Bluetooth security threats and possible countermeasures

MSc Keijo Haataja
Senior assistant
Department of CS
University of Kuopio
Finland

Main references

- My Bluetooth research and teaching
- Bluetooth specifications 1.1 and 1.2
- Robert Morrow, *Bluetooth: Operation and use*, 2002.
- D. M. Bakker, Diane McMichael Gilster, *Bluetooth end to end*, 2002.
- Bluetooth Americas 2003 congress
- BlueZ mailing lists and documentation
- Tuomas Kepanen, *Bluetooth chat software for Linux and Windows: BTChatd, BTChat and BTChatJava*, 2003.
- Matthew Dunn, European Technical Manager, CATC

Contents

- Overview of Bluetooth technology and Bluetooth security
- Bluetooth security threats and countermeasures
- Overview of our Bluetooth laboratory, Bluetooth equipments, and some test data

General

- Open specification for wireless data and voice transfer
- 5x5 mm microchips form ad-hoc networks
- Design goals are simplicity, compatibility, fast data transfer and globality
- 2.4 GHz ISM-band (Industrial Scientific Medicine), f=2402+k MHz, k=0,…,78
- Bluetooth SIG (Bluetooth Special Interest Group) develops technology and brings devices to the market
- Current Bluetooth specification is 1.2

General

- Bluetooth device classes
  - Class 1: max power 100 mW (20 dBm)
  - Class 2: max power 2.5 mW (4 dBm)
  - Class 3: max power 1 mW (0 dBm)
- Reference sensitivity level of Bluetooth device is -70 dBm (or better)
- Range of Bluetooth devices depends on
  - Device class on both ends
  - Sensitivity level on both ends
  - Level of obstacles (none, light, moderate, heavy)

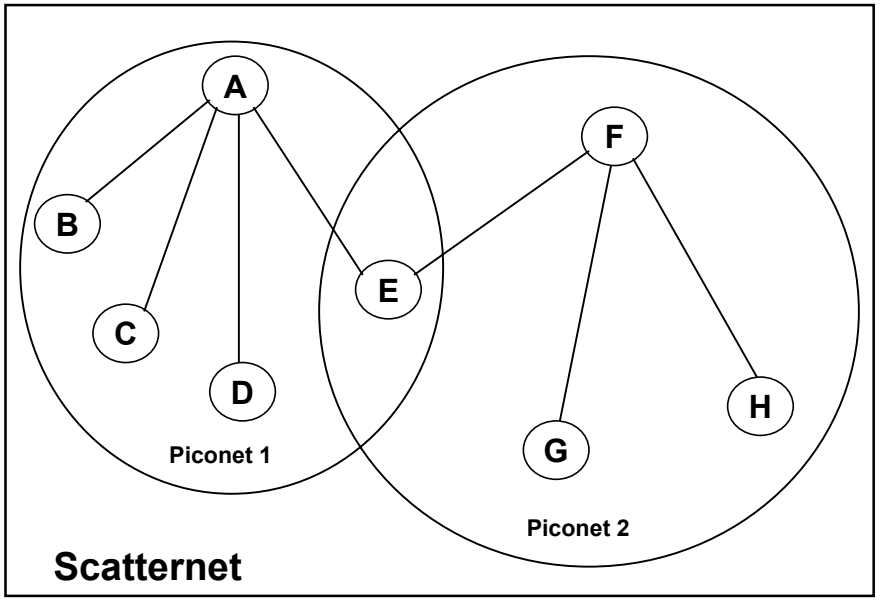
Range of Bluetooth devices

Level of obstacles:	TX power [mW]:	RX sensitivity [dBm]:	Range [m]:
None	1	-70	31
None	1	-80	100
None	100	-70	316
None	100	-80	1000
Light	1	-70	16
Light	1	-80	40
Light	100	-70	100
Light	100	-80	251
Moderate	1	-70	10
Moderate	1	-80	22
Moderate	100	-70	46
Moderate	100	-80	100
Heavy	1	-70	6
Heavy	1	-80	10
Heavy	100	-70	18
Heavy	100	-80	32

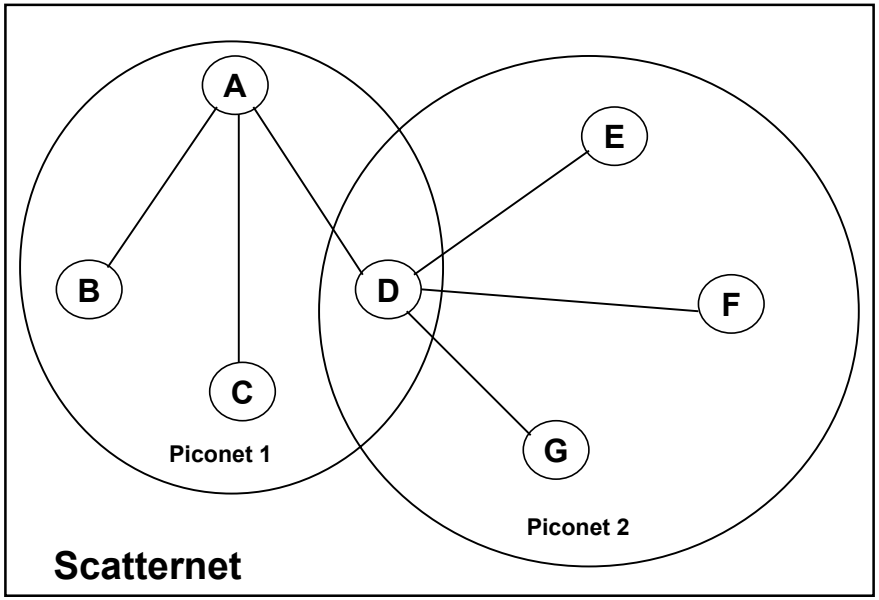
Connection types

- ACL (Asynchronous Connection-Less)
  - Symmetric (max 433.9 kb/s for both directions)
  - Asymmetric (max 723.2 kb/s for send and 57.6 kb/s for receive)
  - Retransmission is used to ensure integrity of data
- SCO (Synchronous Connection-Oriented)
  - Symmetric (64 kb/s for both directions)
  - No retransmission of packets (voice)
- eSCO (extended SCO)
  - Symmetric (64 – 288 kb/s for both directions)
  - Retransmission is used to ensure integrity of data (voice)

