key system in 1978
- Merkle and Hellman published a different solution, later in 1978
- Many proposals have been broken (including the 1978 Merkle-Hellman proposal broken by Shamir)
- Serious candidates today (in public domain)
  - RSA
  - El Gamal
  - Elliptic curve

7

Notation

C = E(KU,M) also $E_{KU}(M)$

M = D(KR,C) also $D_{KR}(C)$

KU : Public (encryption) key, known to all
KR : Private (decryption) key, known only to B
E : Encryption Algorithm
D : Decryption Algorithm
M : Plaintext Message
C : Ciphertext Message

8

Two possible uses

- confidentiality
  - A wants to send message to B
  - A encrypts the message with B's public key
  - A sends the encrypted message to B
  - B decrypts the message with its private key
- authentication, or digital signature
  - A wants to send a message to B so that B is assured that A (and no one else) sent it
  - A encrypts the message with A's private key
  - A sends the encrypted message to B
  - B decrypts the message with A's public key
  - B then knows that only A could have sent it

9

Requirements for Public Key

- Computationally EASY to
  - generate a pair of keys (public KU, private KR)
  - encrypt, given the key KU and the message M
  - decrypt, given the key KR and the encrypted message C
- Computationally INFEASIBLE to
  - determine the private key KR, knowing the public key KU
  - recover the original message (M), given public key KU and the ciphertext C, for the message M
- To make this computation not feasible, key size is no smaller than 512 bits

10

RSA

- public key KU is (n,e)
- secret key KR is (n,d)
- n is (at least) a 200 digit number
- message M is represented as an integer from 0 to n-1
- $C = M^e \bmod n$
- $M = C^d \bmod n$

Why should it be the case that if M is a plaintext and C is a ciphertext and $C = M^e \bmod n$, that

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

How do we know that *there even exist* e and d such that $M^{ed} \bmod n = M$?

11

Modular Arithmetic

- We say that a is congruent to b modulo n, written $a = b \bmod n$, if $a - b = k*n$ for some integer k
- If $b < n$, b is also called the residue of a modulo n
- $a^{-1} = x \bmod n$ if $a*x = 1 \bmod n$
- $a^{-1} = x \bmod n$ has a unique solution if a and n are relatively prime

Examples

$12 = 2 \bmod 5$; $2 = 12 \bmod 10$; $12 = 0 \bmod 6$

Properties

$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
$(a - b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$
$(a * b) \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$
$(a * (b + c)) \bmod n = ((a*b) \bmod n) + (a*c) \bmod n)) \bmod n$

We exploit these properties when we calculate $a^x \bmod n$

$a^{16} \bmod n = (((a^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n$

12

Algorithm: exponentiation by repeated squaring and multiplication

- Computing M^e (mod n) takes at most $2*\log_2(e)$ multiplications and $2*\log_2(e)$ divisions
- Step 1. Let $e_k, e_{k-1}, \ldots e_1, e_0$ be binary rep. of e
- Step 2. Set the variable C to M
- Step 3. Repeat 3a and 3b for i=k-1,…,0 :
  - Step 3a. Set C to the remainder of C^2 when divided by n
  - Step 3b. If $e_i = 1$, then set C to the remainder of C*M when divided by n
- Step 4. Halt. Now C is the encrypted form of M

13

Theory behind RSA

Theorem (Euler and Fermat)
If p, q primes , n = p*q , and if gcd(x,n)=1 then:

$x^{\phi(n)} = 1 \bmod n$

for this choice of p and q, $\phi(n)=(p-1)*(q-1)$

If $e*d=1+q*\phi(n)$ (e and d inverses mod $\phi(n)$)

then

$C^d = M^{e*d} = M^{1+q*\phi(n)} =$

$M^1 *(M^{\phi(n)})^q = M^1*(1)^q = M^1 \bmod n = M$

14

How to find large primes

- 100-digit to 200-digit primes recommended
- large primes can be found efficiently using probabilistic algorithms due to Solvay and Strassen: generate odd 100-digit random numbers until a prime is found
  - about 115 will be tested by prime number theorem

