Methods for Accessing a Link Key

Foreword

Since the advent of Simple Secure Pairing (SSP), Bluetooth engineers have faced a new challenge in debugging Bluetooth devices. SSP simplifies the end user's pairing experience while providing greater security for the connection, but it inherently presents obstacles to Bluetooth engineers seeking to decrypt analyzer-captured Bluetooth traffic (of course, these obstacles are intentionally present for anyone seeking to do such decryption).

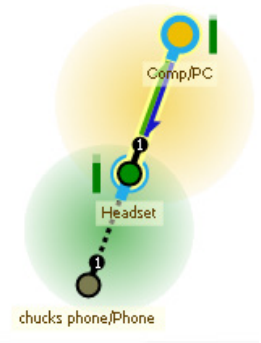
At the center of the challenge is gaining access to the link key. This document will explore some of the methods available to locate and access this key. The Ellisys Bluetooth Explorer 400 Analyzer is uniquely capable of capturing encrypted Bluetooth traffic without *a priori* knowledge of the link key. Encrypted traffic captured using the Ellisys analyzer can be decrypted at a later time when the link key becomes available, and therefore fully understood by the Bluetooth engineer.

The link key is a 128-bit random number and is a "shared secret" between devices; it is never transmitted over the air. It is created during the pairing process, is stored and kept secret, and is used at every connection for authentication and derivation of the encryption key. Even a portion of the pairing sequence, where the link key is created, is encrypted.

The link key is just one of the parameters used to generate the encryption key. See EEN_BT07 - Secure Simple Pairing Explained for details on how SSP works.

Since the link key is never transmitted over the air, the Bluetooth engineer must be aware of the various methods available to access this key. This Expert Note will cover some of these methods. Note that for Bluetooth Smart (Bluetooth Low Energy), things work a bit differently, in that there is in fact an over-the-air transmission of a critical key – see the section on Bluetooth Smart (Low Energy) below.

Key Concepts

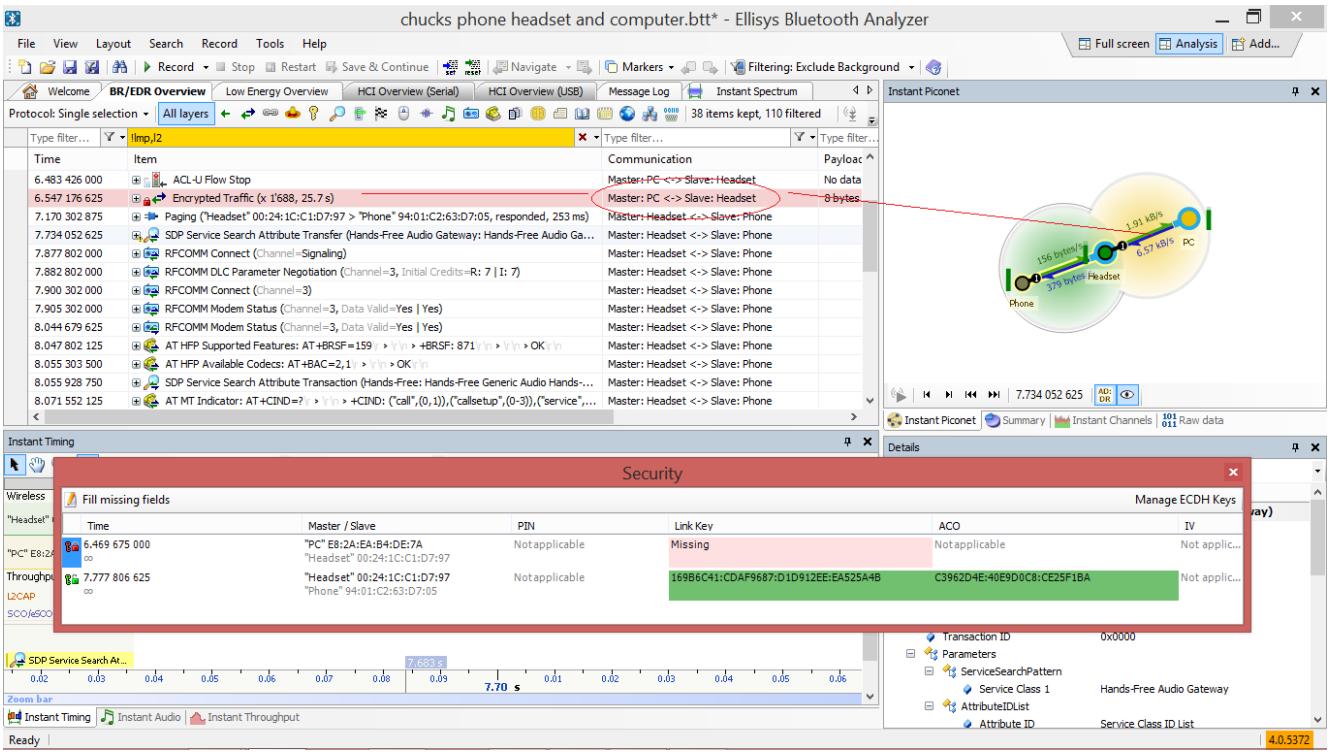


Below and right, the Ellisys Bluetooth Explorer software is used to illustrate some key concepts. In the illustration, we see three devices (a phone, headset, and a PC) comprising two connected piconets (one scatternet). The PC is master to the headset, and the headset is dual-role: master to the phone as well as slave to the PC.

The connection between the headset and the phone has been fully decrypted, as the link key has been obtained (as indicated in the Security pane). The connection from the headset to the PC however is still encrypted, because the link key has not yet been obtained.

The link key between the phone and the headset was obtained using a vendor-specific method and is described in the Android stack section in Vendor-specific methods, described later in this document.

Note that the analyzer is actually capturing the encrypted traffic (PC to headset) and displaying this on a single line. This line will expand into decrypted traffic once the key is provided to the software, either during the capture or after it is saved. See the section on Vendor-specific methods later in this document for details on how this (PC to headset) encrypted traffic is decrypted.