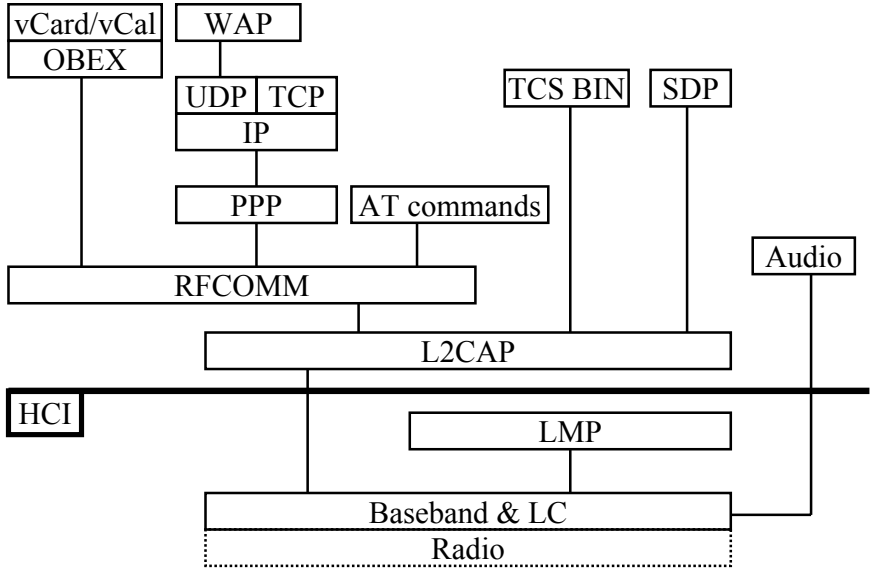
Bluetooth topology (ACL link)



Bluetooth topology (SCO/eSCO link)



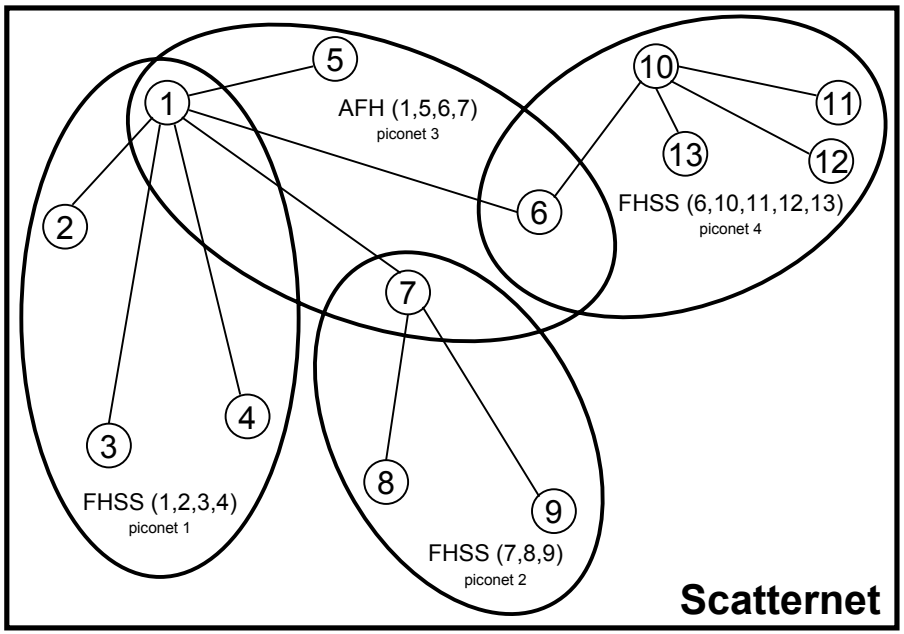
Bluetooth protocol stack



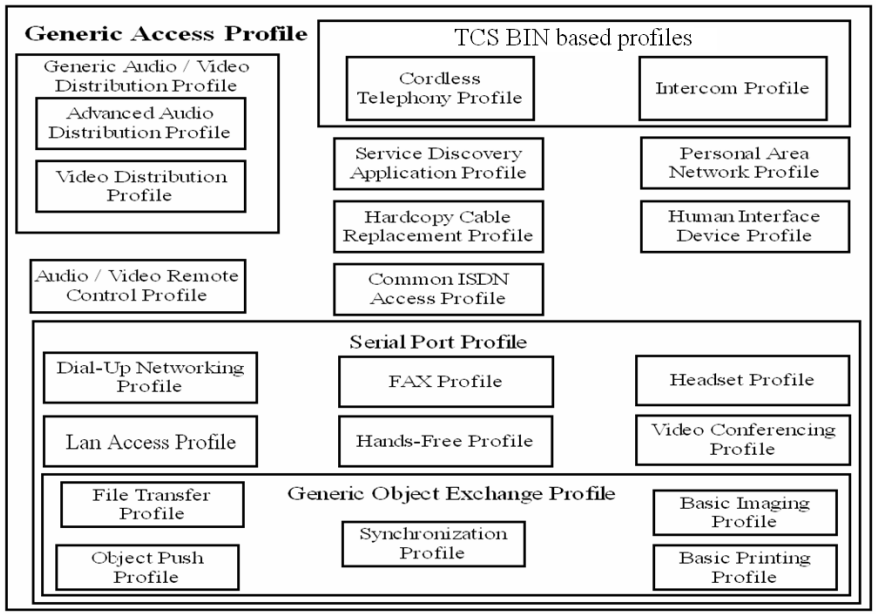
Frequency hopping

- FHSS (Frequency Hopping Spread Spectrum), f=2402+k MHz, k=0,…,78
  - Maximum hopping rate is 1600 hops per second
  - Bluetooth 1.0/1.1/1.2 devices use FHSS
- AFH (Adaptive Frequency Hopping) supports dynamic replacement of frequencies
  - Maximum hopping rate is 800 hops per second
  - 59 ”bad” channels can be switched off during the communication session => Hopping sequence can be as small as 20 frequencies
  - Bluetooth 1.2 devices use AFH

Bluetooth hybridtopology (FHSS & AFH)



