Entity

Authentication II

ECE 4156/6156 Hardware-Oriented Security and Trust

Spring 2024

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

Reading Assignment

- Take good notes during this lecture!
- Introduction to Modern Cryptography, Chapter 10
- G. Lowe, “An attack on the Needham-Schroeder public-key authentication protocol,” Information Processing Letters, Vol. 56, Issue 3, Nov. 1995, pp. 131-133.

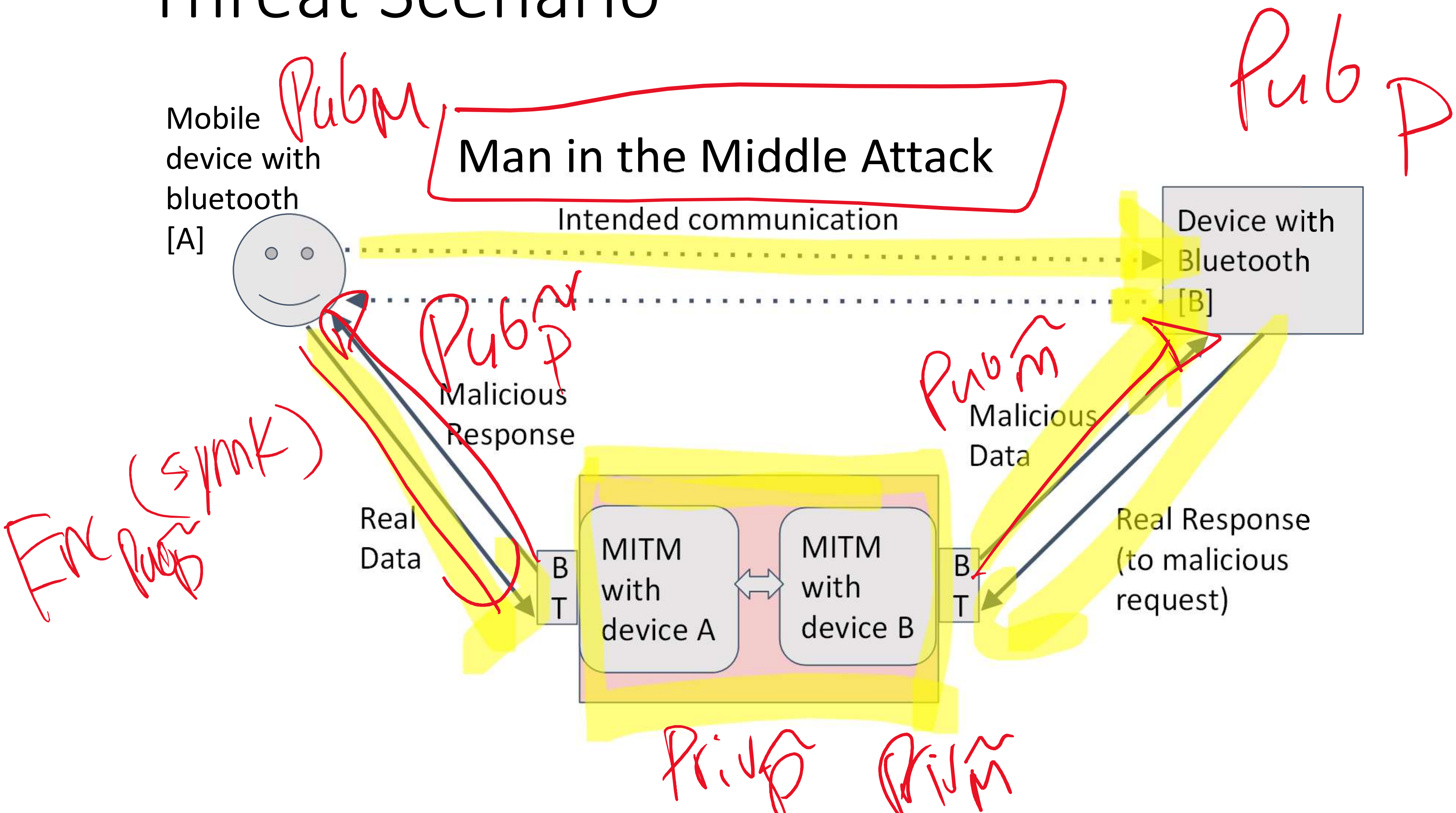
Protocols

- A protocol is a series of steps involving two or more parties designed to accomplish a task.
  - Everyone involved in the protocol must know the protocol and all of the steps to follow in advance
  - Everyone involved in the protocol must agree to follow it
  - The protocol must be unambiguous, the steps must be well defined, and there must be no change of misunderstanding
  - The protocol must be complete, i.e., there must be a specified action for every possible situation