Standard methods

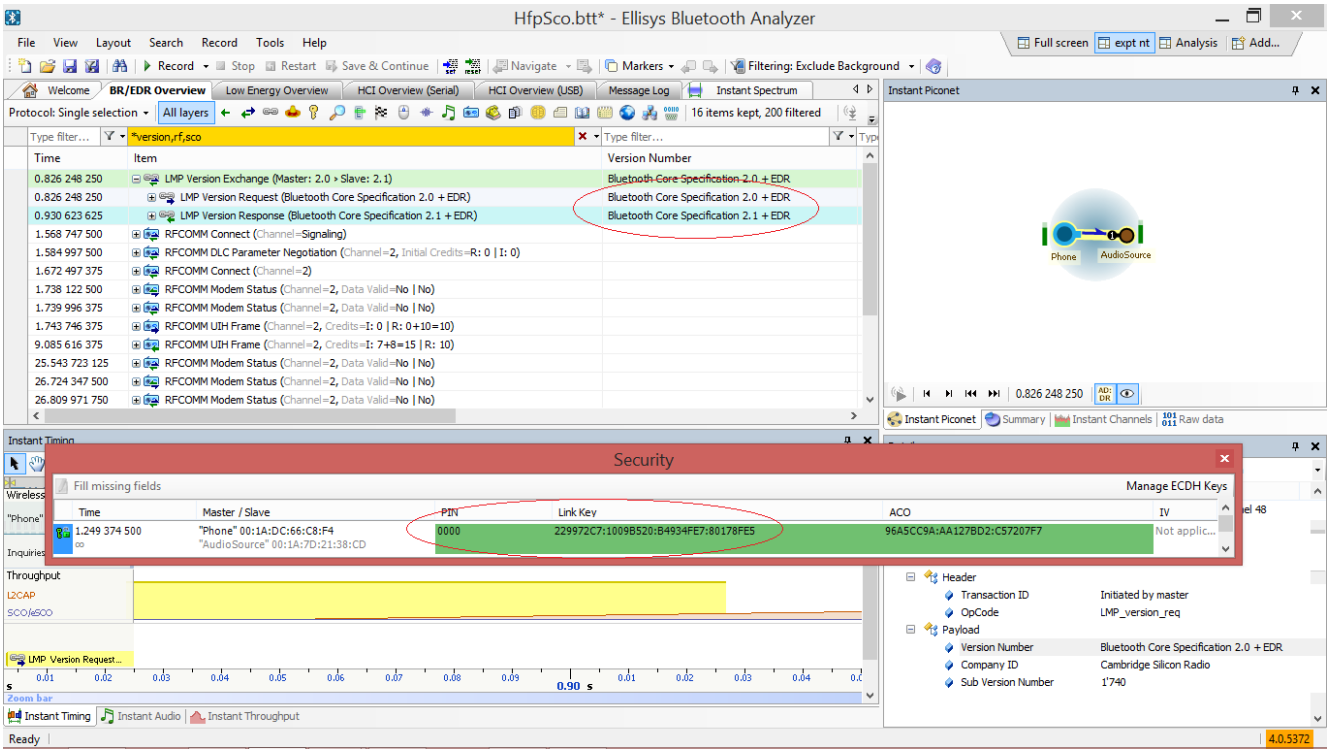
The following methods work independently of the Bluetooth radio or stack manufacturer used. However, these methods may require low-level access to the Bluetooth hardware and are therefore not always applicable with off-the-shelf products.

PIN-code-based pairing

LMP pairing, also known as PIN-code-based pairing or sometimes called Legacy Pairing, is not at all secure, but as with SSP, it also uses/creates a link key that must be known in order to decrypt the traffic. The Ellisys BEX400 analyzer software can automatically decipher the PIN code and deduce the link key, just by looking at the captured pairing traffic. PIN-code-based pairing is not used for Bluetooth 2.1 devices and higher (SSP-enabled devices), *unless these devices are pairing to a Bluetooth 2.0 or earlier device*. This means you can use the Ellisys analyzer to fully decrypt traffic from a default SSP-enabled device if you pair it with a Bluetooth 2.0 or earlier device, thus forcing the SSP-enabled device to revert to (crackable) PIN-code-based pairing.

Below is an example of a Bluetooth 2.1 device (or higher) pairing with a Bluetooth 2.0 device (or lower), where the pairing reverts to PIN-code pairing (as the Bluetooth 2.0 device has no concept of SSP). The BEX400 software has deciphered the Pin Code of 0000 and calculated the Link Key, with subsequent decryption of the traffic taking place automatically.

Note that if the user had entered the wrong PIN code, the Ellisys software would have advised of this error and indicated the correct PIN code to be entered. Easy.



SSP Debug Mode

SSP Debug Mode is a special mode used by SSP-enabled devices where a well-known Private/Public key pair is used instead of the normal Private/Public key pair. If at least one of the two devices is in SSP Debug Mode, the link key will be deduced automatically by the BEX400 analyzer, which recognizes the well-known Public key transmitted over-the-air. Again, only one of the two devices must be in Debug Mode, not both. If one of the two devices is in Debug Mode, then the analyzer will know its SSP Private Key and will thus be capable of computing the link key with the same algorithms used by each device.

