Misbinding Attacks on Secure Device Pairing

Tuomas Aura, Aalto University, Finland

joint work with Mohit Sethi, Ericsson, and
Aleksi Peltonen, Aalto University

IETF 104, SAAG, Prague

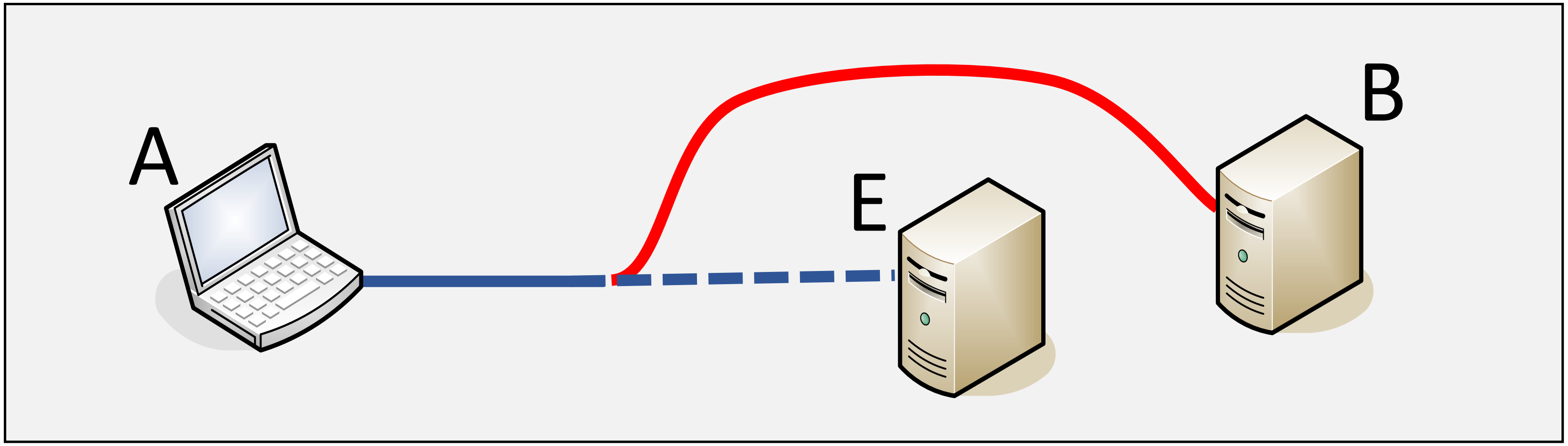
Outline

1. Background:
   misbinding in authenticated key exchange
2. Misbinding in device pairing
   (Bluetooth)
3. Misbinding in connecting devices to cloud
   (EAP-NOOB)

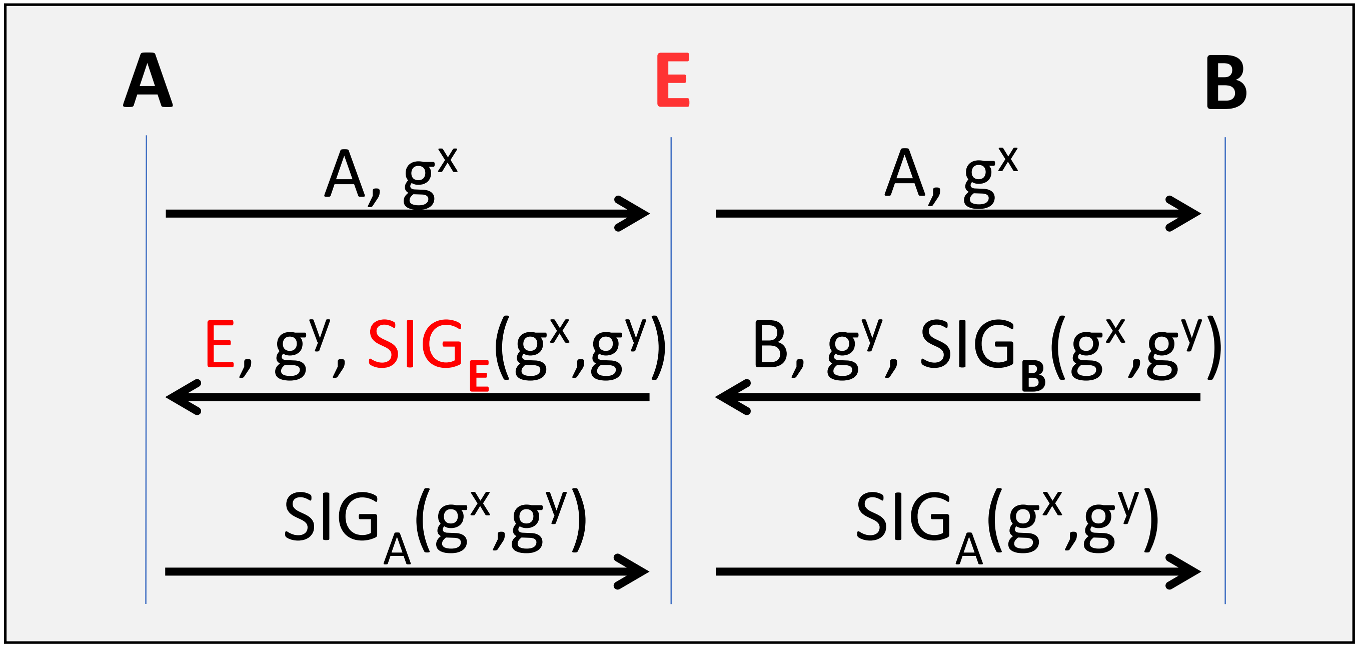
Background: misbinding in authenticated key exchange

Misbinding in key exchange

- **A** thinks it is authenticating to **E**, but it is actually authenticating to **B**
- **E** is dishonest. B can be honest



- Known since 1992 (STS, Diffie et al. 1992) and motivated the SIGMA protocols (IKEv1, IKEv2)
- Named unknown key-share, misbinding, cuckoo



A

E

B

A, g^x

A, g^x

E, g^y, $SIG_E(g^x,g^y)$

B, g^y, $SIG_B(g^x,g^y)$

$SIG_A(g^x,g^y)$

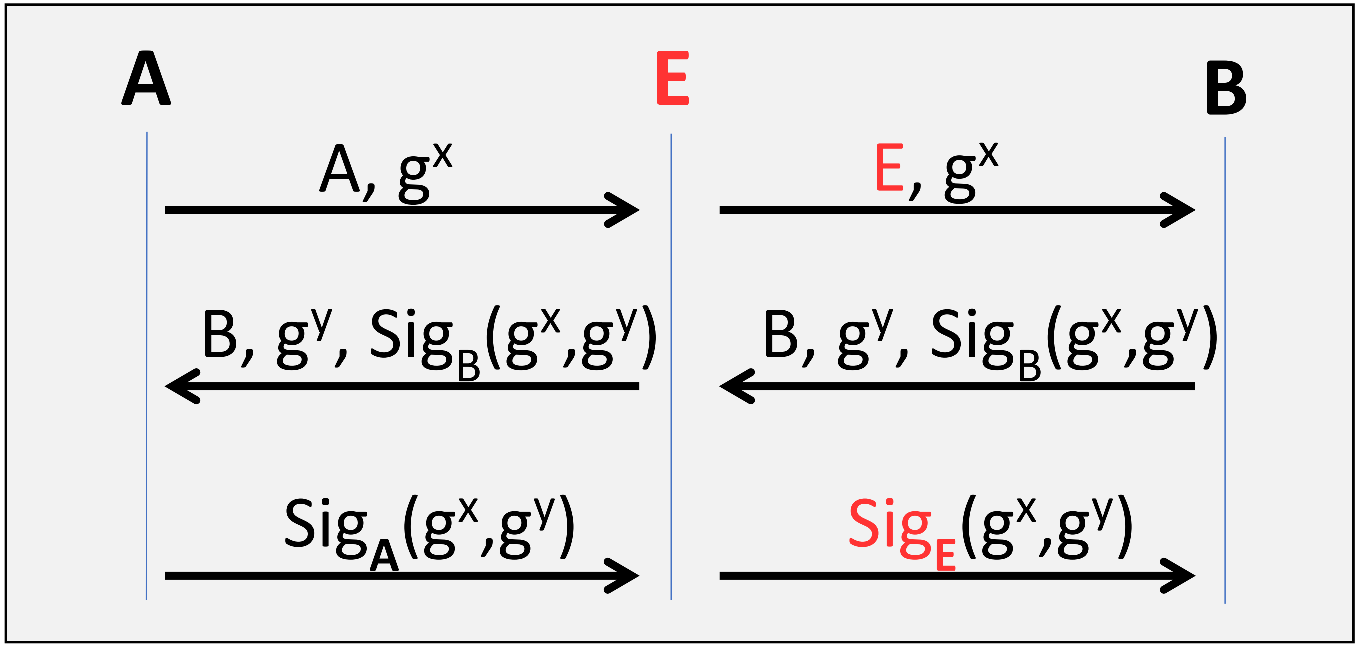
$SIG_A(g^x,g^y)$

Misbinding of responder:

A thinks it is connected to E.

In fact,

A and B are connected



A

E

B

A, g^x

E, g^x

B, g^y, $Sig_B(g^x,g^y)$

B, g^y, $Sig_B(g^x,g^y)$

$Sig_A(g^x,g^y)$

$Sig_E(g^x,g^y)$

Misbinding of initiator:

B thinks it is connected to E.

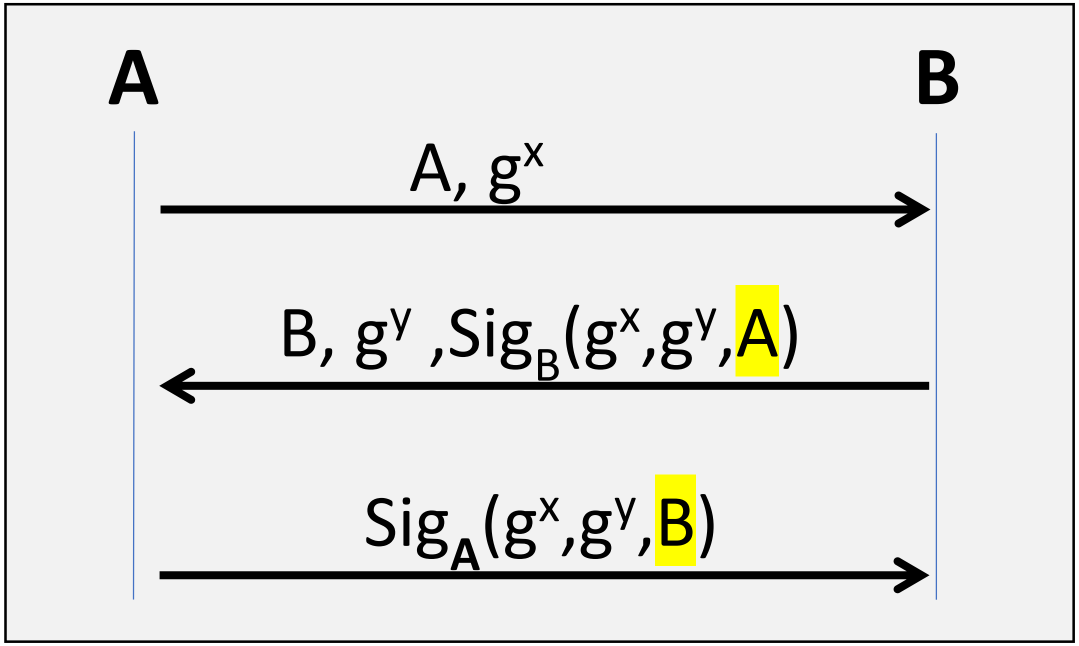
In fact,

A and B are connected.

