

Preventing the Man in the Middle Attack on a Bluetooth Connection Using BlueZ Software

Cameron Bertram and Moradeke Olumogba
Vertically Integrated Projects (VIP)

Secure Hardware

Assoc. Prof. Vincent Mooney

7 November 2017

CREATING THE NEXT®

Outline

- I. **Background**
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Bluetooth Background

- Bluetooth is a standardized protocol for wirelessly sharing relatively small amounts of data.
- Creates short range, ad-hoc networks with other devices such as keyboards, phones, and computers.
- Advantageous because it is low-power and low-cost.
- We are currently running on BlueZ and using Bluetooth 4.0 (LE).



Figure 1. Bluetooth logo.

Bluetooth Background

BlueZ

- Developed by Qualcomm
- Purpose is to implement wireless standards from Bluetooth on Linux devices
- Provides support for Bluetooth layers and protocols^[4]
- Also gives access to a number of libraries and utilities that allow Linux devices to connect with and modify connections with Bluetooth devices

Bluetooth Background

BlueZ (continued)

- Linux distribution bluez-5.47.tar.xz is to be installed on ivy.ece.gatech.edu
 - ivy has Redhat 5.0 installed
 - Redhat 5.0 comes with BlueZ preinstalled
- BlueZ files reside at /usr/include
- Main command are hcitool, hciconfig, rfcomm
- BlueZ appears to create a Link Keys file within the directories it creates for each device it is paired with.
 - Titled by device address

Bluetooth Background

Bluetooth Protocol

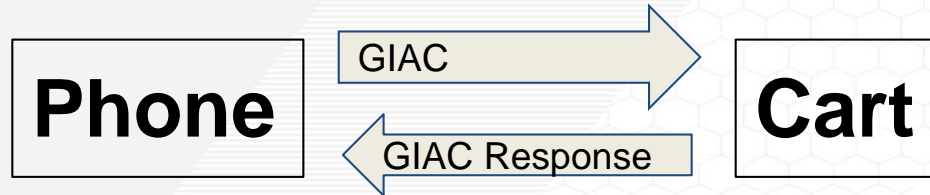
- ‘Scan’ to find the other Bluetooth devices.
- Its name and address appears in alphabetic characters and hexadecimal digits respectively.
- Pair with the device
 - Establishes connections.
 - Link key is generated and stored on devices
 - Encrypts / Decrypts data on both devices.

Bluetooth Background

Pairing Process [3]

1. Discovery/Inquiry

- Initiator broadcasts a General Inquiry Access Code (GIAC) over multiple frequencies until another device receives it.
- GIAC contains the BD_ADDR and the services the device can offer.
- When a listening device hears the message it responds with its own GIAC.



Knows:

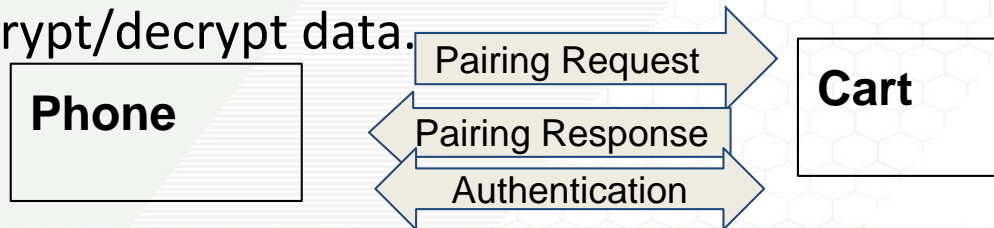
- BD_ADDR
- Service Type
(Printer, Phone etc.)

Bluetooth Background

Pairing Process [3]

2. Paging

- Initiator sends page message to the recipient containing a request to initiate pairing.
- An immediate response is sent back if the device is open to pairing.
- The two devices then exchange information outlining the complexity of the remainder of the pairing process.
- In the key exchange step, the link key is transmitted in order to encrypt/decrypt data.



Knows:

- BD_ADDR
- Service Type (Printer, Phone etc.)
- Encryption Key

Bluetooth Background

Generation and Transfer of Encryption Keys^[6]

- After plaintext initial pairing, all succeeding communication becomes encrypted.
- Devices agree on Temporary Key (TK) with value based on I/O.
 - Numeric Comparison
 - Passkey Entry
 - OOB
 - Just Works
- TK is used to create and then encrypt the Short Term Key (STK).