First Attempt to Communicate Securely

- Alice and Bob agree on a cryptosystem
- Alice and Bob agree on a symmetric key
- Alice takes her plaintext message and encrypts it using the encryption algorithm and the key, creating a ciphertext message
- Alice sends the ciphertext to Bob
- Bob decrypts the ciphertext message with the same algorithm and key and reads it

Threat Scenario

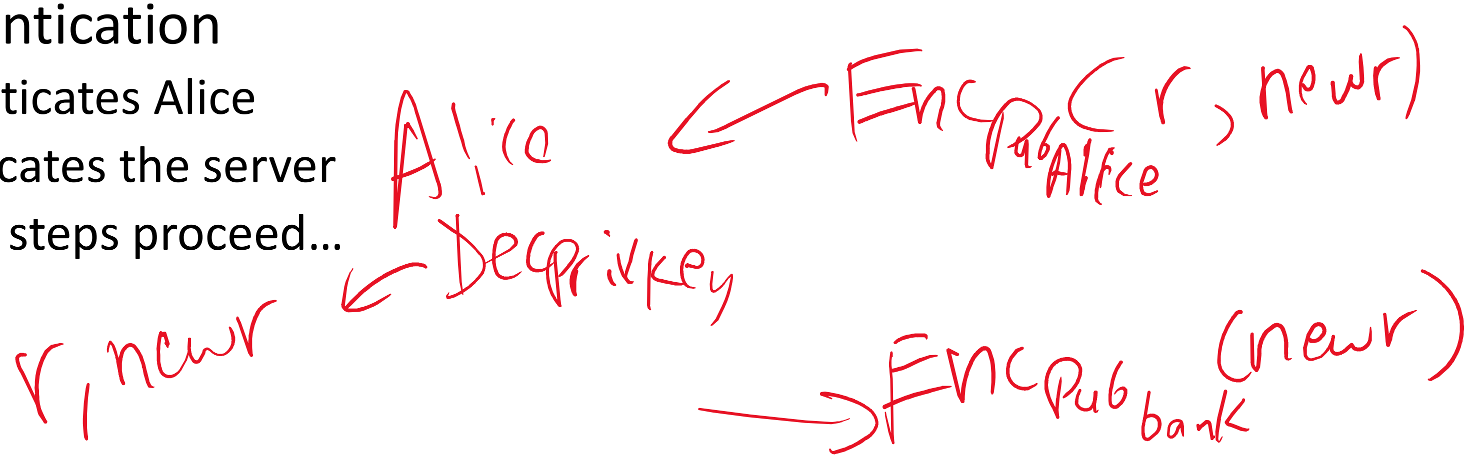


Number used only Once (NONCE)

- Authentication with asymmetric cryptography
  - Server sends Alice a random number (a “nonce”) in plaintext
  - Alice encrypts the nonce with her private key and sends it back to the server along with her name
  - The server uses Alice’s public key to decrypt the message and verify that the nonce sent by Alice is correct
  - Now the server can proceed with the next steps, e.g., by sending Alice a session key (e.g., a 128-bit AES key) encrypted with Alice’s public key