15

Generation of RSA Keys

- choose 2 large (100 digit) primes p and q
  - care must be exercised in choosing p and q, otherwise insecurities may result (p-1, p+1, q-1, q+1 should have large prime factors)
- compute n = p * q
- choose e relatively prime to (p-1)*(q-1)
- compute d so that e*d = 1 mod (p-1)*(q-1)
  (i.e., d = e^{-1} mod (p-1)*(q-1) is the inverse of e)
  - If the factorization of n into p*q is known, this is easy to do.
- publish (n,e)
- keep (n,d) secret (and destroy p and q)
  - How hard is it to compute d given only (n,e)?
  - Not known. But it is not harder than factoring n into p*q.
  - Therefore the security of RSA is no better than complexity of the factoring problem

16

RSA Keys —Example

- choose 2 large (100 digit) prime numbers p and q
  p = 47, q = 71
- compute n = p * q
  n = p*q = 3337
- choose e relatively prime to (p-1)*(q-1)
  for example e = 79 has no factors in common with
  (47-1) * (71-1) = 46 * 70 = 3220
- compute d = e^{-1} mod (p-1)*(q-1) = 79^{-1} mod 3220 = 1019
  (the inverse of a number modulo n can be computed using the extended Euclidean algorithm)
- publish (3337, 79)
- keep d=1019 , p=47 , q=71 secret

17

Example: Confidentiality

Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$

- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - 07^{17} mod 77 = 28
  - 04^{17} mod 77 = 16
  - 11^{17} mod 77 = 44
  - 11^{17} mod 77 = 44
  - 14^{17} mod 77 = 42
- Bob sends 28 16 44 44 42 received by Alice
- Alice uses private key, $d = 53$, to decrypt message:
  - 28^{53} mod 77 = 07
  - 16^{53} mod 77 = 04
  - 44^{53} mod 77 = 11
  - 44^{53} mod 77 = 11
  - 42^{53} mod 77 = 14
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

18

Integrity/Authentication

Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$

- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49 received by Bob
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key; if (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

19

Example: Both

- Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)
  - Alice's keys: public (17, 77); private: 53
  - Bob's keys: public: (37, 77); private: 13
- Alice enciphers HELLO (07 04 11 11 14):
  - $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  - $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- Alice sends 07 37 44 44 14

20

Practical aspects of RSA

- today's computers cannot directly handle numbers larger than 32- or 64-bits
- need multiple precision arithmetic that requires libraries to handle large numbers
- RSA Key Size
  - key size should be chosen conservatively
  - cryptographers can stay ahead of (factorization) cryptanalysts by increasing the key size
  - Until 1989, factorization attacks based on "high school mathematics." Then sophisticated attacks have extended factorization to larger numbers (usually of a specific form). At present it appears that 130 digit numbers can be factored in several months using lots of idle workstations.

21

"Famous" RSA Cracking Attempts

- Breaking the RSA key requires factoring or brute force...
  - RSA inventors offered $100 reward for finding a plaintext sentence enciphered via RSA-129
  - 129 digit modulus is approx. 429 bit binary number
  - RSA predicted 40 quadrillion years was needed
  - 1993: Lenstra (Bellcore) and Atkins (MIT) attempted the 1977 RSA factoring challenge
  - 1600+ workstations * eight months = success !
  - A particular private key was identified that matched the public key.
  - Reward for cracking code given to Free Software Foundation (Richard Stallman)
- Blacknet Key attack
  - Muffett, Leyland, Lenstra and Gillogly managed to use enough computation power (approx. 1300 MIPS) to factor the key in 3 months.
  - Used to decrypt a publicly-available message encrypted with that key.
  - Attack done in secrecy
- RSA-640 cracked announced Nov 2005
- RSA-704 still not factored (2010) ~~30K$~~ (only 212 decimal digits!)
- RSA-768 cracked 12 Dec 2009

Challenge no longer active

22

RSA Versus DES

- fastest implementations of RSA can encrypt kilobits/second
- fastest implementations of DES can encrypt megabits/second
- this 1000-fold difference in speed is likely to remain independent of technology advances
- it is often proposed that RSA be used for secure exchange of DES keys
- the key size of DES is 64 bits (56 bits plus 8 parity bits)
- key size of RSA is selected by the user
  - Casual use 384 bits
  - Commercial use 512 bits
  - Military use 1024 bits
- many implementations choose n to be 154 digits (512 bits) so the key (n,e) is 1024 bits

23

El Gamal

- A variant of the Diffie-Hellman key distribution scheme, allowing secure exchange of messages
- Published in 1985 by T.ElGamal
- Like Diffie-Hellman its security depends on the difficulty of computing discrete logarithms
- Key Generation
  - Select a large prime p (~200 digit), and value g a primitive element mod p
    - Bob has a secret number $Priv_B$
    - Bob compute $Pub_B = g^{PrivB} \bmod p$ which is made public