Solution to misbinding:

be explicit about identities

ISO 9798-3



A

B

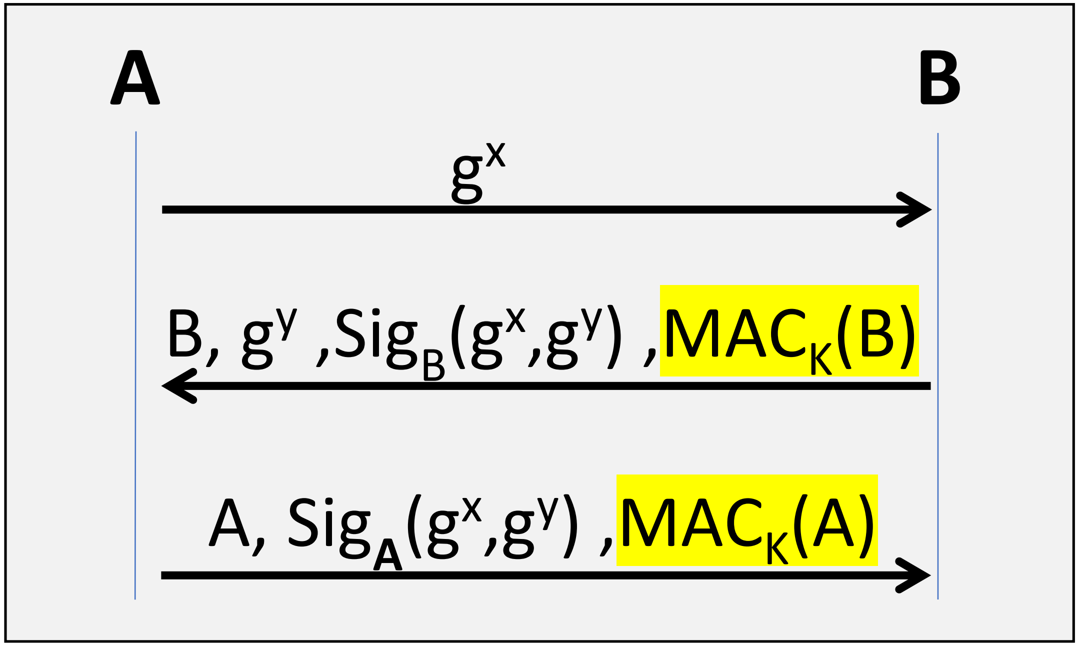
A, g^x

B, g^y ,$\text{Sig}_B(g^x,g^y,A)$

$\text{Sig}_A(g^x,g^y,B)$

SIGMA

(slightly better protection in case of an incompetent CA)



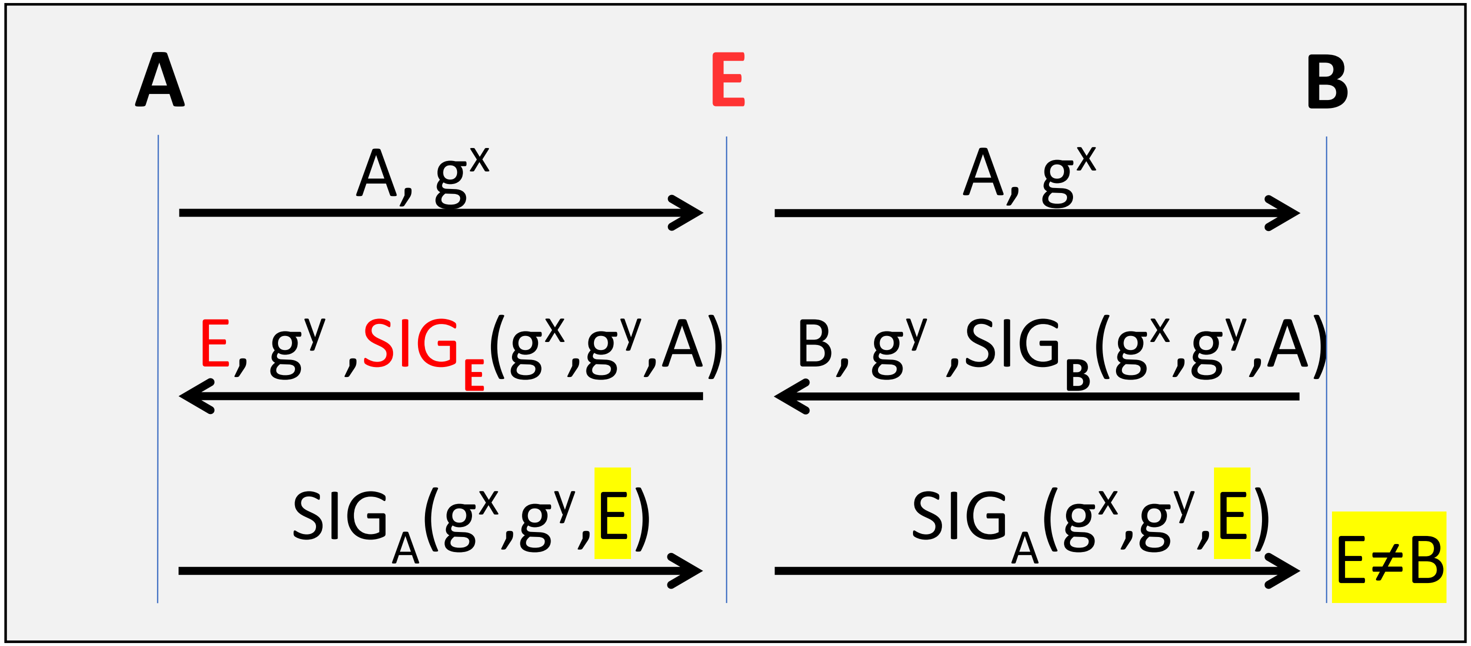
A

B

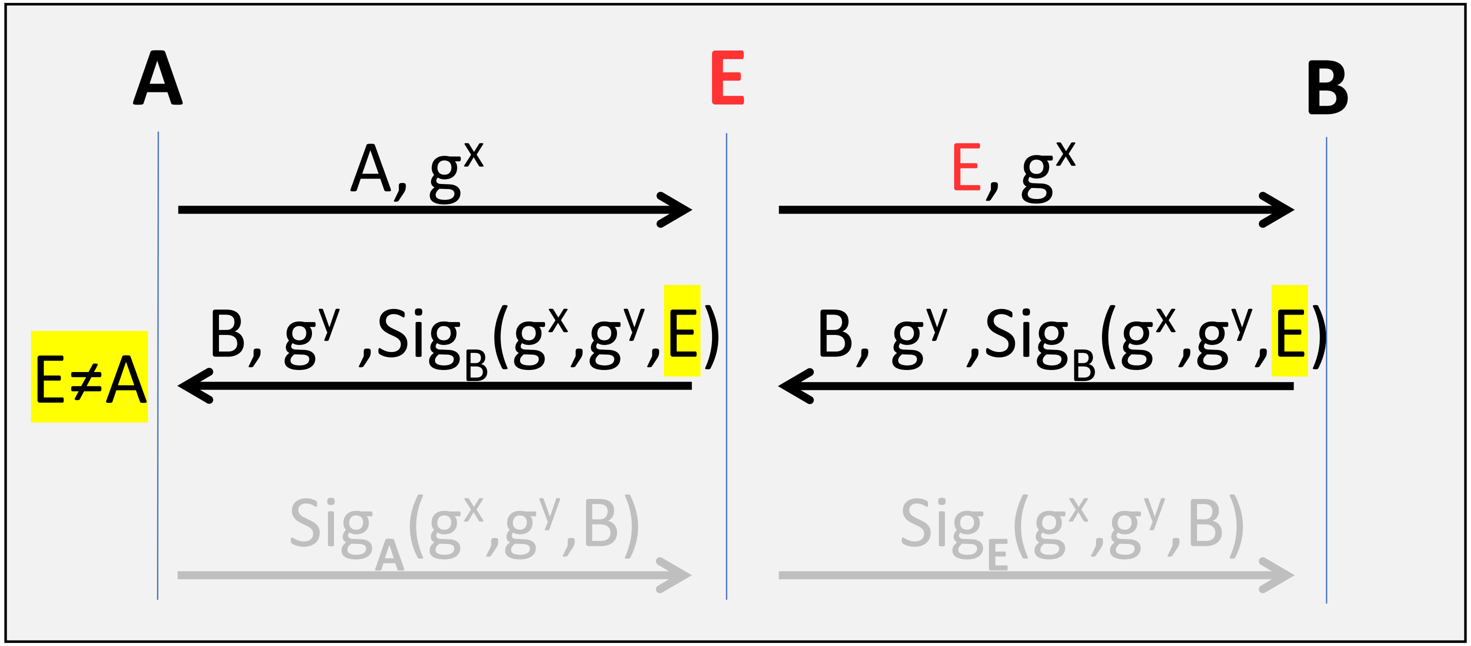
g^x

B, g^y ,$\text{Sig}_B(g^x,g^y)$,$\text{MAC}_K(B)$

A, $\text{Sig}_A(g^x,g^y)$,$\text{MAC}_K(A)$



Detecting misbinding of responder



Detecting misbinding of initiator

How serious is it? (1)

- Seriousness difficult to grasp:
  - No failure of confidentiality. Victim wants to talk with the malicious party E, an thus attacker would get all the secrets even without misbinding
  - Problem related to data authentication. Victim is confused about to who it at the other end of the secure connection
- Attack scenarios in literature are artificial:
  - A is commander, E and B fighter jets. E has been compromised by the enemy. A tells E to self-destruct, but the command goes to B *[Hugo Krawczyk]*
  - A connects to bank B and, over the secure session, deposits an electronic cheque. Bank B thinks the cheque was deposited by E *[Diffie et al.]*

How serious is it? (2)

- Well-defined problem in formal verification: failure of a correspondence property:

  If A and B share session key K,
  *A should think it shares the key K **with B**.*

- Easy to prevent in most protocols: bind endpoint identifiers to the key
- However, must have authenticated identifiers (e.g. certificates) and the other endpoint must know what id to expect