Bluetooth profiles



Bluetooth profiles

- Profile defines the capabilities of Bluetooth device
- All profiles are based on GAP (Generic Access Profile) => Inner profiles use outer profiles to define their own functionality
- Profile
  - Decreases the amount of needed protocols and set boundaries to the parameter values
  - Defines execution order of needed processes
  - Offers compatibility between devices

BlueZ

- Official protocol stack for Bluetooth in Linux environments
  - Kernel modules of BlueZ are included in the Linux kernel
  - http://www.bluez.org
    - Libraries and tools
    - Documentation and mailing lists for getting help
  - Open source
  - Wide support for different Bluetooth devices
  - Fast development work
  - Support for UART, USB and PCMCIA devices
  - HCI emulation can be used in case of no physical devices
  - SCO, eSCO, L2CAP, SDP, RFCOMM, OBEX
  - Configuration and testing tools

Bluetooth security

- Because Bluetooth is a wireless communication system, there is a definite possibility that its transmissions could be deliberately intercepted or jammed, or false information passed to the piconet members!
- To provide usage protection for the piconet, the system must establish security at several protocol levels
- Unlike many network protocols that leave the issue of security up to attached software modules, Bluetooth offers built-in security measures at the link level

Security threats

- Threats in distributed networks can be roughly divided into three categories:
  - **Disclosure threat**: Leakage of information from the target system to an eavesdropper that doesn't have authorization to access the information.
  - **Integrity threat**: Deliberate alteration of information in an attempt to mislead the recipient.
  - **DoS (Denial of Service) threat**: Blocking of access to a service, making it either unavailable or severely limiting its availability to an autorized user.

Overview of Bluetooth security

- Security within Bluetooth itself covers three major areas: **Authentication, authorization** and **encryption**
- The process of *authentication* proves the identity of one piconet member to another => The results of authentication are used for determining client's *authorization* to access various services on a server
- The process of *encryption* is used to encode the information being exchanged between devices such that eavesdroppers (even other members of the same piconet) cannot read its contents
- These three security processes are implemented within several layers of the Bluetooth protocol stack, so security is often termed as a cross-layer function

Overview of Bluetooth security

- LC (Link Controller) has random number generation capability, and includes methods for managing security keys and providing the mathematical operations for authentication and encryption
- LM (Link Manager) includes several commands for handling security issues
- L2CAP (Logical Link Control and Adaptation Protocol) can initiate security procedures when a channel connection attempt is made
- HCI (Host Controller Interface) handles security communication between host and Bluetooth module
- Bluetooth security can be enhanced by using some third party encryption method (e.g. 3DES, Blowfish, RSA, ...) as an extra security in addition to the Bluetooth built-in security

Security levels

- GAP specifies how Bluetooth security is organized
- Security begins when a user decides how a device will implement its discovery and connectability options
- Different combinations of these capabilities can be divided into three general categories (*security levels*):
  - **Silent**: The device will never accept any connections. It simply monitors Bluetooth traffic.
  - **Private**: The device cannot be discovered. Connections will be accepted if the device's BD_ADDR (Bluetooth Device Address) is known by the prospective master.
  - **Public**: The device can be both discovered and connected to.

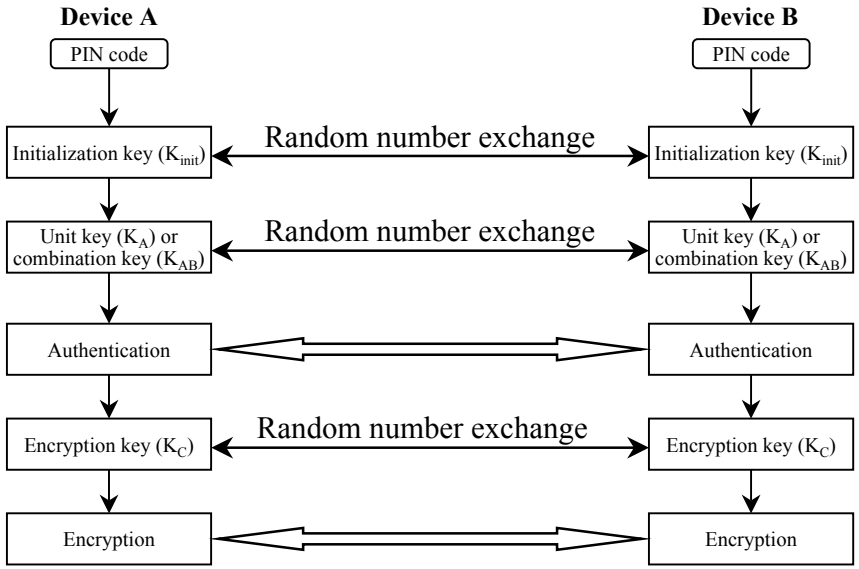
Security modes

- GAP defines three different *security modes* that a device can implement (a device can be only in one security mode at a time):
  1. **Nonsecure**: A device will not initiate any security measures, so communication takes place without authentication or encryption.
  2. **Service-level enforced security**: Two devices can establish an ACL link in a nonsecure manner. Security procedures are initiated when a L2CAP channel request is made.
  3. **Link-level enforced security**: Security procedures are initiated when the ACL link is being established.

Summary of Bluetooth security operations

- The philosophy behind Bluetooth security is to build a chain of events, none of which provides meaningful information to an eavesdropper, but all must occur in a specific sequence for security to be set up successfully
- The devices begin with a common PIN (Personal Identification Number) code upon which several 128 bit keys are built
- During key generation and authentication, the devices transmit random numbers to each other that provide no information to an eavesdropper

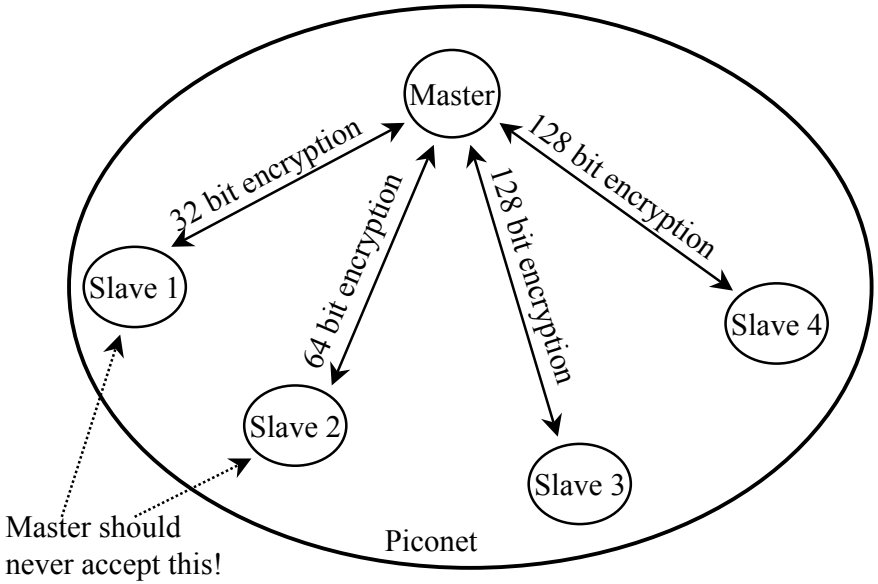
Summary of Bluetooth security operations



