Lecture 6:
Public Key Encryption

Stephen Huang

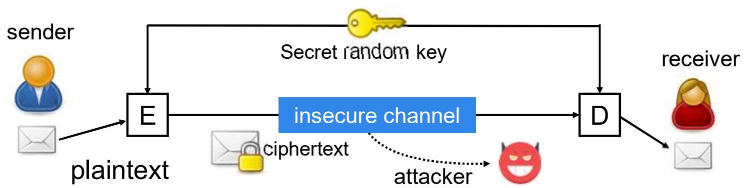
Content

1. Public-Key Cryptography
2. RSA Encryption
3. ElGamal Encryption
4. Elliptic Curve Cryptography
5. Conclusion of Encryption

A **passive** attack attempts to learn or make use of information from the system but does not affect system resources.

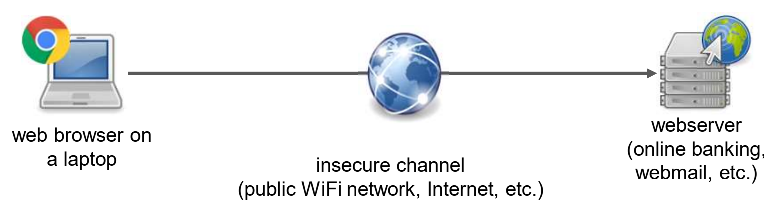
An **active** attack attempts to alter system resources or affect their operation.

Secret-Key Encryption



- Sender and receiver know the secret key → can encrypt/decrypt
- Attacker does not know the secret key → cannot encrypt/decrypt
- Exchanging or agreeing on a key
  - either using a secure side channel
  - or before communicating over the insecure channel

Practical Problem: Key Exchange



- Key Exchange: How or when can the two endpoints exchange a secret key?

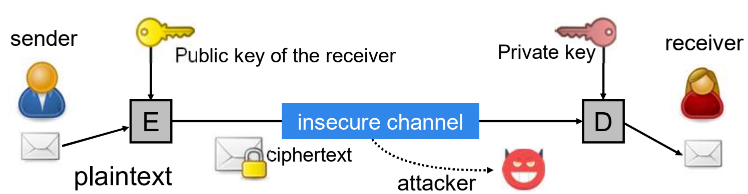
From Classical to Modern Era

- Both the RSA algorithm and the Diffie-Hellman key exchange algorithm were introduced in 1977.
- These new algorithms were revolutionary because
  - they represented the first viable cryptographic schemes where security was based on the **theory** of numbers;
  - it was the first to enable secure communication between two parties **without** a shared secret.

1. Public-Key Cryptography

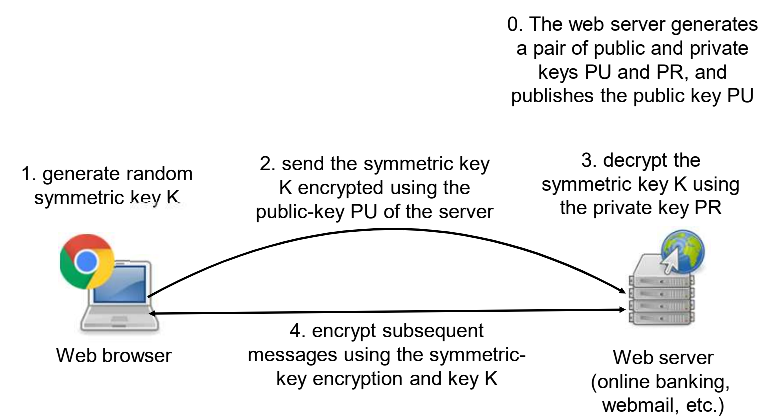
- In 1976, Whitfield Diffie and Martin Hellman proposed a fundamentally different approach to cryptography
  - The first documented discovery was made by the British intelligence agency in 1970
- Public-key cryptography: instead of using a single secret key, use a pair of private and public keys
  - also called asymmetric-key cryptography
- Only the private key needs to be secret; the public key does not
- Public-key cryptography solves multiple problems
  - public-key encryption → key exchange
  - digital signatures → non-repudiation (cannot deny)

Public-Key Encryption



- Everyone knows the public key → sender can encrypt
- Receiver knows the private key → receiver can decrypt
- Attacker does not know the private key → attacker cannot decrypt
- Public key can be published
  - attacker may know the public key