Misbinding in device pairing

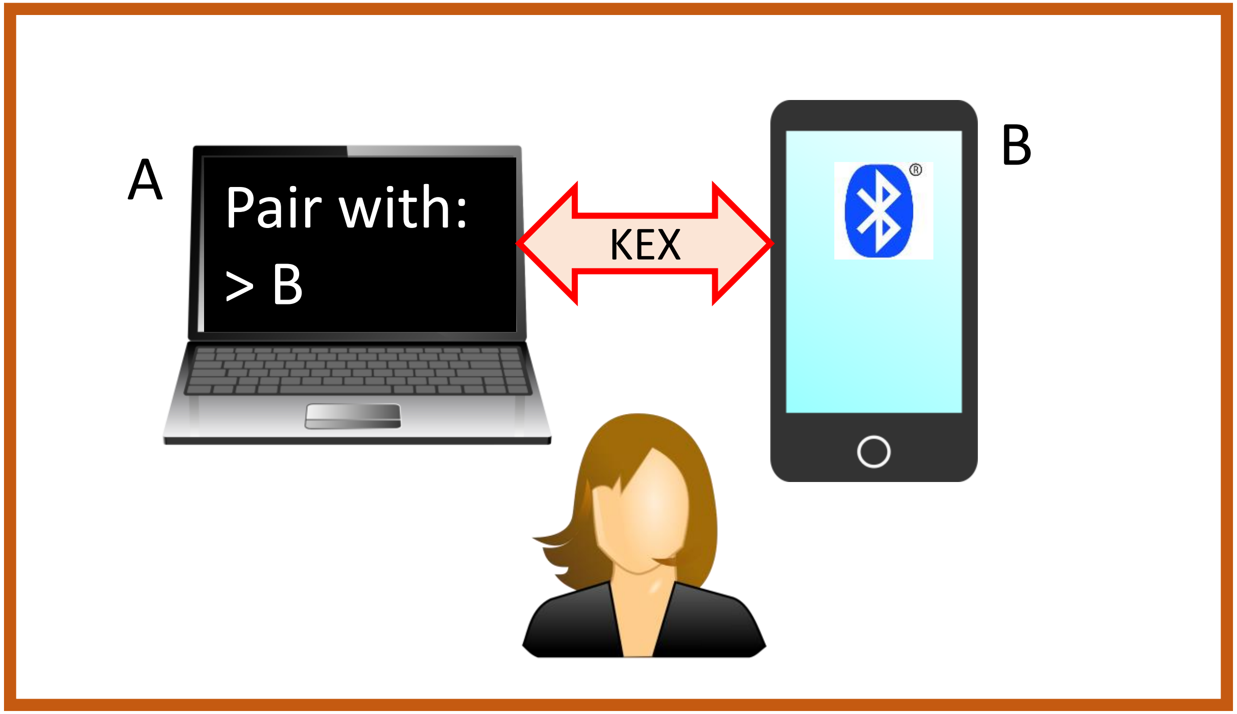
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK → Paired!

A
Pair with:
> B
B

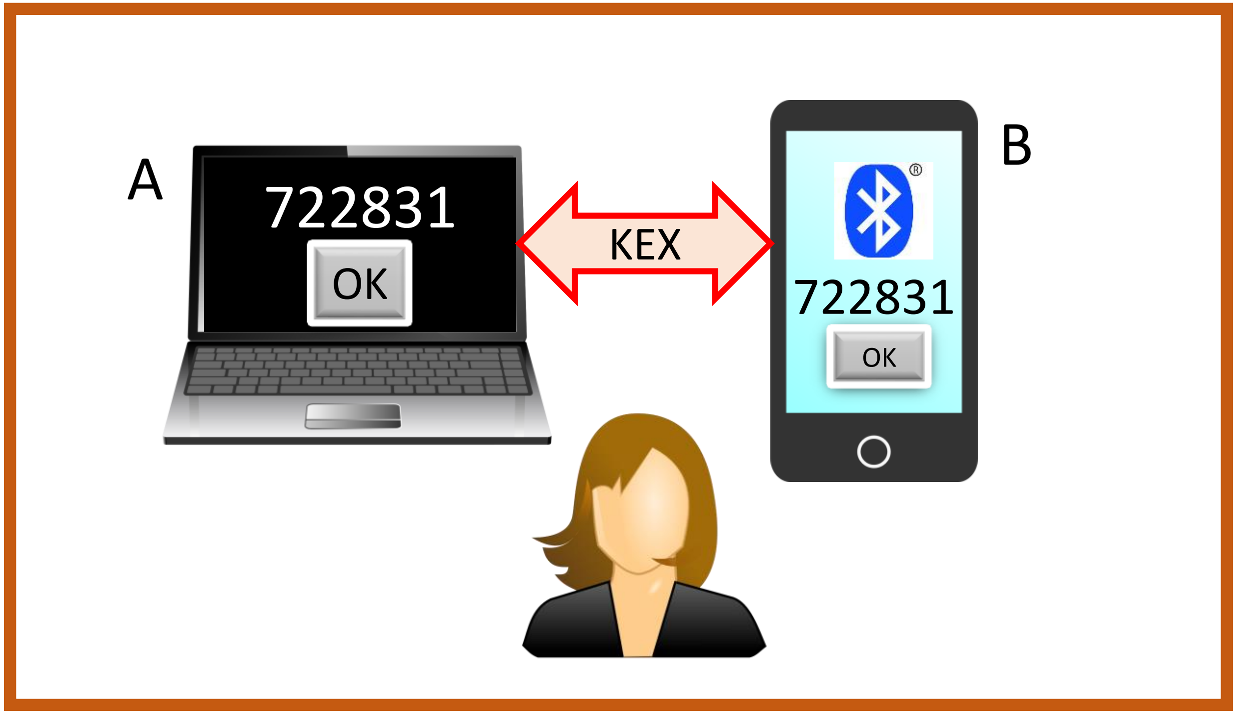
Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK → Paired!



Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK → Paired!



Bluetooth numeric comparison

1. Make device B discoverable
2. On device A, search and select B
3. Key exchange in background
4. Compare 6-digit codes and press OK ➔ Paired!

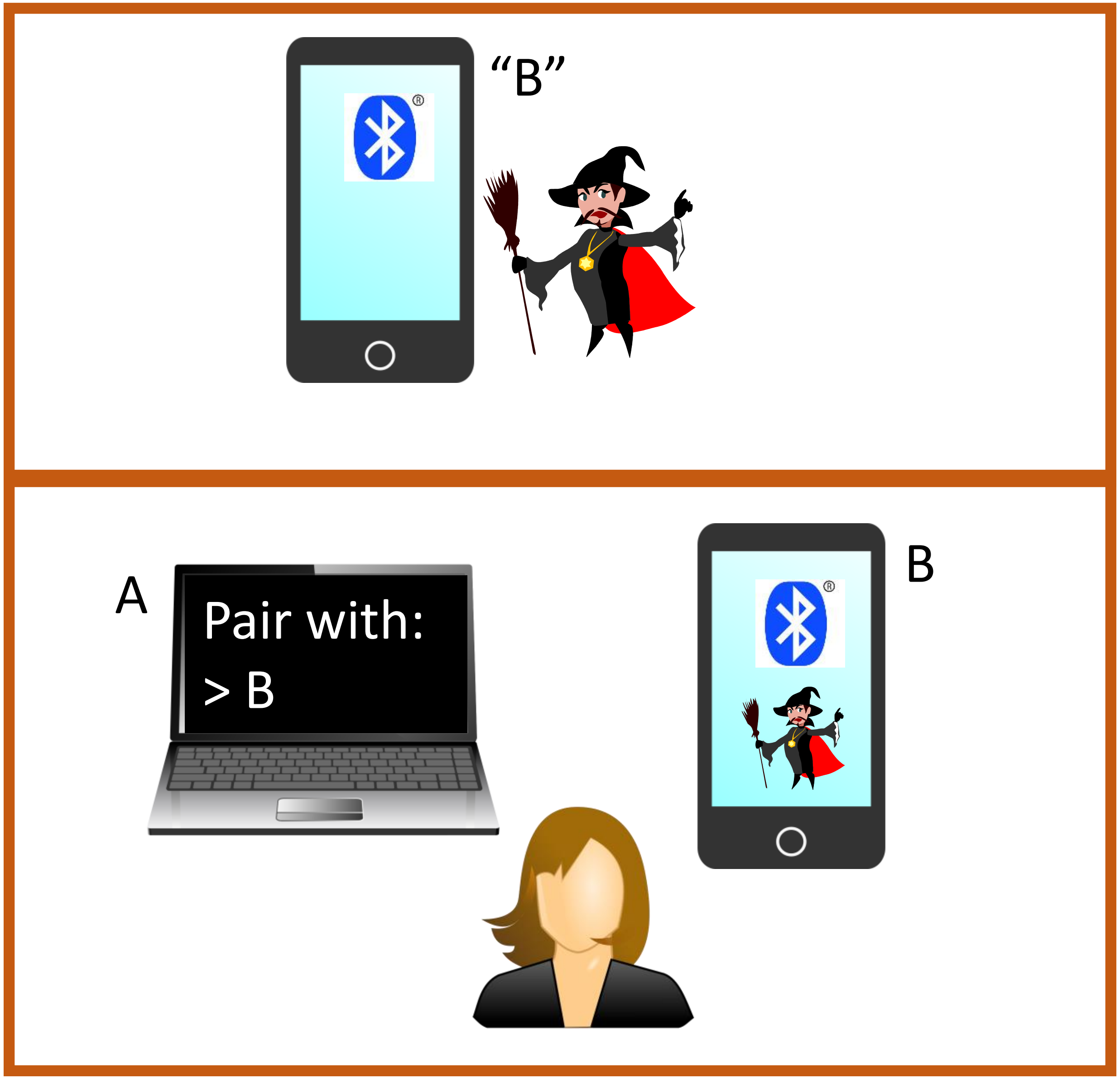
A
Paired!
Paired
B
Paired!

Misbinding in Bluetooth

A
Pair with:
> B
B

Device B is compromised (malicious app)