24

El Gamal

To send a message to Bob

- To **encrypt** a message **M** into ciphertext **C**
  - Selects a random number r, 0 < r < p
  - Computes the message key K = $Pub_B{}^r$ mod p
  - Compute the ciphertext pair: C = (c1,c2)
    - $c_1 = g^r$ mod p, c_2 = K*M mod p
- To **decrypt** the message C
  - Extract the key **K=**c_1^{PrivB} mod p = $g^{r*PrivB}$ mod p
  - Extracts **M** by solving for M in the equation

    c_2 = K * M mod p

25

Elliptic Curve Cryptography

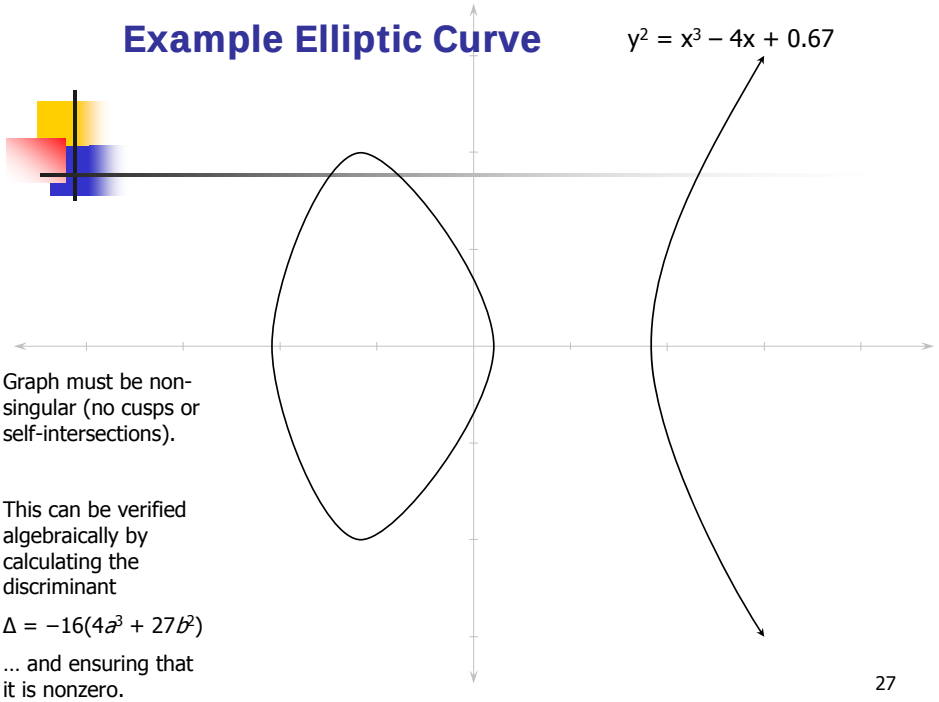
- majority of public-key crypto use integer/polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller key sizes
- an elliptic curve is defined by an equation in two variables x and y, with real coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define a "zero point" O
- More info

  http://www.ruhr-uni-bochum.de/itsc/tanja/summerschool/slides.html

26

Example Elliptic Curve

$y^2 = x^3 - 4x + 0.67$



Graph must be non-singular (no cusps or self-intersections).

