COMP3260/6360 Data Security

Lecture 8

Prof Ljiljana Brankovic

Lecture Overview

1. Public-Key Cryptography

2. RSA
   a) The underlying mathematics
   b) Security of RSA

3. ElGamal Cryptography

4. Diffie-Hellman Key Exchange

Public-Key Encryption

- Chapter 9 Public Key Cryptography and RSA
- Chapter 10, section 10.2 ElGamal Cryptographic System
- Chapter 14, Diffie-Hellman key exchange
- Original paper on RSA by Rivest, Shamir and Adleman

Note that in-text references and quotes are omitted for clarity of the slides. When you write as essay or a report it is very important that you use both in-text references and quotes where appropriate.

Public-Key Cryptography

In 1976, Diffie and Hellman proposed ***public-key cryptography***.

Encryption key and decryption key are *not* the same.

Each user A has a public encryption procedure E_A which may be placed in a public directory, and a private decryption procedure D_A which they keep secret.

Public-key cryptosystem has following properties:

1. $D(E(M)) = M$
2. Both E and D are easy to compute.
3. It is not easy to compute D from E.
4. $E(D(M)) = M$

Like conventional cryptosystem, public-key cryptosystem can provide both **confidentiality** and **authenticity**. Unlike conventional cryptosystem, public-key cryptosystem can also provide a method of implementing **digital signatures**, and it does not need an exchange of secret key prior to private communication.

Confidentiality

1. Bob (B) wants to send a private message to Alice (A).
2. First Bob retrieves E_A from the public directory.
3. Then Bob enciphers M obtaining $E_A(M)$ and he sends it to Alice.
4. Alice deciphers $E_A(M)$ by computing $D_A(E_A(M)) = M$

Confidentiality is provided by step 3: Alice is the only one who can decipher $E_A(M)$.

Advantages:

1. Encryption key (public key) can be sent as a plaintext or be placed in the public directory.
2. There is no need for distribution of secret decryption key.

Signatures

Authenticity can be provided by the means of digital signature.

1. Bob receives a message M signed by Alice.
2. Bob must be able to validate Alice's signature on M.
3. Nobody can forge Alice's signature.
4. A judge or third party can check whether it is Alice's signature or not.

Signature must be both message and signer dependent.

How does Alice send a signed message M to Bob?

1. Alice first 'signs' a message M by computing $D_A(M) = S$ for authenticity.
2. Then Alice encrypts S by computing $E_B(S)$ for confidentiality.
3. Bob first compute $S = D_B(E_B(S))$.
4. Then Bob obtains M by computing $E_A(S) = E_A(D_A(M)) = M$

Alice can not later deny having sent Bob this message, since no one else could have created $S = D_A(M)$.

Bob can not forge Alice's signature since he does not know D_A.

RIVEST-SHAMIR-ADLEMAN (RSA) SCHEME

In 1978, Rivest, Shamir and Adleman published the first method of realizing public-key cryptography.

- The encryption key is a pair of positive integers (e, n).
- The decryption key is a pair of positive integers (d, n).
- Message M is an integer between 0 and $n-1$.
- The encryption procedure E is $C = E(M) = M^e \; mod \; n$.
- The decryption procedure D is $M = D(C) = C^d \; mod \; n$.
- Encryption does not increase the size of a message; both message and the ciphertext are integers in the range $[0, n-1]$.

The Underlying Mathematics

Euler's generalization of Fermat's theorem: For every a and n such that $\gcd(a, n) = 1$ we have $a^{\varphi(n)} mod\ n = 1$.

- We choose $n = p \times q$, where p and q are primes.
- We calculate $\varphi(n) = (p-1)(q-1)$.
- We choose d such that $\gcd(d, \varphi(n)) = 1$.
- We compute e from $(e \times d)\ mod\ \varphi(n) = 1$.