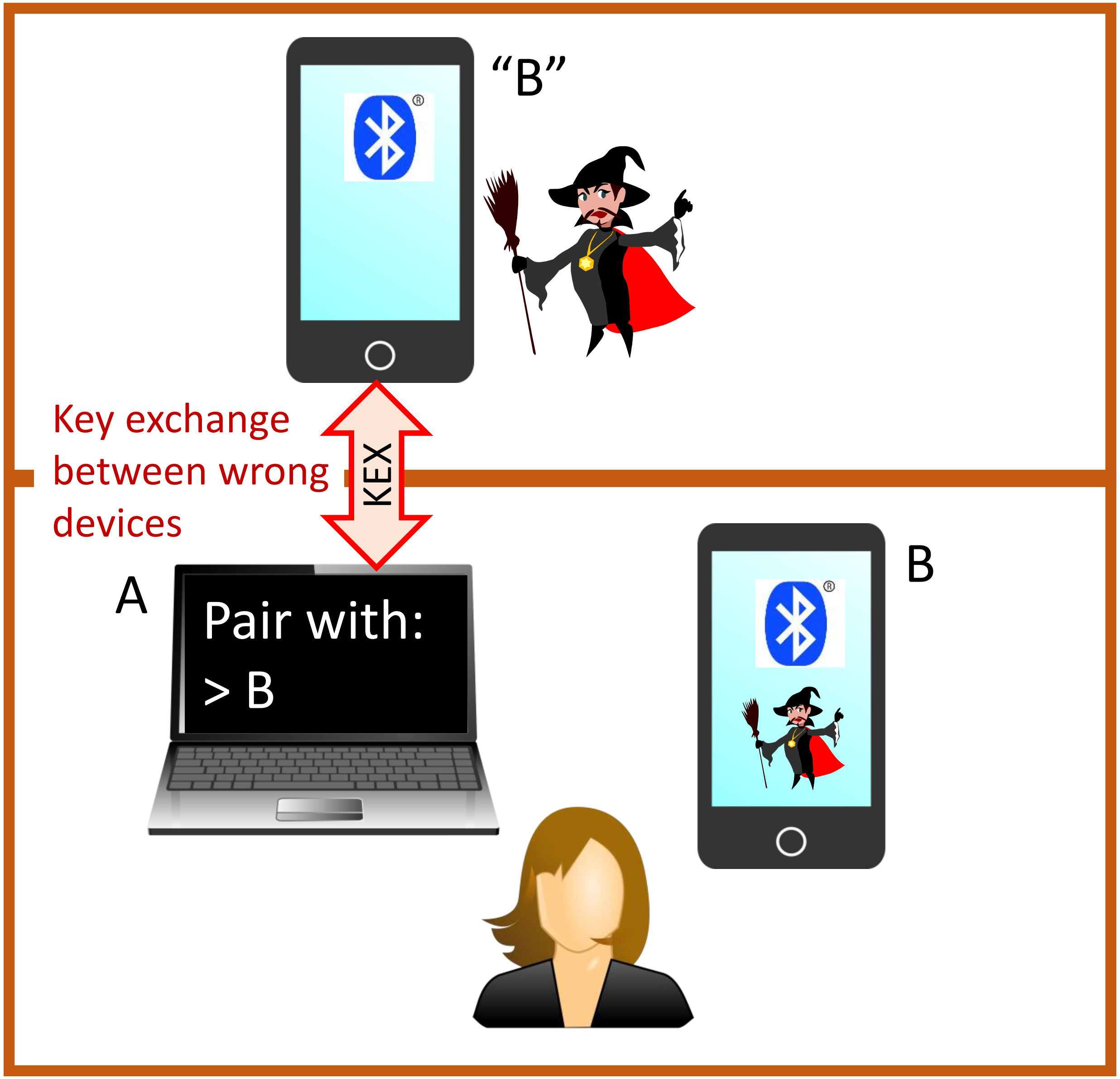
Misbinding in Bluetooth



Attacker
has another
device
named "B"

Device B is
compromised
(malicious app)

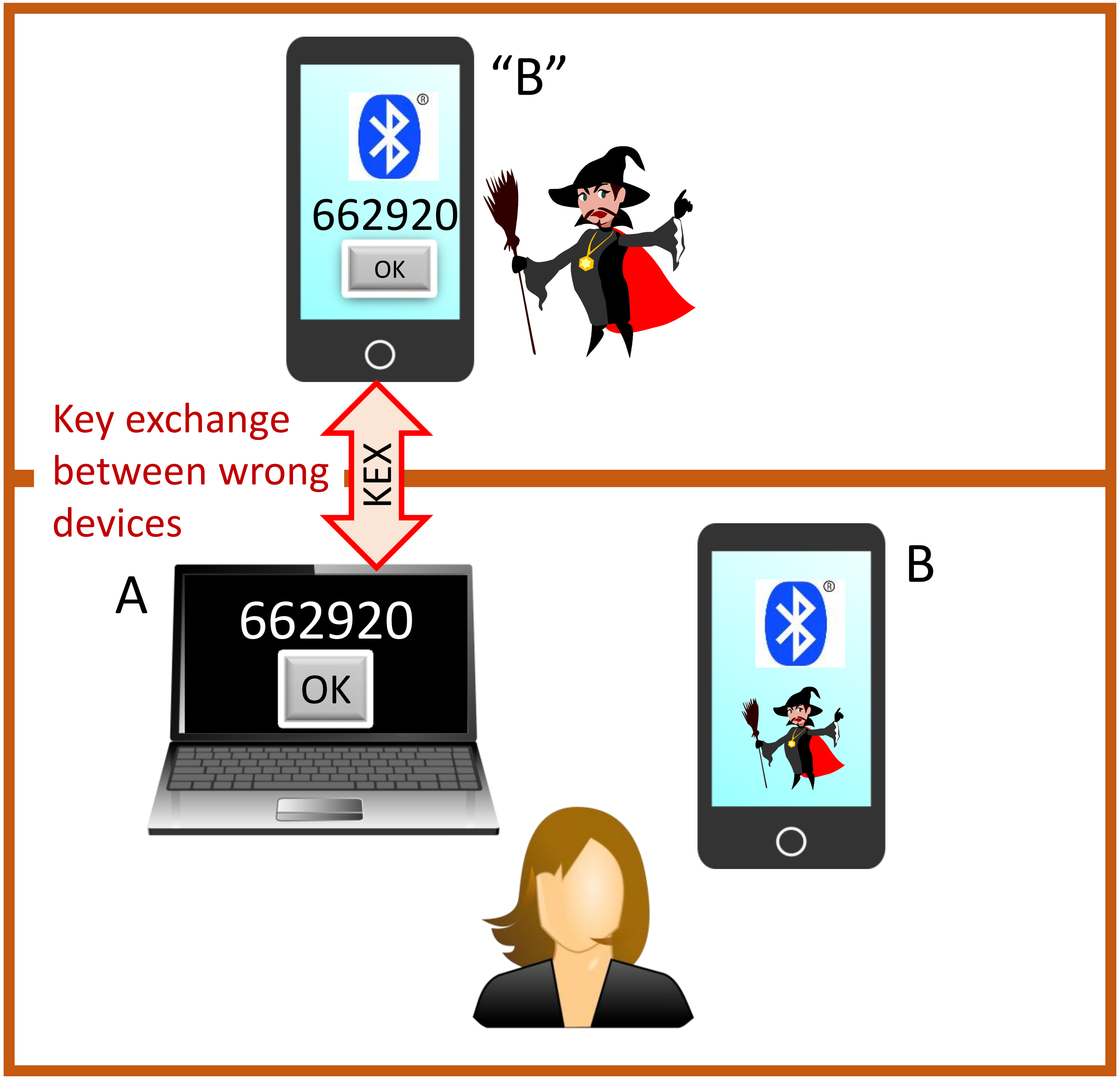
Misbinding in Bluetooth



Attacker has another device named "B"

Device B is compromised (malicious app)

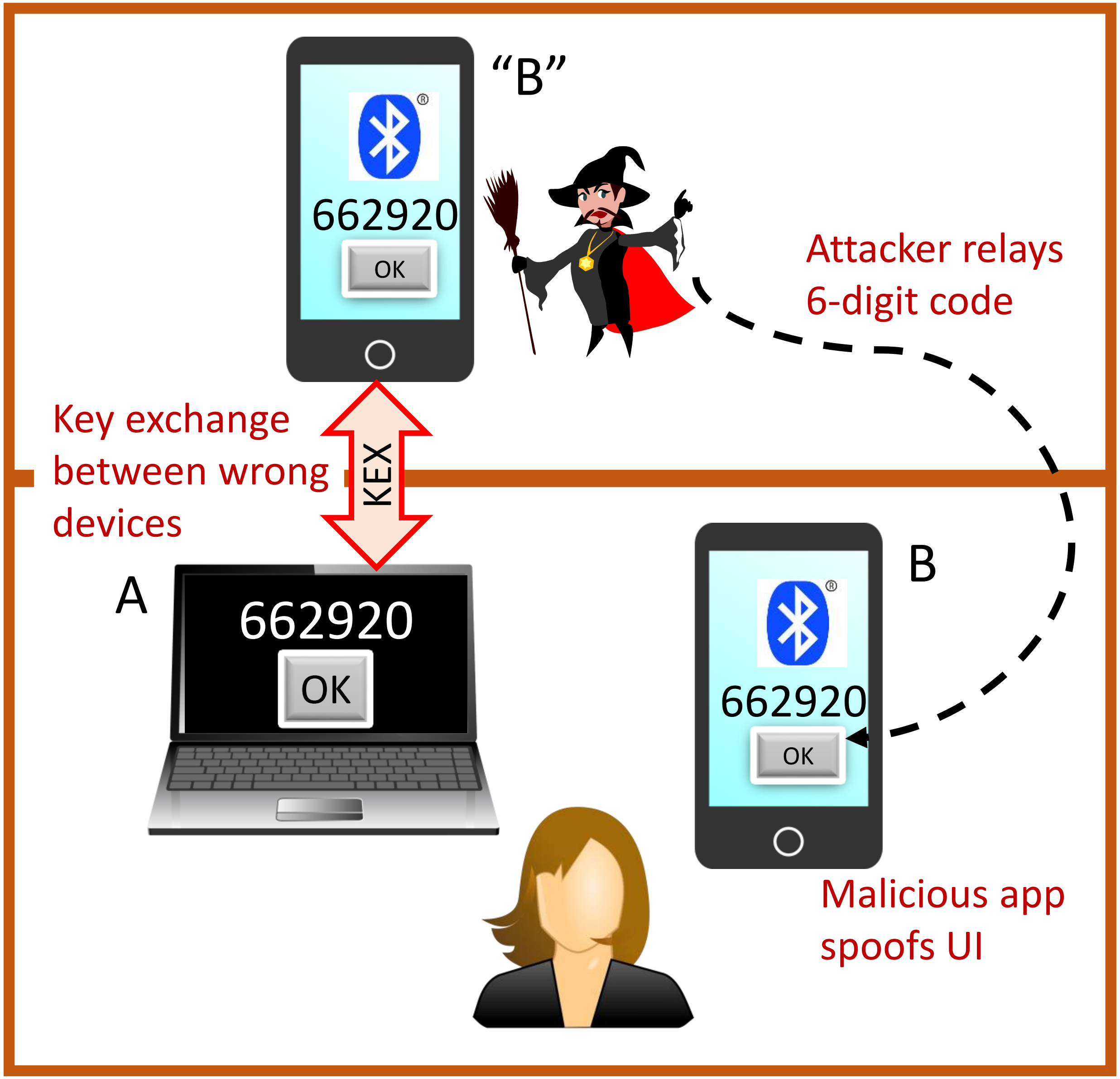
Misbinding in Bluetooth



Attacker has another device named "B"

Device B is compromised (malicious app)