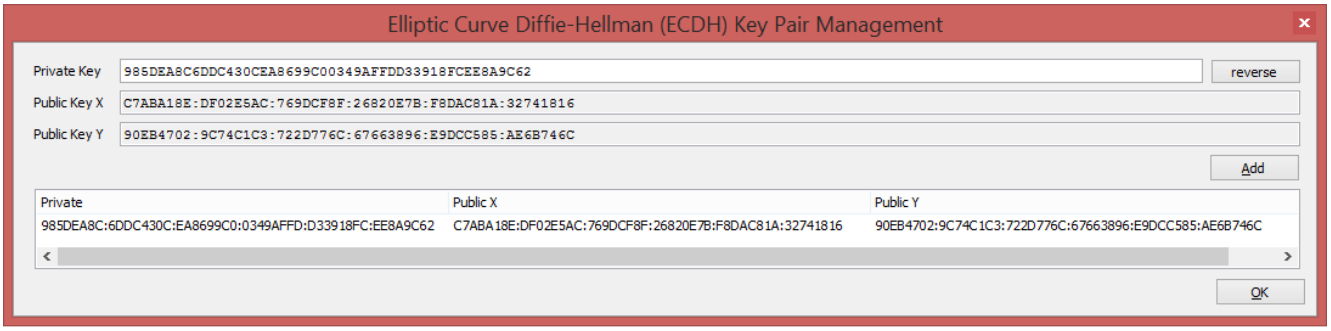
Note that some manufacturers have made the choice to not accept an SSP pairing with other devices that are in Debug Mode when their device is not in Debug Mode. This is a non-standard behavior which unfortunately diminishes the usefulness of Debug Mode, since in such cases both devices have to be put in Debug Mode.

SSP Private Key Injection

As explained above, SSP Debug Mode is just a way of using a known SSP Private Key. Even when not using SSP Debug Mode, if the Private Key is known then it can simply be provided to the protocol analyzer software by the user. By knowing the Private Key, the analyzer software is capable of automatically determining the link key whenever a pairing involving the known Private Key is detected. This provides great usability, since it avoids having to ever enter the link key manually.

This approach also provides an added level of security to the SSP Debug Mode, as only the analyzer configured with the specific SSP Private Key is capable of determining the link key; other sniffers in the area will not be able to decrypt the traffic.

This feature (entry of the Private Key) is provided by the Ellisys analyzer software using the "Manage SSP Key" feature, located in the Security pane, illustrated below.



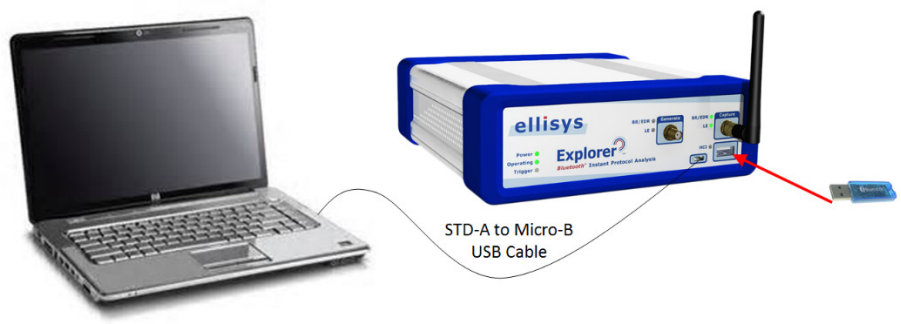
HCI Capture

While the link key is not transmitted over the air, the good news is that it is transmitted between the radio chip and the host, over the host controller interface (HCI) and is accessible over this medium. Even better, the Ellisys analyzer can capture HCI traffic concurrently with the air traffic, hence, the analyzer has access to the link key this way.

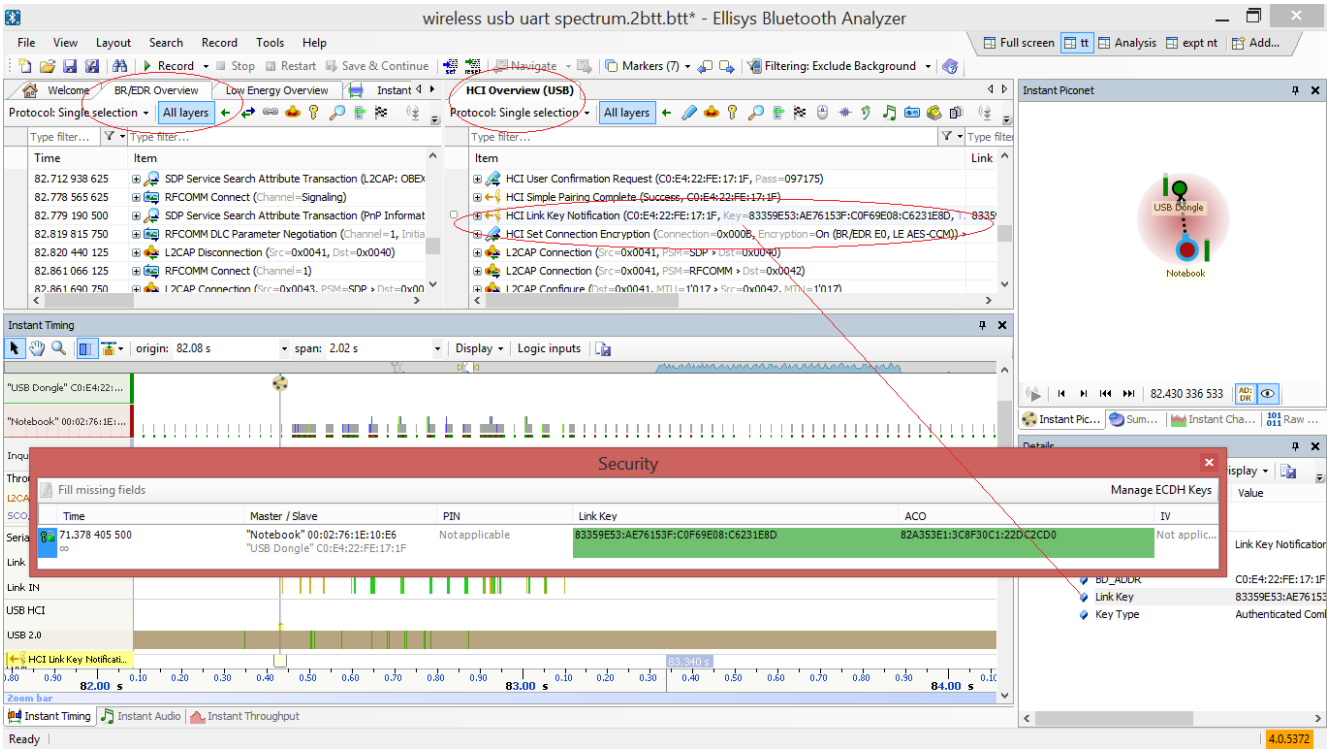
Access to HCI is easily available on prototypes or developer kits (SDKs) used during development stages, but such access will likely not be available on most off-the-shelf devices. One exception is USB HCI, such as that used by Bluetooth dongles. For example, using a PC with an attached USB dongle is an easy way to get access to the HCI. This is done by attaching the dongle to the BEX400 analyzer at the STD-A port (under the antenna), then connecting a USB Micro-B to USB STD-A cable from the adjacent Micro-AB receptacle on the analyzer to the PC. The analyzer software will capture the air traffic concurrently with the USB HCI traffic, and will extract the link key automatically in order to decrypt the air traffic.

