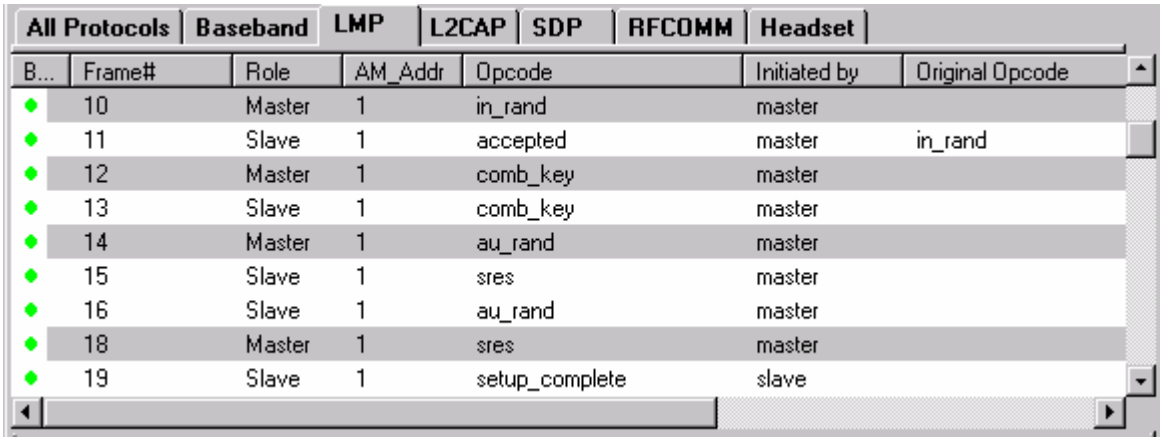
Decryption in FTS4BT

How Encryption Works in Bluetooth:

Bluetooth devices on an encrypted link share a common link key in order to exchange encrypted data. This link key is derived from a shared PIN code, the master's Bluetooth slot clock, the master's BD_ADDR and a random number that is passed between the two devices. The sequence of events, or pairing process, is shown in the FTS4BT Frame Display below.



All Protocols | Baseband | LMP | L2CAP | SDP | RFCOMM | Headset

B...	Frame#	Role	AM_Addr	Opcode	Initiated by	Original Opcode
●	10	Master	1	in_rand	master	
●	11	Slave	1	accepted	master	in_rand
●	12	Master	1	comb_key	master	
●	13	Slave	1	comb_key	master	
●	14	Master	1	au_rand	master	
●	15	Slave	1	sres	master	
●	16	Slave	1	au_rand	master	
●	18	Master	1	sres	master	
●	19	Slave	1	setup_complete	slave	

Frame 10 is the LMP_in_rand where the random number generated by the master is passed to the slave. The slave acknowledges that it has accepted the number in frame 11.

In frames 12 and 13, the combination keys are passed between master and slave. In frame 14, the master sends its LMP_au_rand. This is the random number that has been encrypted using the link key that master has calculated. The slave then responds with frame 15, an LMP_sres confirming that it was able to compute the same number. That process is repeated in the other direction (slave to master) in frames 16 and 18. Then both master and slave send setup_complete messages and they're ready to send encrypted data.

How FTS4BT Decrypts Data:

In order for FTS to decrypt an encrypted Bluetooth conversation, FTS must have the same link key being used by the devices being sniffed. Since this link key is never sent over the air, FTS must have all of the same information the devices being sniffed have so that it can calculate the same link key that each of the two devices does. To decrypt successfully, FTS must capture:

- The LMP_in_rand
- Both LMP_comb_keys
- Both LMP_au_rand/LMP_sres pairs.

If any of these are missed, decryption will fail. If you capture encrypted data and find that everything captured after the LMP_start_encryption_request is in error, look back at the LMP frames previous to that and you'll probably find one or more of these missing.