FLORIDA TECH
FLORIDA'S STEM UNIVERSITY™

Department of Computer Science

CSE 4820: Wireless and Mobile Security

10. Bluetooth Security

Dr. Abdullah Aydeger
Location: Harris Inst #310
Email: aaydeger@fit.edu

Outline

- Bluetooth
  - Encryption and Authentication
  - Traditional Pairing
  - Secure Simple Pairing

Recall: Bluetooth Basics

Device 1 and 2 form a piconet; they are channel hopping in step with each other.

Device 1 (master)	1	8	5	4	7	6	10	2	9	12	3	11
Device 2 (slave)	1	8	5	4	7	6	10	2	9	12	3	11

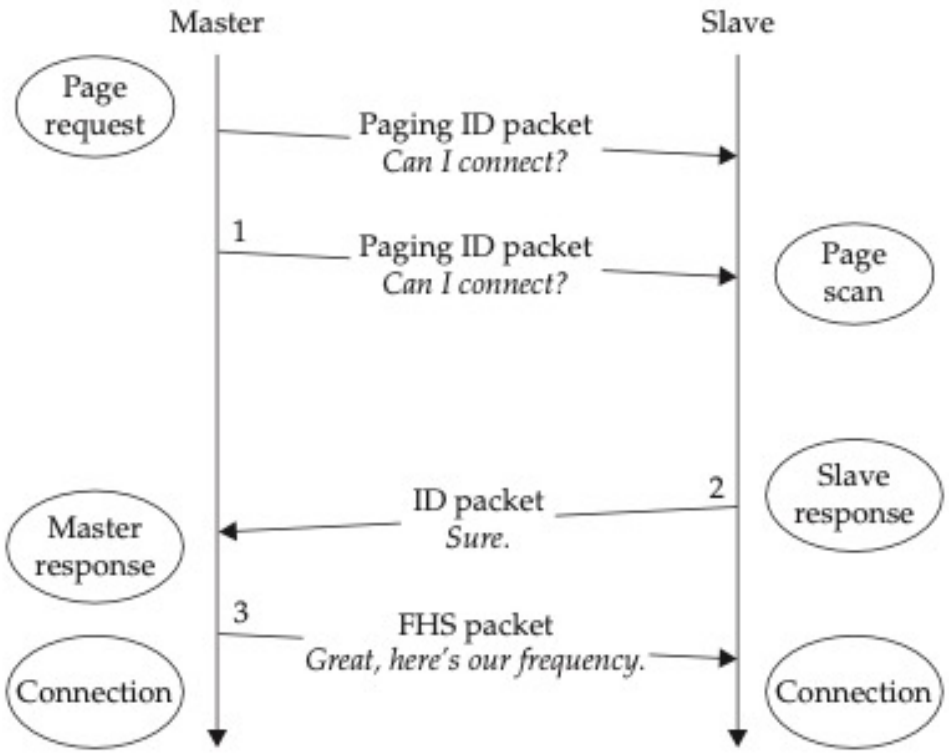
Device 3 is not part of the piconet; it is unaware of the channel-hopping sequence in use by the other devices.

Device 3	6	4	5	10	1	2	6	3	11	8	9	7

- Defines 79 channels across the 2.4-GHz ISM band, each channel occupying 1-MHz of spectrum
- Devices hop across these channels at a rate of 1600 times a second (every 625 microseconds)
- This channel-hopping technique is Frequency Hopping Spread Spectrum (FHSS), and the user can achieve a rate of 3 Mbps of bandwidth across 100 meters
  - FHSS provides robustness against noisy channels by rapidly changing frequencies
  - Later revisions of the standard have added support for adaptive hopping, which allows noisy channels to be detected and avoided all together

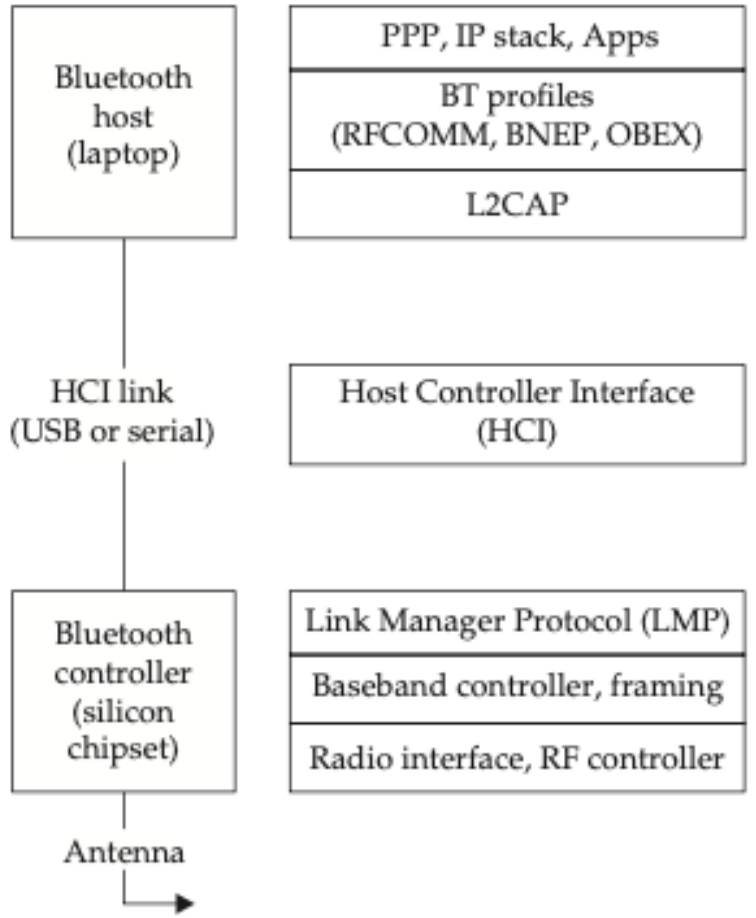
Recall: Bluetooth: Connection Establishment

