Principles of Cryptography: Part 5

Gaurav S. Kasbekar
Dept. of Electrical Engineering
IIT Bombay

References

- J. Kurose, K. Ross, "*Computer Networking: A Top Down Approach*", Sixth Edition, Pearson Education, 2013
- C. Kaufman, R. Perlman, M. Speciner, "*Network Security: Private Communication in a Public World*", Pearson Education, 2nd edition, 2002

Diffie-Hellman Algorithm

Diffie-Hellman Algorithm

- Public key cryptosystem
- Was invented before RSA
- Unlike RSA, cannot be used for encryption or to create digital signatures
- Allows two individuals, say Alice and Bob, to agree on a shared secret key, even though they can only exchange messages that can be overheard by intruders
- After they have agreed upon a shared key, it can be used for communication using symmetric-key cryptography

Diffie-Hellman Algorithm (contd.)

- There are numbers p and g, where
  - ❑ p is a large (e.g., 2048-bit) prime number
  - ❑ g is a number less than p
- p and g can be publicly known
  - ❑ e.g.: Alice may choose p and g and send them over the channel to Bob (p and g may be overheard by intruders)
- Then each of Alice and Bob independently chooses a large number less than p at random and keeps it secret
  - ❑ Let S_A and S_B denote Alice's and Bob's secret number, respectively, where $S_A < p$ and $S_B < p$

Diffie-Hellman Algorithm (contd.)

- Alice computes $T_A = g^{S_A} \bmod p$; Bob computes $T_B = g^{S_B} \bmod p$
- Alice sends T_A to Bob and Bob sends T_B to Alice
- Alice computes ${T_B}^{S_A} \bmod p$ and Bob computes ${T_A}^{S_B} \bmod p$
- **Theorem**: ${T_B}^{S_A} \bmod p = {T_A}^{S_B} \bmod p$
- **Proof**:
  - LHS = $(g^{S_B} \bmod p)^{S_A} \bmod p = (g^{S_B})^{S_A} \bmod p = g^{S_A S_B} \bmod p$
  - Similarly, RHS = $g^{S_A S_B} \bmod p$
- Thus, both Alice and Bob agree on the same number $g^{S_A S_B} \bmod p$ (which is the shared key)
- **Terminology**: S_A and T_A are known as Alice's private and public key, respectively; S_B and T_B are known as Bob's private and public key, respectively

Example

- $p = 23; g = 5$
- $S_A = 4; S_B = 3$
- Then T_A:
  - 4
- T_B:
  - 10
- ${T_A}^{S_B} \text{ mod } p$:
  - 18
- ${T_B}^{S_A} \text{ mod } p$:
  - 18
- In this example, the shared key is 18

Security of Diffie-Hellman

- Even though an intruder may know $g, p, T_A = g^{S_A} \bmod p$ and $T_B = g^{S_B} \bmod p$:
  - ❑computationally infeasible for him/ her to calculate $g^{S_A S_B} \bmod p$
- Problem of finding S_A using g, p and $g^{S_A} \bmod p$ known as "*Discrete Logarithm Problem*"
- If intruder could compute discrete logarithms efficiently, then he/ she could find S_A, S_B and hence $g^{S_A S_B} \bmod p$ efficiently
- However, no efficient algorithm for finding discrete logarithms is known

Additional Requirements on p and g

- The security of Diffie-Hellman is compromised unless p and g satisfy the following additional properties:

1) $g^x \bmod p$ must not equal 1, unless x is a multiple of $(p-1)$

- ❑ Reason:
  - ○ If $g^x \bmod p = 1$ for a small value of x, then to find S_A using $g^{S_A} \bmod p$ by brute force, an intruder only needs to try out a small number of values of S_A
  - ○ E.g., if $g^3 \bmod p = 1$, then $g^4 \bmod p = g \bmod p$, $g^5 \bmod p = g^2 \bmod p$, $g^6 \bmod p = 1$, $g^7 \bmod p = g \bmod p$, etc.