Bluetooth Background

Link Key Creation and Transfer

- Once encrypted by the STK, devices exchange values used to make a link key.
- Link Key results from a function with specific inputs.
 - Static random 128 bits called the Encryption Root
 - 16 bit Diversifier which is device specific
 - Function is called a Diversifying Function and is based off AES -128
- The two devices will continue to use that link key to encrypt communication until instructed not to do so.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

NFC Background

- **NFC (near field communication)**: is a subset of RFID and interacts at a frequency of 13.56 MHz
 - **PCD (Proximity Coupling Device)**: active device (i.e. reader)
 - **PICC (Proximity Inductive Coupling Card)**: passive device (i.e. card, passive tags)
- Based on **ISO/IEC 18092 N** and **ISO/IEC 14443**



Figure 2. This is one of NFC's logos.

NFC Background

NFC Data Exchange

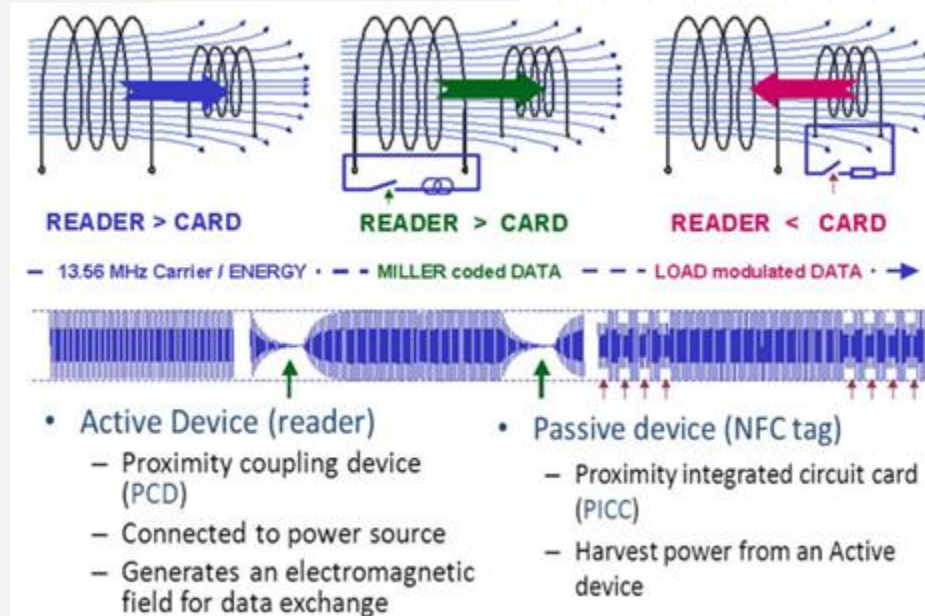


Figure 3. Data exchange process, showing how a reader powers up a card (passive tag).

NFC Background

Operating Modes

- 1)**Read/Write**: obtains stored data or write new data for future use
- 2)**Tag/Card Emulation**: active device acts as passive to communicate information
- 3)**Peer-to-Peer**: two active devices send information between each other

