Interactive decryption of DECT phone calls

Patrick McHardy
kaber@trash.net

Andreas Schuler
dect@badterrorist.com

Erik Tews
TU Darmstadt
Hochschulstrasse 10
64289 Darmstadt, Germany
e_tews@cdc.informatik.tu-darmstadt.de

ABSTRACT

DECT[3] is a widely deployed standard mostly used for short range wireless phones. So far, no method has been published which is able to recover the audio signal in a call that is encrypted and lasts only for a few minutes.

In this paper, we present a method that recovers the audio signal sent from the phone to its base station in an encrypted call. To do so, we use a replay-attack against the phone to recover the key streams, which were used to encrypt the call. The method is applicable to short calls too, where not enough keystreams are available to recover the ciphers key using a key recovery attack[8] on DSC. The method is fast and practical and can be executed at very low cost.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; D.4.6 [**Security and Protection**]: Metrics, Cryptographic controls; E.3 [**DATA ENCRYPTION**]: Code breaking

Keywords

DECT, DSC, Replay attacks, Keystream recovery

1. INTRODUCTION

With more than 800 million devices sold worldwide[1], Digital Enhanced Cordless Telecommunications (DECT)[3] is one of the most common standards for short range cordless telephones. Besides for phones, DECT is used for many other applications like wireless payment systems, traffic control, access control and room monitoring. DECT networks usually consists of a single or multiple base stations named DECT Fixed Part (FP) in the standard and phones named DECT Portable Part (PP) linked with these base stations. For most residential use cases, only a single base station is operated with a small number of phones. A single base station can cover a single house or up to a few hundred meters in the open field. European systems operate at 1880 to 1900 MHz and have a maximum transmit power of 250 mW, while the North American version operates at 1920 to 1930 MHz and just uses a maximum transmit power of 100 mW. DECT systems can scale to many base stations and phones, and also support roaming as GSM does.

[1]http://www.etsi.org/WebSite/NewsandEvents/201004_CATIQ.aspx

To protect sensitive data transmitted over DECT, the standard provides authentication (DECT Standard Authentication Algorithm, DSAA) as well as confidentiality (DECT Standard Cipher, DSC). Both algorithms were specially designed for DECT and are only available to DECT device manufactures who sign a non-disclosure agreement. DSAA is responsible for the initial pairing of a new phone with its base station. It is also used for authentication of phones and base stations and for key derivation to generate a session key (Cipher Key (CK)) for DSC from the User Authentication Key (UAK).

DSC is used for encryption. It is a stream cipher, which takes an Initialization Vector (IV) and a session key (cipher key, CK) to generate a key stream (cipher stream, CS) from it. Besides the actual payload (voice data), parts of the the control traffic (C-channel traffic) are also encrypted, which for example contain the dialed number.

1.1 Previous attacks on DECT

First attacks on DECT were made public at the end of 2008 and the first paper *Attacks on the DECT authentication mechanisms*[5] by Lucks, Schuler, Tews, Weinmann and Wenzel was published at CT-RSA 2009. They reverse engineered DSAA and implemented a low-cost DECT sniffer, which uses a DECT PCMCIA card. Many combinations of phones and base stations on the market did not use DSC at all, always sending their communications in clear. This can be easily be eavesdropped on using their passive DECT sniffer.

All other devices examined did not request authentication of the base station, but the base station did request authentication of the phones and enabled encryption. To attack these phones, they implemented a rogue base station impersonating the original base station. When a phone was connected to their base station, the base station did not request encryption, so the phone did send it's data in clear. Because no phone requested authentication of the base station or enforced encryption, none of the phones noticed that the base station was not in possession of the correct UAK. However, this method can only be used to intercept calls originating from the phone, but not from the original base station. These methods were later improved by [7] and [6].

To counter these attacks, new phones were developed, which always require encryption. If a base station does not enable encryption when a new call is established, the call will be automatically dropped. Such phones are immune against these attacks.