Note that HCI access is required on one of the two devices only, not both, as the link key is the same for both devices.

Below is a typical setup involving USB HCI capture from a Bluetooth dongle. The analyzer captures both the air traffic as well as the HCI traffic, concurrently. Note that in normal circumstances, the dongle is extended away from the antenna using a 6-inch USB STD-A to STD-A extender cable (as otherwise the dongle may present an overly strong signal when situated very close to the antenna).

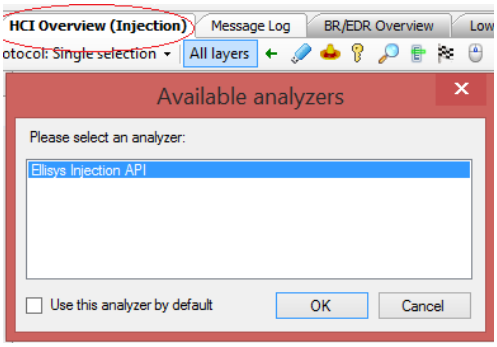


In the picture below, the BEX400 analyzer has captured USB HCI concurrently with BR/EDR traffic, with the link key being automatically extracted from the HCI capture and used to decrypt the BR/EDR traffic. A similar approach can be used to capture other forms of HCI, such as UART or SPI, using a special cable at the rear of the BEX400 analyzer.



Ellisys HCI Injection API

The Bluetooth Explorer 400 software application supports an HCI Injection API, selectable in the Available Analyzers dialog in the Record menu. The Injection API uses UDP (User Datagram Protocol) to push HCI traffic to the Bluetooth Explorer 400 software, which is displayed live in the HCI Overview (Injection), *just as if this traffic was being captured* on one of the standard HCI capture ports on the analyzer.



Link keys captured with this method are stored by the analyzer software and associated with the device connection, so that any subsequent captures of air traffic between the devices is automatically decrypted.

The User Manual for the Bluetooth Explorer 400 provides a link to download explanatory documents and sample code.

Ellisys Remote Control API

The Ellisys Remote Control API can be used to programmatically inject link keys (obtained from any source) into the Ellisys BEX400 software application. For instance, it is possible to build a Remote Control API client that monitors Windows Registries for changes and pushes new link keys into the analysis application.

The link provided below contains an introductory guide, the plug-in DLL, and sample code written in C#, using Microsoft Visual Studio 2010. Compatibility is Visual Studio 2005 or higher.

http://www.ellisys.com/better_analysis/bex400a_remote_api.zip

Vendor-specific methods

The BEX400 analyzer captures all Bluetooth traffic in the vicinity, even encrypted traffic. The analyzer's software allows the user to manually input a link key, either during capture, or even on a saved capture, in order to decrypt the traffic. When standard link key access methods (such as HCI capture or injected HCI) are not applicable, it may still be possible to extract the link key from devices with vendor-specific methods, and then enter the link key into the analyzer's Security pane for decryption.

In these cases, it is required to have access to special features of the radio chip, the operating system, the SDK, or the host stack. For example, access to a Windows registry, where a link key may be stored, or to developer modes available on some mobile phones.

By no means is the list below complete - it is a work in progress. Ellisys encourages readers to submit suggestions on methods to find link keys across various devices, stacks, etc. for inclusion in this document. See information at the end of this document on how to contact Ellisys. Ellisys also strongly suggests checking with your device manufacturer before attempting any changes to your system software.

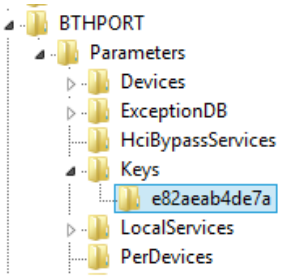
For developers, most Bluetooth stack providers or chip developers will include an API or an SDK that provides access to the system registers that likely will include an option to read Bluetooth link keys. Contact your supplier as needed for details.

Windows Bluetooth stack

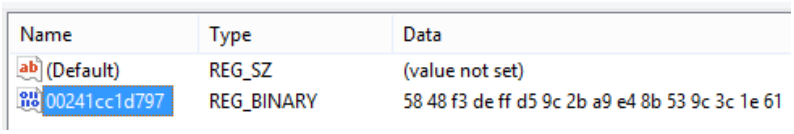
The link key for Windows-based systems (using the Windows stack) is often accessible from the Windows registry. In many cases, this is located in the BTHPORT folder as shown below, associated with the applicable Bluetooth address (i.e., the Bluetooth device to which the system is connected). Note that this folder may require administrative permissions for access, and that such permissions may be changed in order to gain access (right-click the folder for this option).