Application Example



- Secure against passive attacks

Public-Key Encryption Scheme

- A public-key encryption system is a triplet of algorithms (G, E, D)
  - Key generation G(): randomized algorithm, outputs (PU, PR)
  - Encryption E(PU, M): takes public key PU and plaintext M, outputs ciphertext C
  - Decryption D(PR, C): takes private key PR and ciphertext C, outputs plaintext M
- Requirements
  - for every (PU, PR) that was output by G, D(PR, E(PU, M)) = M
  - G is efficiently computable, E is efficiently computable given PU and M, and D is efficiently computable given PR and C
  - given only PU and C, an attacker cannot efficiently compute M

Symmetric vs. Asymmetric-Key

	Symmetric-key encryption	Asymmetric-key encryption
Typical design	series of substitutions and permutations	hard mathematical problems
Key	completely random	special structure, expensive to generate
Recommended key size	128 - 256 bits	2048 - 15360 bits
Performance	fast	slow

2. RSA Encryption

- Developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman
  - in 1973, Clifford Cocks, an English mathematician working for a British intelligence agency, described an equivalent system (however, this was classified until 1997)
- For their work on public-key cryptography, Rivest, Shamir, and Adleman received a Turing Award in 2002.
- One of the most widely accepted and implemented general-purpose approach to public-key encryption.
- Idea
  - represent fixed-length plaintext **P** and ciphertext **C** as numbers
  - encryption: $C = P^e \bmod n$
  - decryption: $P = C^d \bmod n$, where private key **d** is such that $(P^e)^d = P$

RSA Mathematical Background

- Prime: an integer $p > 1$ is a prime number if its only positive divisors are 1 and p.
- Greatest common divisor: $\gcd(a, b)$ of integers a and b is the largest positive integer c that is a divisor of both a and b.
  - a and b are relatively prime if $\gcd(a, b) = 1$
  - if a and m are relatively prime, then a has a multiplicative inverse a^{-1} in modulo m.
- <u>Integer factorization problem</u>: decompose a non-prime number into a product of smaller integers.
  - widely believed to be a computationally hard problem (cannot be solved efficiently, i.e., in polynomial time).
  - however, this hardness has not been proven.

What is a Totient

1. total + quotient -> totient
2. For an integer $n \geq 1$, $\varphi(n)$ is the number of relative prime numbers less than n.
3. We consider 1 to be relative prime to all integer numbers.
4. $\boxed{\varphi(n) = n-1}$ if n is a prime.
5. The totient function is multiplicative: $\boxed{\varphi(ab) = \varphi(a)\,\varphi(b)}$ if gcd(a,b)=1.

$\varphi(n)$ for $1 \leq n \leq 100$

+	1	2	3	4	5	6	7	8	9	10
0	1	1	2	2	4	2	6	4	6	4
10	10	4	12	6	8	8	16	6	18	8
20	12	10	22	8	20	12	18	12	28	8
30	30	16	20	16	24	12	36	18	24	16
40	40	12	42	20	24	22	46	16	42	20
50	32	24	52	18	40	24	36	28	58	16
60	60	30	36	32	48	20	66	32	44	24
70	70	24	72	36	40	36	60	24	78	32
80	54	40	82	24	64	42	56	40	88	24
90	72	44	60	46	72	32	96	42	60	40

RSA Key Generation

1. Pick 2 large and random prime numbers, p and q, $p \neq q$.
2. Calculate n = p×q.
3. Calculate Euler's totient function $\varphi(n) = (p - 1)\times(q - 1)$.
4. Pick e such that $\gcd(e, \varphi(n)) = 1$ and $1 < e < \varphi(n)$.
5. Calculate d, so that $d\times e = 1 \bmod \varphi(n)$
   (d is the multiplicative inverse e^{-1} of e in mod $\varphi(n)$)
6. Let the public key pair be PU = (e, n)
7. Let the private key pair be PR = (d, n)

RSA Key Generation

1. Select two prime numbers (P, Q)
2. Calculate Produce N = P×Q
3. Calculate Totient, $\varphi(n)$ T = (P-1)×(Q-1)
4. Select a Public Key E,
   - It must be prime
   - It must be less than the totient
   - It must NOT be a factor of the totient
5. Select a private key D
   - The product of D and E, divided by T, must result in a remainder of 1, (D×E) mod T = 1.