Security entities

Entity	Description	Length (bits)	Status
PIN	Personal Identification Number	8, 16, …, 128	Private
BD_ADDR	Bluetooth Device Address	48	Public
IN_RAND	Random number for generating K_{init}	128	Public
LK_RAND	Random number for generating K_{AB}	128	Private
K_{init}	Initialization key	128	Private
K_A	Unit key (not recommended)	128	Private
K_{AB}	Combination key	128	Private
K_{master}	Master key for broadcast messages	128	Private
AU_RAND	Random number for authentication	128	Public
SRES	Authentication result	32	Public
EN_RAND	Random number for generating K_C	128	Public
ACO	Authenticated Ciphering Offset for generating K_C	96	Private
K_C	Encryption key	8, 16, …, 128	Private

An example of a bad security



Encryption

- From a security standpoint, one of the major shortcomings of wireless is the fact that its transmissions can be heard over relatively large distances away from the simple line-of-sight path between endpoints
- Indeed, one of the characteristics of most ad-hoc wireless devices is their use of omnidirectional antennas so that their signals can be received by other devices, regardless of where they are relative to each other
- Sending information in all directions facilitates eavesdropping as well, so sensitive data should be encrypted to prevent its unauthorized use

Bluetooth encryption strengths

- The encryption key can be as long as 128 bits, compared to 40 or 64 bits for many other encryption implementations, greatly reducing the chance that an eavesdropper will be able to guess the key
- The encryption key is built from several secret entities derived over a period of time, reducing the change that it can be rederived by an eavesdropper
- The cipher stream generator is reinitialized at the beginning of every transmitted packet => This prevents an intruder from intercepting several copies of the same cipher stream

Bluetooth piconet vulnerability

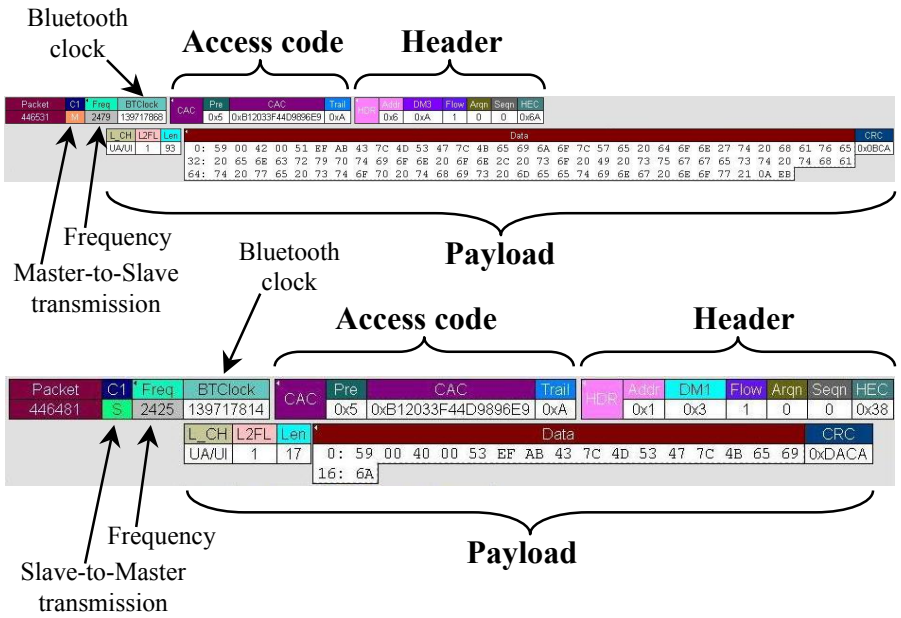
- The distance over which the Bluetooth network can be intercepted or disrupted can be significantly greater than that over which normal communication takes place
- An intruder can also place a bug nearby to intercept and record Bluetooth activity over a long period of time
- The bug can be retrieved later on and its contents analyzed, perhaps using powerful computation techniques in an attempt to break the cipher on encrypted data

Disclosure threat

- Disclosure threat is especially serious => The presence of an eavesdropper is often not seen because of the physical separation of eavesdropping equipment from the communicating devices
- Unencrypted transmissions are easy prey for an eavesdropper, but even encrypted packets can be recorded for later cryptographic analysis

Sensitivity level of eavesdropper's radio:	Target piconet TX power:	Range of Vulnerability:
-100 dBm	1 mW	250 m
-100 dBm	100 mW	1580 m

Example of packet interception with protocol analyzer



Encryption weaknesses

K_C length (bits):	Average search time at 2^{20} trials per second:	Average search time at 2^{40} trials per second:
8	≈ 122 microseconds	≈ 116 picoseconds
16	≈ 31 milliseconds	≈ 30 nanoseconds
24	≈ 8 seconds	≈ 8 microseconds
32	≈ 34 minutes	≈ 2 milliseconds
40	≈ 6 days	≈ 500 milliseconds
48	≈ 4 years	≈ 128 seconds
56	≈ 1090 years	≈ 9 hours
64	≈ 278922 years	≈ 97 days
72	≈ 71 million years	≈ 68 years
80	≈ 18 billion years	≈ 17433 years
128	≈ 162×10^{30} years	≈ 155×10^{24} years

Integrity threat

- Several methods can be employed by spoofer in his attempt to access the piconet:
  - Exploiting a weakness in the PIN and initialization process
  - Exploiting a weakness in the unit key
  - Attempting to pair repeatedly with different authentication responses
  - Acting as the verifier in a one-way authentication
  - Using a stronger RF signal to displace an active piconet member

Example: Access via RF capture

- An example how spoofer uses access via RF capture to download a sensitive file:

1. Spoofer eavesdrops until he can identify the BD_ADDR of a server S on which the target file F is located. He also identifies the BD_ADDR of a device A that seems to have the proper authorization to access that file.
2. Spoofer eavesdrops long enough to learn how A accesses the files on S, and he gathers any other information that he may need to eventually access file F himself.
3. When spoofer is ready to carry on the attack to the next level, spoofer waits until A connects to S and is properly authenticated. Device A may have no intention of downloading file F during this particular session.
4. At an appropriate time during the communication session between S and A, spoofer captures A's channel by assuming the identity of A and transmitting signals that are at least 11 dB stronger at S's reveicer than what A was sending.
5. Continuing to pose as A, spoofer then requests the desired information from S.