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\BTHPORT\Parameters\Keys`

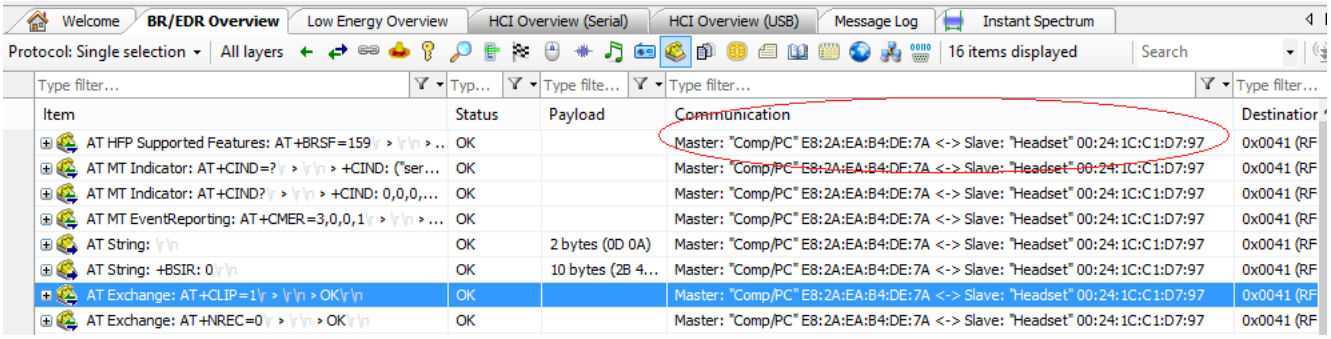
In our case here (at right), the computer (a WIN8 PC) shows a registry directory in `BTHPORT/Parameters` labeled "e82aeab4de7a" which is the BD ADDR of the PC.



Below, we see the contents of this directory, with "00241cc1d797" shown as the BD ADDR of the connected device, the headset in this case. The data associated with the headset is the link key. We simply copy this link key and paste it into the Ellisys analyzer application (in the Security pane), and as shown below, the traffic between these devices is immediately decrypted.



Below, we see the analyzer has now decrypted the traffic (filtered to show only the AT Protocol). Note the Communication column, which lists the two communicating devices and their BD ADDRs.



Many Windows systems will use a third-party Bluetooth stack instead of the Microsoft stack, such as Toshiba, Widcomm, etc. Registry access to these third-party stacks may not be quite as obvious, so one approach is to force the system to revert from the third-party stack to the Windows stack. The user should set a system restore point, do an uninstall of the third-party stack, and then reboot. The system should then default to the Microsoft stack.

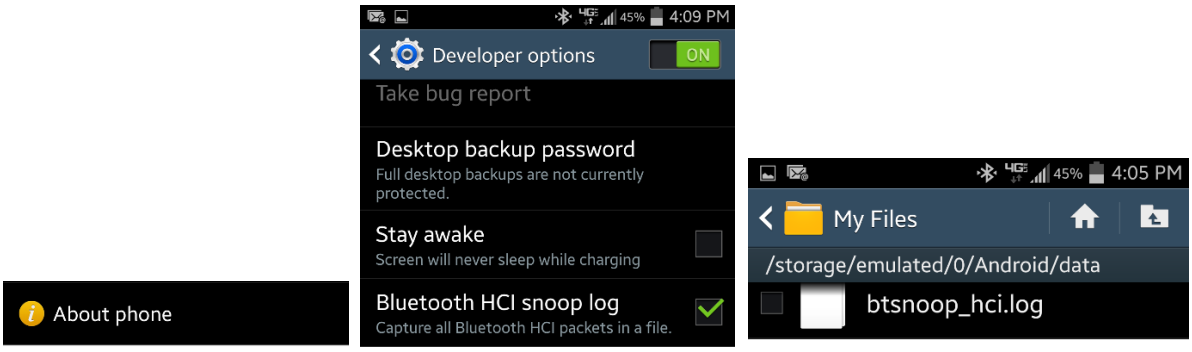
In some cases, browsing online forums may be useful. The URL below is one example:
http://superuser.com/questions/229930/finding-bluetooth-link-key-in-win7-to-double-pair-a-device-on-dualboot-computer

Android stack

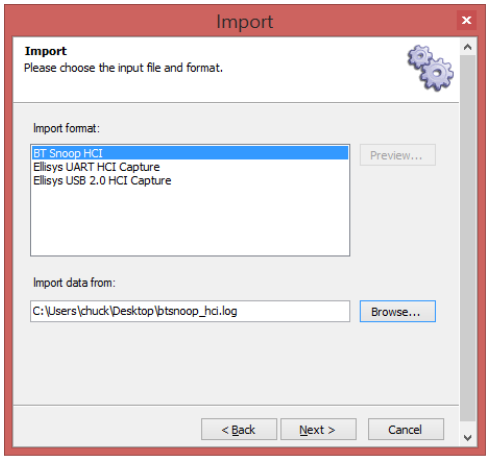
Later versions of the Android operating system provide a Developer Mode, which can be enabled through the *Settings/About Phone* menu by depressing the *Build Number* selection several times repeatedly (typically seven times). Consult your phone's User Manual to be sure you have Developer Mode available and for specific instructions.

The Developer Mode includes a variety of features including an option to enable a log file containing HCI traffic. This file is in BT Snoop format. This file, which will contain link keys from pairing events, can be easily imported to the BEX400 analyzer software. The analyzer software will store these link keys and will associate these with the devices using them, so subsequent air captures of the connected devices will be automatically decrypted.

