

Authentication II

Cryptographic Hardware for Embedded Systems

ECE 3170

Fall 2026

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

Reading Assignment

- Please continue reading Chapter 3 of the course textbook by Schneier

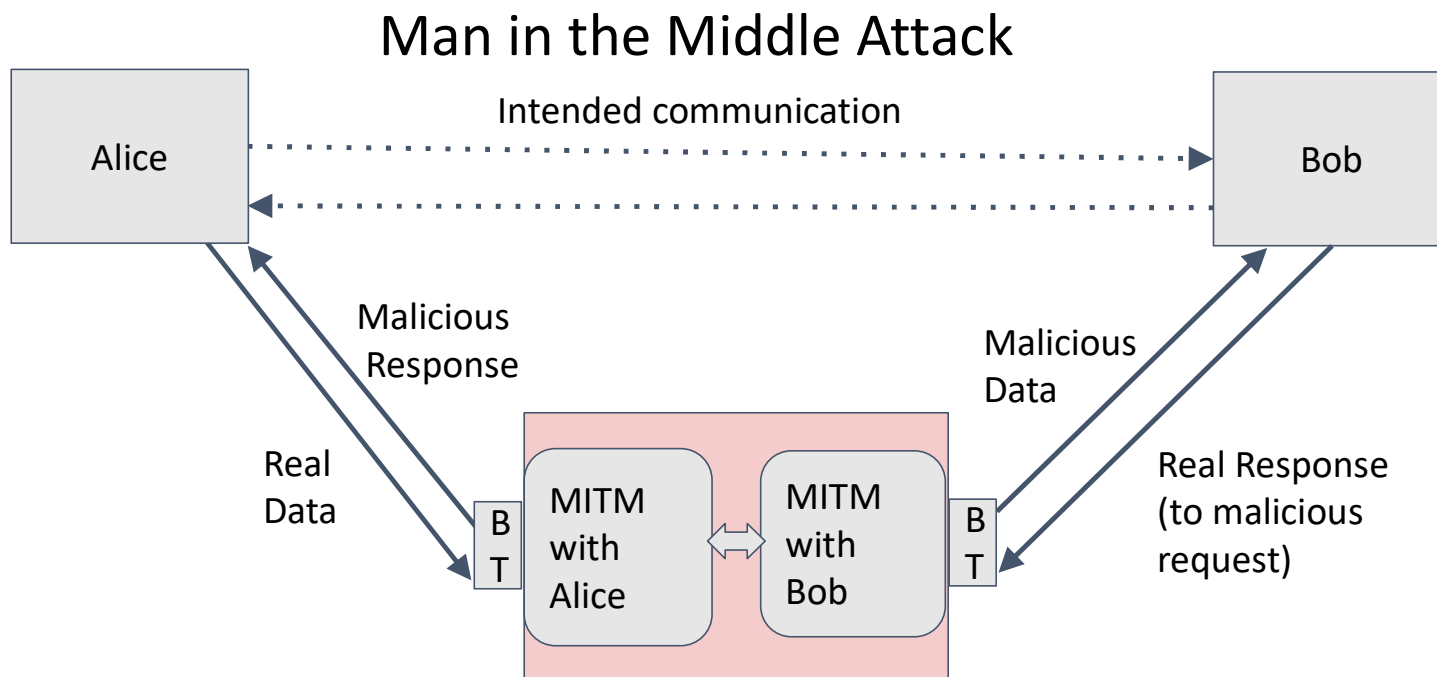
Characters in Use in This Lecture

- Alice is an honest participant in the protocols
- Bob is an honest participant in the protocols
- Mallory is a malicious active attacker
- Trent is a trusted arbiter

Interlock Protocol

- The interlock protocol aims to foil a Man-in-the-Middle attack
 - 1) Alice sends Bob her public key
 - 2) Bob sends Alice his public key
 - 3) Alice encrypts a message for Bob but only sends half of the message
 - 4) Bob encrypts a message for Alice but only sends half of the message
 - 5) Alice sends the rest of her message to Bob
 - 6) Bob puts together both halves of Alice's message and decrypts it; then he sends the rest of his message to Alice
 - 7) Alice puts together both halves of Bob's message and decrypts it

Threat Scenario



Ways to Require Both Halves For Decryption

- Send every other bit with each half
- Use an initialization vector (IV) which is not sent until the second half is sent
- The first “half” could be a hash of the message

What Exactly Does the Interlock Protocol Stop?

Key Exchange with Digital Signatures

- Trent signs Alice's key and Bob's key
 - $Dec_{Private_Trent}(Public_{Alice})$
 - $Dec_{Private_Trent}(Public_{Bob})$
- Note that even if Mallory later breaks into Trent's server and obtains Trent's private key $Private_{Trent}$, Mallory cannot obtain session keys exchanged between Alice and Bob
 - Why not?
- However, Mallory can now potentially carry out Man-in-the-Middle attacks for those who have yet to obtain public keys of others

Password Comparison Using a Hash

- A server need not store every user's password, only the hash
- Alice sends the server her password
- The server calculates the hash of the password
- The server compares the calculated hash value with a stored value of the hash
- If the server is compromised, Alice's password is not revealed

However...

- Most passwords are “weak”
- Dictionary attack
 - Rather than try all possible ASCII combinations, use a dictionary
 - E.g., try all words of eight letters or less, including proper names, and add random ASCII characters to make the length equal to eight
 - Can try also with the first letter capitalized or not

For Every Attack There is a Countermeasure

- Salt
 - Before applying the one-way function (i.e., the hash), use a random number
 - Append the random number to the end of the password
 - XOR the password with a random number of equal length
 - Store each password's random number (i.e., "salt") in a different location
 - E.g., in a separate filesystem with separate access privileges

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...

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
 - $E_K(T+1)$

Needham-Schroeder (1978)

- Alice to Trent: A, B, R_A
- Trent to Alice: $E_A(R_A, B, K, E_B(K, A))$
- Alice to Bob: $E_B(K, A)$
- Bob to Alice: $E_K(R_B)$
- Alice to Bob: $E_K(R_B - 1)$

An Attack on Needham-Schroeder

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

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