Cryptographic Protocols 1

Luke Anderson

luke@lukeanderson.com.au

5th May 2017

University Of Sydney

Overview

Crypto-Bulletin

Massive Google Docs phishing attack hits Gmail users

https://www.itnews.com.au/news/massive-google-docs-phishing-attack-hits-gmail-users-460436

Watch out IT admins: you're a hacker's new target

https://www.itnews.com.au/news/watch-out-it-admins-youre-a-hackers-new-target-459590

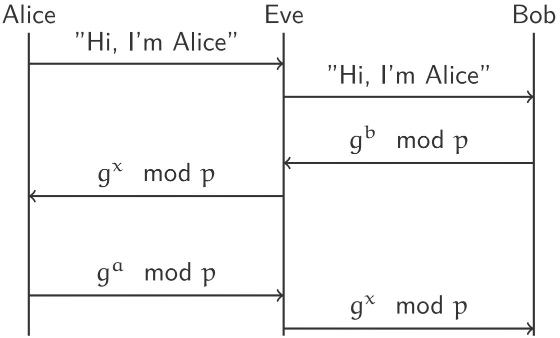
Blind Trust in Email Could Cost You Your Home

https://krebsonsecurity.com/2017/04/blind-trust-in-email-could-cost-you-your-home/

Problem with Diffie-Hellman

Man-in-the-Middle

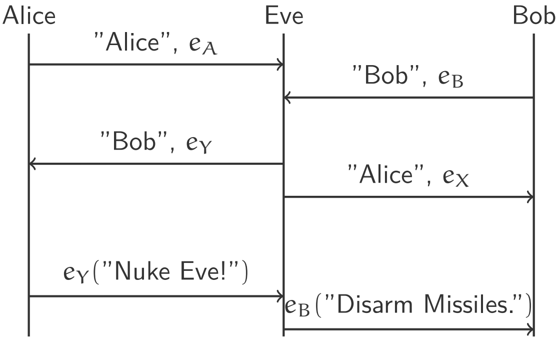
Suppose Alice and Bob are performing Diffie–Hellman key exchange, but a third party (Eve) actually owns the channel, and can intercept traffic.



- Shared secret for Alice/Eve is $g^{ax} \mod p$.
- Shared secret for Eve/Bob is $g^{bx} \mod p$.
- Eve owns the channel and neither Alice nor Bob know!

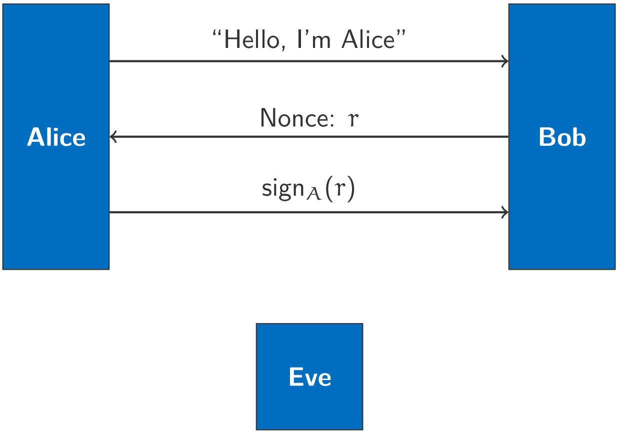
Man-in-the-Middle

Example:



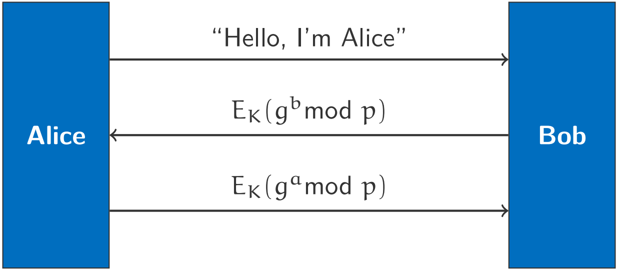
Eve owns the channel!

Session Hijacking



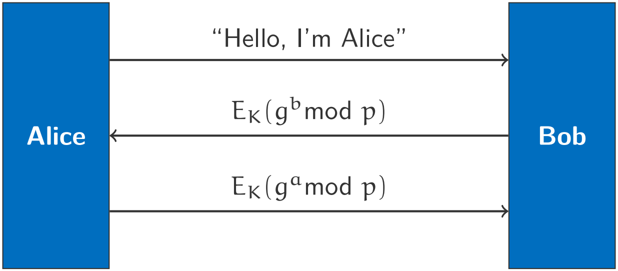
Eve owns the channel!

EKE — Encrypted Key Exchange



- The shared secret is $g^{ab} \mod p$.
- A (possibly low entropy) shared key k is used to form the high entropy shared secret.
- Prevents eavesdropping, and active attacks on DH.
- Maintains forward secrecy.
- What is the obvious problem?

EKE — Encrypted Key Exchange



- The shared secret is $g^{ab} \mod p$.
- A (possibly low entropy) shared key k is used to form the high entropy shared secret.
- Prevents eavesdropping, and active attacks on DH.
- Maintains forward secrecy.
- What is the obvious problem?