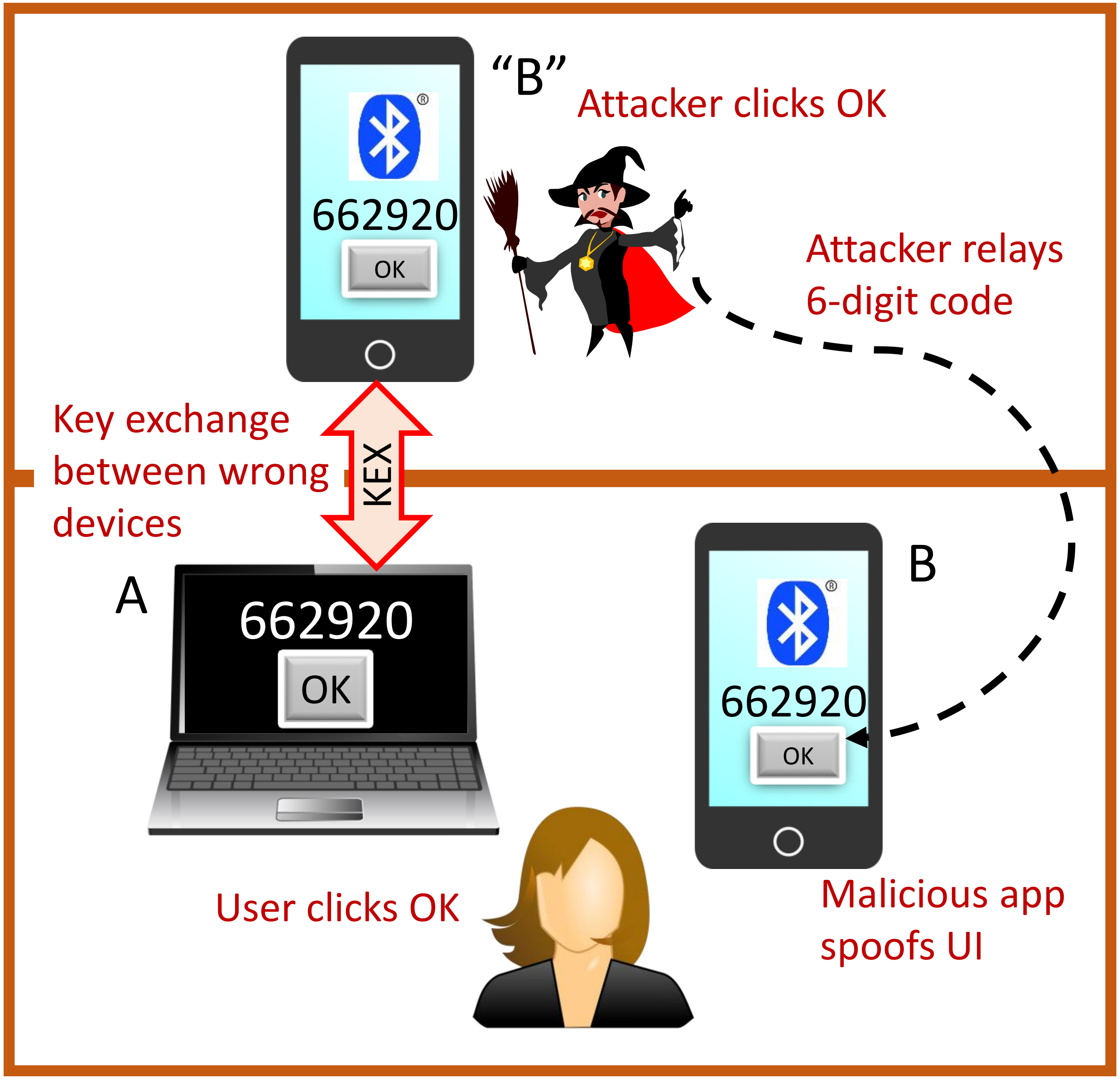
Misbinding in Bluetooth


"B"
662920
OK
Attacker relays 6-digit code
Key exchange between wrong devices
KEX
A
662920
OK
B
662920
OK
Malicious app spoofs UI

Attacker has another device named "B"

Device B is compromised (malicious app)

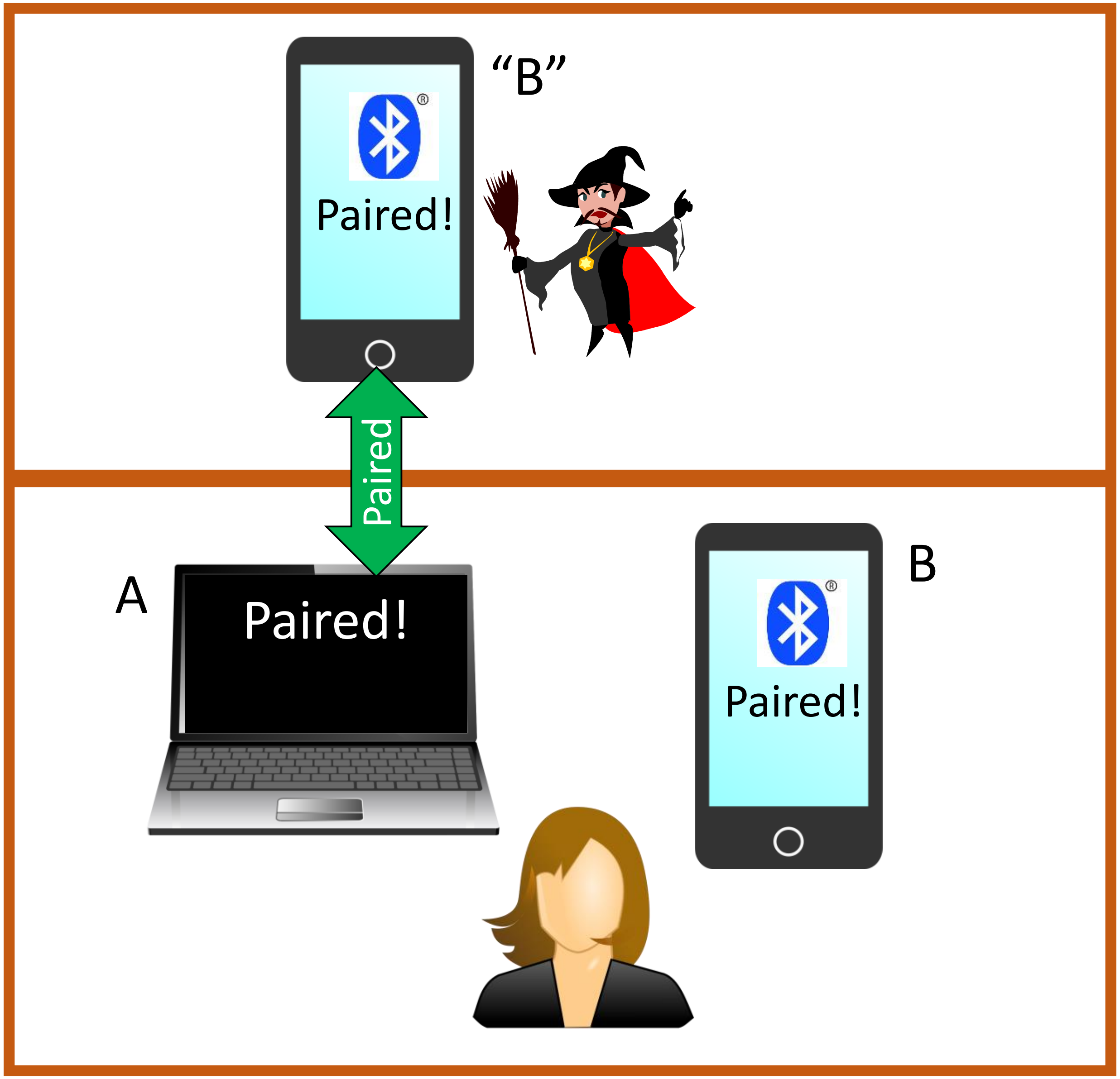
Misbinding in Bluetooth



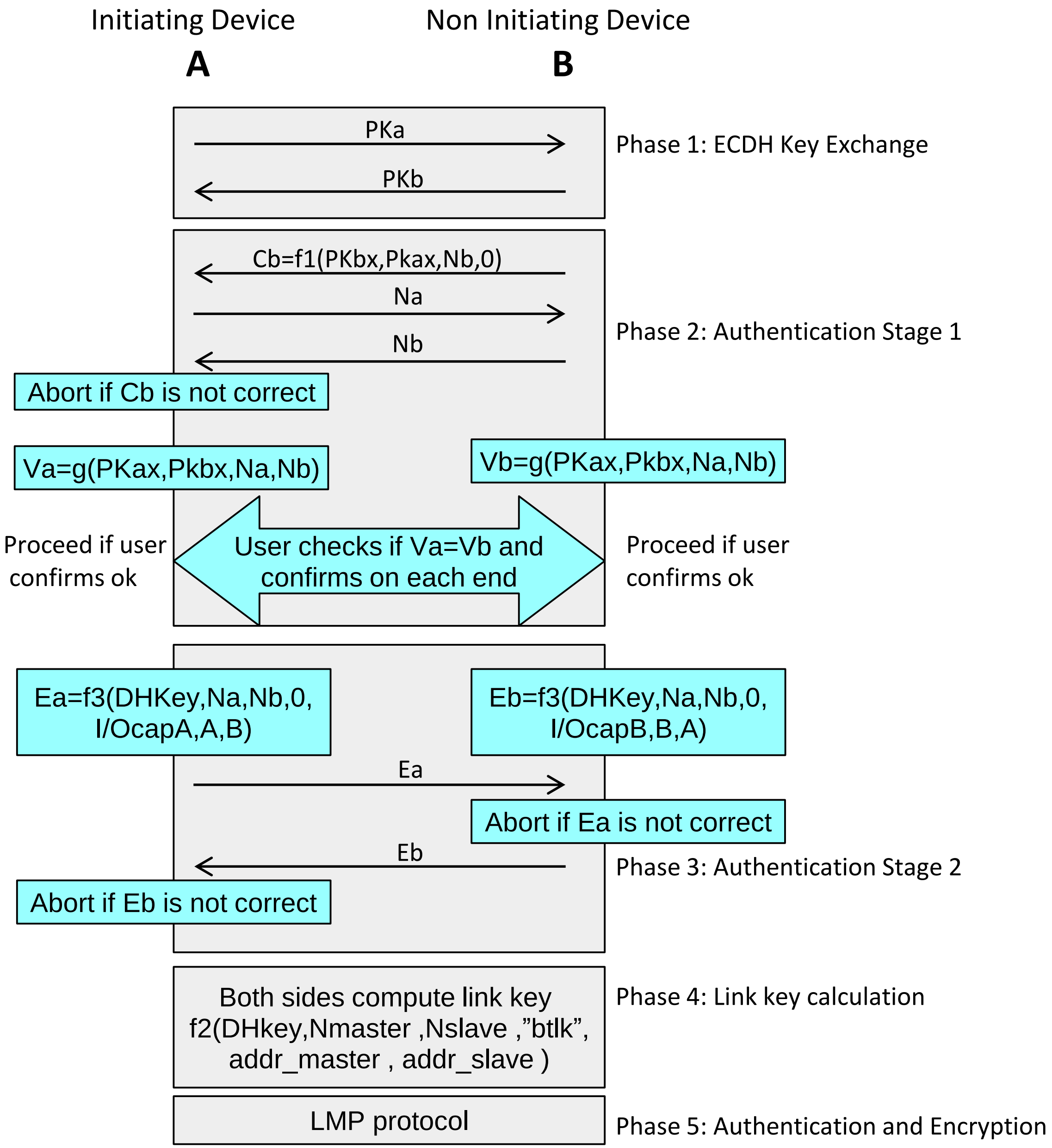
Attacker has another device named "B"

Device B is compromised (malicious app)

Misbinding in Bluetooth



Wrong devices paired!