- The diagram covers this in some detail
  - The most important thing to remember about "paging" or connection establishment is that in order to establish a connection you must know the target's BD_ADDR, and that device must be interested in accepting connections



Recall: Bluetooth Protocol Overview

- The organization of layers in the Bluetooth stack and where each layer is typically implemented:
  - The controller is responsible for frequency hopping, baseband encapsulation, and returning the appropriate results back to the host
  - The host is responsible for higher-layer protocols
  - The Host Controller Interface (HCI) link is used as the interface between the Bluetooth host (your laptop) and the Bluetooth controller (the chipset in your Bluetooth dongle)

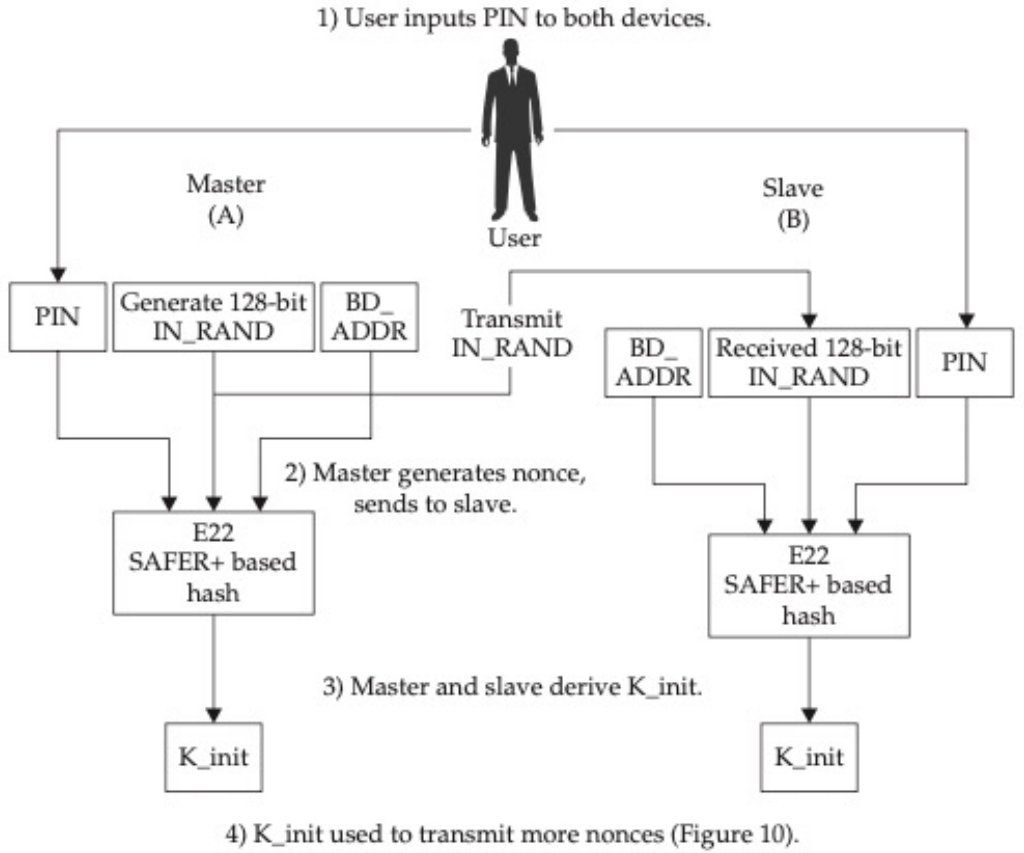


Bluetooth Encryption and Authentication

- Are built into the Bluetooth standard
  - Both are largely handled on the controller chip itself, not directly accessible to the Host Controller Interface layer
- Bluetooth devices can authenticate in two different ways:
  - Traditional pairing
  - Secure Simple Pairing (SSP)
    - SSP was added in version 2.1 of Bluetooth

Traditional Pairing Process: Link Key Creation

- Was superseded by the release of Bluetooth 2.1 by Secure Simple Pairing
  - The tradition process is still used in many Bluetooth devices
- When two devices connect for the first time, a link key is derived from a BD_ADDR, a PIN code, and a random number
  - Once both sides in the exchange have created K_init, they use it to generate a stronger link key