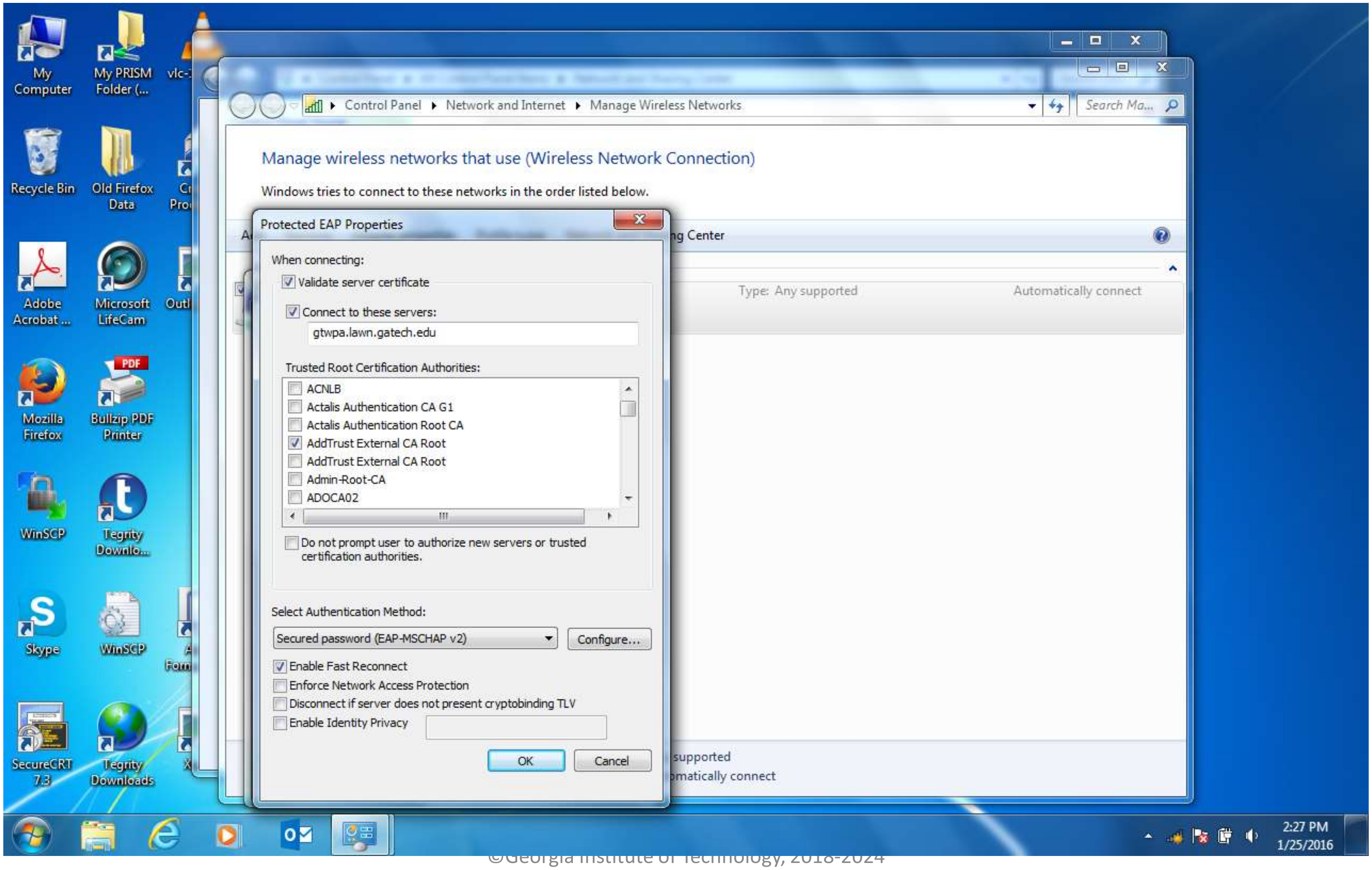
Actually…

- The previous slide presented one-way authentication, e.g., Alice authenticated herself to the server
- What about communication pretending to be from the server but really from another entity?
- Two-way authentication
  - Server authenticates Alice
  - Alice authenticates the server
  - Then the next steps proceed…