Public key crypto would be great, but we need a way of obtaining Bob’s public key without contacting him...

Definitions

A protocol is said to have *perfect forward secrecy* if disclosure of long-term keys does not compromise past (short term) sessions.

- Ephemeral keys (such as are generated during Diffie–Hellman) give this automatically.
- Encrypting everything directly with an RSA public key does **not** have forward secrecy.

A protocol is vulnerable to a *known-key attack* if disclosure of past session keys allows an attacker to compromise future session keys (including actively impersonating).

Needham-Schroeder Protocol

Needham–Schroeder Protocol

The Needham–Schroeder protocol facilitates key exchange using a trusted third party (TPP).

Alice and Bob want to set up a session key for communication. All parties share a key with Trent, the TPP.

1. Alice sends Trent a request to talk to Bob: (A, B, r_A), where r_A is a nonce.
2. Trent sends Alice a session key, and a ticket to give to Bob: $E_{k_{AT}}(r_A, B, k_{AB}, E_{k_{BT}}(k_{AB}, A))$.
3. Alice sends Bob the ticket: $(E_{k_{BT}}(k_{AB}, A), E_{k_{AB}}(s_A))$.
4. Bob challenges Alice: $E_{k_{AB}}(s_A - 1, r_B)$, where r_B is a nonce.
5. Alice responds to Bob's challenge: $E_{k_{AB}}(r_B - 1)$.

Needham–Schroeder Protocol

The Needham–Schroeder protocol facilitates key exchange using a trusted third party (TPP).

Alice and Bob want to set up a session key for communication. All parties share a key with Trent, the TPP.

1. Alice sends Trent a request to talk to Bob: (A, B, r_A), where r_A is a nonce.
2. Trent sends Alice a session key, and a ticket to give to Bob: $E_{k_{AT}}(r_A, B, k_{AB}, E_{k_{BT}}(k_{AB}, A))$.
3. Alice sends Bob the ticket: $(E_{k_{BT}}(k_{AB}, A), E_{k_{AB}}(s_A))$.
4. Bob challenges Alice: $E_{k_{AB}}(s_A - 1, r_B)$, where r_B is a nonce.
5. Alice responds to Bob's challenge: $E_{k_{AB}}(r_B - 1)$.

What if Eve gets a hold of k_{AT} somehow?

Needham–Schroeder Protocol

Problem: Bob has no guarantee that k_{AB} is fresh. Old session keys are valuable, as they do not expire.

1. Suppose Eve manages to get k_{AT}.
2. She can now read all of Alice's messages and impersonate her to everyone else.
3. Alice needs to revoke her key, but the only person that can make this have an effect is Trent.
4. **Key revocation is a major problem**.

Needham-Schroeder's problem is that it assumes all users of the system are good guys, and the goal is to keep the bad guys from getting in — the "eggshell" model.

Kerberos

Kerberos is a protocol which builds on the Needham–Schroeder protocol, and allows nodes within an insecure network to prove identity to each other.

Extra reading: Designing an Authentication System: a Dialogue in Four Scenes.

Public Key Management

Public Key Management using Certification Authorities

A public key certificate binds a public key to its owner:

1. Alice sends her public key to the CA (Trent).
2. The CA produces a certificate for Alice.
3. Alice sends her public key and certificate to Bob.
4. Bob verifies the certificate using Alice's public key.
5. Bob sends encrypted messages to Alice using the key.

$$\text{Certificate} = \text{sign}_{CA}[\text{X.500: name, org, address, pubkey, expires, ...}]$$

- Everyone must be able to verify the CA's public key, so it is shipped with OS's, browsers, etc.
- The CA is a trusted party: it has the ability to issue signatures to whomever.

Public Key Certificate Generation

1. Alice generates a public/private keypair.
2. Alice sends the public key to the CA.
3. The CA challenges Alice to see if she knows the private key.
4. The CA generates a certificate and sends it to Alice.

Important Note:

1. The CA never learns Alice's private key.
2. Important for forward secrecy.
3. A compromise of the CA can still lead to people pretending to be Alice.

Certificate Revocation

Alice's certificate may need to be revoked.

- Her private key is stolen or otherwise compromised.
- She changes jobs (or trustworthiness).

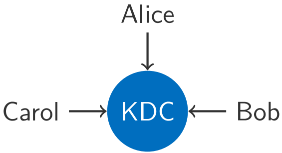
This is a major problem with the CA system.

- May require daily certification-validation information (slow, cumbersome).
- Use expiration date field.
- Use of a certificate revocation list (CRL) which is circulated (like bad credit cards.)

Trust Models

Symmetric Keys

- TTP must be online (used every session).
- TTP is a juicy target (knows passwords).
- No forward Secrecy.



Asymmetric Keys

- TTP is offline (only used in generation).
- TTP only knows public keys.
- TTP has forward secrecy.
- Not as fast (e..g SSL/TLS, PGP, ...)