2) $\frac{(p-1)}{2}$ must also be prime

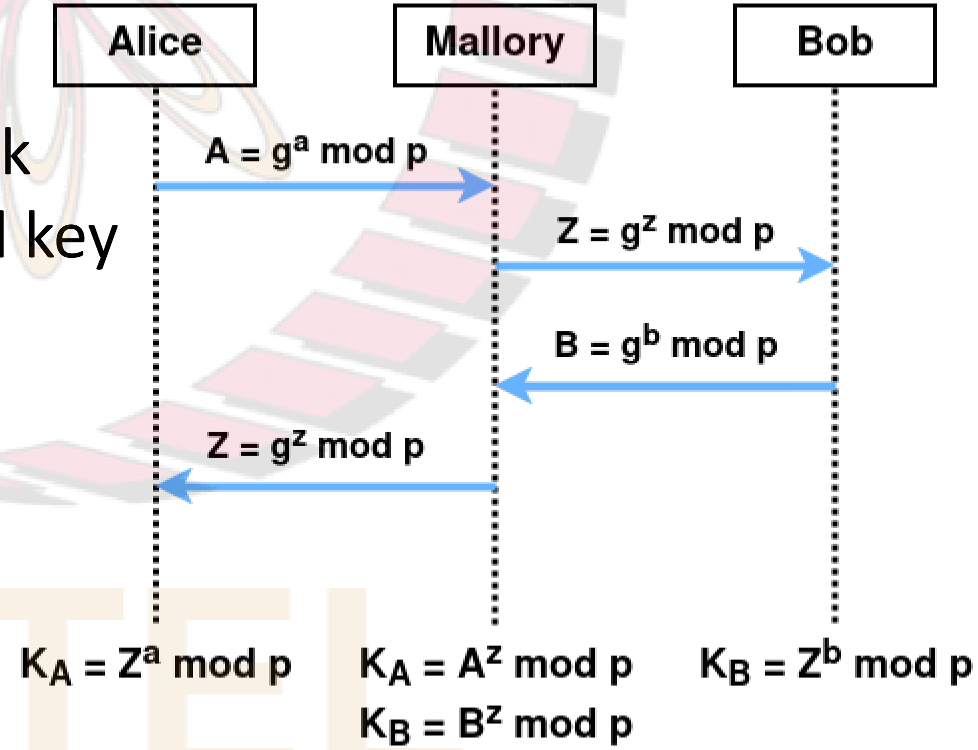
- ❑ A prime that satisfies this property called "*safe prime*"
- ❑ Reason:
  - ○ If this property not satisfied, it may be possible to efficiently compute discrete logarithms using "Pohlig-Hellman algorithm" (details omitted)

Man-in-the-Middle Attack

- Diffie-Hellman algorithm insecure when there can be an intruder (“man-in-the-middle”) on the channel between Alice and Bob, who can modify messages sent from Alice to Bob or vice-versa
- Suppose g and p publicly known
- When Alice receives T_B, there is no way for her to find out whether:
  - ❑ it was really sent by Bob or not
  - ❑ it was modified during transit from Bob to Alice
- Similar uncertainty when Bob receives T_A

Man-in-the-Middle Attack (contd.)

- In fig., Mallory is intruder who performs man-in-the-middle attack
- At end of attack:
  - ❑ Alice has secret shared key $g^{az} \bmod p$ with Mallory
  - ❑ Bob has secret shared key $g^{bz} \bmod p$ with Mallory
- However, Alice and Bob think that they have secret shared key with each other



Man-in-the-Middle Attack (contd.)

- After intruder has performed man-in-the-middle attack:
  - ❑ when Alice sends an encrypted message to Bob, intruder can read it and modify it before forwarding it to Bob
  - ❑ similarly can read and modify messages from Bob to Alice
- Hence, the above form of the Diffie-Hellman algorithm is only secure against “passive attacks”, in which the intruder just watches messages being transmitted between Alice and Bob

Defences Against Man-in-the-Middle Attack

- Suppose after completing the Diffie-Hellman algorithm, Alice encrypts and transmits the established shared key to Bob to prove that she is indeed Alice
- Will Bob be able to detect man-in-the-middle attack?
- No:
  - ❑the intruder can encrypt and send the shared key established between himself/ herself and Bob to Bob
  - ❑Bob will think it was sent by Alice

Defences Against Man-in-the-Middle Attack (contd.)

- Suppose p and g are public
- Recall: S_B and T_B are known as Bob's private and public key, respectively
- The public key of every user is stored in a database like a telephone directory
- When Alice wants to establish a secret key with Bob, she just looks up T_B and computes ${T_B}^{S_A} \bmod p$; Bob looks up T_A and computes ${T_A}^{S_B} \bmod p$
- Recall that ${T_B}^{S_A} \bmod p = {T_A}^{S_B} \bmod p$; thus, Alice and Bob have agreed upon this shared secret key
- Does this defence work?
  - ❑ Yes
- Later we will study how the database in which public keys are stored (called "Public Key Infrastructure (PKI)") can be implemented