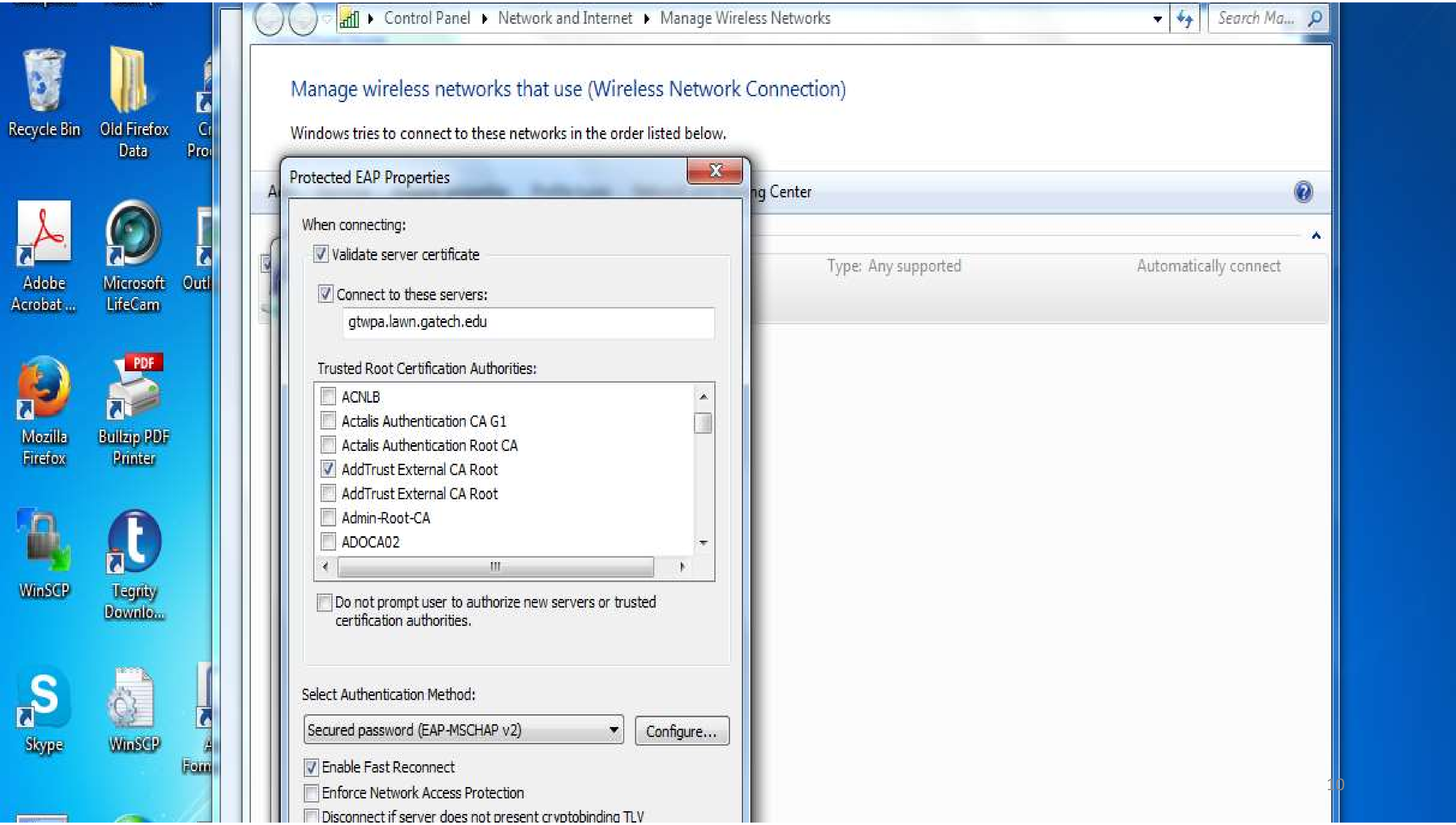


A Second Attempt to Communicate Securely

- A public key cryptosystem infrastructure is made widely available
- Alice obtain's Bob's public key from the infrastructure
  - E.g., using a Certificate Authority (CA) or a Trusted Third Party (TTP)
- Alice encrypts her message using Bob's public key and sends the message to Bob
- Bob then decrypts Alice's message using his private key



Control Panel ▸ Network and Internet ▸ Manage Wireless Networks
Manage wireless networks that use (Wireless Network Connection)
Windows tries to connect to these networks in the order listed below.
Protected EAP Properties
When connecting:
Validate server certificate
Connect to these servers:
gtwpa.lawn.gatech.edu
Trusted Root Certification Authorities:
ACNLB
Actalis Authentication CA G1
Actalis Authentication Root CA
AddTrust External CA Root
AddTrust External CA Root
Admin-Root-CA
ADOCA02
Do not prompt user to authorize new servers or trusted certification authorities.
Select Authentication Method:
Secured password (EAP-MSCHAP v2)
Configure...
Enable Fast Reconnect
Enforce Network Access Protection
Disconnect if server does not present cryptobinding TLV
Enable Identity Privacy
OK
Cancel
Type: Any supported
Automatically connect
2:27 PM
1/25/2016


Control Panel ▸ Network and Internet ▸ Manage Wireless Networks
Search Ma...
Manage wireless networks that use (Wireless Network Connection)
Windows tries to connect to these networks in the order listed below.
Type: Any supported
Automatically connect
Protected EAP Properties
When connecting:
Validate server certificate
Connect to these servers:
gtwpa.lawn.gatech.edu
Trusted Root Certification Authorities:
ACNLB
Actalis Authentication CA G1
Actalis Authentication Root CA
AddTrust External CA Root
AddTrust External CA Root
Admin-Root-CA
ADOCA02
Do not prompt user to authorize new servers or trusted certification authorities.
Select Authentication Method:
Secured password (EAP-MSCHAP v2)
Configure...
Enable Fast Reconnect
Enforce Network Access Protection
Disconnect if server does not present cryptobinding TLV
Recycle Bin
Old Firefox Data
Adobe Acrobat ...
Microsoft LifeCam
Mozilla Firefox
Bullzip PDF Printer
WinSCP
Tegrity Downlo...
Skype
WinSCP

Trent has all sym. keys of everyone

Needham-Schroeder (1978)