In 2009, the DECT Standard Cipher was reverse engineered by Nohl, Tews, and Weinmann[8, 9]. They also presented a key recovery attack on the DSC, which recovers the key in a few minutes to hours using a fast computer. However, this attack needs about 2^{15} key streams produced with the same key and different Initialization Vectors, which must be available to the attacker. They proposed two methods to recover that many key streams:

- If only silence is transmitted from the base station to the phone (for example a voice mail system is being called and the caller leaves a long message on the system), then the plaintext of the payload will be known and these key streams can be extracted in about 10-11 minutes. Also this method does not recover the first 40 bits of these key streams, which are only used to encrypt control traffic (see Figure 3). This renders the attack less efficient on these key streams.

- Some phones show a call duration counter on their display, which is implemented on the base station. This counter is updated once per second using C-channel messages. Because the plaintext of those messages is mostly constant, this can be used to recover about 5 key streams per second. This makes the recovery of 2^{15} key streams in about two hours possible.

So far, no method has been published to decrypt a short DECT call, without iterating over all 2^{64} possible keys for the DSC, which is made on a phone, where encryption and authentication are enforced, and no other implementation flaws are present.

1.2 Our contribution

In this paper, we propose a practical and efficient method to recover the audio data sent from a phone to the base station during an encrypted DECT call. We use a replay attack against the phone to recover the keystreams which were used to encrypt the call. These keystreams can be used to decrypt the call. Our method requires nearly no computational power and can recover the data from an x seconds phone call in about $2.8 * x$ seconds. This value is based on experiments, and depends on the implementation of the attack and the phone being attacked.

Because some phones have bad echo cancellation, this may also include the audio sent from the base station to the phone, which is played by the phone speaker and recorded by the phone microphone again.

2. DECT PROTOCOL

To understand our attack, we give a brief overview of the DECT protocol. The DECT standard allows a wide variety of features, which can be implemented, however just a small subset of them is usually implemented in a standard consumers phone. A DECT network consists of a single or multiple base stations, named **DECT Fixed Part (FP)**, and a number of phones, named **DECT Portable Part (PP)**.

2.1 Radio protocol

The DECT protocol can be divided into 5 layers: Physical Layer, MAC Layer, Data Link Control Layer, Network Layer and the actual speech and audio coding. Encryption takes place in the MAC Layer, however encryption is negotiated at the Network Layer.

To allow multiple devices to transmit on the same frequency, DECT uses TDMA (Time Division Multiple Access), a period of 10 ms, as known as **frame**, is divided into 24 **time slots**. In every frame, one of the first 12 time slots is used for transmissions from the base station to the phone (FP $\rightarrow$ PP), and the time slot 12 time slots later is used for transmissions from the phone to the base station (PP $\rightarrow$ FP). In every time slot, 480 bits could be transmitted. A single DECT **full slot packet (P32)** has a total length of 32+64+320+4+4 = 424 bits and is divided into an S-, A-, B-, X- and an optional Z-Field. The remaining 56 or 60 bits are a guard period between the time slots.

- The S-field is only a static preamble of a packet, which is used by the receiver to synchronize on the signal.

- The A-field contains the packet header and can transport control traffic. Control traffic is separated into different logical channels, namely the C, M, N, P, and Q channels. A tag in the A-field header determines which channel is embedded in the packet, and for example C-channel messages are usually split over several packets, because they usually don't fit in a single A-field.

- The B-field contains the actual payload, for example the voice data, when a phone call is made over DECT.

- The X and Z fields are checksums to detect transmission problems.

An overview of the particular fields is given in Figure 1 and Figure 3. Base stations usually broadcast a beacon once per period. Phones synchronize their timer on this beacon.