Example

1. Pick p = 7, q = 13, n = pq = $\boxed{7 \text{ x } 13 = 91.}$
2. Compute (p-1)(q-1) = 72.
3. Select e = 5, for 5 and 72 are relative prime.
4. Select d = 29, d x e = 145 mod 72 = 1.
5. Public key (n, e) = (91, 5)
6. Private key (n, d) = (91, 29)

RSA Encryption and Decryption

- The sender represents the plaintext as a series of numbers < n. Encrypt/decrypt each number separately.
- Encryption: Given a plaintext M (M<n),

  $C = M^e \bmod n$
- Decryption: Given a ciphertext C (C<n)

  $M = C^d \bmod N$
- Euler's Theorem: if a and n are relative prime, then $a^{\varphi(n)} = 1 \bmod n$.
- Prove that the decryption is correct.

Example

- Continue from the last one.
- Public key (n, e) = (91, 5)
- Private key (n, d) = (91, 29)
- Encryption:
  M = 10, C = p^e mod n = 10^5 mod 91 = 82
- Decryption:
  M = C^d mod n = 82^{29} mod 91 = 10

The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n.

Consistency Proof

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

Since e×d = 1 mod φ(n), we have that

e×d = 1 + φ(n)×i where i is some integer

So, $C^d \bmod n = M^{1+\varphi(n)\times i} \bmod n$

$= M \times M^{\varphi(n)\times i} \bmod n$

$= M \times 1^i \bmod n$

$= M \bmod n$

Example 1

- Encryption

  $C = M^e mod\ N$

  $= 60^{29}\ mod\ 133$

  $= 86$
- Decryption

  $M = C^d\ mod\ N$

  $= 86^{41}\ mod\ 133$

  $= 60$

Name	Symbol	Value
Prime	p	7
Prime	q	19
Product	n	133
Totient	φ(n)	108
Public Key	e	29
Private Key	d	41
Message	M	60
Cipher	C	86

Example 2

- Encryption

$$C = M^{d} \bmod N$$
$$= 60^{41} \bmod 133$$
$$= 72$$

- Decryption

$$M = C^{e} \bmod N$$
$$= 72^{29} \bmod 133$$
$$= 60$$

Name	Symbol	Value
Prime	p	7
Prime	q	19
Product	n	133
Totient	φ(n)	108
Public Key	e	29
Private Key	d	41
Message	M	60
Cipher	C	86

Diffie-Hellman Example

- This is a simplified example to illustrate the concept.

Common Number = 2

Random Number = 4

$2^4 = 16$

32

$32^4 = 1,048,576$

Random Number = 5

$2^5 = 32$

16

$16^5 = 1,048,576$

How Secure is RSA?

- Security lies in the difficulty of factoring semi-prime numbers (133 = 7×19).
  - Try 1909 …
- 1991, RSA Lab released 54 semi-primes of various sizes and asked for factors,
  - As of February 2020, the largest number factored is 829 bits.
  - The 1024-bit number has never been factored.

RSA Factoring Challenge

- RSA Laboratories published a list of RSA moduli in 1991

Number of Bits	Number of Decimal Digits	Year Achieved
330	100	1991
576	174	2003
640	193	2005
768	232	2009

- According to NIST, 15360-bit RSA keys are equivalent to 256-bit symmetric keys in strength

How Secure is RSA?

1024 bit Semi-Prime number:

135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

- 1024 bit RSA keys was recommended standard since 2002
- 2048 bit RSA Keys is recommended standard since 2015

RSA Conclusion

- Security
  - Best-known attack (if implemented properly): integer factorization of the modulus n
  - 829-bit keys have been broken. 1024-bit keys might become breakable soon
  - comparable symmetric-key security (*e.g.*, AES)

Symmetric (*e.g.*, AES)	RSA
80 bits	1024 bits
128 bits	3072 bits
256 bits	15360 bits

- Efficiency: very slow → use it to encrypt a secret key and then switch to symmetric-key encryption

3. ElGamal Encryption

- Proposed by Taher Elgamal in 1984.
- Developed from the public-key cryptographic key exchange proposed by Diffie and Hellman in 1976.
- Security is based on the difficulty of computing discrete logarithms
  - discrete logarithm problem: given *g*, *y*, and *p*, find an *x* that satisfies

$$y = g^x \bmod p$$

  - widely believed to be a computationally hard problem
- Example: Find an integer x for $1 = 2^x \bmod 5$.