NFC Background

APDU

Application Protocol Data Unit: command code sent between cards/readers

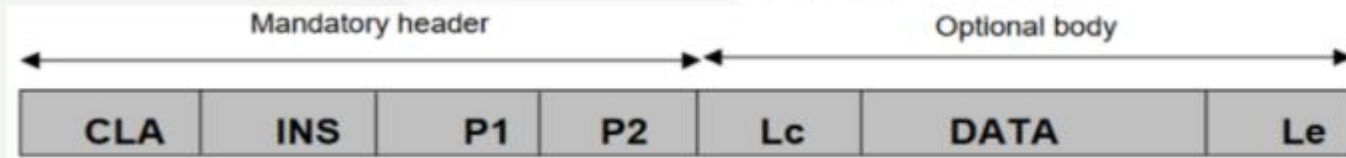


Figure 4. Command APDU



Figure 5. Response APDU

NFC Background

APDU: Instruction List

Table 4.1 — Commands in the alphabetic order

Command name	INS	See
ACTIVATE FILE	'44'	Part 9
APPEND RECORD	'E2'	7.3.7
CHANGE REFERENCE DATA	'24'	7.5.7
CREATE FILE	'E0'	Part 9
DEACTIVATE FILE	'04'	Part 9
DELETE FILE	'E4'	Part 9
DISABLE VERIFICATION REQUIREMENT	'26'	7.5.9
ENABLE VERIFICATION REQUIREMENT	'28'	7.5.8
ENVELOPE	'C2', 'C3'	7.6.2
ERASE BINARY	'0E', '0F'	7.2.7
ERASE RECORD (S)	'0C'	7.3.8
EXTERNAL (/ MUTUAL) AUTHENTICATE	'82'	7.5.4
GENERAL AUTHENTICATE	'86', '87'	7.5.5
GENERATE ASYMMETRIC KEY PAIR	'46'	Part 8
GET CHALLENGE	'84'	7.5.3
GET DATA	'CA', 'CB'	7.4.2
GET RESPONSE	'C0'	7.6.1
INTERNAL AUTHENTICATE	'88'	7.5.2

Figure 6. Table of instruction commands that are used to read, write values found in the MiFare 1k card.

NFC Background

MiFare 1k Card Memory Storage

Sector	Block	Byte Number within a Block														Description	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13		14
15	3	Key A				Access Bits				Key B						Sector Trailer 15	
	2																Data
	1																Data
	0																Data
14	3	Key A				Access Bits				Key B						Sector Trailer 14	
	2																Data
	1																Data
	0																Data
:	:																
:	:																
:	:																
1	3	Key A				Access Bits				Key B						Sector Trailer 1	
	2																Data
	1																Data
	0																Data
0	3	Key A				Access Bits				Key B						Sector Trailer 0	
	2																Data
	1																Data
	0																Manufacturer Block

- The last block of each sector is the “sector trailer”
- First block is private and cannot be accessed
- Access bits determine operation

Figure 7. Diagram of Mifare 1k Memory Storage with unofficial storage of 1040 bytes of memory

NFC Background

APDU Command and APDU Response

; [1] Load (Mifare Default) key in reader (key location 0)

FF 82 00 00 06 **D3 F7 D3 F7 D3 F7** (9000)

; [6] Authenticate sector 1, Block 5 with key at location 0

FF 86 00 00 05 01 00 05 60 00 (9000)

; [7] Store a value "1" into block 5

FF D7 00 05 05 00 00 00 00 01 (9000)

; [8] Read the value block 5

FF B1 00 05 04 [xx xx xx xx] (9000)

- [xx xx xx xx] : is the data field found in APDU Response

- Sub-Experiment: Padding with 0's and 1's

Figure 8. APDU Command and Response to writing a value of '01' into Sector 1, Block 1 (Sector 0, Block 5)

NFC Background

Reading and Writing

Initially, we decided to replace a value found in a block with the Bluetooth address of the Panda-Bluetooth-4.0-Nano-Adapter.

- 1.The data body is represented in hexadecimal format
- 2.We need to change two things: Lc and Data Field

Command APDU						
Header (required)				Body (optional)		
CLA	INS	P1	P2	Lc	Data Field	Le

Figure 9. The Lc determines how many bytes are in the data field.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. **Problem Definition**
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Problem Definition

Health Center Scenario:

- A patient agrees to have health information recorded by a shopping cart.
- Health data is transferred from the cart to the patient's phone using Bluetooth.
- The phone transfers the data to medical professionals for analysis.

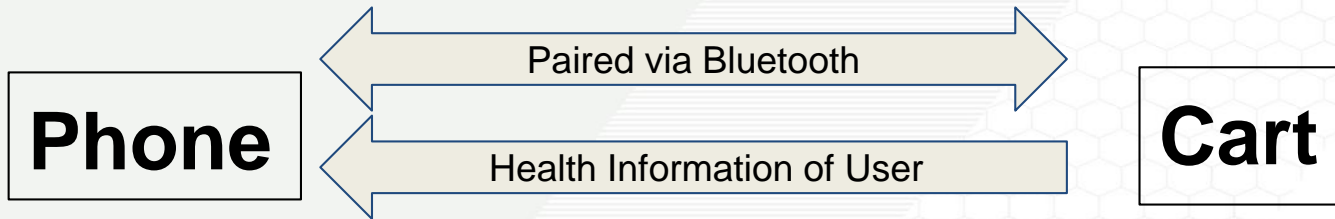


Figure 10. Simplified diagram showing the bluetooth protocol between user's phone and a grocery store's shopping cart.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. **Prior Work**
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Prior Work

Accomplishments: Read and write a set of bits to the MiFare 1k Classic Card using the ACR122U Scripting Tool provided by the manufacturer.