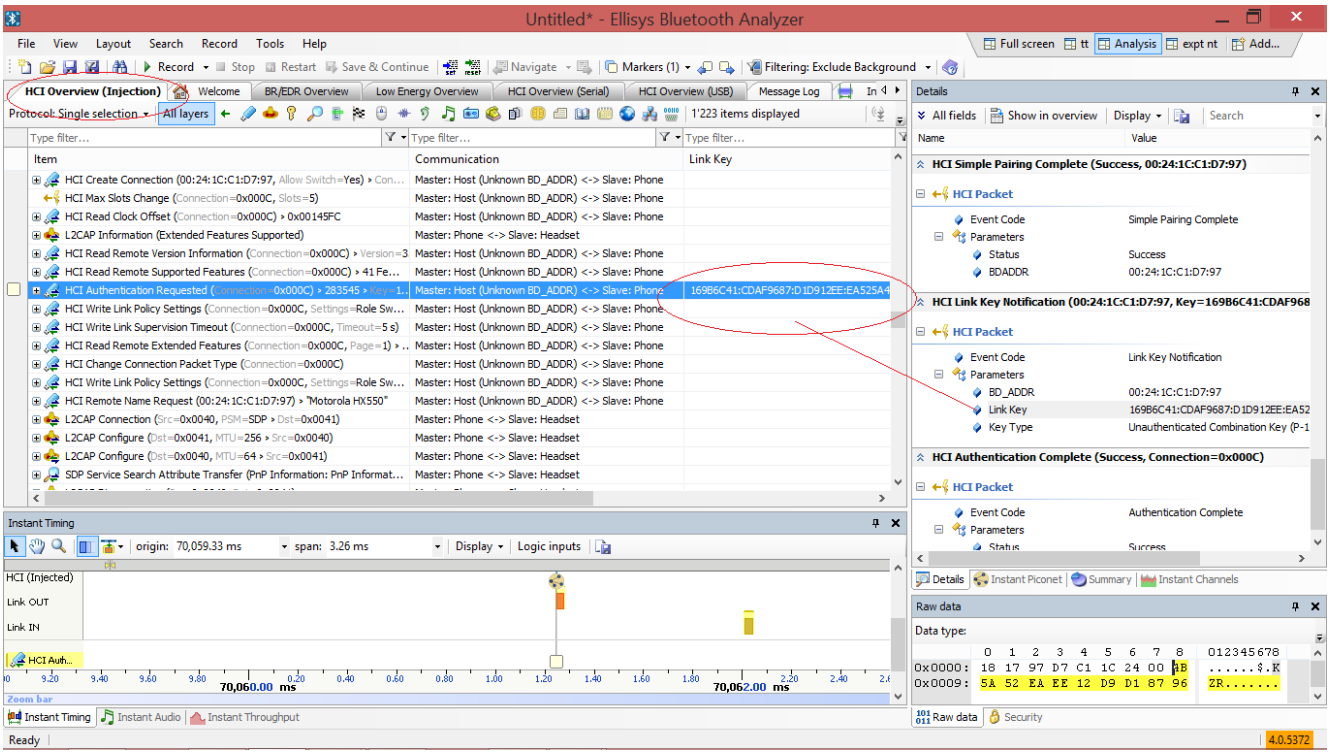
The pictures below depict an abbreviated sequence to enable the Developer Mode, and the resulting BT Snoop HCI log file. The file can be accessed over the phone's USB connection or it can be shared in a variety of ways, such as by E-Mail.



The screenshot below shows the BEX400 Import menu, with BT Snoop HCI selected for import. The BT Snoop file created by the phone is referenced as the location from which to do the import.



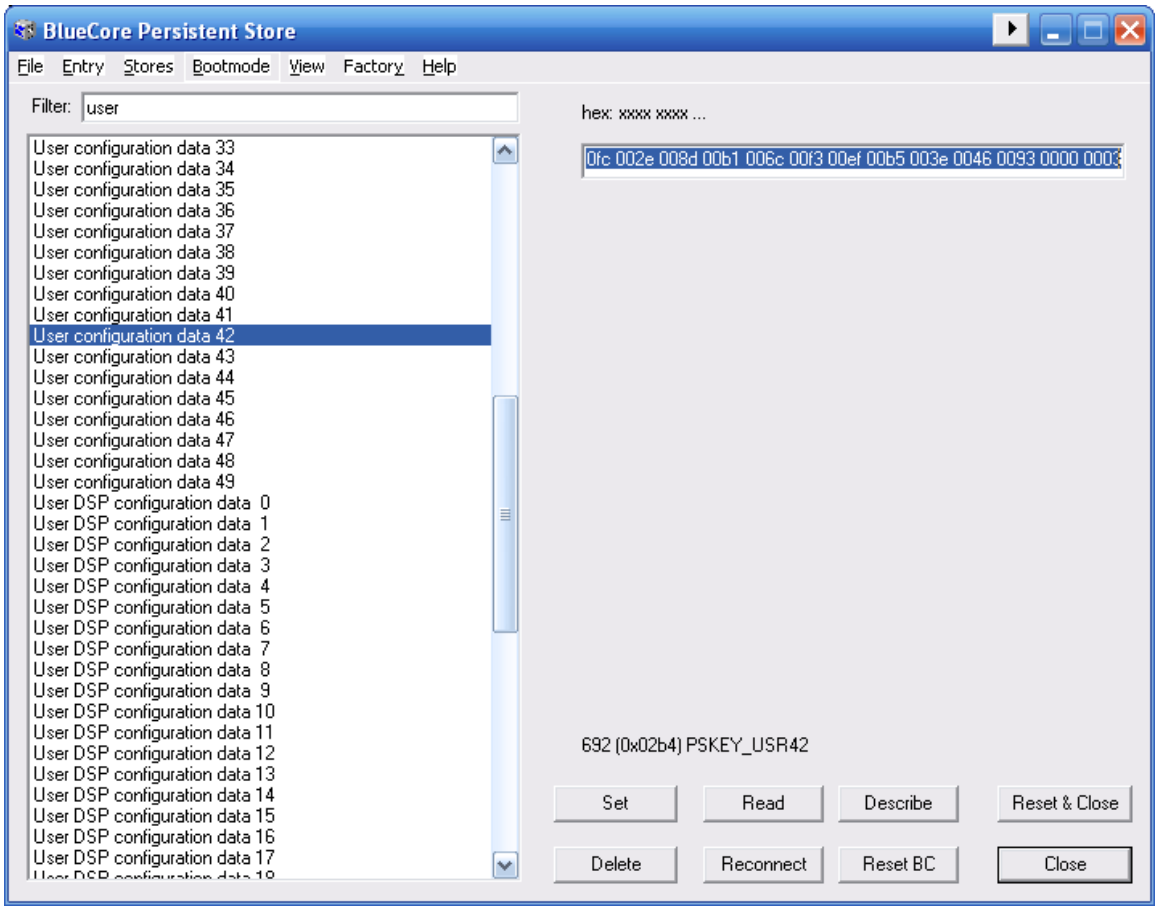
Below, the resulting capture after the phone's BT Snoop log has been imported by the BEX400 application. This shows the link key established between the Phone and the Headset. This key will be stored by the analyzer software and used to decrypt traffic on captures of subsequent connections between the phone and headset.