Encryption: $D\big(E(M)\big) = E(M)^d\ mod\ n = (M^e\ mod\ n)^d\ mod\ n = M^{e\times d}\ mod\ n = M$

Decryption: $E(D(M)) = D(M)^{\,e}\ mod\ n = (M^d\ mod\ n)^e\ mod\ n = M^{e\times d}\ mod\ n = M$

Proof: $M^{e\times d}\ mod\ n = M^{k\times\varphi(n)+1}\ mod\ n$, because $(e \times d)\ mod\ \varphi(n) = 1$

$$= M \times M^{k\times\varphi(n)} mod\ n$$

$$= M \times \left(M^{\varphi(n)} mod\ n\right)^k mod\ n$$

$$= M \times 1^k\ mod\ n$$

$$= M$$

If $\gcd(M, n) \neq 1$, the equation$M^{e\times d}\ mod\ n = M$ still holds.

The Underlying Mathematics

To compute d from e one should know $\varphi(n)$.

$$e \times d \bmod \varphi(n) = 1$$

Recall that n is public, but p and q are not.

It is very difficult to compute $\varphi(n)$ without knowing p and q.

It is very difficult to find p and q (to factor n).

How to Encrypt and Decrypt Efficiently

Using fast exponentiation algorithm (exponentiation by repeating squaring and multiplication), computing $M^e \bmod n$ requires $O(\log e)$ steps.

The encryption time per block increases no faster than $O(m^3)$, where m is the number of digits in n.

How to Find p and q

Each user must choose p and q to create their own encryption and decryption keys.

The authors of RSA recommend that n be about 200 digits long, so p and q should have about 100 digits each.

To find a 100 digit random prime number, generate 100 digit random odd numbers until a prime number is found.

Prime number theorem describes how prime numbers are spaced and states that around N, for large enough N, there is on average one prime in $\ln N$ numbers.

About $\frac{\ln 10^{100}}{2} = 115$ numbers will be tested before a prime is found.

Choosing p, q and e and computing d

How to Find p and q: To gain additional protection against sophisticated factoring algorithms:

1. p and q should differ in length by a few digits;

2. both $(p-1)$ and $(q-1)$ should contain a large prime factor;

3. $\gcd(p-1, q-1)$ should be small.

How to Choose d: It turnes out that d easy to choose - any prime number greater than $\max(p, q)$ will do.

How to Compute e: Note that e can be computed using Euclid's algorithm for computing $\gcd$ extended to compute inverses. Number of steps will be less than $2\log_2 n$.

Security of RSA

- No techniques exist to prove that an encryption algorithm is secure.
- All obvious approaches for breaking RSA are at least as difficult as factoring n.
- Factoring large numbers is a well known difficult problem that has been worked on by many mathematicians.
- The fastest general-purpose factoring algorithm (Number Field Sieve) can factor n in approximately
  - $O(e^{1.9\sqrt[3]{\lg n}\sqrt[3]{(\lg \lg n)^2}})$
- RSA-200 was factored using the above algorithm in May 2005. The time taken for factoring was equivalent to 55 years on a single $2.2GHz$ CPU.
- RSA-640 (193 decimal digits) was factored in November 2005. The time taken for factoring was equivalent to 30 years on a single $2.2GHz$ CPU.

Security of RSA

- In Dec 2009 RSA-768 (232 decimal digits) was factored using number field sieves (Kleinjung et al. 2010):

1230186684530117755130494958384962720772853569595334792197322
4521517264005072636575187452021997864693899564749427740638459
2519255732630345373154826850791702612214291346167042921431160
2221240479274737794080665351419597459856902143413

- The time taken was equivalent to 1500 years on a single core $2.2GHz$ AMD processor with $2GB$ RAM. Actually took over 3 years.