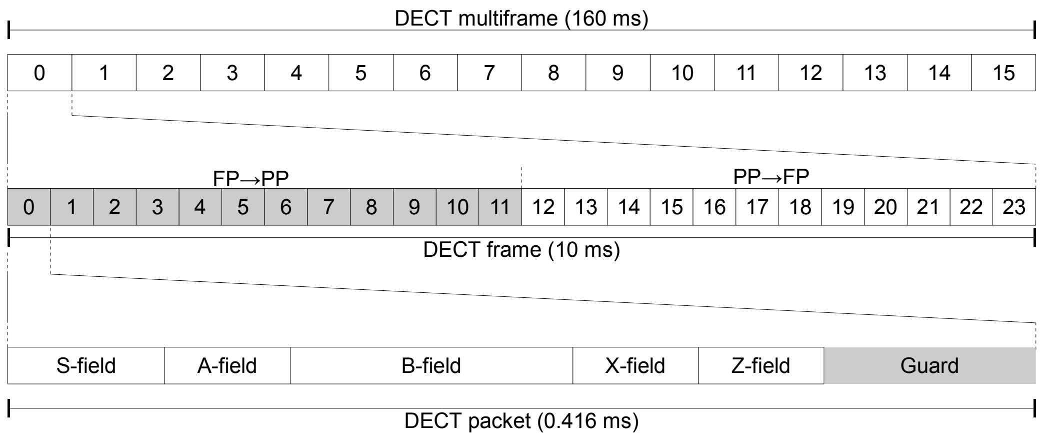
Most of the time, a phone only passively listens to the broadcasts of a base station. When there is traffic, for example a call is active or the base station needs to update the display of the phone, the phones establish a connection with the base station. A base station can request a new connection from the phone by broadcasting an `LCE-PAGE-REQUEST` message.

The DECT standard makes heavy use of timers, to specify how long certain procedures may take. Only one timer, namely the `LCE.01` timer is of importance for the attack. When a connection is not needed anymore by any upper protocol layers, the `LCE.01` timer is started, which runs for 5 seconds. If there is no more activity on the connection within these 5 seconds, the connection is terminated.

An existing connection between a phone and a base station does not necessarily mean that a call is active. Instead, a base station might, for example, establish a connection just to update a phone display state to indicate that a new voicemail has arrived or a text has been received by the base station. All phones we examined so far send packets with all bits set to 1 (**0xff** in hex) in the B-field when there is no audio data present in the connection.

DECT supports sending shorter packets (the standards also specifies the P00 format), if no payload present, but this is an optional feature, the base station and the phone

Figure 1: DECT TDMA structure



must support. Even if the phone would be able to send shorter packets, the base station could indicate that it is not capable of processing such packets and the phone would need to switch to P32 packets with B-fields.

2.2 Authentication and key derivation

Usually, phone and base station share a 128 bit symmetric key named UAK. For authentication, two different procedures are defined:

2.2.1 *Authentication of a phone by base station*

First, a base station can request authentication from a phone. To do so, the base station chooses two random numbers RS and RAND_F and sends them in an `AUTHENTICATION-REQUEST` message to a phone. Now the phone computes a response to this challenge using the DSAA algorithms A11 and A12 with the UAK and these two random numbers as input. The result RES1 is transmitted in an `AUTHENTICATION-RESPONSE` message to the base station, which performs the same computations and compares the received RES1 with the locally computed result XRES1. In addition to that a 64 bit cipher key CK is also generated by A12, which can be used for encryption later on. See Figure 2 for details.

2.2.2 *Authentication of a base station by a phone*

A phone can also request authentication from a base station. To do so, it picks just a single 64 bit random number RAND_P and sends it to the base station. The base station picks another 64 bit random number RS, and computes a response RES2 to the challenge sent by the phone using the DSAA algorithms A21 and A22 with UAK, RAND_P and RS as input. RES2 and RS are transmitted to the phone, which compares it to the locally computed result. No cipher key is generated and the procedure does not affect the generated cipher key from the previous paragraph.

None of the devices examined [5, 6] used this feature, so it has no importance for the rest of this paper. We will discuss this in Section 6.2 as a countermeasure.

2.3 Encryption

Even when there is no encryption in use, a base station continuously broadcasts a multiframe number embedded in a Q-channel message. Every packet sent and received by the base station has a frame number, which is a function depending on the last broadcasted multiframe number and the time elapsed since this event. Every packet in the same frame shares the same frame number.

For encryption, DECT defines a stream cipher, the DECT Standard Cipher (DSC). DSC takes a 64 bit initialization vector IV and a 64 bit cipher key (CK) as input and generates a key stream (cipher stream, CS) of arbitrary length from it. The IV used is the frame number zero-padded to 64 bit length. Usually, 720 bits of key stream (cs0 . . . cs719) are generated for each IV. The first 360 bits (cs0 . . . cs359) are used to encrypt the packet sent from the base station to the phone (FP → PP). The remaining 360 bits (cs360 . . . cs719) are used to encrypt the packet sent from the phone to the base station (PP → FP) in the same frame. To encrypt a packet, the first 40 bits of key stream (cs0 . . . cs39) are used to encrypt C-channel messages in the A-field, if the A-field contains C-channel traffic. If no C-channel traffic is present in the packet, the first 40 bits (cs0 . . . cs39) are silently discarded. The remaining 320 bits (cs40 . . . cs359) are XORed with the B-field to encrypt the payload present in the B-field. An overview of the process is given in Figure 3.