Primitive Root modulo n

- A number g is a primitive root modulo n if every number coprime to n is congruent to a power of g modulo n.
- The number 3 is a primitive root modulo 7.

$$\begin{array}{lllllllll}
3^1 & = & 3^0 \times 3 & \equiv & 1 \times 3 & = & 3 & \equiv & 3 \pmod 7 \\
3^2 & = & 3^1 \times 3 & \equiv & 3 \times 3 & = & 9 & \equiv & 2 \pmod 7 \\
3^3 & = & 3^2 \times 3 & \equiv & 2 \times 3 & = & 6 & \equiv & 6 \pmod 7 \\
3^4 & = & 3^3 \times 3 & \equiv & 6 \times 3 & = & 18 & \equiv & 4 \pmod 7 \\
3^5 & = & 3^4 \times 3 & \equiv & 4 \times 3 & = & 12 & \equiv & 5 \pmod 7 \\
3^6 & = & 3^5 \times 3 & \equiv & 5 \times 3 & = & 15 & \equiv & 1 \pmod 7
\end{array}$$

ElGamal Key Generation

1. Pick a large prime q.
2. Pick an integer α such that α is a primitive root of q.
3. Pick a random integer X such that $1 < X < q-1$.
4. Compute $Y = \alpha^X \bmod q$.
5. Let the public key be PU = (q, α, Y)
6. Let the private key be PR = (q, α, X) or simply X.

https://www.ques10.com/p/33937/el-gamal-cryptography-algorithm/

Primitive Root

- A number α is a primitive root if $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{(q-1)}$ are different mod q.

$n = 7$

$3^1 = 1\times3 = 3 \equiv 3 \pmod 7$
$3^2 = 3\times3 = 9 \equiv 2 \pmod 7$
$3^3 = 2\times3 = 6 \equiv 6 \pmod 7$
$3^4 = 6\times3 = 18 \equiv 4 \pmod 7$
$3^5 = 4\times3 = 12 \equiv 5 \pmod 7$
$3^6 = 5\times3 = 15 \equiv 1 \pmod 7$

$\alpha = 1, 2, \ldots, 6$

n = 14, congruence class = {1, 3, 5, 9, 11, 13}

```
 1: 1
 3: 3, 9, 13, 11, 5, 1
 5: 5, 11, 13, 9, 3, 1
 9: 9, 11, 1
11: 11, 9, 1
13: 13, 1
```

$\alpha = 3$ or 5

Encryption and Decryption

- Encryption: given plaintext M (M < q),
  1. Pick a random integer k such that 0 < k < q - 1
  2. Compute $K = Y^k \bmod q$, using public key Y
  3. Let the ciphertext be (C_1, C_2), where
     $C_1 = \alpha^k \bmod q$
     $C_2 = K \cdot M \bmod q$
- Decryption: given ciphertext (C_1, C_2),
  1. compute $K = (C_1)^X \bmod q$, using private key X
  2. compute $M = C_2 \cdot K^{-1} \bmod q$
- Consistency: $K = (C_1)^X = (\alpha^k)^X = (\alpha^X)^k = Y^k = K \bmod q$

From key generation:
$Y = \alpha^X \bmod q$
PU = (q, α, Y)
PR = (q, α, X)

ElGamal Example

- Key generation
  - Pick prime q = 19, primitive root α = 10, and integer X = 5.
  - Compute $Y = \alpha^X = 10^5 = 100000 = 3 \bmod 19$.
- Encryption: given plaintext M = 17
  - Pick k = 6 and compute $K = Y^k = 3^6 = 729 = 7 \bmod 19$.
  - Compute $C_1 = \alpha^k = 10^6 = 1000000 = 11 \bmod 19$
  - Compute $C_2 = K \cdot M = 7 \cdot 17 = 119 = 5 \bmod 19$
- Decryption
  - compute $K = (C_1)^X = 11^5 = 161051 = 7 \bmod 19$
  - compute $K^{-1} = 7^{-1} = 11 \bmod 19$
  - compute $M = C_2 \cdot K^{-1} = 5 \cdot 11 = 55 = 17 \bmod 19$

ElGamal Security and Efficiency

- Computing discrete logarithms is widely believed to be a computationally hard problem.
  - recovering private key X: requires computing the logarithm of Y to base α in modulo q
  - recovering factor k: requires computing the logarithm of C_1 to base α in modulo q
- Efficiency
  - ciphertext is twice as long as the plaintext
  - encryption requires two exponentiations, while decryption requires only one
    → decryption is faster

4. Elliptic Curve Cryptography

- Problem with public-key cryptography based on modular arithmetic.