Goal 1: obtain shared session key

Goal 2: keep Bob's master key from Alice, vice versa

- Alice to Trent: A, B, N_A (NOTE: Trent is a Trusted Third Party or TTP!)
- Trent to Alice: $E_{K_A}(N_A,B,K,E_{K_B}(K,A))$
- Alice to Bob: $E_{K_B}(K,A)$
- Bob to Alice: $E_K(N_B)$
- Alice to Bob: $E_K(N_B\text{-}1)$

session key

two-way auth. w/ session key

Kerberos

- Alice sends Trent her identity and Bob's: A,B
- Trent generates key K and adds a timestamp T plus a lifetime L; he then encrypts two messages as follows and sends them to Alice
  - $E_A(T,L,K,B)$; $E_B(T,L,K,A)$
- Alice then uses K to send Bob her identity and timestamp, plus Trent's message
  - $E_K(A,T)$; $E_B(T,L,K,A)$
- Bob creates a message consisting of the timestamp plus one, encrypts it in K, and sends it to Alice
  - $E_K(T+1)$

session

Alice is entity authenticated to Bob

Bob is entity auth. to Alice ⇒ 2-way

Old laptop, old tower comp., old phone

An Attack on Needham-Schroeder

Assume all prior comm. eavesdropped

Also assume Mallory can intercept & redirect messages

- Mallory obtains an old session key K
- Mallory to Bob: $E_B(K,A)$ → Bob
- Bob to Alice: $E_K(N_B)$
    - Mallory intercepts this message and decrypts it with K
- Mallory to Bob: $E_K(N_B\text{-}1)$

RECALL!

Alice to Trent: A, B, N_A

Trent to Alice: $E_{K_A}(N_A,B,K,E_{K_B}(K,A))$

Alice to Bob: $E_{K_B}(K,A)$

Bob to Alice: $E_K(N_B)$

Alice to Bob: $E_K(N_B\text{-}1)$

Goal: discovery of an old session key only compromises the old session

Public-Key Needham-Schroeder

Trent has everyone's public keys

- 2.1 Alice to Trent: A, B
- 2.2 Trent to Alice: $E_{T_{priv}}(B_{pub}, B)$
- 2.3 Alice to Bob: $E_{B_{pub}}(N_A, A)$
- 2.4 Bob to Trent: B, A
- 2.5 Trent to Bob: $E_{T_{priv}}(A_{pub}, A)$
- 2.6 Bob to Alice: $E_{A_{pub}}(N_A, N_B)$ → $E_{A_{pub}}(B, N_A, N_B)$
- 2.7 Alice to Bob: $E_{B_{pub}}(N_B)$

An Attack on Public-Key Needham-Schroeder

Assumption: Alice does not know Mallory is malicious

- Assumption: Alice talks to Mallory
- 1.1 Alice to Trent: A, M
- 1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
- 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$

Same
Same
Same

RECALL!

1.1 Alice to Trent: A, M
1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
1.4 Mallory to Trent: M, A
1.5 Trent to Mallory: $E_{T_{priv}}(A_{pub}, A)$
1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_M)$
1.7 Alice to Mallory: $E_{M_{pub}}(N_M)$

An Attack on Public-Key Needham-Schroeder

- Assumption: Alice talks to Mallory
- 1.1 Alice to Trent: *A*, *M*
- 1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
- 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
- 2.3 Mallory(Alice) to Bob: $E_{B_{pub}}(N_A, A)$

Mallory makes up this msg.

RECALL!

1.1 Alice to Trent: *A*, *M*

1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$

1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$

1.4 Mallory to Trent: *M*, *A*

1.5 Trent to Mallory: $E_{T_{priv}}(A_{pub}, A)$

1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_M)$

1.7 Alice to Mallory: $E_{M_{pub}}(N_M)$

Bob thinks this has happened:

2.1 Alice to Trent: *A*, *B*

2.2 Trent to Alice: $E_{T_{priv}}(B_{pub}, B)$

2.3 Alice to Bob: $E_{B_{pub}}(N_A, A)$

An Attack on Public-Key Needham-Schroeder

- Assumption: Alice talks to Mallory
- 1.1 Alice to Trent: A, M
- 1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
- 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$

Fake invented message

- 2.3 Mallory(Alice) to Bob: $E_{B_{pub}}(N_A, A)$
- 2.4 Bob to Trent: B, A
- 2.5 Trent to Bob: $E_{T_{priv}}(A_{pub}, A)$
- 2.6 Bob to Mallory(Alice): $E_{A_{pub}}(N_A, N_B)$

RECALL!