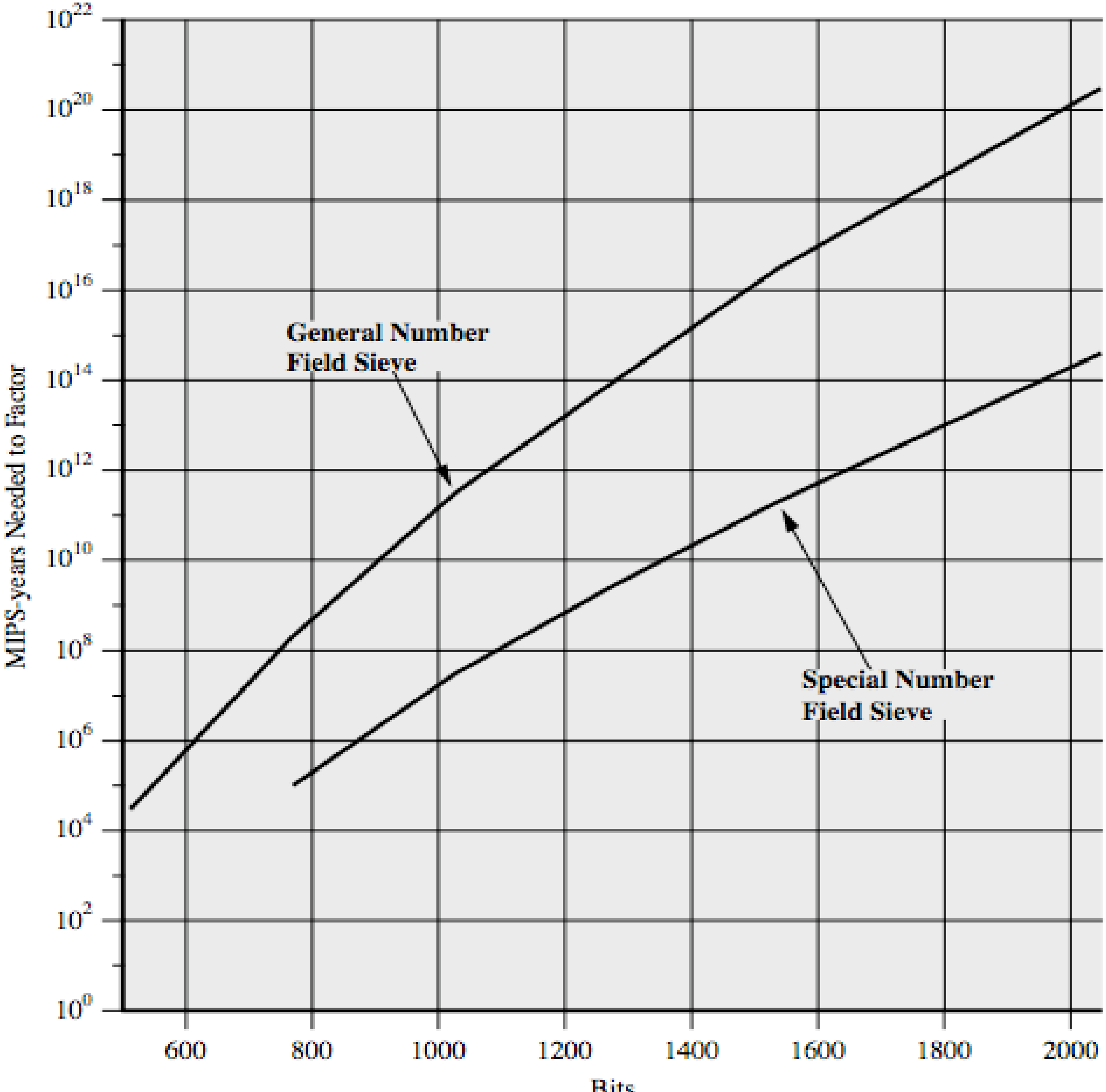
- Today, n with less than 300 decimal digits is no longer secure; 1024-bit RSA is also being phased out – you should use 2048-RSA at least.