This can be verified algebraically by calculating the discriminant

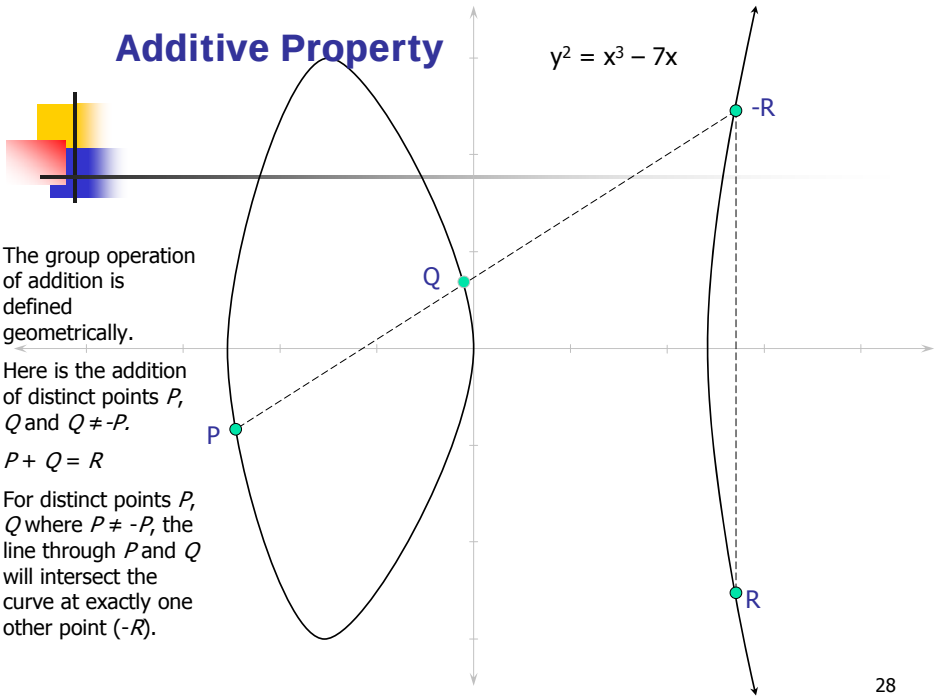
$\Delta = -16(4a^3 + 27b^2)$

... and ensuring that it is nonzero.

27

Additive Property

$y^2 = x^3 - 7x$



The group operation of addition is defined geometrically.

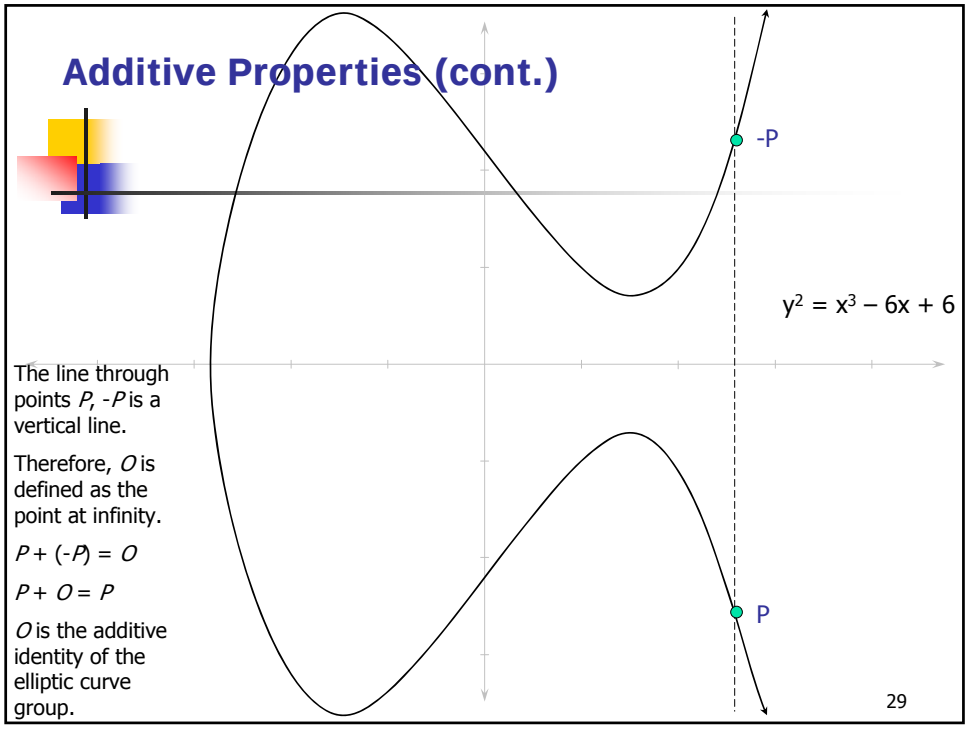
Here is the addition of distinct points *P*, *Q* and $Q \neq -P$.

$P + Q = R$

For distinct points *P*, *Q* where $P \neq -P$, the line through *P* and *Q* will intersect the curve at exactly one other point (*-R*).

28

Additive Properties (cont.)



The line through points P, $-P$ is a vertical line.

Therefore, O is defined as the point at infinity.