1.1 Alice to Trent: A, M
1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
1.4 Mallory to Trent: M, A
1.5 Trent to Mallory: $E_{T_{priv}}(A_{pub}, A)$
1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_M)$
1.7 Alice to Mallory: $E_{M_{pub}}(N_M)$

did not happen

2.1 Alice to Trent: A, B
2.2 Trent to Alice: $E_{T_{priv}}(B_{pub}, B)$
2.3 Alice to Bob: $E_{B_{pub}}(N_A, A)$
2.4 Bob to Trent: B, A
2.5 Trent to Bob: $E_{T_{priv}}(A_{pub}, A)$
2.6 Bob to Alice: $E_{A_{pub}}(N_A, N_B)$

intercepted by Mallory

An Attack on Public-Key Needham-Schroeder

- Assumption: Alice talks to Mallory
- 1.1 Alice to Trent: A, M
- 1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
- 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
- 2.3 Mallory(Alice) to Bob: $E_{B_{pub}}(N_A, A)$
- 2.4 Bob to Trent: B, A
- 2.5 Trent to Bob: $E_{T_{priv}}(A_{pub}, A)$
- 2.6 Bob to Mallory(Alice): $E_{A_{pub}}(N_A, N_B)$
- 1.4 Mallory to Trent: M, A
- 1.5 Trent to Mallory: $E_{T_{priv}}(A_{pub}, A)$
- 1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_B)$
- 1.7 Alice to Mallory: $E_{M_{pub}}(N_B)$
- 2.7 Mallory(Alice) to Bob: $E_{B_{pub}}(N_B)$

RECALL!

1.1 Alice to Trent: A, M
1.2 Trent to Alice: $E_{T_{priv}}(M_{pub}, M)$
1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
1.4 Mallory to Trent: M, A
1.5 Trent to Mallory: $E_{T_{priv}}(A_{pub}, A)$
1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_M)$
1.7 Alice to Mallory: $E_{M_{pub}}(N_M)$

2.1 Alice to Trent: A, B
2.2 Trent to Alice: $E_{T_{priv}}(B_{pub}, B)$
2.3 Alice to Bob: $E_{B_{pub}}(N_A, A)$
2.4 Bob to Trent: B, A
2.5 Trent to Bob: $E_{T_{priv}}(A_{pub}, A)$
2.6 Bob to Alice: $E_{A_{pub}}(N_A, N_B)$
2.7 Alice to Bob: $E_{B_{pub}}(N_B)$

Solution to PK Needham-Schroeder Attack

- Include identities with nonces!
- 2.6 Bob to Mallory(Alice): $E_{A_{pub}}(B, N_A, N_B)$
    - 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
    - 2.3 Mallory(Alice) to Bob: $E_{B_{pub}}(N_A, A)$
    - 2.6 Bob to Mallory(Alice): $E_{A_{pub}}(B, N_A, N_B)$
    - 1.6 Mallory to Alice: $E_{A_{pub}}(B, N_A, N_B)$
    - 1.7 Alice does not proceed

Recall!

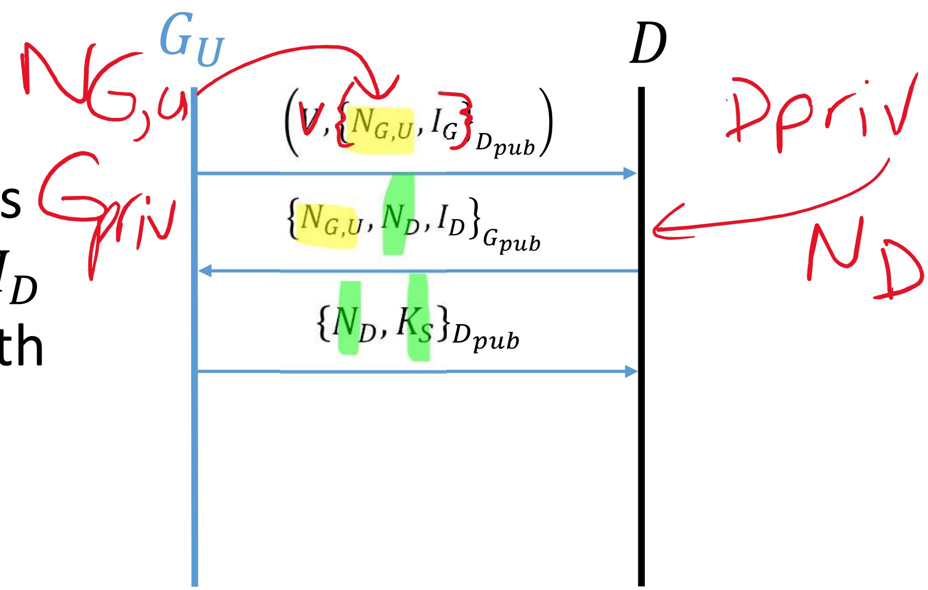
- 1.3 Alice to Mallory: $E_{M_{pub}}(N_A, A)$
- 2.3 Mallory(Alice) to Bob: $E_{B_{pub}}(N_A, A)$
- 2.6 Bob to Mallory(Alice): $E_{A_{pub}}(N_A, N_B)$
- 1.6 Mallory to Alice: $E_{A_{pub}}(N_A, N_B)$
- 1.7 Alice to Mallory: $E_{M_{pub}}(N_B)$
- 2.7 Mallory(Alice) to Bob: $E_{B_{pub}}(N_B)$