Initiating Device

A

Non Initiating Device

B

PKa

PKb

Phase 1: ECDH Key Exchange

Cb=f1(PKbx,Pkax,Nb,0)

Na

Nb

Abort if Cb is not correct

Va=g(PKax,Pkbx,Na,Nb)

Vb=g(PKax,Pkbx,Na,Nb)

Phase 2: Authentication Stage 1

Proceed if user confirms ok

User checks if Va=Vb and confirms on each end

Proceed if user confirms ok

Ea=f3(DHKey,Na,Nb,0, I/OcapA,A,B)

Eb=f3(DHKey,Na,Nb,0, I/OcapB,B,A)

Ea

Abort if Ea is not correct

Eb

Phase 3: Authentication Stage 2

Abort if Eb is not correct

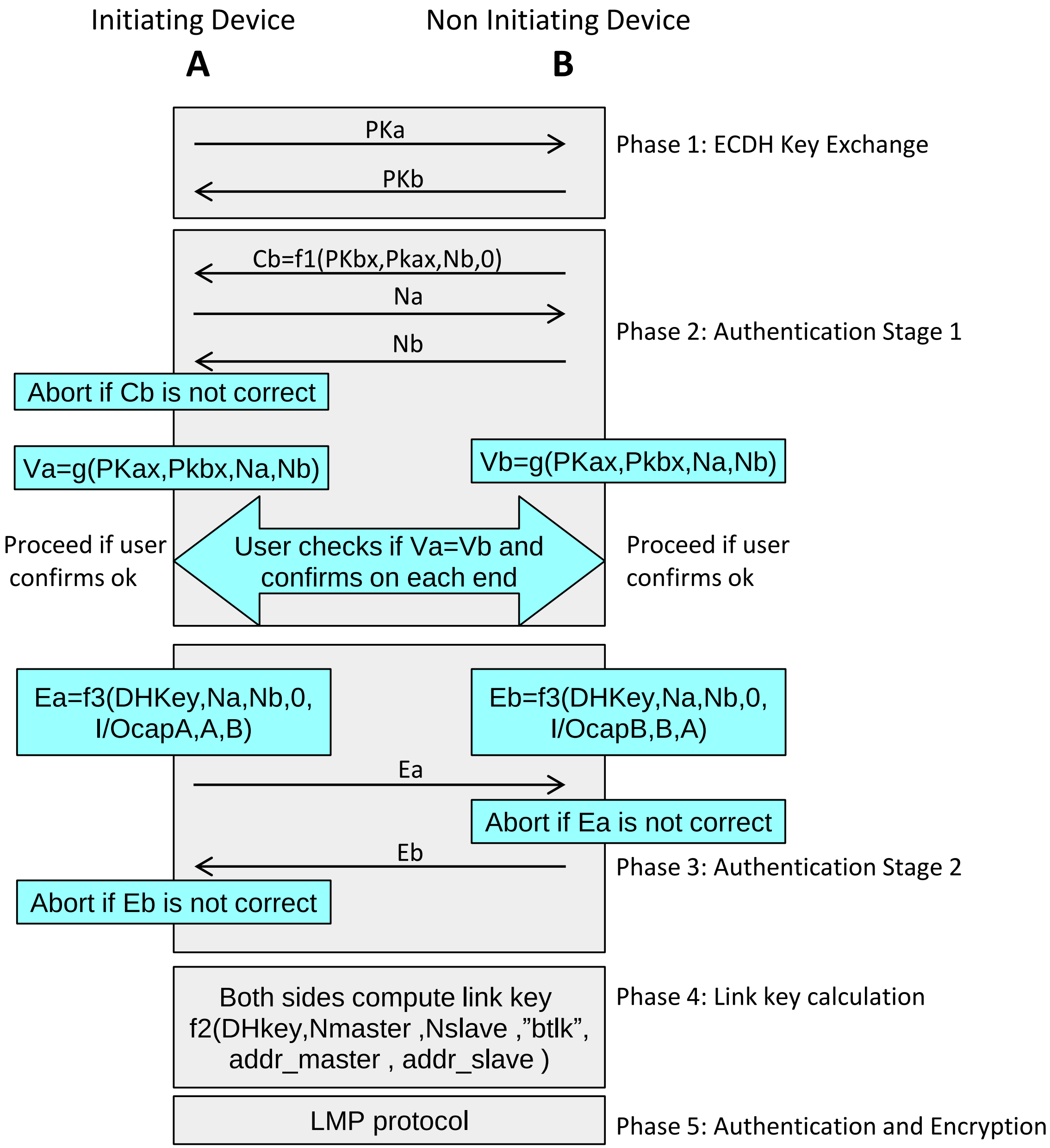
Both sides compute link key f2(DHkey,Nmaster ,Nslave ,”btlk”, addr_master , addr_slave)

Phase 4: Link key calculation

LMP protocol

Phase 5: Authentication and Encryption

- Why does Bluetooth not detect misbinding?
- Could it?



Initiating Device

A

Non Initiating Device

B

PKa

PKb

Phase 1: ECDH Key Exchange

Cb=f1(PKbx,Pkax,Nb,0)

Na

Nb

Phase 2: Authentication Stage 1

Abort if Cb is not correct

Va=g(PKax,Pkbx,Na,Nb)

Vb=g(PKax,Pkbx,Na,Nb)

Proceed if user confirms ok

User checks if Va=Vb and confirms on each end

Proceed if user confirms ok

Ea=f3(DHKey,Na,Nb,0, I/OcapA,A,B)

Eb=f3(DHKey,Na,Nb,0, I/OcapB,B,A)

Ea

Abort if Ea is not correct

Eb

Phase 3: Authentication Stage 2

Abort if Eb is not correct

Both sides compute link key f2(DHkey,Nmaster ,Nslave ,"btlk", addr_master , addr_slave)

Phase 4: Link key calculation

LMP protocol

Phase 5: Authentication and Encryption

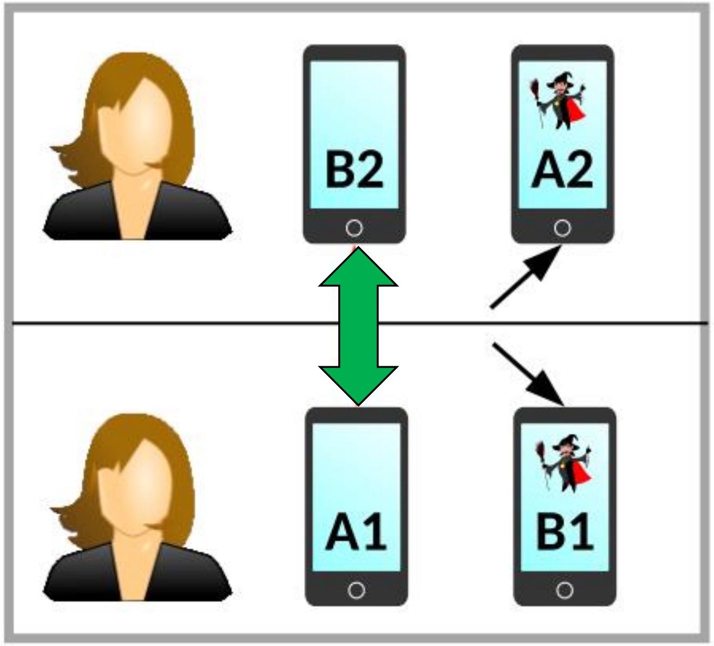
- Why does Bluetooth not detect misbinding?
- Could it?
- Devices have no verifiable identifiers!
- Authentication based only on physical access

Formal modeling

- Previous security analysis of Bluetooth had not detected misbinding
- We modeled Bluetooth numeric comparison and other pairing protocols with ProVerif
  - Physical channel defines device identity
  - Check correspondence between user intention and completed pairing

→ Can detect misbinding

- Analysis yielded a new double-misbinding case:



Lessons

- All device-pairing protocols are vulnerable if devices have no verifiable identifiers and authentication is based only on physical access

- Trusted path issue: attacker can spoof the pairing UI on the compromised device
  - Trusted path (e.g. hard-wired reset button) would prevent malicious apps from spoofing the critical UI
  - Device UIs are difficult to standardize, and attacker could still replace or modify the hardware

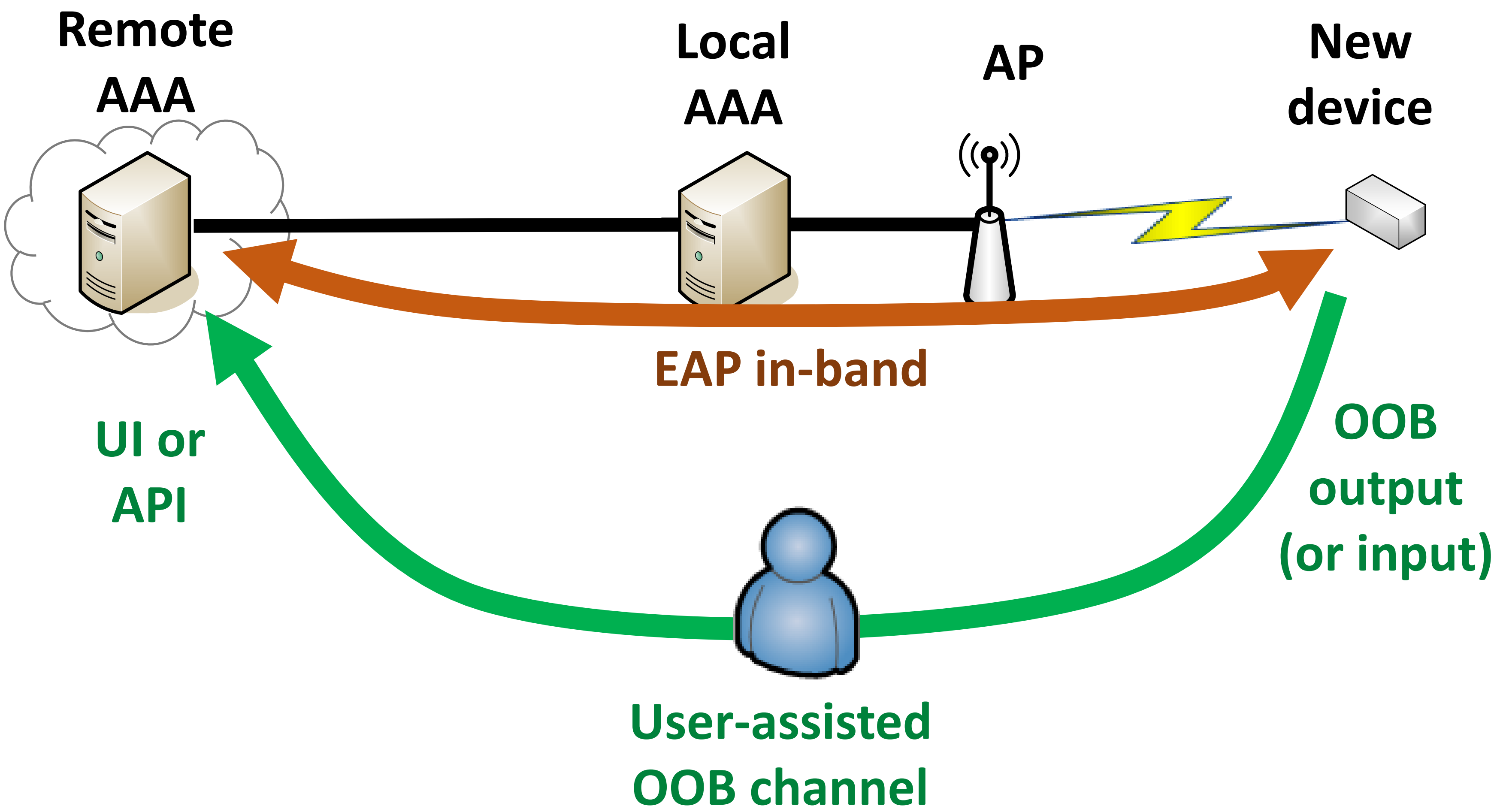
Misbinding in connecting devices to cloud (EAP-NOOB)