- 1) We understand what each byte from the command and response APDU is (i.e. difference between CLASS and INS)
- 1) Used the previous semester's examples to write a HEX value of "01" into a byte of the MiFare's memory storage.

Prior Work Cont.

APDU Command and APDU Response

; [1] Load (Mifare Default) key in reader (key location 0)
FF 82 00 00 06 **D3 F7 D3 F7 D3 F7** (9000)

; [6] Authenticate sector 1, Block 5 with key at location 0
FF 86 00 00 05 01 00 05 60 00 (9000)

; [7] Store a value "1" into block 5
FF D7 00 05 05 00 00 00 00 01 (9000)

; [8] Read the value block 5
FF B1 00 05 04 [xx xx xx xx] (9000)

- [xx xx xx xx] : is the data field found in APDU Response
- Sub-Experiment: Padding with 0's and 1's

Figure 11. APDU Command and Response to writing a value of '01' into Sector 1, Block 1 (Sector 0, Block 5)

Prior Work Cont.

- Wrote a 10 bit number to smart card and read that number on another device.
- Using an AES encrypt tool^[4] taking the 10 bit number as an input, both devices generated identical keys.
- The transfer was between a Mac laptop and Windows laptop.
- These keys were used to encrypt a text file which was then sent to the other device via Bluetooth and then decrypted using its own key.



Figure 12. Example of a Text file sent via Bluetooth which is encrypted before transmission. It was successfully encrypted, sent, decrypted, and read.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. **Experimental Method**
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

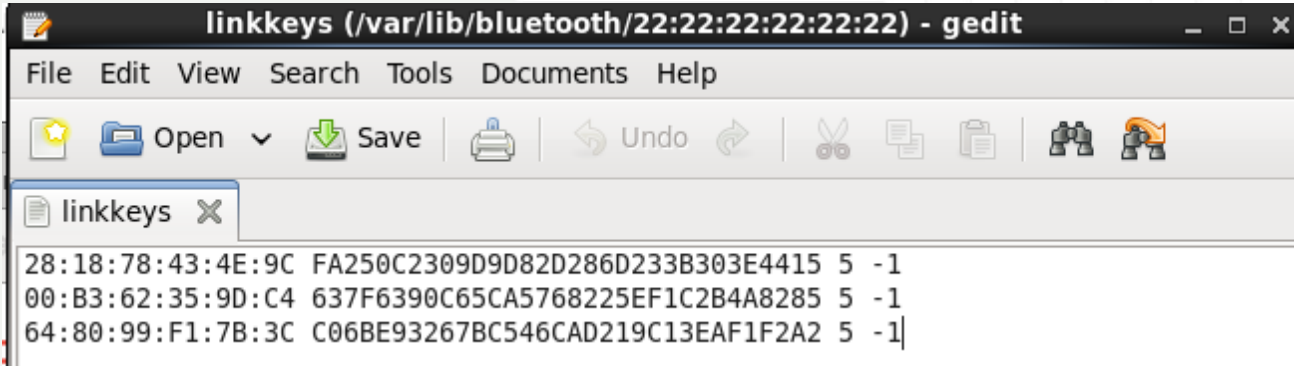
Experimental Method

Necessary Linux Permissions

- In order to proceed with our experiments, we obtained administrative access to specific Linux commands:
 - hcitool: Manages Bluetooth connections
 - hciconfig: Configures Bluetooth devices
 - Rfcomm: Establishes a Bluetooth connection
- Given access to link key directory via `chmod -R /var/lib/Bluetooth/<BD_ADDR>/linkkeys`
 - ECE Help Desk provided our user ID's access to this subdirectory upon boot

Experimental Method Cont.