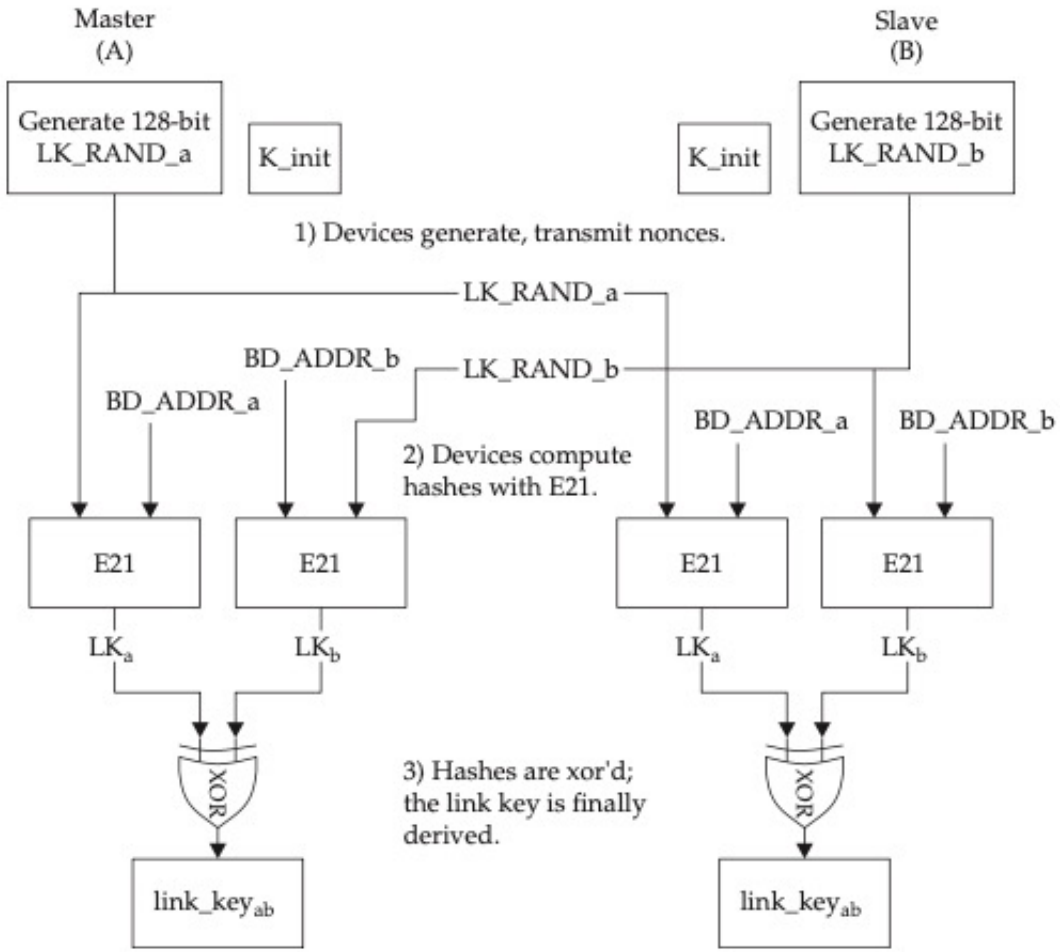


Traditional Pairing Process: PIN

- Some devices (like wireless earphones) the PIN is fixed and cannot be changed
  - In such cases, the fixed PIN is entered into the peer device
- If two devices have a fixed PIN, they cannot be paired, and therefore cannot communicate

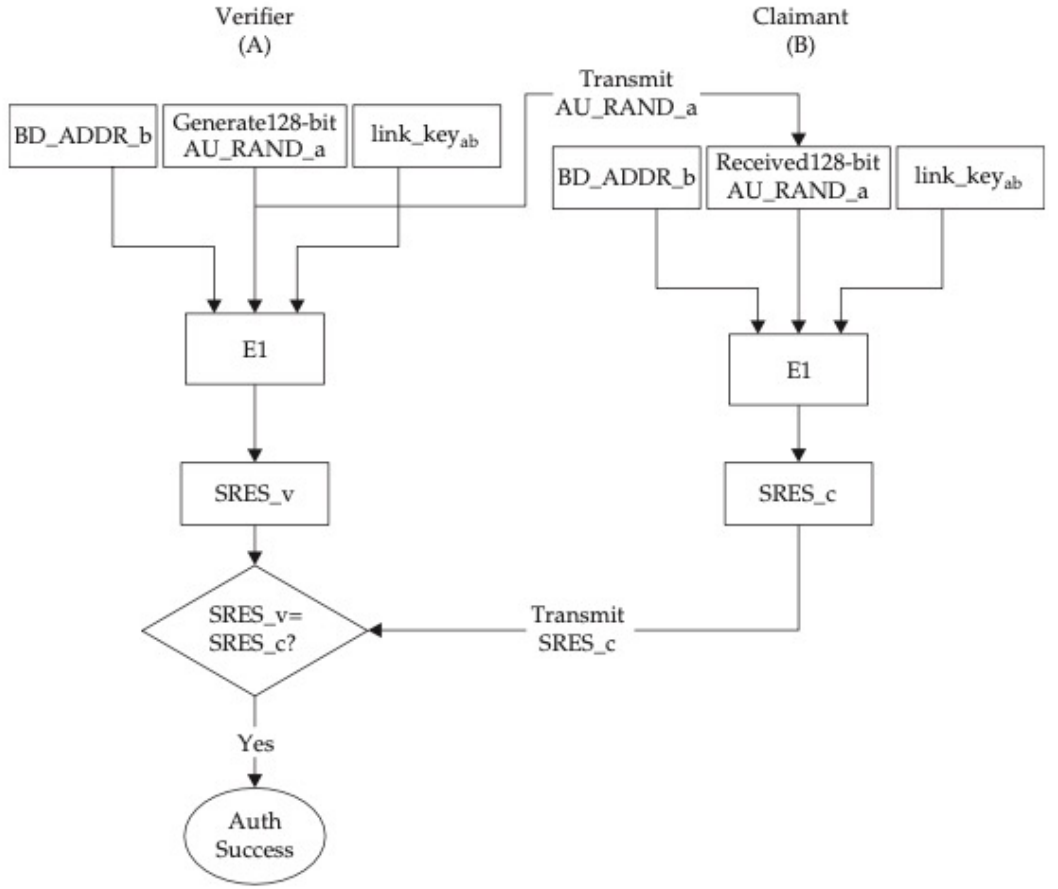
Traditional Pairing Process: Derivation of the Link Key