Security of RSA

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

Decimal Digits	Bits	Date
100	332	1 April 1991
110	365	14 April 1992
120	398	9 June 1993
129	428	26 April 1994
130	431	10 April 1996
140	465	2 February 1999
155	512	16 August 1999
160	530	1 April 2003
174	**576**	3 December 2003
200	663	9 May 2005
193	**640**	2 November 2005
232	**768**	12 December 2009
180	596	8 May 2010
190	629	8 November 2010
212	**704**	2 July 2012
210	696	26 September 2013
220	729	13 May 2016
230	762	15 August 2018
240	795	2 December 2019
232	768	17 February 2020
250	829	28 February 2020

Security of RSA



Security of RSA

- Computing $\varphi(n)$ without factoring does not appear to be easier than factoring n since it enables the cryptanalyst to easily factor n.
- Determining d without factoring n or computing $\varphi(n)$ does not seem to be easier then factoring n since once d is known n could be factored easily.

Example 1

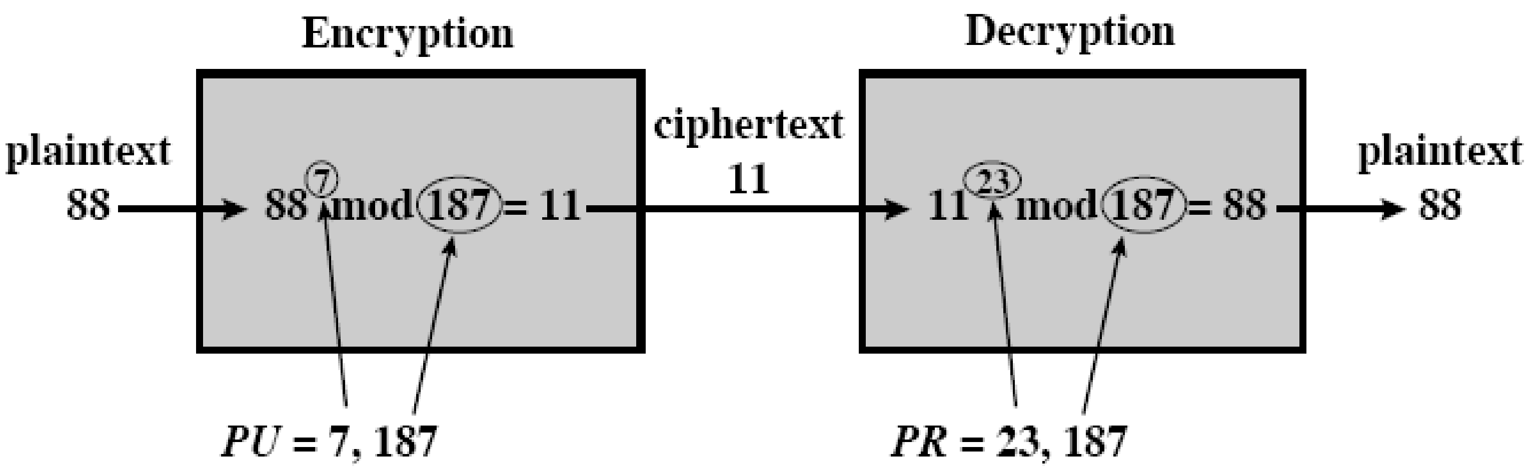


Figure 9.6 Example of RSA Algorithm

$n = 187, e = 7, d = 23$

Example 1

Generating value for previous slide.

1. Select two prime numbers, $p = 17$ and $q = 11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\varphi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select a prime number d such that $d > \max(p, q)$, and $d < \varphi(n) = 160$; we choose $d = 23$.
5. Determine e such that $d \times e \ mod \ \varphi(n) = 1$. Hence $e = 7$, since $7 \times 23 \ mod \ 160 = 1$.

