Key Exchange & Diffie-Hellman

Gaspare FERRARO
CyberSecNatLab
Matteo ROSSI
Politecnico di Torino

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Prerequisites

- Lectures:
  - *CR_0.1 - Number Theory and modular arithmetic*
  - *CR_1.1 – Introduction to cryptography and classical ciphers*

Recap

- At this point we know:
  - How symmetric ciphers work
  - How to protect communications using a shared secret key
- Goal of this lecture:
  - Find a way to exchange keys
  - Expose the possible issues in key-exchange

Outline

- The Key Exchange problem
- The Diffie-Hellman protocol
- Issues

Outline

- The Key Exchange problem
- The Diffie-Hellman protocol
- Issues

The Key Exchange Problem

- Problem settings:
  - N users that want to communicate with each others
  - Communication must be independent from the others but in a shared channel:
    - User C can see the messages exchanged by user A and user B
    - User C cannot decrypt them
- Naïve solution: every user stores $N - 1$ different keys shared with every single other user

A Better Solution

- The naïve solution doesn't seem efficient as the number of keys grows quadratically with the number of users
- A better way to solve this problem, is to have a trusted 3rd party (TTP):
  - Every user has to remember a single key to speak with the TTP
  - The TTP manages the creation of shared keys between users

TTP – Toy Example

- Alice wants to speak with Bob using a TTP, who knows both Alice's key k_a and Bob's key k_b:
  - Alice says to TTP: "I want to talk with Bob“
  - TTP randomly creates a key k_{AB}
  - TTP sends to Alice $E(k_A, k_{AB})$ and $E(k_B, k_{AB})$
  - Alice sends to Bob the second one and they start communicating

TTP - Issues

- This kind of scheme has two main problems:
  - It relies on the TTP being always online
  - The TTP knows all the keys: it is a single point of failure for the whole system
- There are contexts in which TTP makes sense:
  - Inside a company/university or a whatever closed environment
  - A similar reasoning is in fact at the basis of Kerberos (See lecture CP_1.3 - Kerberos)

The Key Exchange Problem

- Key question: can we generate online keys without a TTP?
- Some protocols have been proposed during the years:
  - Merkle puzzles (1974)
  - Diffie-Hellman (1976)
  - RSA (1977)
  - Identity-based encryption (2001)
  - Functional encryption (2011)

Merkle Puzzles

- Merkle puzzles are a way to exchange keys using block ciphers
  - The idea is that Alice sends N puzzles to Bob, containing the messages *"This is message X. This is the symmetrical key Y"*
  - Bob randomly choose one of them, solve it, and gives back the number X to Alice
  - Alice and Bob both know which puzzle X Bob solved and its symmetrical key Y, but whoever listen to the conversation do not
- Issue^1: to make this work, the number of puzzles N must be very big
- Issue^2: An eavesdropper need only a linear time factor of O(N), compared to Alice and Bob, to find the key exchanged
- Our goal is to have a key exchange algorithm which needs exponential time to be broken

Outline

- The Key Exchange problem
- The Diffie-Hellman protocol
- Issues

The Diffie-Hellman Protocol

“We stand today on the brink
of a revolution in cryptography.”

[*Whitfield Diffie and Martin Hellman,*
“New directions in Cryptography”, November 1976]

The Diffie-Hellman Protocol

- Diffie and Hellman in 1976 give a better solution using number theory
  - Alice and Bob agree (publicly!) on a prime number p and a generator g in $\{2, 3, \ldots, p-1\}$
  - Alice generates a number a in $\{2, 3, \ldots, p-1\}$ and Bob does the same with b
  - Alice sends the value $A = g^a \; mod \; p$ and, in the same way, Bob sends $B = g^b \; mod \; p$

The Diffie-Hellman Protocol

- At this point:
  - Alice knows a and B
  - Bob knows b and A
  - They both can calculate $g^{ab} = A^b = B^a \ mod \ p$, that will be their shared key
  - An eavesdropper that only knows $g^a \ mod \ p$ and $g^b mod \ p$ can't compute $g^{ab} \ mod \ p$ efficiently!

The Diffie-Hellman Protocol

- Example:
  - Take $p = 37$ and $g = 2$
  - Alice generates the number 7 and sends $2^7 mod\ 37 = 17$
  - Bob generates the number 21 and sends $2^{21} mod\ 37 = 29$
  - Both can compute $2^{7*21} = 29^7 = 17^{21} = 8\ mod\ 37$
  - 8 will be Alice and Bob's shared key

How hard is to break DH?

- It is believed that the only way to break DH is to recover one of a and b, computing so a discrete logarithm
- The best-known algorithm for discrete logarithms is the General Number Field Sieve (GNFS) that runs in $O(e^{(\sqrt[3]{n})})$

How hard is to break DH?

- To give some practical numbers:
  - Breaking DH with p of 1024 bits is roughly equivalent to break a block cipher with 80-bit security
  - If p is of 3072 bits it is equivalent to a 128-bit block cipher (like AES-128!)
  - ...and so on
- These numbers are even better in the case of variations of the classic DH: for example, a 256-bit instance of DH over Elliptic Curves is enough to get the security of AES-128

Outline

- The Key Exchange problem
- The Diffie-Hellman protocol
- Issues

Issues

- There are two main issues in using Diffie-Hellman:
  - Reaching high level of security requires very big keys
  - Diffie-Hellman Protocol is vulnerable to active attacks like the *Man-in-the-Middle*

Insecurity against Man-in-the-Middle

- An eavesdropper, who can listen and modify the communication, can:
  - Intercept $g^a mod\ p$ from Alice and substitute it with $g^{a'} mod\ p$
  - Do the same with Bob, using $g^{b'} mod\ p$
  - Craft the keys $(g^{a'b} mod\ p, g^{ab'} mod\ p)$ (now Alice and Bob have different “shared” keys!)
  - Decrypt every communication, read it, and re-encrypt it with the correct key (without letting know Alice and Bob anything!)

Key Exchange & Diffie-Hellman

Gaspare FERRARO
CyberSecNatLab
Matteo ROSSI
Politecnico di Torino

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it