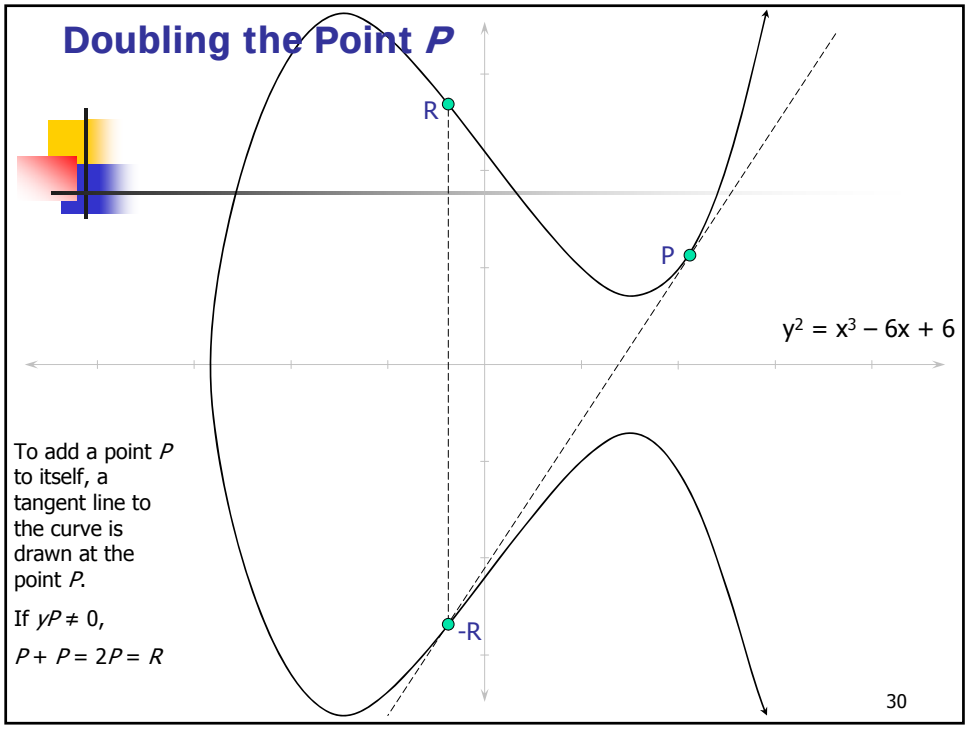
$P + (-P) = O$

$P + O = P$

O is the additive identity of the elliptic curve group.

29

Doubling the Point *P*



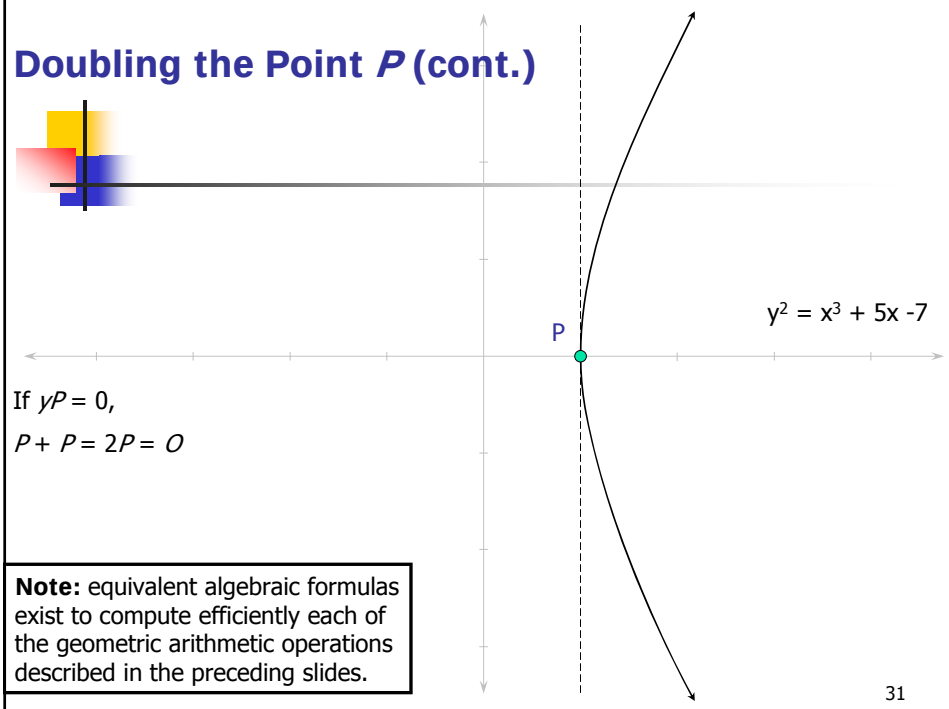
To add a point P to itself, a tangent line to the curve is drawn at the point P.