Example 1

Encrypting plaintext $M = 88$:

$$
\begin{aligned}
C &= M^e \mod n \\
&= 88^7 \mod 187 \\
&= 88\,(88)^6 \mod 187 \\
&= 88\,(7744)^3 \mod 187 \\
&= 88\,(77)^3 \mod 187 \\
&= 88 \times 77\,(77)^2 \mod 187 \\
&= 6776\,(5929) \mod 187 \\
&= 44 \times 132 \mod 187 \\
&= 5808 \mod 187 \\
&= 11 \mod 187
\end{aligned}
$$

$$C = 11$$

Decrypting ciphertext $C = 11$:

$$
\begin{aligned}
M &= C^d \mod n \\
&= 11^{23} \mod 187 \\
&= 11\,(11)^{22} \mod 187 \\
&= 11\,(121)^{11} \mod 187 \\
&= 11 \times 121\,(121)^{10} \mod 187 \\
&= 1331\,(14641)^5 \mod 187 \\
&= 22\,(55)^5 \mod 187 \\
&= 22 \times 55\,(55)^4 \mod 187 \\
&= 1210\,(3025)^2 \mod 187 \\
&= 88\,(33)^2 \mod 187 \\
&= 95832 \mod 187 \\
&= 88 \mod 187
\end{aligned}
$$

$$M = 88$$

Example 2

Consider the RSA encryption scheme with public keys $n = 55$ and $e = 7$.

a) Encipher the plaintext $M = 10$.

b) Break the cipher by finding p, q and d.

c) Decipher the ciphertext $C = 35$.

Solution:

a)

$$E(M) = M^e \, mod \, 55$$
$$E(10) = 10^7 \, mod \, 55 = 10$$

b)

$$n = p \times q$$

$n = 55$, it follows that $p = 5$ and $q = 11$

$$\varphi(n) = (5-1)(11-1) = 40$$

$$e * d \, mod \, \varphi(n) = 1$$

$7 * d \, mod \, 40 = 1$, it follows that $d = 23$

c)

$$D(C) = C^d \, mod \, n$$
$$D(35) = 35^{23} \, mod \, 55 = 30$$

ElGamal Cryptography

- ElGamal is a public-key cryptosystem that uses exponentiation in a finite (Galois) fields.
- Security of ElGamal is based difficulty of computing discrete logarithms
- To understand ElGamal and its security, we need the following concepts: primitive root and discrete logarithm.

ElGamal Cryptography

A <u>primitive root</u> a of a prime number p is an integer whose powers mod p generate all the integers from 1 to $p-1$.

<u>Example 3</u>:

Is 2 is a primitive root of 5?

$2^1 \bmod 5 = 2$

$2^2 \bmod 5 = 4$

$2^3 \bmod 5 = 3$

$2^4 \bmod 5 = 1$

<u>Answer:</u> Yes, since the powers of $2 \bmod 5$ generate all the integers from 1 to 4.

<u>Example 4</u>: Is 4 is a primitive root of 5?

$4^1 \bmod 5 = 4$

$4^2 \bmod 5 = 1$

$4^3 \bmod 5 = 4$

$4^4 \bmod 5 = 1$

<u>Answer:</u> No, since the powers of $4 \bmod 5$ generate only 1 and 4 and not all the integers from 1 to 4.

ElGamal Cryptography

Discrete logarithm: For a given integer b, prime p and a primitive root a of p, discrete logarithm i is a unique integer such that $1 \le i \le p-1$ and $b = a^i \bmod p$.

Example 5. Find a discrete logarithm of 3 for the base 2 modulo 5.

Answer.

$$2^1 \bmod 5 = 2$$
$$2^2 \bmod 5 = 4$$
$$\underline{2^3 \bmod 5 = 3}$$

Thus discrete logarithm of 3 for the base 2 modulo 5 is 3.

ElGamal Cryptography

- In addition to each user's public and private keys, ElGamal also has global public elements, a prime number q and a primitive root a of q.