EAP-NOOB

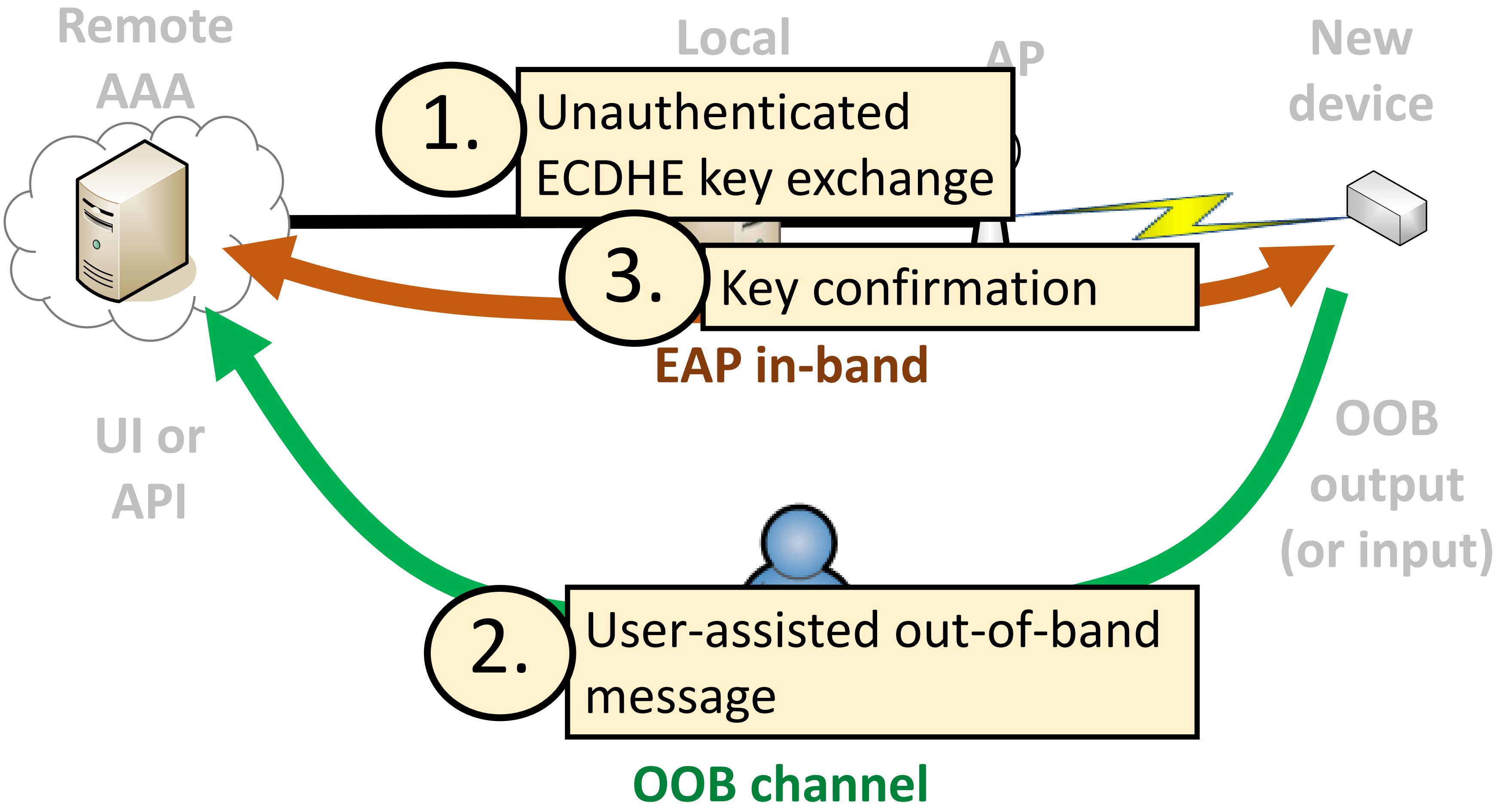
- EAP method for bootstrapping devices out-of-the-box without professional administration and without pre-established device credentials or identifiers
- User-assisted out-of-band (OOB) authentication
  - One OOB message in one direction between peer and server, e.g. scanning a dynamic QR code or NFC tag
- OOB authentication registers a new peer device. Once registered, reauthentication without user interaction