DoS threat

- The classic DoS threat involves disruption of the piconet such that legitimate throughput is either slowed considerably or shut down completely
- This threat can be accomplished in several different ways and at different levels of Bluetooth protocol stack
- At the physical layer (PHY), an intruder can either capture the channel from a legitimate piconet member or jam the piconet entirely
- Attacks on higher protocol layers try exploit some of their characteristics in an attempt to occupy the attention of one or more members of the piconet such that they're unable to service other devices in a timely manner

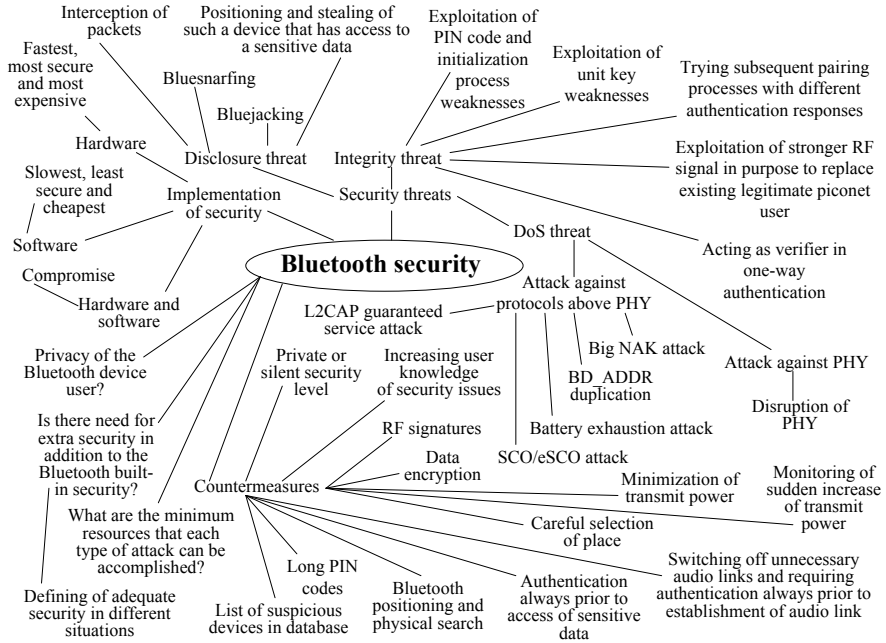
Disrupting the PHY

- Let's assume that the target piconet receivers all have a desired signal power level of either -60 dBm or -40 dBm, and jammer can transmit either 100 mW (Class 1 device) or 1 W (Bluetooth transmitter with gain antenna)
- Range of susceptibility for jammer on a target piconet is shown below:

Minimum jamming signal power at target receiver:	Jammer's TX power:	Range of susceptibility:
-40 dBm	100 mW	6.3 meters
-40 dBm	1 W	16 meters
-60 dBm	100 mW	40 meters
-60 dBm	1 W	100 meters

Disrupting protocols above the PHY

- Aside from occuring at the PHY, DoS attacks can also take place within the framework of protocols within the data link layer and higher
- Here is a list of possible attacks on some of the higher protocols:
  - Big NAK (Negative Acknowledgement) attack
  - BD_ADDR duplication
  - SCO/eSCO attack
  - L2CAP guaranteed service attack
  - Battery exhaustion attack



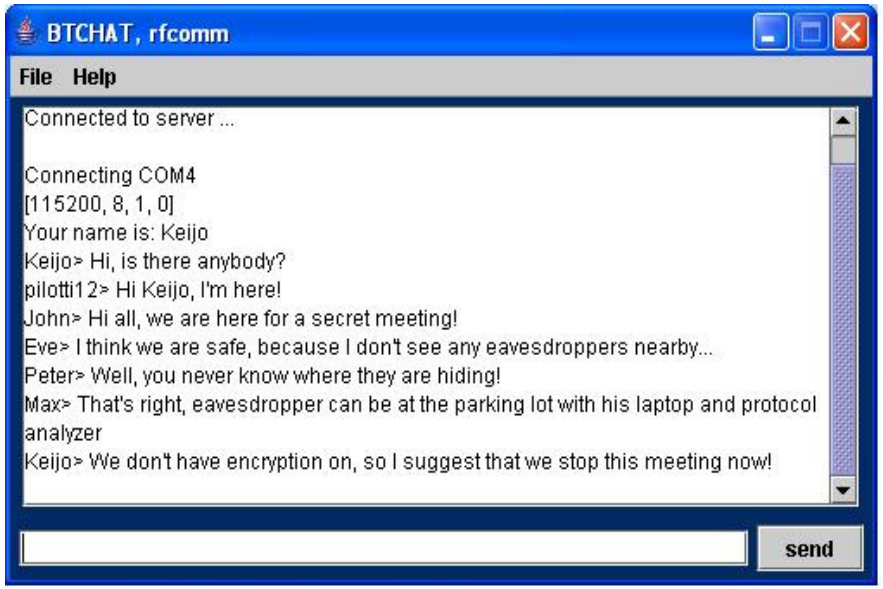
Bluetooth laboratory

- Department of CS in University of Kuopio has Bluetooth laboratory:
  - 17 computers with *Windows XP* and *Linux Fedora Core 2*
  - Protocol analyzer system (sniffing/jamming of Bluetooth piconets/scatternet can be done) with two Bluetooth radio modules => Same as two separate analyzers but better because of a common Bluetooth clock => Enables also to sniff hybridpiconets (Bluetooth 1.1 & 1.2 devices) or scatternet
  - Bluetooth development kits, Bluetooth USB adapters, Bluetooth PCMCIA cards, Bluetooth HF devices, Bluetooth mobile phones, Pocket PCs with Bluetooth CF cards (or with embedded Bluetooth chips), ...
  - Bluetooth chat software (at the moment supports only Bluetooth data link)
    - BTChatd (Bluetooth Chat server for Linux)
    - BTChat (Bluetooth Chat client for Linux)
    - BTChatJava (Bluetooth Chat java client for Linux and Windows)