Notation

- D: target device
- G_U: updating organization
- (G_{pub}, G_{prv}): updating organization key pair
- (D_{pub}, D_{prv}): device key pair
- N_G, N_D: organization and device nonces
- I_G, I_D: organization and device identifiers
- V: incoming update version number
- K_S: symmetric key
- U: update image
- H: hash of the update image
- H_U: update hashes sent by G_U
- $\{M\}_{D_{pub}}$: message M is encrypted using key D_{pub}
  - Notation is common to both symmetric and asymmetric encryption
- $(G \rightarrow D : M)$: organization G sends M to device D
- $(G \leftarrow D : M)$: device D sends M to organization G

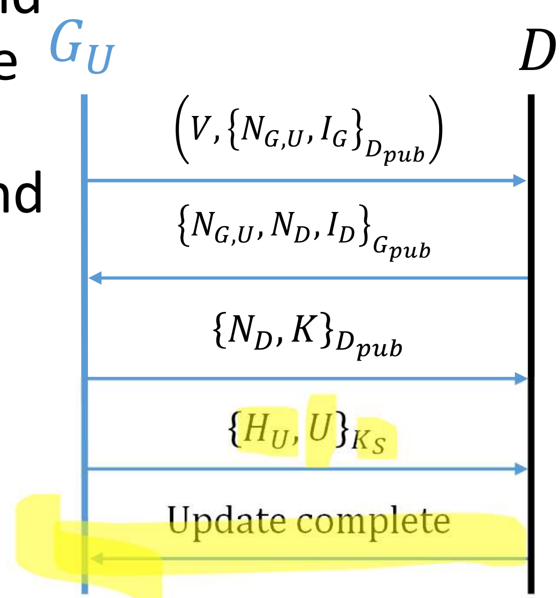
Authentication Phase Using Public Key Crypto

1. Organization nonce N_G and identifier I_G sent to device
2. Device retrieves N_G, and appends its own nonce N_D and identifier I_D
3. Finally, organization responds with N_D and symmetric key K_S



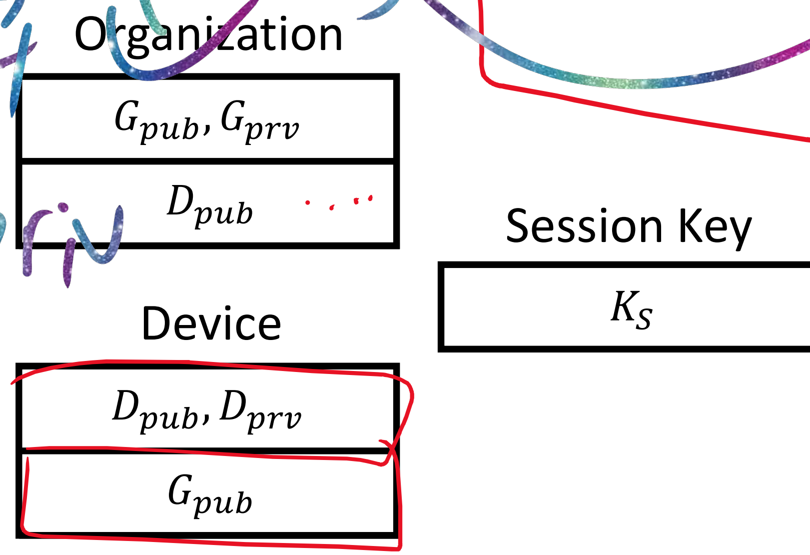
Update Phase Using Symmetric Key Crypto

4. Organization sends update U and hash of the update H_U using the and symmetric key K_S
5. Device decrypts the message and checks that the (keyless) hash value H_U is obtained on the update U
6. Finally, D sends an encrypted message indicating that the update is complete



Long Term Asymmetric Keys, Short Term Symmetric Session Key

- New symmetric session key generated by updating organization on every update
  - Shared during authentication phase
- Advantages
  - Decryption of update code faster than asymmetric
  - Higher security
- Disadvantages
  - Device has a higher implementation overhead in order to support asymmetric as well as symmetric crypto