draft-aura-eap-noob

EAP-NOOB architecture



EAP-NOOB protocol



1. Unauthenticated ECDHE key exchange

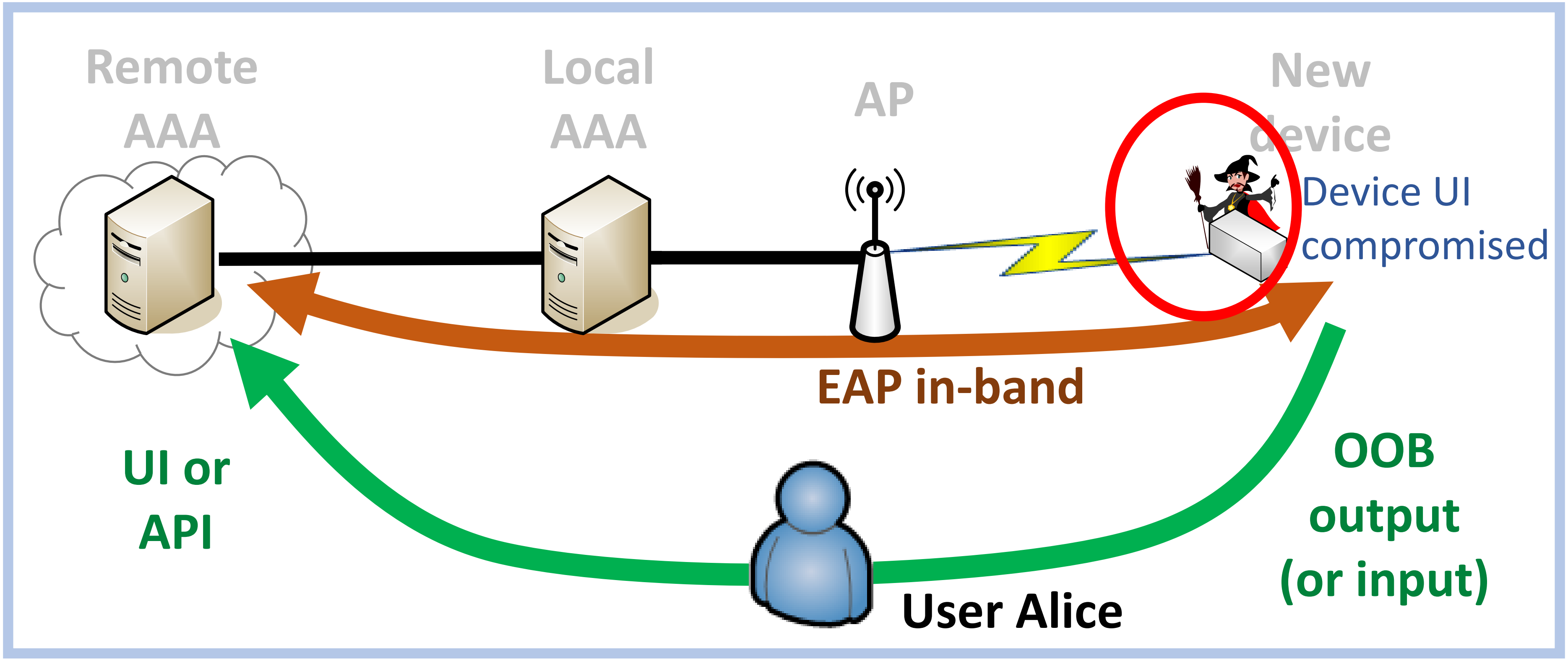
2. User-assisted out-of-band message

3. Key confirmation

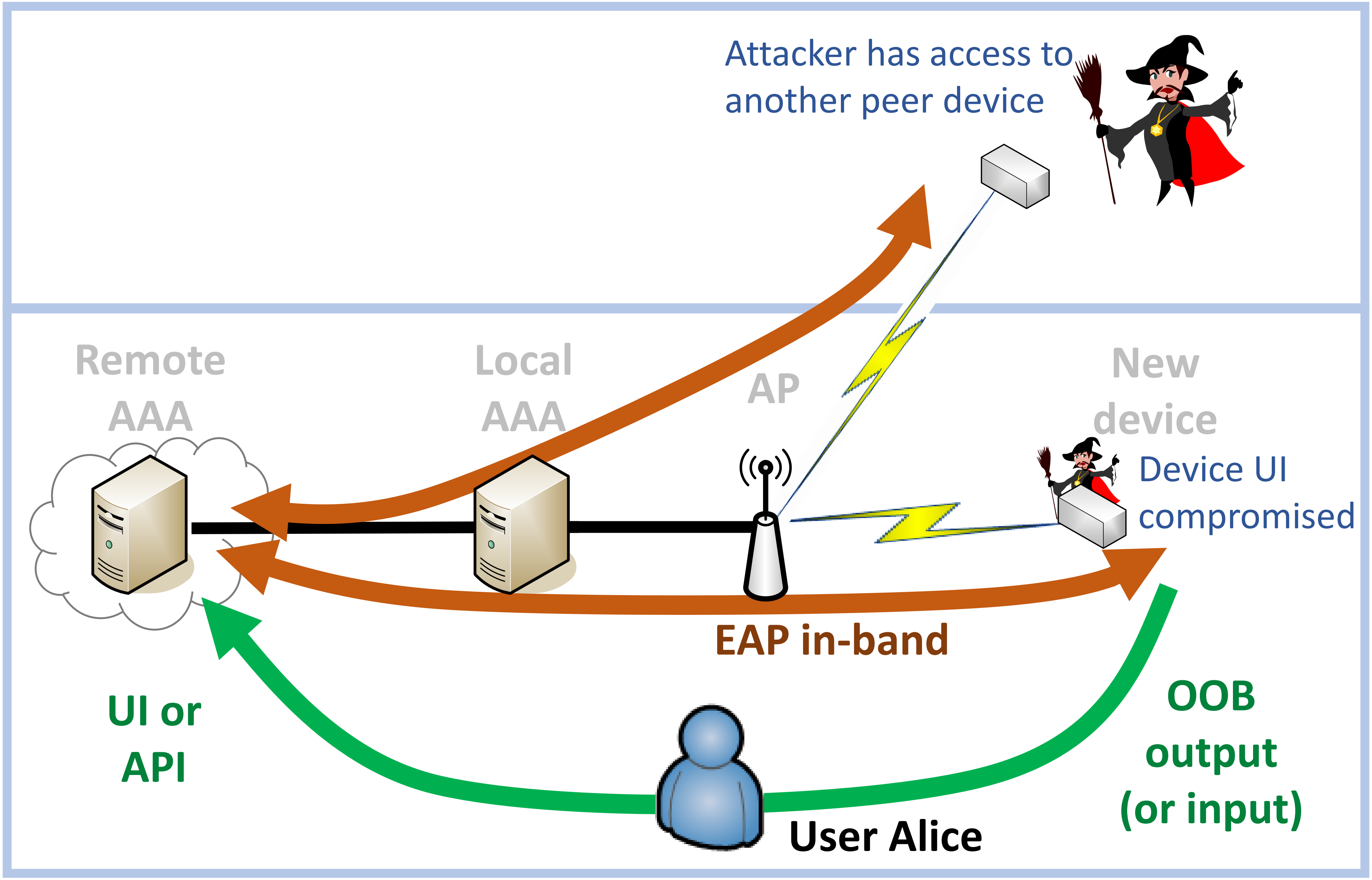
EAP in-band

OOB channel

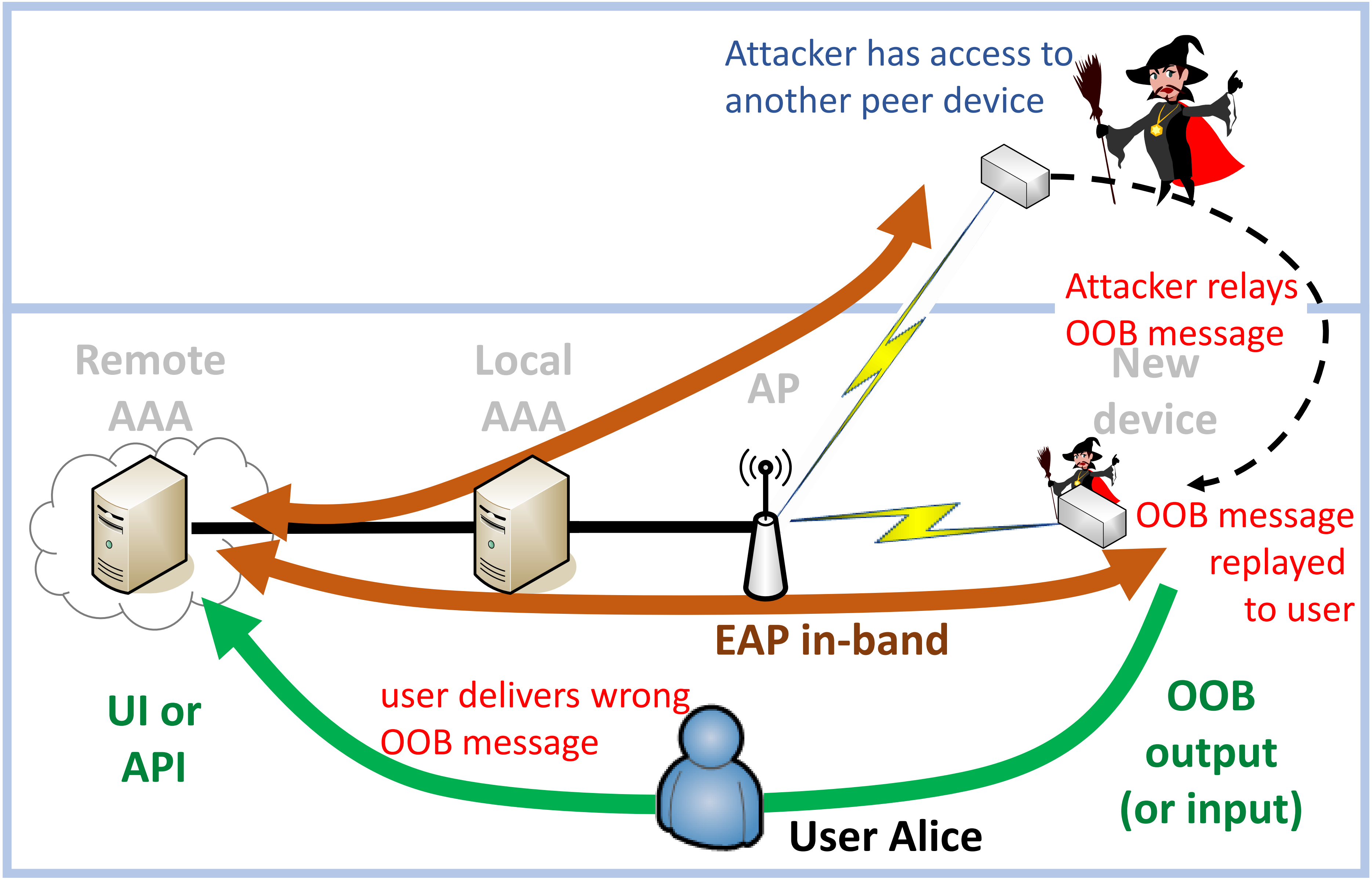
Misbinding in EAP-NOOB



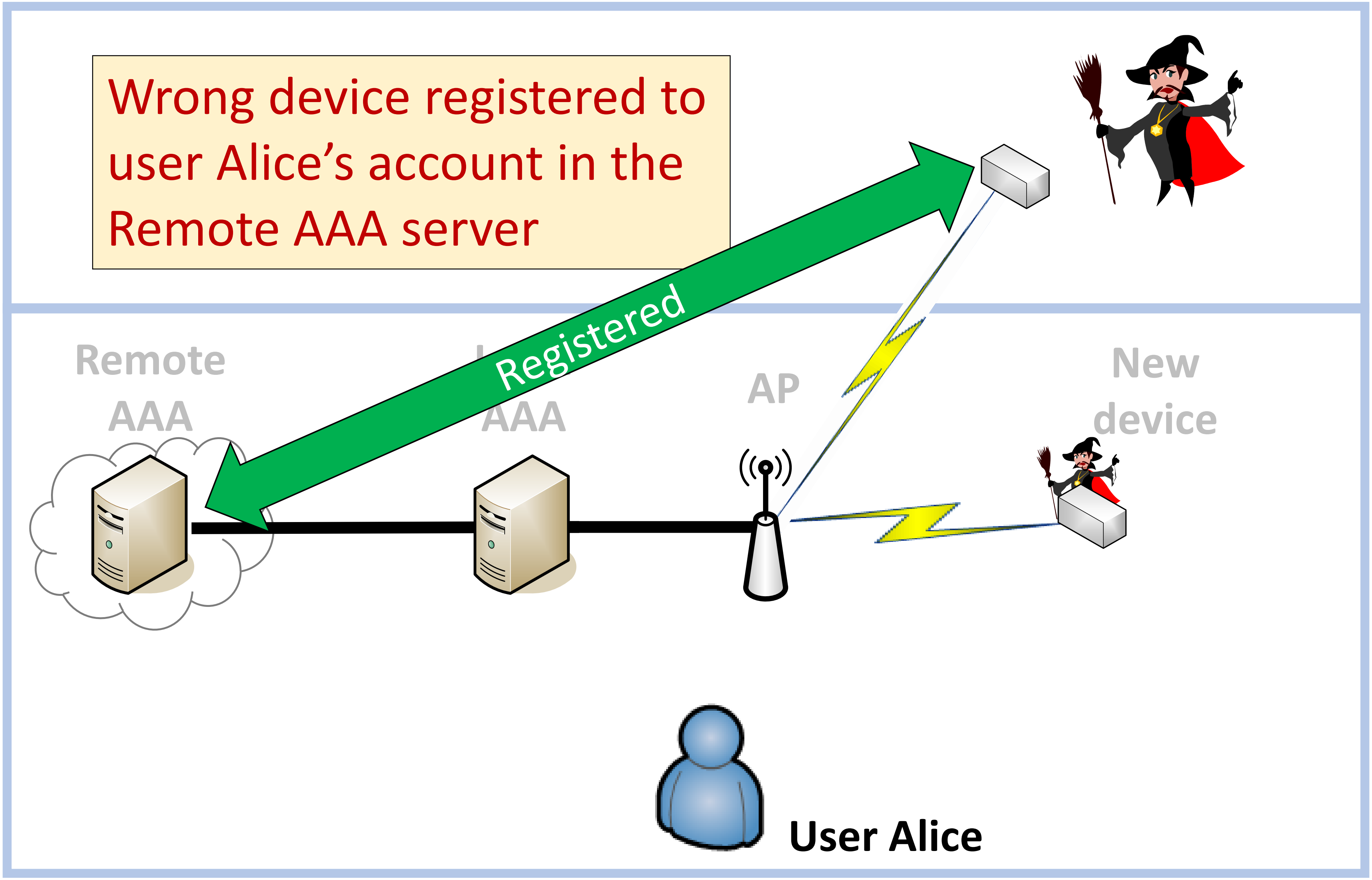
Misbinding in EAP-NOOB



Misbinding in EAP-NOOB


Attacker has access to another peer device
Attacker relays OOB message
Remote AAA
Local AAA
AP
New device
OOB message replayed to user
EAP in-band
UI or API
user delivers wrong OOB message
OOB output (or input)
User Alice

Misbinding in EAP-NOOB



Wrong device registered to user Alice’s account in the Remote AAA server

Why misbinding in EAP-NOOB?

- User physically identifies the the peer device; no other authentication

- Not a flaw in this specific protocol: Inherent weakness in pairing-like protocols that rely on user’s physical access for authentication

- Misbinding of the server not possible because typical OOB channels use web certificates, and user or app checks the server name

Misbinding and trusted execution

- Misbinding-like cuckoo attacks are known in trusted-computing
- Cryptographic authentication of TPM/TEE does not prove that the secure execution takes place inside a the user-chosen physical device
  - Compromised device with fake number plate or fake UI can cause misbinding
- Relevant to two IETF WGs:
  - Remote ATtestation ProcedureS (rats)
  - Trusted Execution Environment Provisioning (teep)

Mitigation and summary

Mitigating misbinding

- Cryptographically bind session keys to context data
  - Persistent non-modifiable device identifiers and hw info
  - Channel binding to wireless MAC addresses

  → Harder to trick user, and attacker may be forced to modify hardware or perform active MitM in the access network

- Preventing software-based UI spoofing
  - Specify a trusted path for the devices (e.g. reset button)

- Knowing your devices
  - Device certificates to attest make, model, serial number
  - Asset tracking: user or admin has prior knowledge of the devices, identifiers and intended deployment

Summary

- All device-pairing and bootstrapping protocols are vulnerable to misbinding if
  - Device authentication is based on physical access
  - Device identity not cryptographically authenticated, or if the verifier does not know which identifier is correct
- Several ways to mitigate the threat, but complete prevention will require redefining the assumptions (or goals) of device pairing and registration

Discussion question: Should we now tell everyone that Bluetooth pairing is inherently insecure, or similarly for TPM/TEE provisioning?

Full report: https://arxiv.org/abs/1902.07550