- First segment is the Bluetooth Address for the device.
- Second segment is a 32 hex digit link key.



```
linkkeys (/var/lib/bluetooth/22:22:22:22:22:22) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
linkkeys x
28:18:78:43:4E:9C FA250C2309D9D82D286D233B303E4415 5 -1
00:B3:62:35:9D:C4 637F6390C65CA5768225EF1C2B4A8285 5 -1
64:80:99:F1:7B:3C C06BE93267BC546CAD219C13EAF1F2A2 5 -1
```

Figure 13. 'Linkkeys file' showing all the bluetooth devices 'ivy' is connected to along with their shared link keys.

Experimental Method Cont.

- Write a link key then initiate a file transfer
- File was able to be sent

```
[molumogba3@ivy 22:22:22:22:22:22]$ vi linkkeys
[molumogba3@ivy 22:22:22:22:22:22]$ more linkkeys
00:B3:62:35:9D:C4 637F6390C65CA5768225EF1C2B4A8285 5 -1
28:18:78:43:4E:9C BB5A09854404034B4405E1AB9C018576 5 -1
E0:06:E6:BF:8A:E8 9E18F9F891AB02DC37D6471FF6A4DAF8 5 -1
64:80:99:F1:7B:3C A9D7F6E5C3E4AB90C2E7AD6CB3E7F8F7 5 -1
[molumogba3@ivy 22:22:22:22:22:22]$
```

Figure 14. New link key is written to share with bluetooth address 64:80:99:F1:7B:3C

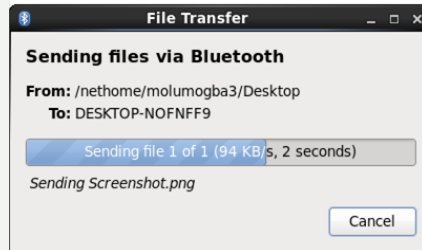


Figure 15. Successful transfer of file after link key change.

Experiment Method Cont.

Test to see whether link key was actually transferred:

- Wrote a new link key
- Sent a file
- Read the link key of the receiving device.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. **Results**
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Results

- The link key was manually written into the file 'linkkeys'.
- The syntax is <bluetooth address> <32-hexadecimal digit link key> <channel> ...
 - a) Reasoning is based on syntax for rfcomm available on its 'man' file
 - i) From rfcomm, bdaddress and channel are parameters written in similar order, 32-bit key is what we expect based on Bluetooth documentations.
- The bluetooth address refers to the other device paired with the computer.

```
config      eir lastseen linkkeys names          sdp
[molumogba3@ivy 00:1A:7D:DA:71:02]$ more linkkeys
28:18:78:43:4E:9C 098A3BCA38B7CDE16D013D8FE2097057 5 -1
[molumogba3@ivy 00:1A:7D:DA:71:02]$ □
```

Figure 16. Screenshot of linkkeys file (under the first shell).

Results Cont.

- A screenshot from the computer was sent via bluetooth and it transferred to the other device after altering the link key.



Figure 17. GUI indicating successful transfer of file after manually writing a new link key into linkkeys

Results Cont.

Test to see whether link key was actually transferred:

- Wrote a new link key
- Sent a file
- Read the link key of the receiving device.
 - Did not change

```
00:1A:7D:DA:71:02 098A3BCA38B7CDE16D013D8FE2097057 5 -1
```

Figure 18. Example of a 'Linkkeys file' of the receiving device showing the bluetooth address of sending device and link key shared between the two. It was the same link key as before alteration.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion**
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Discussion

Suggested Approach 1:

- 1) Bluetooth Address
 - a) Eliminates the need for either party to broadcast their bluetooth address.
 - b) Without knowing addresses, the attacker cannot pose as either device although it can eavesdrop.