To enable encryption, the base station sends a `CIPHER-REQUEST` message to the phone, indicating that it requests ciphering. The phone either confirms that using a mac layer message, or sends a `CIPHER-REJECT` message back to the base

Figure 2: Authentication of a DECT PP

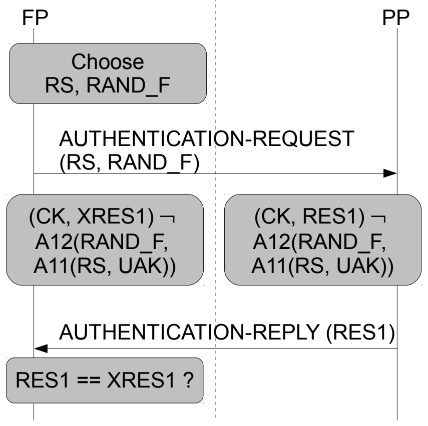
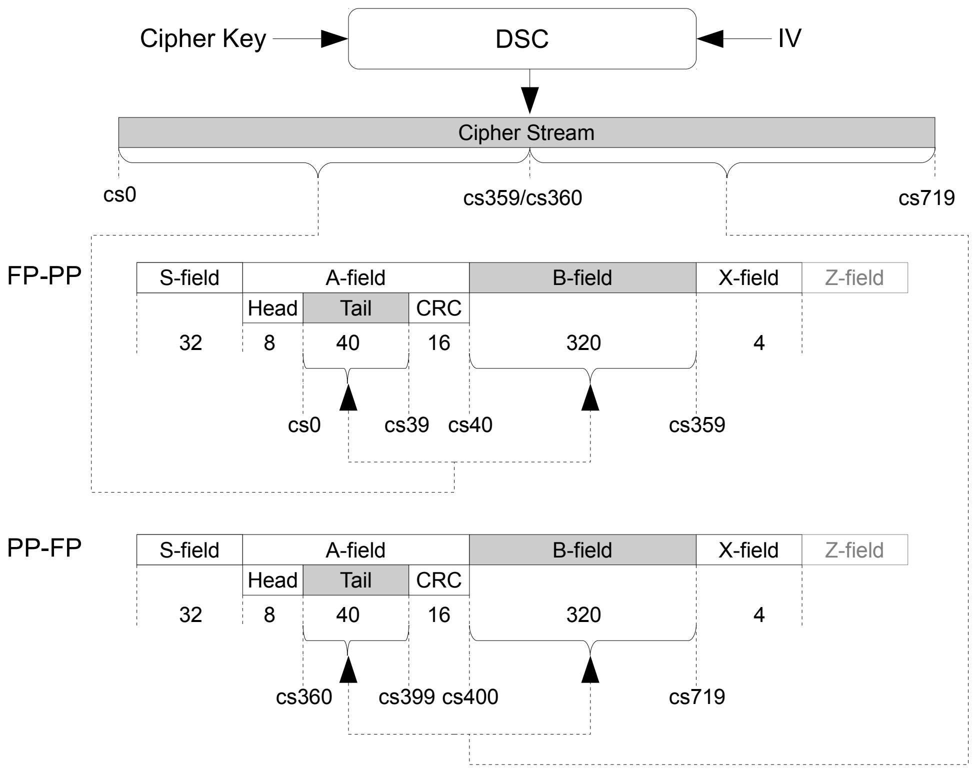

FP
PP
Choose RS, RAND_F
AUTHENTICATION-REQUEST (RS, RAND_F)
(CK, XRES1) ← A12(RAND_F, A11(RS, UAK))
(CK, RES1) ← A12(RAND_F, A11(RS, UAK))
AUTHENTICATION-REPLY (RES1)
RES1 == XRES1 ?