If $yP \neq 0$,

$P + P = 2P = R$

30

Doubling the Point *P* (cont.)



If $yP = 0$,

$P + P = 2P = O$

Note: equivalent algebraic formulas exist to compute efficiently each of the geometric arithmetic operations described in the preceding slides.

31

Elliptic Curve Groups over Finite Fields

- To provide fast and precise arithmetic for cryptographic applications, finite fields are used in place of the real numbers.
- For example, $y^2 \bmod p = x^3 + ax + b \bmod p$ has an underlying field of F_p if a and b are in F_p.
- The algebraic formulas derived from the geometric arithmetic of elliptic curves over real numbers can be adapted for elliptic curves over finite fields.

32

Algebraically

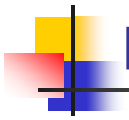
Adding distinct points P and Q

- When $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are not negative of each other, $P + Q = R$ where
  $s = (y_P - y_Q) / (x_P - x_Q)$
  $x_R = s^2 - x_P - x_Q$ and $y_R = -y_P + s(x_P - x_R)$
  Note that s is the slope of the line through P and Q.

Doubling the point P

- When y_P is not 0,
  $2P = R$ where
  $s = (3x_P^2 + a) / (2y_P)$
  $x_R = s^2 - 2x_P$ and $y_R = -y_P + s(x_P - x_R)$
  Recall that a is one of the parameters chosen with the elliptic curve and that s is the tangent on the point P.

33



Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need "hard" problem equiv. to discrete log
  - Q=kP, where Q,P belong to a prime curve
  - It is "easy" to compute Q given k,P
  - but "hard" to find k given Q,P
  - known as the elliptic curve logarithm problem

34

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_p(a,b)$
- select base point $G=(x_1,y_1)$ with large order n s.t. $n \times G = 0$
- A & B select private keys $priv_A < n$, $priv_B < n$
- compute public keys: $Pub_A = priv_A \times G$, $Pub_B = priv_B \times G$
- compute shared key: $K = priv_A \times P_B$, $K = priv_B \times P_A$
  - same since $K = (priv_A * priv_B) \times G$

35

ECC Encryption/Decryption

- several alternatives, will consider simplest
- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $priv_A < n$
- compute public key $Pub_A = priv_A \times G$
- encrypt P_m : $C_m = \{k \times G,\ P_m + k \times Pub_B\}$, k random
- decrypt C_m by computing:

$$P_m + k \times Pub_B - priv_B \times (k \times G) =$$
$$P_m + k \times (priv_B \times G) - priv_B \times (k \times G) = P_m$$

36

ECC Security

- relies on elliptic curve logarithm problem
- compared to factoring, can use much smaller key sizes than with RSA
  - 160 bit ECC public key claimed to be equivalent to 1024 bit RSA public key
- for equivalent key lengths, computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

37

Comparison of Public-key crypto

key comparison symmetric vs. asymmetric cryptosystems

symmetric key size	ECC over Zp size of p	ECC over GF(2^n) size of n	RSA modulus size
80	192	163	1024
112	224	233	2048
128	256	283	3072

comparison between RSA and Elliptic Curve Cryptography for comparable security levels

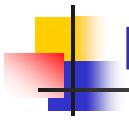
	1024-bit RSA	163 bit ECC
certificate size key and signature	over 256 bytes	over 62 bytes
key generation (ms)	285, 630	397
signature generation (ms)	20,208	528
signature verification (ms)	900	1,142

38

Message Authentication

- **message authentication** is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- three alternative functions used:
  - message encryption (as for secrecy)
  - message authentication code (MAC)
  - hash function

39



Message Encryption

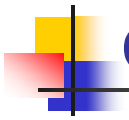
- message encryption, used mainly for confidentiality, also provides a measure of authentication
- if symmetric encryption is used then:
  - receiver knows sender must have created it since only sender and receiver know key used
  - content cannot have been altered
  - if message has suitable structure, redundancy or a checksum used to detect any changes

40

Message Encryption

- if public-key encryption is used:
  - encryption provides no confidence in the sender since anyone potentially knows public-key
  - however if
    - sender **signs** (encrypts) message using his/her private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

41



Cryptographic Checksums

- Also known as Message Authentication Code MAC or hash functions
- Mathematical function to generate a set of k bits from a set of n bits (where $k \leq n$).
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is "parity"
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

42

Definition

Cryptographic checksum h: $A \rightarrow B$:

1. For any $x \in A$, $h(x)$ is easy to compute
2. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
3. It is computationally infeasible to find two inputs x, $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$
   - Alternate form (stronger): Given any $x \in A$, it is computationally infeasible to find a different $x' \in A$ such that $h(x) = h(x')$.