- Both devices have generated a 128-bit link key
- This link key will be stored alongside its peer's BD_ADDR, either on disk or in nonvolatile storage
- When the two devices want to communicate, they will go through a handshaking process
- The link-key generation will only happen each time the devices are paired (typically once)
  - For later authentication purposes, possession of this link key is used, not the PIN



Proving Link-Key Possession

- Bluetooth devices utilize possession of the link key to verify they are communicating with a device they have previously paired with
- Proving possession of this link key authenticates the peer
- The same exchange takes place if the link key was created using SSP or the traditional pairing technique



$link\_key_{ab}$ was derived during the pairing process.
This portion of the protocol is very similiar to the four-way handshake used by WPA; instead of verifying possession of the PMK, we are verifying possession of the link_key.

Proving Link-Key Possession

- In this exchange, there are two entities, the *Verifier and the Claimant*
- The Verifier verifies that the Claimant has possession of the previously negotiated link key
  - The Verifier does this by transmitting a challenge in the clear (AU_RAND_a)
- Upon receiving the challenge, the Claimant computes the keyed hash of the challenge using an algorithm called E1 and the link key
  - The Claimant transmits the results (called the Signed Response, or SRES_c) back to the Verifier
  - Meanwhile, the Verifier computes the same hash
- If the Claimant's SRES_c matches the Verifier's computed hash, the Claimant has proven possession of the link key

Proving Link-Key Possession

- If mutual authentication is desired, the two devices reverse roles and repeat the process
- Once the devices have authenticated each other by verifying possession of the link key, they will likely proceed to derive an encryption key
  - The encryption key is derived from a hashing function, which will take the link key as input

Traditional Pairing Process: Security

- If an attacker observes the traditional link-key generation step, as well as the link-key authentication step, all he needs to do to derive the link key is to brute-force the PIN until he generates the observed value of SRES
- Once he gets the SRES to match, he possesses both the PIN and the link key

Secure Simple Pairing (SSP)

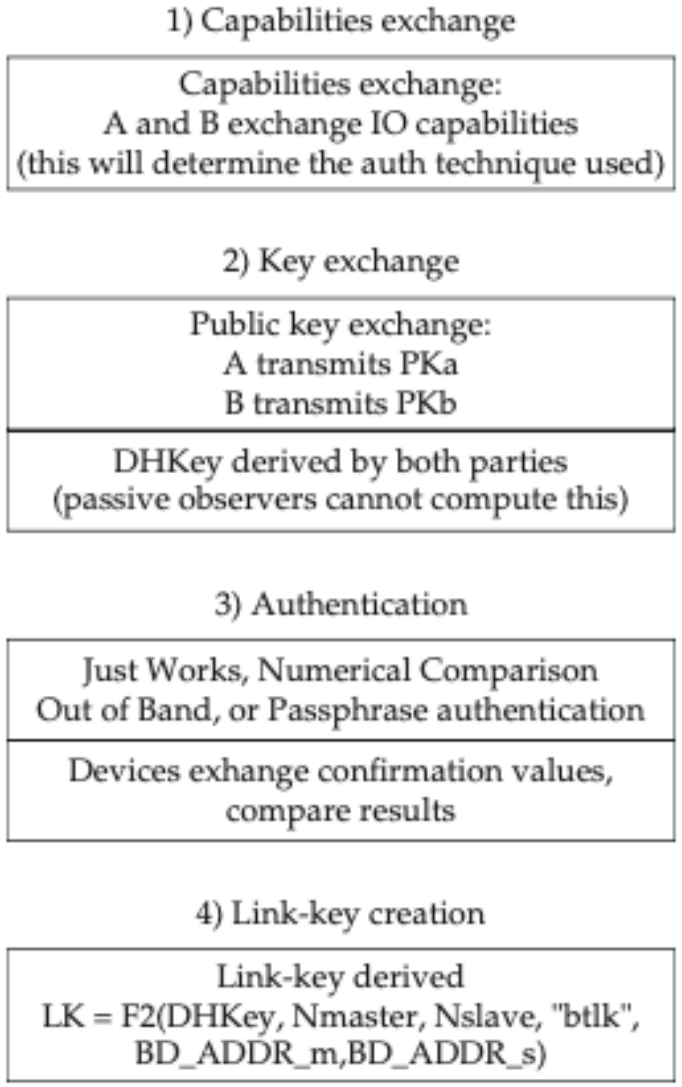
- The problem with the traditional pairing scheme is that a passive attacker who observes the pairing exchange can typically brute-force the PIN in seconds
- SSP attempts to prevent a passive observer from retrieving the link key
  - SSP accomplishes this by using public key crypto, specifically Elliptic Curve Diffie-Hellman
- A Diffie-Hellman key exchange allows two peers to exchange public keys and then derive a shared secret that an observer will not be able to reproduce
- The resulting secret key is called the *DHKey*
  - Ultimately, the link key will be derived from the *DHKey*