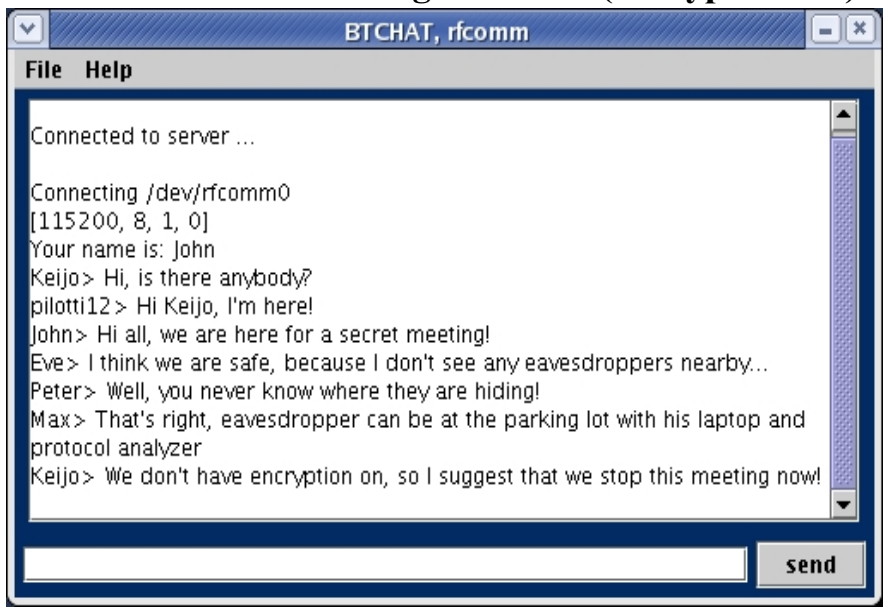
BlueZ and BTChatd startup (encryption off)

```
May 27 10:25:25 itmw-ope24 hcid[7092]: HCI daemon ver 2.4 started
May 27 10:25:25 itmw-ope24 bluetooth: hcid startup succeeded
May 27 10:25:25 itmw-ope24 bluetooth: sdpd startup succeeded
May 27 10:25:31 itmw-ope24 btchatd: Serial Port service registered
May 27 10:25:31 itmw-ope24 btchatd: Logging chat to /tmp/btlog.txt
May 27 10:25:31 itmw-ope24 btchatd: BtCHATD, BlueZ RFCOMM chat server running!
May 27 10:25:31 itmw-ope24 btchatd: Waiting for connection on RFCOMM channel 10
May 27 10:25:31 itmw-ope24 btchatd: Tuomas Kepanen, kepanen@cs.uku.fi
May 27 10:25:55 itmw-ope24 btchatd: Connection from [00:05:4E:00:68:64]
May 27 10:26:52 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F5]
May 27 10:27:07 itmw-ope24 btchatd: Connection from [00:05:16:48:12:4C]
May 27 10:27:32 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F2]
May 27 10:27:48 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F3]
May 27 10:28:02 itmw-ope24 btchatd: Connection from [00:05:16:48:10:58]
May 27 10:29:32 itmw-ope24 btchatd: Connection from [00:05:16:48:12:55]
```

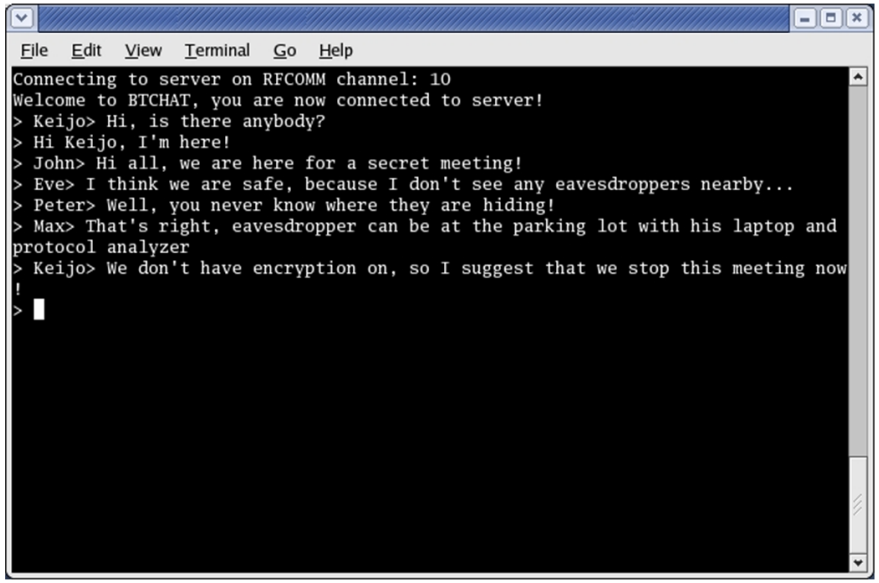

BTChatJava running on Windows (encryption off)



BTChatJava running on Linux (encryption off)



BTChat running on Linux (encryption off)



Packet interception with protocol analyzer (encryption off)

Packet	C1	Freq	BTClock	CAC	Pre	CAC	Trail	HDR	Addr	DM3	Flow	Arqn	Seqn	HEC	L_CH	L2FL	Len
446531	M	2479	139717868	CAC	0x5	0xB12033F44D9896E9	0xA	HDR	0x6	0xA	1	0	0	0x6A	UA/UI	1	93

Data	CRC	Ack'd	Idle	Time Stamp
0: YDBDQDDC|MSG|Keijo|We don't have 32: encryption on, so I suggest tha 64: t we stop this meeting now!DD	0x0BCA	Ack	358.800 µs	00811.712 1418