Figure 3: Key streams used in DECT (P32 full packet)



station, indicating that it is not capable of enabling ciphering. After ciphering has been confirmed by the phone, the base station sends a mac layer message back to the phone to confirm. The next packet send or received will be encrypted.

3. EXECUTION OF THE ATTACK

The attack is executed in two phases: In the first phase outlined in Section 3.1, the call is passively recorded. If the call would be unencrypted, the attack would be finished after this step. For encrypted calls, a second phase outlined in Section 3.2 is executed, which decrypts the call.

3.1 Recording the encrypted call

In the first phase, a call is recorded in its encrypted form. To do so, the attacker needs to be in communication range of the victim's phone and base station. If he is able to receive signals from multiple base stations, he needs to pick the right one, or just record all calls originating from any base station. To pick the right one, he might call the victim's land line, and see which base stations signal an incoming call to their phones. It is not required that the victim takes the call, only that the base station signals to at least one of the phones that there is an incoming call. Repeating this procedure should narrow down the set of possible base stations to a single one. Alternatively the attacker could use a directional antenna pointing at the victim's base station and pick the one with the strongest signal.

The attacker now records all packets send between the base station and the phones. We decided to use the tools written by the authors of [5] for this purpose. For simplification, we assume that only one phone is present, however the attack could also be carried out if multiple phones are present at the victim's base station.

After the call has been completed, the attacker is ready to execute the second phase of the attack. Because the second phase of the attack will disrupt communication between the phone and it's base station, the attacker may decide to delay the second phase to a later time, if he expects more calls to be made in the next time he is interested in.

3.2 Decryption of the call

The key idea in the second phase of the attack is, to use a replay attack against the phone to recover all keystreams, which were used to encrypt the original call. We first set up our base station impersonating the original base station of the victim.

3.2.1 Jamming the base station

To make the victim's phone loose communication with its original base station, one can use a utility named *hijack*, that is part of *libdect*, which broadcasts frames in the same channel and time slot as the original base station suggesting that the phone should change it's frequency and time slot. When the phone has received such a packet, it will switch to the frequency and time slot we suggested and switch to our base station. Checking whether the phone has locked on our base station can be done by periodically broadcasting a `LCE-PAGE-REQUEST` message from our base station. When the phone answers to the `LCE-PAGE-REQUEST`, it has locked on our base station.

DECT base stations are usually built in a way that they can detect another base station broadcasting on their frequency in their time slot. They do that by periodically skipping some of their broadcast packets and instead switching to listen mode in this time slot. If a signal is received there, another system must be broadcasting on the same frequency and time slot and the time slot or the frequency is changed. When we broadcast a request that the phone should switch to the frequency and time slot of our faked base station, our packet will not collide with the packed send by the base station and will very likely be received by the phone correctly.

3.2.2 Recovering the key streams

In the next step, we need to set the correct cipher key on the phone, which was used in the call we would like to decrypt. If no further call has been made and the phone was not switched off in the meantime, we could skip this step. Executing the step will only cost less than a second of additional attack time. Of course we do not know the key, otherwise, we could easily decrypt the call from our capture ourselves. However, the key was derived from the UAK and two random numbers RAND_F and RS using the A11 and A12 algorithms at the begin of the recorded call. We simply send again the `AUTHENTICATION-REQUEST` message, which was exchanged at the beginning of the recorded call. The phone will respond with an `AUTHENTICATION-REPLY` message containing the response of the challenge we send.

If the response, which is included, is not equal to the response in our recording, we might have picked the wrong phone, recorded incorrect data or the UAK on the phone has been altered. Therefore, the following steps would be unlikely to succeed:

