

Authenticity

Authentication II

Cryptographic Hardware for Embedded Systems

ECE 3170

Fall 2026

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

UAV Alice

MUAV

UAV Bob

77

Starting
pt.

No
public
verified
keys

Reading Assignment

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

Launch

UAV₁

Alice

Bob
UAV₂

new just fixed
motherboard

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

only one
attacker

CA

Trusted
Third
Party
TTP

Also
no TTPs

$C[2047:0]$
odd $c = C[2047]$ $C[2045]$ $C[2043]$ $C[2041]$
even $c = C[2046]$ $C[2044]$ $C[2042]$ $C[2040]$
message = 128 bits

Interlock Protocol

NO TTP

needed each other's public keys

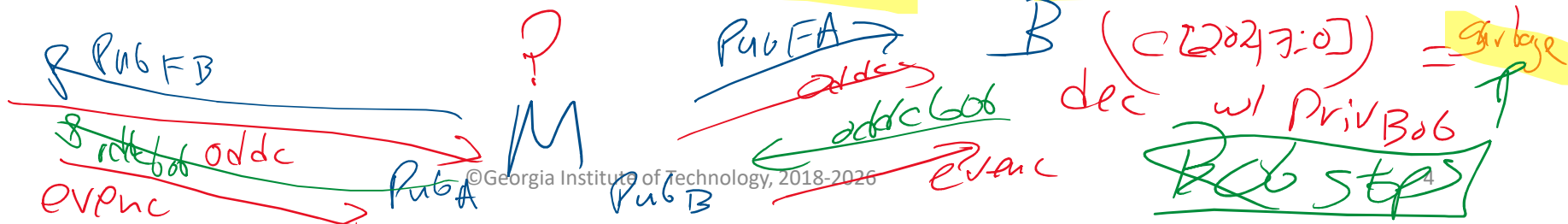
- The interlock protocol aims to foil a Man-in-the-Middle attack

- Alice sends Bob her public key Pub_A
- Bob sends Alice his public key Pub_B
- Alice encrypts a message for Bob but only sends half of the message
- Bob encrypts a message for Alice but only sends half of the message
- Alice sends the rest of her message to Bob
- Bob puts together both halves of Alice's message and decrypts it; then he sends the rest of his message to Alice
- Alice puts together both halves of Bob's message and decrypts it