Secure Simple Pairing

- By using a Diffie-Hellman key exchange, a strong pool of entropy is used for deriving the link key
  - Solving the biggest problem with the standard pairing derivation, where the sole source of entropy was a PIN only a few digits in length
- SSP adds another layer of authentication to the pairing process
  - SSP depends on the devices both having some form of input and output you can use to communicate with
  - These IO capabilities are explicitly negotiated during the SSP exchange

Secure Simple Pairing: Overview

- Capabilities Exchange
- Key Exchange
- Authentication
- Link-key creation



Secure Simple Pairing: Capabilities Exchange

- During this exchange, the link managers on both devices trade information on whether they have the capability to input or display information
- This exchange is important because the Just Works authentication method, which is used when there is no better option, doesn't provide protection against active MITM attacks

Secure Simple Pairing: Capabilities Exchange

- Tables summarizing the capability information exchanged by devices:

Capability	Description
No input	Device cannot indicate yes or no, lacking any input capability.
Yes/No	Device has a button that the user can activate to indicate yes or no.
Keyboard	Device can input values 0–9, as well as indicate yes or no.

Capability	Description
No output	Device cannot display a 6-digit number.
Numeric output	Device can display a 6-digit number.

Secure Simple Pairing: Capabilities Exchange

- Some combinations of IOCapabilities cannot provide a defense against a MITM attack
- When a link key is generated and the authentication used can protect against MITM attacks, the device key is called “authenticated”
- If the authentication scheme cannot protect against MITM attacks, the link key is said to be “unauthenticated”

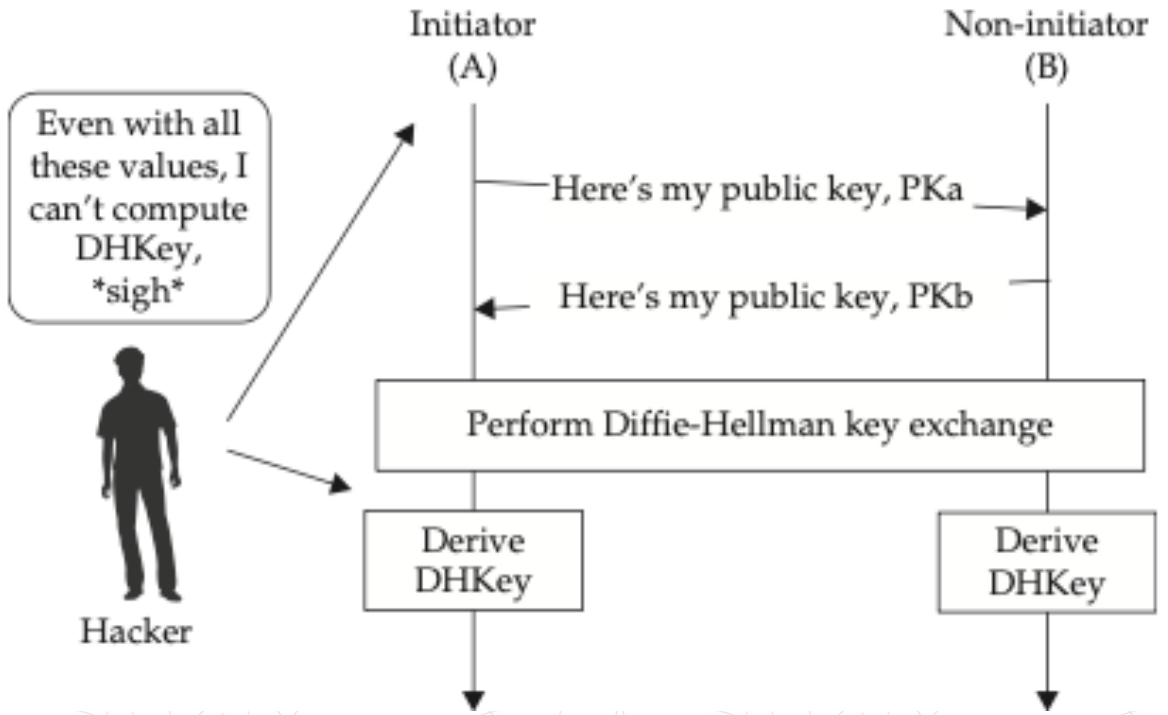
Secure Simple Pairing: Capabilities Exchange