43

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have 2 objects in it.
  - Application: if there are 25 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

44

MAC Properties

- a MAC is the value of a cryptographic checksum
  - condenses a variable-length message M to a fixed-sized authenticator using a secret key K
- cryptographic checksum is a many-to-one function
  - potentially many messages may have the same MAC but finding these needs to be very difficult
- needs satisfy the following:
  - knowing a message and MAC, it is unfeasible to find another message with same MAC
  - MACs should be uniformly distributed
  - MAC should depend equally on all bits of the message

45

Symmetric Ciphers for MACs

- can use any block cipher in chaining mode and use the final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits (16≤M≤64) of final block
- but final MAC is now too small for security

46

Hash Functions

- condenses an arbitrary message to a fixed size message digest
- usually assumed that the hash function is public and not keyed (unlike MAC which is keyed)
- hash used to detect changes to message
- can be used in various ways with message
- most often to create a digital signature

One-way Hash Functions

- **WEAK** Given M and H(M) it should be difficult to find M' such that H(M')=H(M)
  - (weak collision resistance)
- **STRONG** It should be difficult to find any two M1 and M2 such that H(M1)=H(M2)
  - (strong collision resistance)

47

One-way Hash Functions

- Authenticity of a message can be checked by computing H(M)=h, and comparing with the transmitted h
- This requires that either
  - h be transmitted over a more secure channel than M, e.g., M is a disk transmitted by mail, h is transmitted via telephone, or
  - h be digitally signed (which may be easier than signing M)

48

Chances of Success

- Hash function generates 64-bit digest ($n = 2^{64}$), randomly distributed and diffused
- Chance that a randomly chosen message maps to a given hash value is 1 in n or 2^{-64} : seems secure
- but by **birthday attack** it is not: (digest of size m)
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- Conclusion: strong hash functions should produce at least 128 bits

49

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i}[H_{i-1}]$
  - and use final block as the hash value
  - similar to DES-CBC but without a key
- resulting hash is too small (64-bit)
  - due to direct birthday attack
- other variants also susceptible to attack

50

Current Generation Non-keyed Message Digest Algorithms

- **MD5 (Message Digest 5)** by R.Rivest
  - 128 bit message digest
  - falling out of favor
  - 64 bits birthday attack
- **SHA-1 (Secure Hash Algorithm)** by NIST
  - 160 bit message digest
  - slightly slower than MD5 but more secure
  - 80 bits birthday attack (65K longer time)(broken Feb 2005, with collision in 2**69 instead of 2**80)
  - After 2010 usable only for HMACs, KDFs and RNGs
- **SHA-2 Family** by NIST (2006) not used much
  - 224, 256, 384 or 512 bit message digest
- **SHA-3** under development by NIST (exp. 2012)

51

Current Generation MAC

- HMAC-MD5, HMAC-SHA
  - IETF standard
  - general technique for constructing a MAC from a message digest (unkeyed) algorithm
- Older MACs are based on secret key encryption algorithms (notably DES) and are still in use
  - DES based MACs are 64 bit and not considered strong anymore

52

HMAC

- Make keyed cryptographic checksums using keyless cryptographic checksums
- *h* keyless cryptographic checksum function that takes data in blocks of *b* bytes and outputs blocks of *l* bytes. *k′* is cryptographic key of length *b* bytes
  - If short, pad with 0 bytes; if long, hash to length *b*
- *ipad* is 00110110 repeated *b* times
- *opad* is 01011100 repeated *b* times
- HMAC-*h*(*k*, *m*) = *h*(*k′* ⊕ *opad* || *h*(*k′* ⊕ *ipad* || *m*))
  - ⊕ exclusive or, || concatenation

53

Key Points

- Two main types of cryptosystems: classical and public key
- Classical cryptosystems encipher and decipher using the same key
  - Or one key is easily derived from the other
- Public key cryptosystems encipher and decipher using different keys
  - Computationally infeasible to derive one from the other
- Cryptographic checksums provide a check on integrity

54