For information on the Android Developer SDK, consult the link below:
http://developer.android.com/sdk/index.html

CSR Persistent Store

For developers working with CSR devices and tools, there is good news: the Ellisys BEX400 Explorer application natively supports the format of CSR's PSTool utility. The keys are stored in configuration data 42 to 49:



The hex data is directly supported by the Ellisys software and can be provided in the Link Key field of the Security view as is, without further conversion. The Ellisys software will recognize CSR format and will convert it automatically.

Nokia Mobile Phones

Some Nokia models can be put in Debug Mode by typing *#2873#

Linux/BlueZ Bluetooth stack

BlueZ is official Linux Bluetooth protocol stack. This stack is also used on many Android platforms (prior to Android 4.2). See www.bluez.org for information on utilities that may be useful for accessing link keys.

The link below has information on the BlueZ Utils application, which features an HCI tool that may also be useful.
https://www.linux.com/news/hardware/peripherals/44623-working-with-bluetooth-connecting-to-all-those-cool-devices

`/var/lib/Bluetooth/[BD_ADDR]/linkkeys`

Widcomm Bluetooth stack

Note that Widcomm was acquired by Broadcom. Widcomm was the first Bluetooth stack used by Windows. Similar to the Windows stack section above, try the location in the registry below for Widcomm.

```
HKEY_CURRENT_USER\Software\Widcomm\BTConfig\LinkKeys
```

Bluetooth Smart (Low Energy)

Security features for Bluetooth Smart (per specification version 4.0) differ from those used by BR/EDR. A Bluetooth Smart pairing procedure results in the generation of a long term key (LTK), which is derived using a process that varies significantly from the process used by BR/EDR that results in generation of a link key. Bluetooth Smart devices are architecturally simpler than BR/EDR devices as they are intended for simpler tasks, so there is generally less processing and storage capabilities on these devices, and in fact a simpler and less secure approach to privacy.

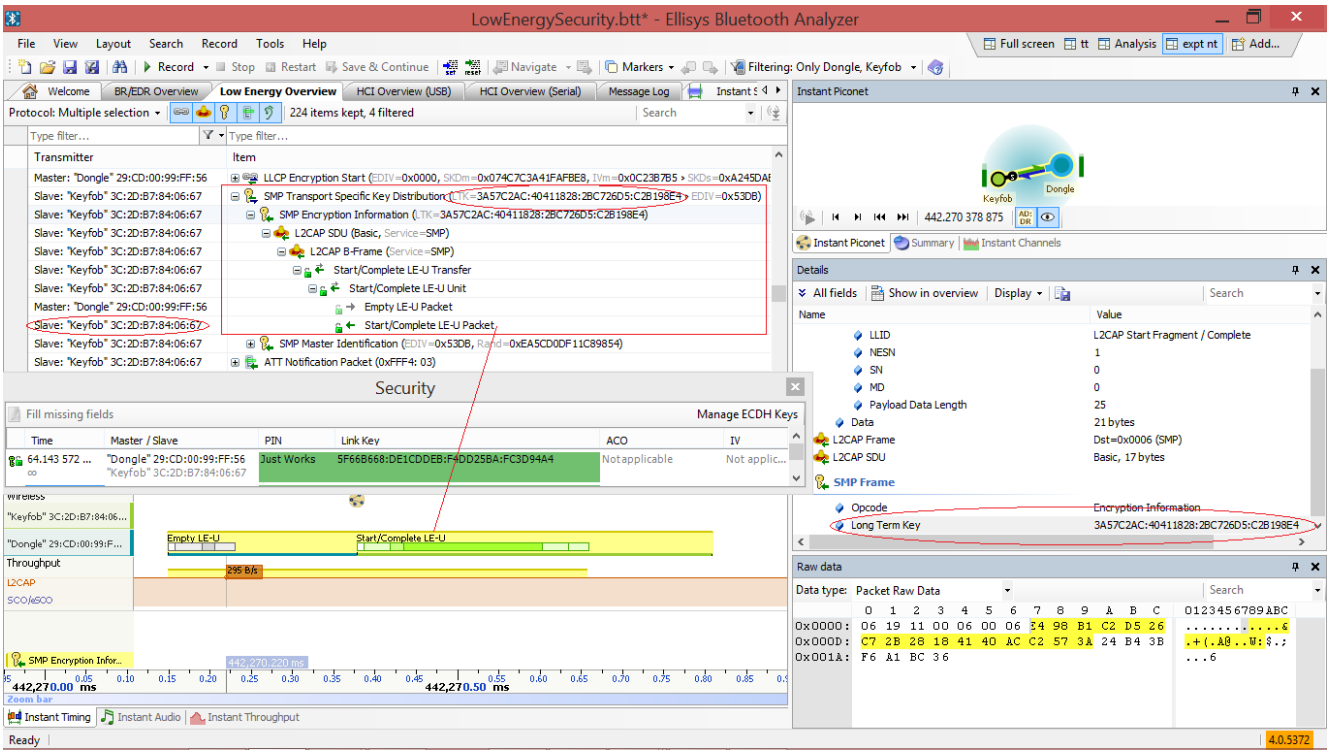
See the section below titled Secure Connections for Bluetooth Smart for information on future changes involving security for Bluetooth Smart.

With Bluetooth Smart, one device determines the LTK, then sends this to another device using temporary encryption based on a short term key (STK). One benefit to this simpler approach is faster connections and reconnections, a primary design goal of Bluetooth Smart and a major difference to the slower-connecting BR/EDR. A drawback is a lack of eavesdropping protection.