- Many devices use the weaker form of "Just Works" authentication, even when both devices support a more secure authentication scheme
  - This is because devices use the following algorithm to determine which authentication system to use
  - 1. Has any Out-Of-Band (OOB) data successfully been received from the other device? Then use the OOB authentication technique
  - 2. Do both devices support insecure (unauthenticated) link keys? Then use the Just Works (Numeric Comparison with no user confirmation)
  - 3. Does either device require an authenticated link key? Then choose the appropriate scheme

Secure Simple Pairing: Capabilities Exchange

- Notice that if both devices are configured to accept unauthenticated link keys, they will do so, even if they both have extensive IOCapabilities
- Once the IOCapabilities exchange has taken place, the devices should know what authentication technique will be used later
  - The next step is to exchange public keys and derive the DHKey

Key Exchange



- Is the easiest phase
- The devices simply transmit their public keys to each other and then perform a Diffie-Hellman key exchange operation to derive DHKey
- An attacker who captures this entire exchange will still be unable to compute DHKey, due to the cryptographic security of the Diffie-Hellman protocol

Authentication

- Once the DHKey has been derived, there are four possible authentication techniques
- In traditional pairing, you are verifying possession of a link key
  - At this current point in the SSP exchange, you haven't created the link key yet
  - Instead, you are trying to verify that the device you just performed a Diffie-Hellman key exchange with is the device you think it is
- Ultimately, verification of the link key will take place using the same algorithm used in traditional pairing

Authentication: Just Works

- This mode runs the same protocol as numeric comparison, but the user does not actually make a comparison
- This protocol is secure against passive attacks (due to the DHKey exchange), but an active MITM attack can succeed by sending both A and B its own public key and nonces at the right time

Authentication: Just Works

- Was a necessary accommodation to achieve the greatest level of compatibility between electronics design and Bluetooth security
- While other authentication mechanisms offer a greater level of security, they all require some level of Man-Machine Interface (MMI) to display content or collect responses from the end-user
  - This requirement would otherwise limit the scope of deployment options for Bluetooth developers, requiring an authentication mechanism that does not require any input from the user

Authentication: Just Works

- Just Works mechanism is also the easiest to use, leading developers and end-users to choose this authentication mechanism over stronger protocols
- Although the Just Works method protects against the passive eavesdropping attacks that plague legacy PIN authentication, it does not protect against active attacks
- Devices that accept Just Works authentication are incapable of authenticating the identity of the remote entity or defeating MITM attacks, leaving the link-layer exposed to a variety of attacks

Authentication: Just Works

- For example, consider a case where a Bluetooth USB HID interface is used on a PC intended for use with a Bluetooth keyboard
- If the PC is connectable, an adversary could impersonate a legitimate keyboard device and send arbitrary keystrokes to the host, perhaps downloading and running malware-ridden executables from the Internet
- In this example, the PC device is at fault for accepting the Just Works authentication technique when stronger input mechanisms are available
  - Though it isn't unreasonable to foresee a situation where Just Works is used by a device manufacturer to simplify the product setup process