$\{Bmes\} Priv_B$

$\{Amsg\} Priv_A$

A

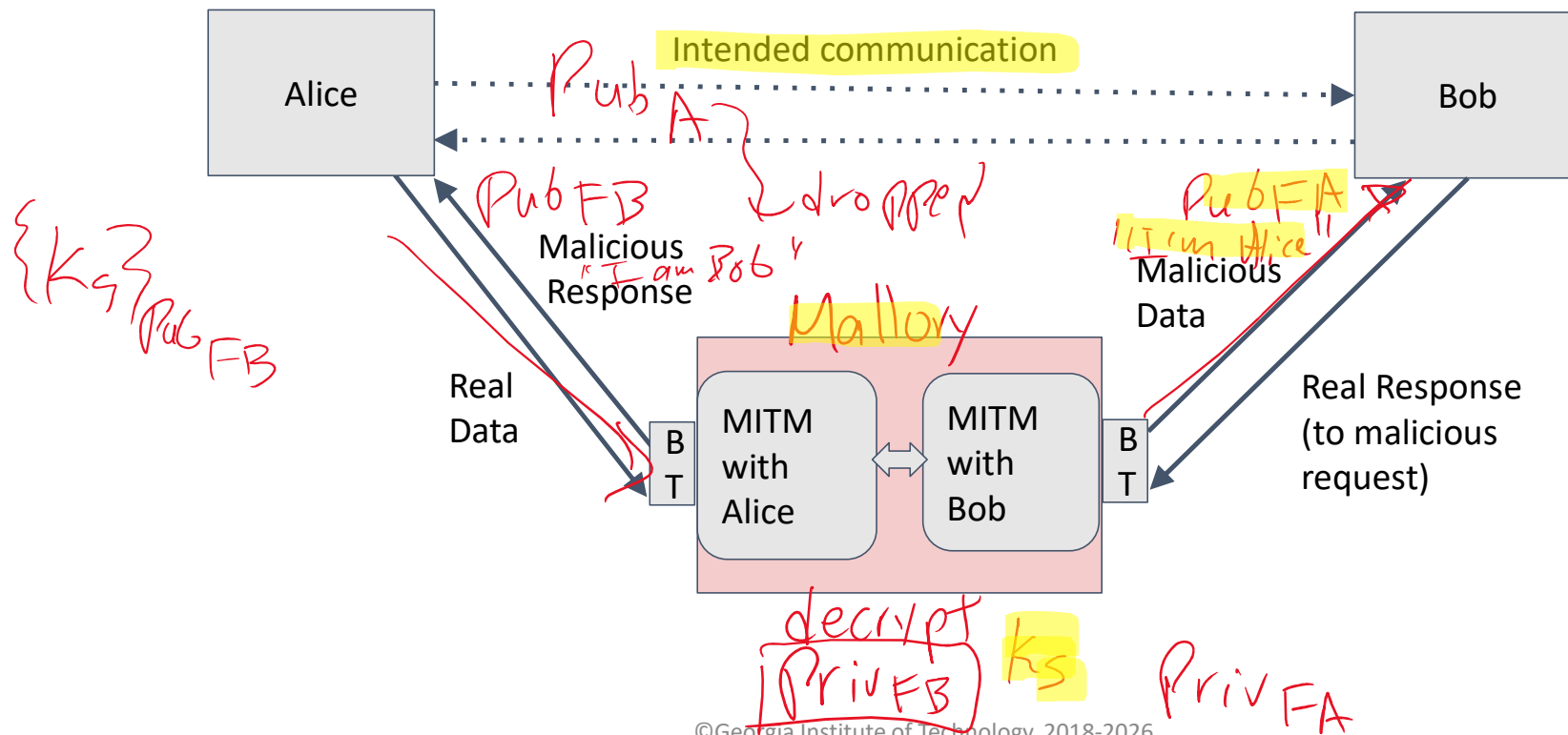


Bluetooth (Not part of class) "just works" legacy Threat Scenario

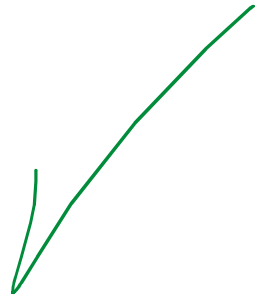
Mallory

Man in the Middle Attack

on Public Key exchange



claim: Interlock Protocol
prevents MITM for
one execution / one "run"
of the protocol



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

If (i) initial messages of
Bob & Alice do not
change (e.g., same day)
and (ii) session reset allowed,
Mallory can break the
Interlock Protocol

3rd time
"hello20260917"

"good morning20260917"

M
succeeds
w/ MITM

Q1: how do Bob & Alice have
messages they know but
others do not?

A1: agreed ahead of time

What Exactly Does the Interlock Protocol Stop?

MITM first time through

None of the protocols taught in EE3170 prevent Denial of Service (DoS)

Can a "fake" UAV
spot Bob?

Answer: not unless Bob's
message can be figured
out

Can Interlock Protocol
prevent replay attacks?

Answer: No, assuming message
does not change

recall: in replay, all transmitted is
recorded & sent exactly as before

In practice, can use time
granularity to milliseconds,
add nonces, then the
messages are never the same

Need: Threat Scenario

If Trent is hacked,
can adversary carry
out future attacks
successfully?

Yes

But past sessions are safe due
to secret session keys not known to Trent

Key Exchange with Digital Signatures

have TTP Trent

- 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

A

B

$\{N_A\}_{Pub_B}$
 $\xrightarrow{\quad}$
 $\{N_A, N_B\}_{K_S}$
 $\xrightarrow{\quad}$
 $\{N_B\}_{K_S}$

K_S

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...

CEO Coo
CIO CISO

- 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

✓
polynomial
time

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

hash("password" + "random")

Number used only Once (NONCE)

⇒ need
high
quality

- 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

random
number

Problem: if RNG broken,
now NONCE can be guessed

gen
(RNG)

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 ^{entity} authentication
 - Server authenticates Alice
 - Alice authenticates the server
 - Then the next steps proceed...

Before RSA, only sym. keys could be used
Kerberos w/ symmetric keys

→ Trent has symmetric keys for everyone

- 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

K_A $\bullet E_A(T, L, K, B); E_B(T, L, K, A)$ K_B

- Alice then uses K to send Bob her identity and timestamp, plus Trent's message T, L, K, B

$\bullet E_K(A, T); E_B(T, L, K, A)$

- Bob creates a message consisting of the timestamp plus one, encrypts it in K , and sends it to Alice T, L, K, A

$\bullet E_K(T+1)$

Result:
Alice + Bob timestamp T , lifetime L ,
 K

Note that Alice does not receive K_B
Bob does not receive K_A

What if a hacker breaks into
the computer w/ Trent's sym. key?

Note: assume all network traffic
captured/archived

Are past sessions available
to attacker in plaintext?

Yes

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: $E_K(R_B - 1)$

Not
Covered
this
year
(already in ECE 4158 H05)

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: $E_K(R_B^{-1})$

Not
covered
this
year
No
test

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

from breaks