So, with Bluetooth Smart, we're using "key transport" (the LTK is transmitted to the other device) instead of "key agreement" (the link key is determined symmetrically by both devices).

Importantly for BEX400 users, the hunt for a key is not really needed, as the analyzer is able to crack the security for Bluetooth Smart. Future evolutions of Bluetooth Smart will make this much more difficult, but the BEX400 architecture will accommodate these changes easily (see Secure Connections for Bluetooth Smart, below).

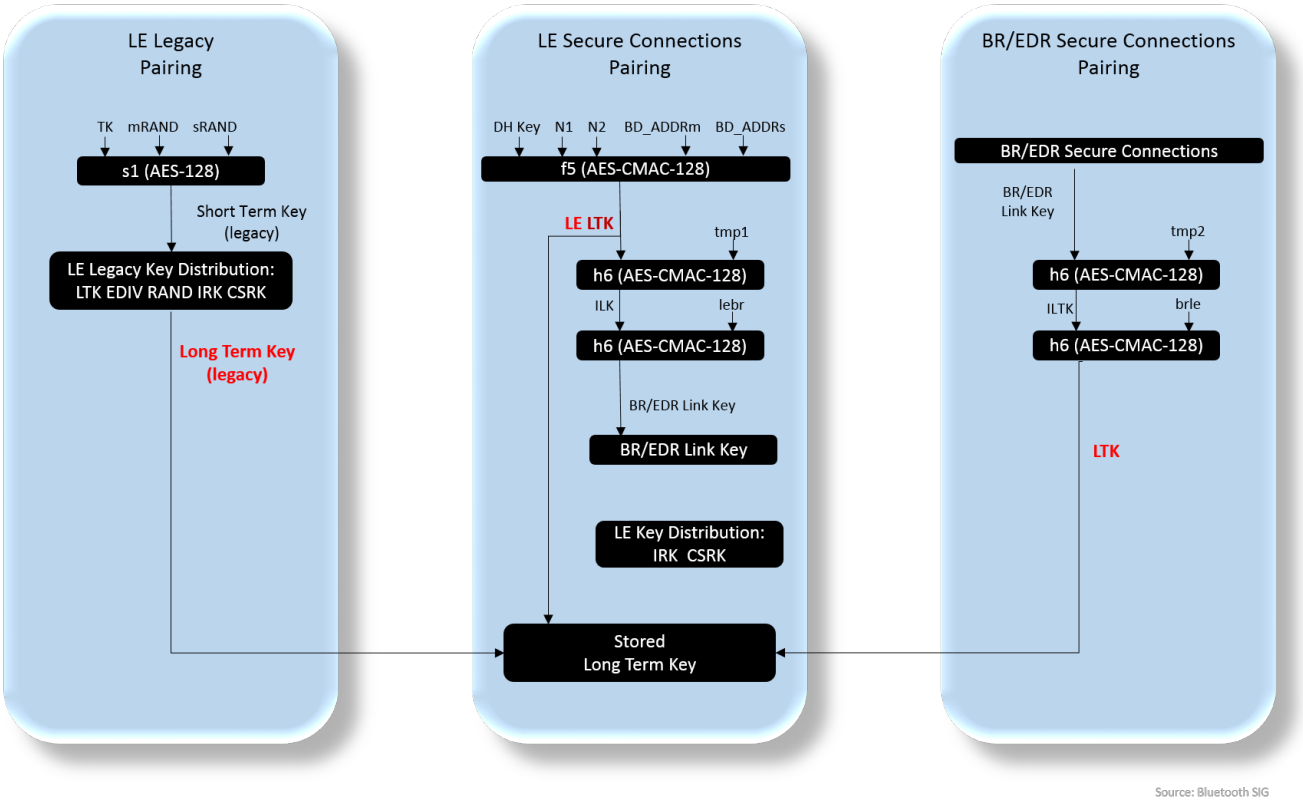
The screenshot below depicts the LTK being sent from a key fob to a dongle. Note that STK-enabled encryption is already in place at the time this packet is sent.



Secure Connections for Bluetooth Smart

An upcoming revision to the Bluetooth specification (v4.2) will add various enhancements, including the addition of LE Secure Connections that will greatly improve security for Bluetooth Smart devices. This addition updates the pairing procedures to utilize the P-256 elliptic curve and FIPS-approved algorithms (AES-CMAC and P-256 elliptic curve) in a way that is similar to BR/EDR Secure Connection SSP. This change also includes provisions for a shared secret key. LE legacy pairing (described in the section above) will still be available. Note that for BR/EDR, Secure Connections was added with the 4.1 version of the specification and consisted mainly of a change in the encryption method; the pairing method still being SSP but with longer Private/Public Keys (256 bits instead of 192 bits).

The diagram below describes the expected architecture for LE Secure Connections as well as a comparison to LE legacy pairing and BR/EDR Secure Connections.



As depicted in the diagram above, LE Secure Connections will also allow a key derived over one transport (e.g., BR/EDR) to be used for another transport (e.g., Bluetooth Smart). So, if you have two devices that both support BR/EDR and Bluetooth Smart, and both support Secure Connections on both of these transports, you just have to pair once. In other words, an LTK generated during the LE pairing process may be converted to a BR/EDR link key for use on the BR/EDR transport, and conversely, a BR/EDR link key generated during the BR/EDR SSP pairing process can be converted to an LTK for use on the LE transport.

Feedback needed

Do you know of other ways to access link keys, or do you have feedback on the implementation of the suggestions in this document? Please share your feedback with us at expert@ellisys.com and we will update this document accordingly (we'll attribute your input in the updated document if you wish). This will be a great help to the Bluetooth community!

Other interesting readings

- EEN_BT03 - Your First Wide-Band Capture
- EEN_BT06 - Bluetooth Security - Truths and Fictions
- EEN_BT07 - Secure Simple Pairing Explained
- More Ellisys Expert Notes available at: http://www.ellisys.com/technology/expert_notes.php

Rev. A. Updated 2014-10-01