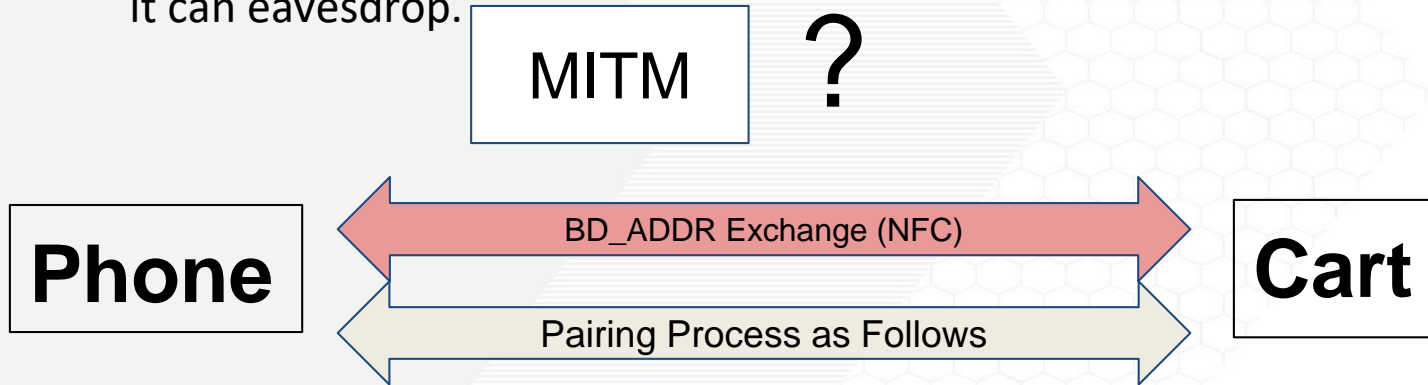


Figure 19. If NFC contains the Bluetooth address of either party, the protocol can be secured.

Discussion Cont.

Suggested Approach 2:

1) Key Transferred OOB

- a) Still lets the attacker pose as the opposite party, but all intercepted data is meaningless without the key (i.e link key.)

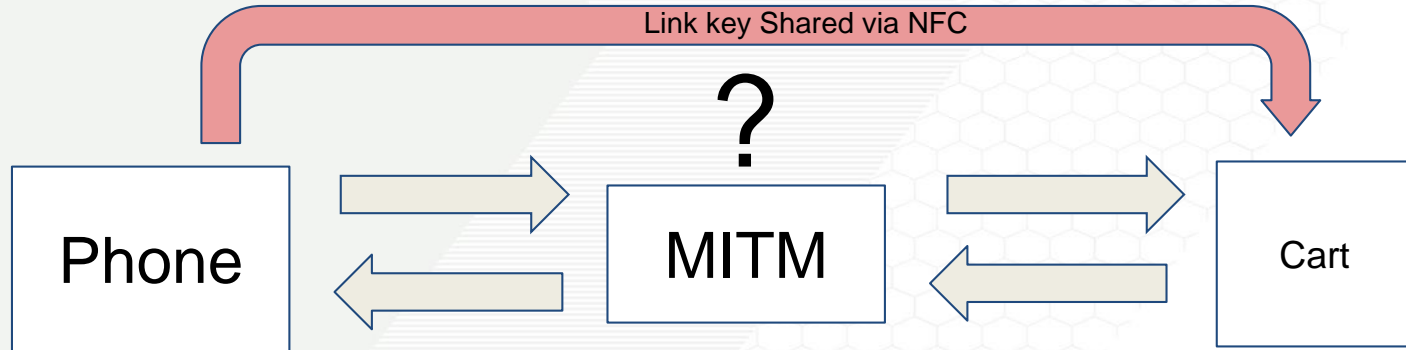


Figure 20. The AES based OOB key is sent via NFC to the opposite party.

Discussion Cont.

Attempts to Overcome Challenges

Try to manually write the link key on both devices

- It was discovered that the link key on the receiving device had not changed although it accepted the file transferred to it.
- Therefore a suggestion was raised to manually write the link key on both devices (both would be Linux Devices.) The receiving device was the newer redhat 6 operating system on 'long.'
- The same permissions were required to enable use of the same commands and reading/writing the linkkeys file.
- Help Desk stated that 'long' had no linkkeys file.

Discussion Cont.