Packet	C1	Freq	BTClock	CAC	Pre	CAC	Trail	HDR	Addr	DM3	Flow	Arqn	Seqn	HEC	L_CH	L2FL	Len
446535	M	2408	139717876	CAC	0x5	0xB12033F44D9896E9	0xA	HDR	0x2	0xA	1	0	1	0xE4	UA/UI	1	93

Data	CRC	Ack'd	Idle	Time Stamp
0: YDADQDDC|MSG|Keijo|We don't have 32: encryption on, so I suggest tha 64: t we stop this meeting now!DD	0x71C1	Ack	358.800 µs	00811.714 6418

Packet	C1	Freq	BTClock	CAC	Pre	CAC	Trail	HDR	Addr	DM3	Flow	Arqn	Seqn	HEC	L_CH	L2FL	Len
446539	M	2402	139717884	CAC	0x5	0xB12033F44D9896E9	0xA	HDR	0x3	0xA	1	0	1	0x8C	UA/UI	1	93

Data	CRC	Ack'd	Idle	Time Stamp
0: YDCDQDDC|MSG|Keijo|We don't have 32: encryption on, so I suggest tha 64: t we stop this meeting now!DD	0x2233	Ack	358.800 µs	00811.717 1418

Packet	C1	Freq	BTClock	CAC	Pre	CAC	Trail	HDR	Addr	DM3	Flow	Arqn	Seqn	HEC	L_CH	L2FL	Len
446543	M	2452	139717892	CAC	0x6	0xB12033F44D9896E9	0xA	HDR	0x4	0xA	1	0	1	0x5F	UA/UI	1	93

Data	CRC	Ack'd	Idle	Time Stamp
0: YDCDQDDC|MSG|Keijo|We don't have 32: encryption on, so I suggest tha 64: t we stop this meeting now!DD	0x2233	Ack	358.800 µs	00811.719 6428

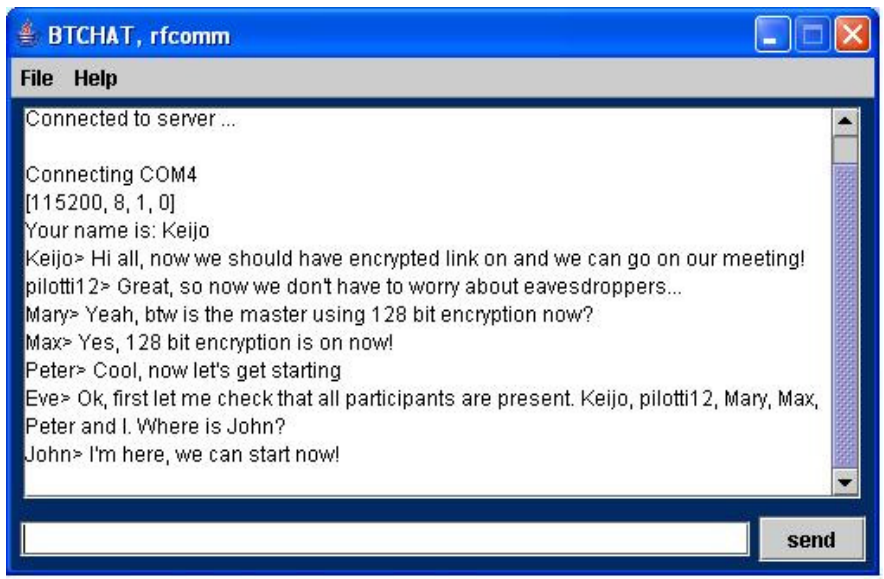
Packet	C1	Freq	CAC	Pre	CAC	Trail	HDR	Addr	DM3	Flow	Arqn	Seqn	HEC	L_CH	L2FL	Len
446547	M	2442	CAC	0x7	0xB12033F44D9896E9	0xA	HDR	0x5	0xA	1	1	1	0xA0	UA/UI	1	93

Data	CRC	Ack'd	Idle	Time Stamp
0: YDCDQDDC|MSG|Keijo|We don't have 32: encryption on, so I suggest tha 64: t we stop this meeting now!DD	0x2233	Ack	358.800 µs	00811.722 1418

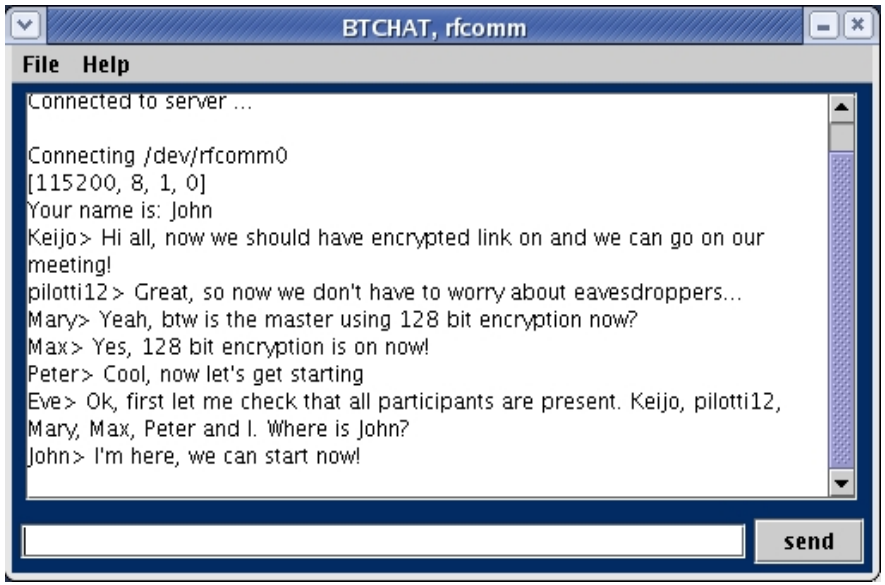
BlueZ and BTChatd startup (encryption on)

```
May 27 12:11:23 itmw-ope24 hcid[7374]: HCI daemon ver 2.4 started
May 27 12:11:23 itmw-ope24 hcid[7374]: Starting security manager 0
May 27 12:11:23 itmw-ope24 bluetooth: hcid startup succeeded
May 27 12:11:23 itmw-ope24 sdpd[7380]: sdpd v1.5 started
May 27 12:11:23 itmw-ope24 bluetooth: sdpd startup succeeded
May 27 12:11:25 itmw-ope24 btchatd: Serial Port service registered
May 27 12:11:25 itmw-ope24 btchatd: Logging chat to /tmp/btlog.txt
May 27 12:11:25 itmw-ope24 btchatd: BtCHATD, BlueZ RFCOMM chat server running!
May 27 12:11:25 itmw-ope24 btchatd: Waiting for connection on RFCOMM channel 10
May 27 12:11:25 itmw-ope24 btchatd: Tuomas Kepanen, kepanen@cs.uku.fi
May 27 12:11:40 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:4E:00:68:64)
May 27 12:11:40 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:4E:00:68:64)
May 27 12:11:40 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
May 27 12:11:40 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:4E:00:68:64
May 27 12:11:40 itmw-ope24 btchatd: Connection from [00:05:4E:00:68:64]
May 27 12:14:14 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F5)
May 27 12:14:14 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F5)
May 27 12:14:14 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
May 27 12:14:14 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:0C:F5
May 27 12:14:14 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F5]
May 27 12:16:13 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:12:4C)
May 27 12:16:13 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:12:4C)
May 27 12:16:13 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
May 27 12:16:13 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:12:4C
May 27 12:16:13 itmw-ope24 btchatd: Connection from [00:05:16:48:12:4C]
May 27 12:17:02 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F2)
May 27 12:17:02 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F2)
May 27 12:17:02 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
May 27 12:17:02 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:0C:F2
May 27 12:17:02 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F2]
```