Authentication: Numeric Comparison

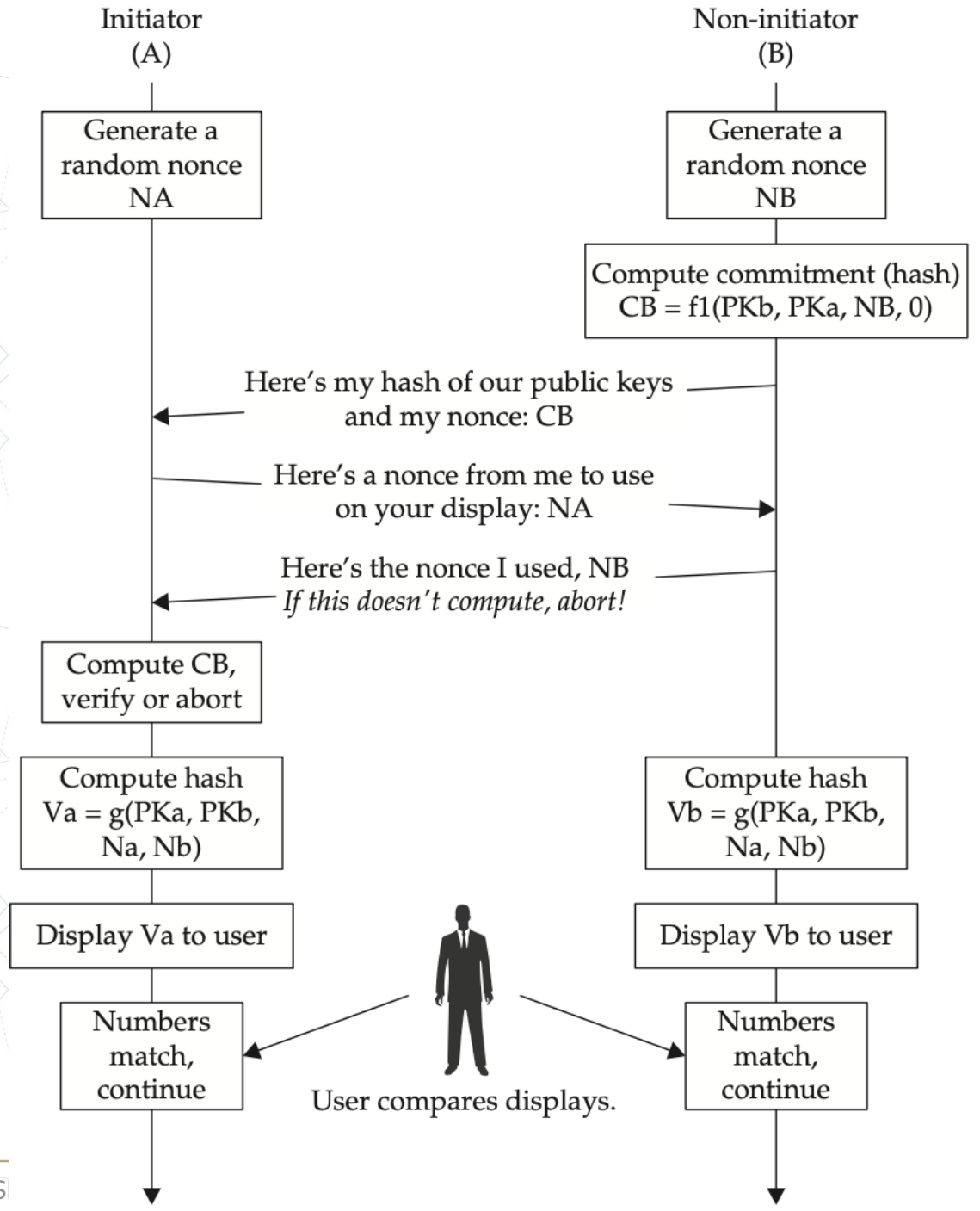
- It it used, if both devices have sufficient IO capabilities to display a six-digit number to the user
- When this technique is used, each device computes a hash of the exchanged public keys, as well as two more nonces
- The devices display six digits of this hash, and the end user is expected to verify that the displayed hashes match and select *Yes or No*

Authentication: Numeric Comparison

- Although this may appear to be similar to the traditional pairing process, cryptographically it is very different
- In this case, the six-digit values displayed to the user are an artifact of the pairing process
  - They are not used as input to any cryptographic functions
  - Rather he is comparing the number, which is a hash generated by both devices