Attempts to Overcome Challenges

Understanding of BlueZ software design architecture to date

- Whenever the 'hcitool key' command is used, it changes the link key on both devices. From one device it is transmitted to the other via Bluetooth.
- By reading the header file titled 'hci.h' it is ascertained that another function within is responsible for changing the link key.
- As of date, the source code is being parsed and a search is underway to find the implementation of that sub-function as it does not seem to be present.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work**
- VIII. Conclusion
- IX. List of References
- X. Appendices

Future Work

- One method for securing a connection would be to install the ACR122U NFC Card Reader Software on Linux in order to develop an automatic way for the link key to be read from a file using the NFC USB or Mifare 1k Card.
- Once the directory is found, permission must be granted to read and write linkkeys on the second linux machine (long) so that transfer of files can be conducted between two linux devices and command line may be used.
- Additionally, it might be useful to debug the existing program for Bluetooth made by a previous semester group. See github
- Find out whether the BlueZ version on 'long' is compatible with that on 'ivy', might be the problem. Type `bluetoothctl --version` or `bluetoothd --version` in command line.
- Conduct the same final experiment between two Linux operating systems.
- Try manually changing the link key and bluetooth address manually after enabling encryption on both devices. Assess whether sending the TK or STK is better than the Link Key.
- It is suggested that understanding the implementation of 'hciconfig putkey' command via the source code will be helpful. Read BlueZ's source code to for its overall software architecture.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Conclusion

- Manipulating the link key still resulted in a completed transfer of a file between paired devices.
- However, the link key did not change at the receiving device, it is not certain what effect manipulating it has.
- It seems to still be transmitting data with the old link key.
- Therefore, it is not yet certain whether the solution of storing a key on NFC for Bluetooth Link Key encryption is preferred.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. **List of References**
- X. Appendices

Bluetooth References

1. "Bluetooth Basics." *Bluetooth Basics*. SparkFun, n.d. Web. 08 Feb. 2017.
2. "Adaptive Frequency Hopping for Reduced Interference between Bluetooth® and Wireless LAN." *Design And Reuse*. Ericsson Technologies, June 2003. Web. 08 Feb. 2017.
3. "BlueZ," *BlueZ: Official Linux Bluetooth protocol stack*. [Online]. Available: <http://www.bluez.org/>. [Accessed: 02-Oct-2017].
4. *AES Crypt - Advanced File Encryption*. Packetizer, Inc, n.d. Web. 19 Apr. 2017.
5. Singh, S. (2015). *Cellphone-based Authentication in Public Spaces*. Atlanta: Secure Hardware Team.
6. J. Padgette, J. Bahr, M. Batra, M. Holtmann, R. Smithbey, L. Chen, and K. Scarfone, "Guide to Bluetooth Security," NIST, publication.

NFC References

7. J. Thrasher. (2013, October 11). *RFID vs. NFC: What's the Difference?* [Online]. Available: <http://blog.atlasrfidstore.com/rfid-vs-nfc>. Accessed: Feb. 04, 2017
8. S. Tamrakar, “NFC Application Security”, Aalto Univ., 2013. [Online]. Available: <http://slideplayer.com/slide/1532054/>. Accessed: Feb. 04, 2017
9. M. Sievänen, “Application Protocol Data Unit”, Telecommunications Software and Multimedia Laboratory, 2003. [Online]. Available: <http://www.tml.tkk.fi/Studies/T-110.497/2003/lecture4.pdf>. Accessed: Dec. 1, 2016

Other References

10. Wu, F., Bozkurt, D., & Mooney, P. (2016). *Bluetooth Man in the Middle Protection in Smart Shopping Cart Scenario*. Atlanta: Secure Hardware Team.
11. Mayberry, L. (2016). *Near Field Communication*. Atlanta: Secure Hardware Team.

Outline

- I. Background
 - a) Bluetooth Background
 - b) NFC Background
- II. Problem Definition
- III. Prior Work
- IV. Experimental Method
- V. Results
- VI. Discussion
- VII. Future Work
- VIII. Conclusion
- IX. List of References
- X. Appendices

Appendices

The following presentations from previous semesters are included as separate powerpoint files. The most relevant presentations are included first (e.g., Appendix A has the highest relevance to this semester's work).

- Appendix A: <Research on Wireless Technology Standards for the Development of Secure Protocols to Prevent Man in the Middle Attacks, April 24 2017>
- Appendix B: <Bluetooth Man in the Middle Protection in Smart Shopping Cart Scenario, December 2 2016>
- Appendix C: <Near Field Communication, December 1 2016>
- Appendix D: <Cellphone-based Authentication in Public Spaces, April 15 2015>