- Each user (e.g., Alice, or A for short) generates their own private and public keys as follows:
  - A chooses a **private key** x_A such that $1 < x_A < q-1$
  - A computes her **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Message Exchange

- Bob encrypts a message to send to Alice:
  - represents message M in range $0 <= M <= q-1$
    - longer messages must be sent as blocks
  - choses random integer k with $1 <= k <= q-1$
  - computes one-time key $K = y_A^k \bmod q$
  - encrypts M as a pair of integers (C_1, C_2) where
    - $C_1 = a^k \bmod q$;
    - $C_2 = K \times M \bmod q$

- Alice then recovers message by
  - recovering key K as $K = C_1^{xA} \bmod q$
  - computing M as $M = C_2 \times K^{-1} \bmod q$

- A unique k must be used each time, otherwise the system would be vulnerable to known plaintext attack.

ElGamal Example

Example 6:

- Use field $GF(19)$: $q = 19$ and $a = 10$
- Alice computes her key:
  - A chooses $x_A = 5$ and computes $y_A = 10^5 \bmod 19 = 3$
- Bob send message $M = 17$ as $(11,5)$ by
  - choosing random $k = 6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;
  - $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{xA} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} \bmod 19 = 7^{-1} \bmod 19 = 11$
  - recover $M = C_2 K^{-1} \bmod q = 5 \times 11 \bmod 19 = 17$

Diffie-Hellman Key Exchange

The security of Diffie-Hellman scheme relays on the difficulty of computing discrete algorithm.

Background - revision:

- A primitive root of a prime number is the one whose powers generate the complete set of residues except 0 (that is, all the integers from 1 to $p-1$).
- For any integer b and a primitive root a of the prime number p there is a unique exponent i such that $b = a^i \bmod p$, where $0 \leq i < (p-1)$.
- The exponent i is referred to as discrete logarithm.

Diffie-Hellman Key Exchange

Example 7:

2 is a primitive root of 5 since:

$2^0 \bmod 5 = 1$

$2^1 \bmod 5 = 2$

$2^2 \bmod 5 = 4$

$2^3 \bmod 5 = 3$

$2^4 \bmod 5 = 1$

4 is not a primitive root of 5 since:

$4^0 \bmod 5 = 1$

$4^1 \bmod 5 = 4$

$4^2 \bmod 5 = 1$

$4^3 \bmod 5 = 4$

$4^4 \bmod 5 = 1$

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Figure 6.16 The Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman Key Exchange with Three or More Parties

- Alice chooses a random large number x and sends Bob
  $X = \alpha^x \bmod q$
- Bob chooses a random large number y and sends Carol
  $Y = \alpha^y \bmod q$
- Carol chooses a random large number z and sends Alice
  $Z = \alpha^z \bmod q$
- Alice sends Bob
  $Z' = Z^x \bmod q$
- Bob sends Carol
  $X' = X^y \bmod q$
- Carol sends Alice
  $Y' = Y^z \bmod q$

Diffie-Hellman Key Exchange with Three or More Parties

- Alice computes $K = Y'^{x} \bmod q$
- Bob computes $K = Z'^{y} \bmod q$
- Carol computes $K = X'^{z} \bmod q$
- $K = \alpha^{xyz} \bmod q$
- Note that α is a primitive root of q.

Next Week

1. Key Management
    1. Distribution of public keys
        - Public Announcement
        - Publicly Available Directory
        - Public-key Authority
        - Public-Key Certificates
    2. Public-Key Distribution of Secret Keys

2. Message Authentication
    - Encryption
    - Massage Authentication Code
    - Hash functions

Chapter 14 from text: Key Management and Distribution
Chapter 11 from text: Cryptographic Hash Functions
Chapter 12 from text: Message Authentication Codes

References

1. R. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, Vol. 21 (2), pp.120–126. 1978.

2. W. Stallings. "Cryptography and Network Security", Global edition, Pearson Education Australia, 2016.