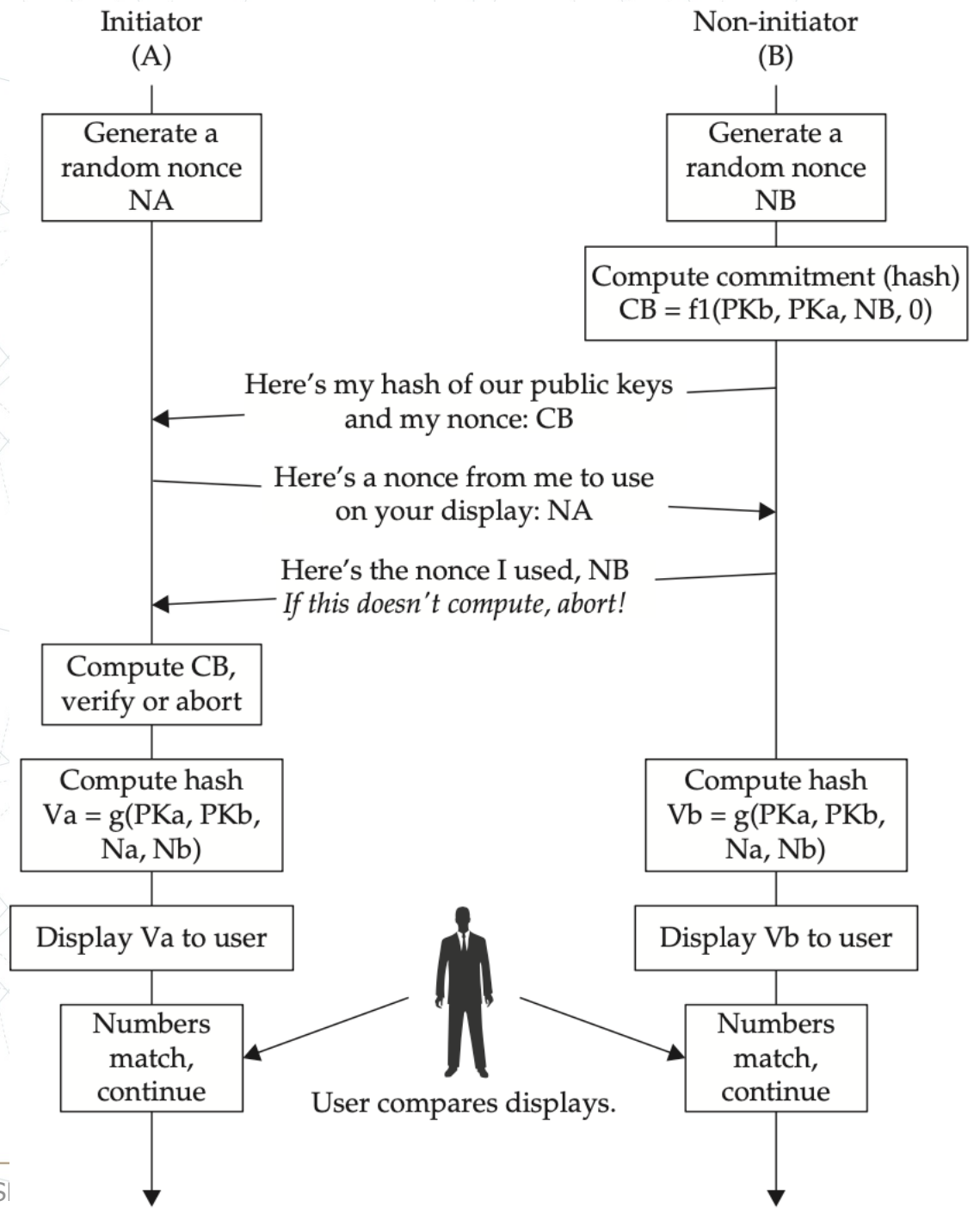
Authentication: Numeric Comparison

- Both devices generate and exchange nonces (a number used once), and then compute a hash of these nonces as well as the public keys exchanged previously
  - This hash is displayed to the user in the form of a six-digit number
- The user is supposed to compare these numbers to verify that they match



Authentication: Numeric Comparison

- If so, the user presses the *Yes* button to indicate pairing should proceed
- If the hashes don't match, then an active attacker has tried to inject keying material or a transmission error has occurred
  - In either case, the pairing shouldn't proceed



Authentication: Out of Band

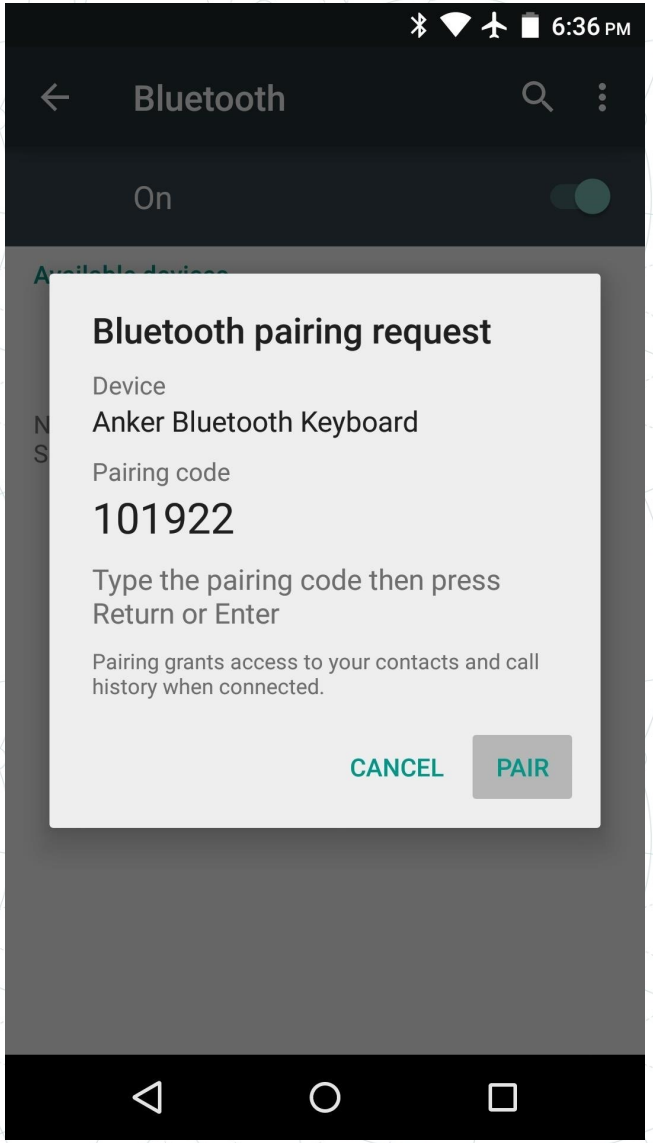
- This mode is used if the devices have some way other than Bluetooth to exchange cryptographic material
  - Near Field Communication (NFC) is described as one possibility
- If the OOB communications technique can be exploited with a MITM attack, this authentication technique will be similarly vulnerable

Authentication: Passkey Entry

- Passkey entry allows Bluetooth devices to authenticate each other by verifying they both possess a shared secret
- This secret can be entered into both devices or generated on one and entered into the other
- A typical use-case for this is a keyboard and computer
  - The computer generates a random PIN, and the user inputs it through the keyboard
  - When implemented poorly, this technique can be severely compromised

Authentication: Passkey Entry

- The goal of the protocol is to verify the passphrase with the other device, one bit at a time
- For every bit in the passphrase, the devices will compute a hash including the bit and transmit the hash to the other side
  - Both sides will transmit eight hashes



FLORIDA TECH

FLORIDA'S STEM UNIVERSITY™

Thank you. Questions?

Dr. Abdullah Aydeger