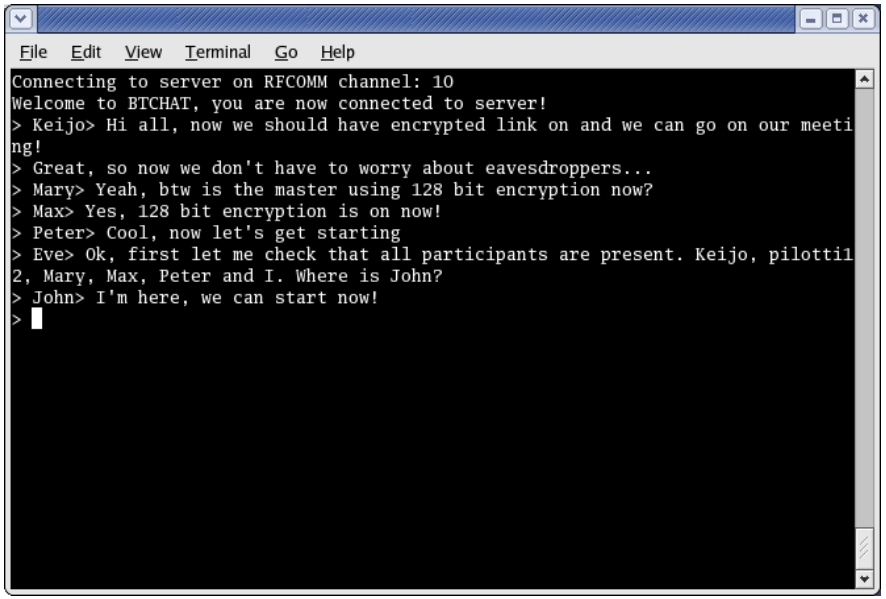

BTChatJava running on Windows (encryption on)



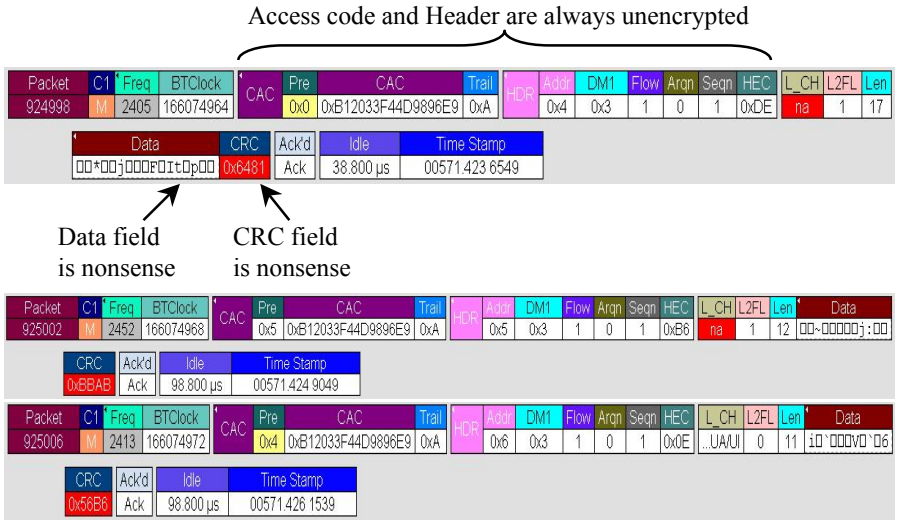
BTChatJava running on Linux (encryption on)



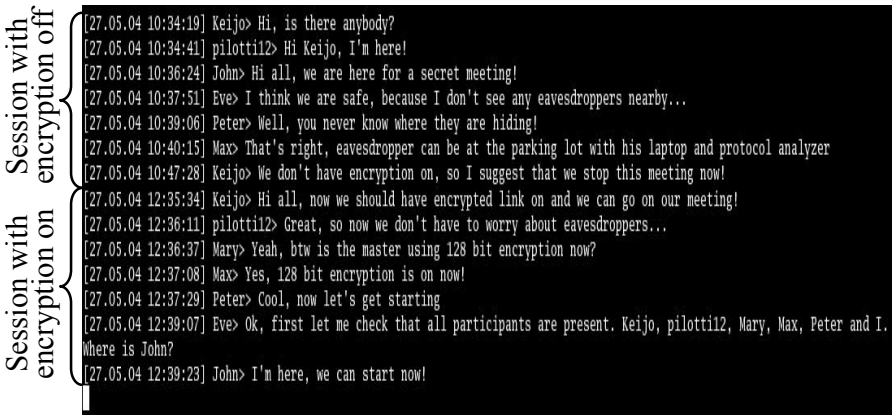
BTChat running on Linux (encryption on)



Packet interception with protocol analyzer (encryption on)



BTChatd logfile (/tmp/btlog.txt)



THANK YOU!

ANY QUESTIONS?