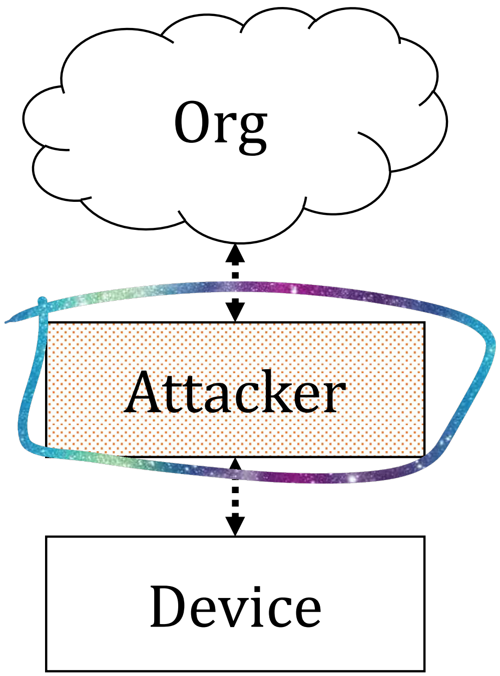


Security Analysis

1. Man in the middle
2. Replay attack
3. Organization spoofing

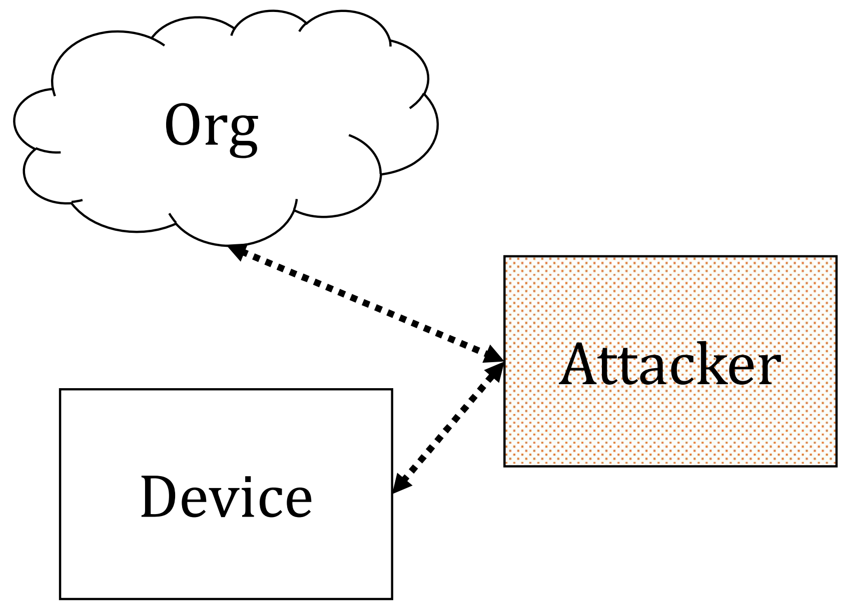
Man in the Middle

- Attacker tries to place himself between the updating organization and the device
- Attack fails because
  1. Authentication requires possession of private key
  2. All communication is encrypted
- Note that the assumption is that the public keys are correct



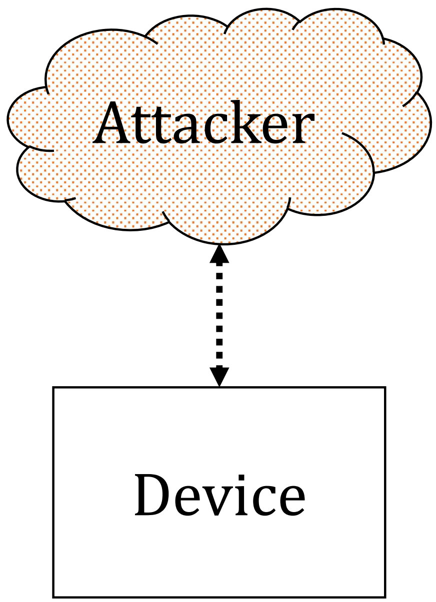
Replay Attack

- Attacker saves previous authentication and replays it
- Replay will be denied
  - Nonce used prevents successful replay



Organization Spoofing

- Attacker claims to be the updating organization
  - Pushes out malicious update
- Authentication will fail
  - Organization public key statically stored on Device
- Device will deny the update



Lessons Learned

- Do not try to be too clever; do not remove important pieces
  - Names
  - Random numbers
  - Timestamps
- Focus on what has worked in the past and has not yet been broken; optimizing a protocol will often break it
- What is your communications need?
  - Client-server
  - Many to many
- Time synchronization can be a big issue
- Recovery