Symmetric (*e.g.*, AES)	RSA
80 bits	1024 bits
128 bits	3072 bits
256 bits	15360 bits

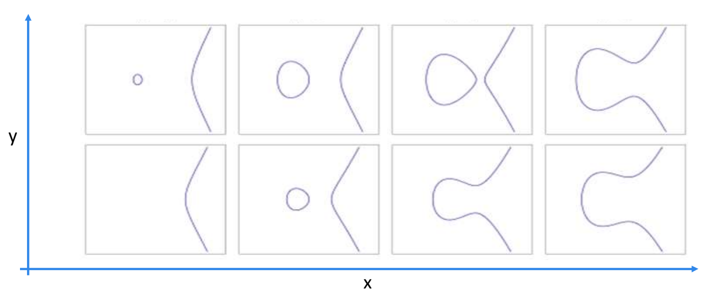
  - very long keys, heavy processing load
- Idea: replace modular arithmetic with operations over elliptic curves.
- Elliptic Curve Cryptography (ECC)
  - First suggested in 1985 but had not been widely used before the mid-2000s.
  - A 160-bit ECC key is comparable in security to a 1024-bit RSA public key.
  - NIST and NSA endorsed ECC as a recommended approach, even for most classified information.

Elliptic Curves

- Elements: points (x, y) that satisfy

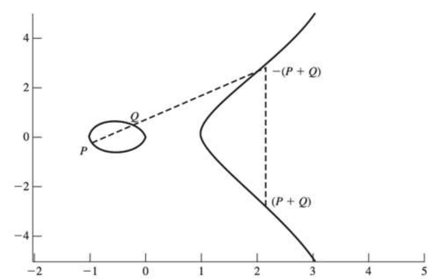
$$y^2 = x^3 + ax + b$$

where x and y are coordinates, a and b are parameters



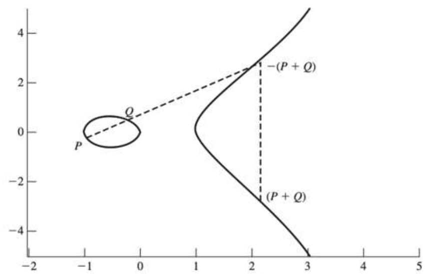
Elliptic Curve Operation

- Operation +
  - operation P + Q: draw a line through P and Q, find the third point of intersection -(P + Q), and mirror that point vertically to get P + Q
  - inverse element -P: mirror point P vertically
  - operation P + P: draw the tangent line and find the another point of intersection, …



Elliptic Curve Operation

- Points of the elliptic curve (and a "point at infinity") with operation + form a commutative group
  - in other words, arithmetic with this operation "works as expected."



Comparison of Key Sizes

Symmetric-key algorithm	RSA	ECC
80	1024	160 - 223
112	2048	224 - 255
192	7680	384 - 511
256	15360	512+

- However, ECC might be more vulnerable to quantum computing attacks

DLP for Elliptic Curves

- *Reminder*: with modular multiplication, it is difficult to find X such that $Y = \alpha^X \bmod q$, given Y, α, and q
  - in other words, it is difficult to determine the "number of operations"
- Discrete Logarithm Problem for elliptic curves: find k such that

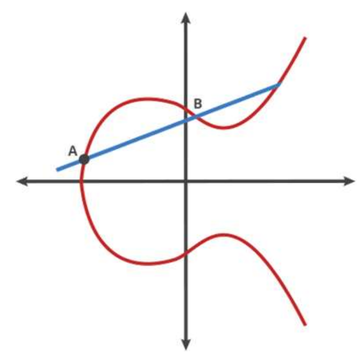
$$Q = k \cdot P,$$

given Q and P, where $k \cdot P = P + P \ldots + P$

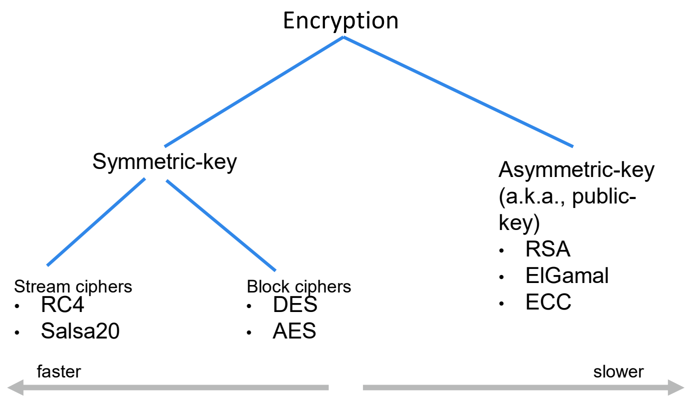
- We can "generalize" ElGamal encryption to elliptic curves in a straightforward manner.

Elliptic Curve

- Horizontal symmetry
- Any non-vertical line will intersect the curve in at most three places.



5. Conclusion of Encryption



Next Topics

- Public-Key Encryption
- Hash Functions
- Integrity