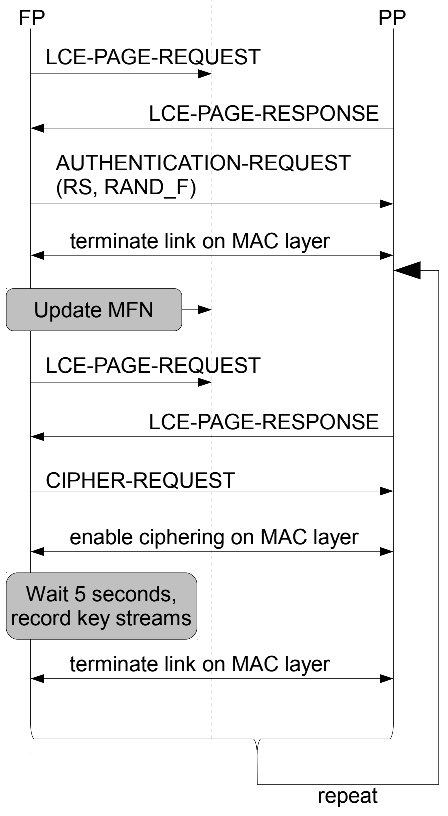
We now repeat the following procedure: Let n be the multiframe number of the first packet in our capture that we have not decrypted so far. We wait just before our base station broadcasts the next update of our multiframe number using a Q-channel message and set the multiframe number of our base station to $n - (t_d/16)$. Here, t_d is the time it takes from sending a `CIPHER-REQUEST` message to the phone until ciphering is enabled in the MAC layer and the first encrypted packet from the phone is sent in 10^{-2} seconds.

Then we broadcast our multiframe number update and send a `CIPHER-REQUEST` message to the phone. The phone will receive our multiframe number update and update its multiframe number accordingly. After having received our `CIPHER-REQUEST` message, the phone will respond with an encryption control MAC layer message stating that it is ready to enable encryption. We now confirm that using a MAC layer message and the phone will signal that it will enable encryption now. The next packet sent by the phone will be encrypted using the same key as the original call and the same initialization vectors (frame numbers) will be used. From now on, we do not send any payload to the phone and wait until the LCE.01 timer on the phone has expired and the link is released. Because we have not established a phone call on the link or did run any other application on it, the B-fields of all frames sent by the phone just contained **0xff** as plaintext. XORing all B-fields of the received frames with **0xff** reveals the key streams used to encrypt the frames in the original call. XORing these key streams with the B-fields in the original call decrypts these B-fields, revealing the audio data sent from the phone to the base station in the original call. We now repeat that procedure until all original call frames have been decrypted.

After the call has been decrypted, we shut down our base station and the phone starts scanning for the original base station and locks on it again.

Choosing t_d too small when implementing the attack results in some keystreams at the beginning of the call not being recovered. Choosing t_d too large results in a small performance decrease of the attack, so one might choose a slightly larger value for t_d in this step.

Figure 4: Attack overview


FP
PP
LCE-PAGE-REQUEST
LCE-PAGE-RESPONSE
AUTHENTICATION-REQUEST (RS, RAND_F)
terminate link on MAC layer
Update MFN
LCE-PAGE-REQUEST
LCE-PAGE-RESPONSE
CIPHER-REQUEST
enable ciphering on MAC layer
Wait 5 seconds, record key streams
terminate link on MAC layer
repeat

4. IMPLEMENTATION

We implemented a proof-of-concept of our attack. As basis, we used a DECT stack for the Linux kernel, which is available on `http://dect.osmocom.org/`. We needed to tweak three parts of the stack:

kernel We made a small change to the kernel code, that makes it possible to update the multiframe number in the kernel from the userland.

libnl We added another function, that passes a new multiframe number from the userland to the kernel.

libdect This library implements all higher level functionality of the kernel stack. We added a small change to libdect so that a program using libdect can pass a new multi frame number to the kernel, using libnl. We also wrote a new program that uses libdect to implement the attack tool. Because libdect already implemented much of the functionality we needed, we were able to implement the tool in 222 lines of C-code.

To enable encryption on the link, we used a cipher key CK with all bytes set to zero (**0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00**), so that the kernel decrypted all the encrypted traffic with this key. To capture the encrypted frames, we used libpcap, which is also part of the DECT stack, which we did not need to modify. To reveal the keystream, we decrypted the received B-field with the cipher key **0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00** again, which revealed the original B-field received and then XORed the B-field with **0xff** to reveal the keystreams.

We used a Siemens Gigaset AS150 phone to test our code. During implementation and execution, we observed the following effects:

- The phone did accept changes of the multiframe number without any problems, when the phone was in idle mode. However, it sometimes took the phone a few seconds until it was responsive again, after having changed the multiframe number.
- Changing the multiframe number after having established a connection, but before sending the `CIPHER-REQUEST` message did cause the phone to drop the connection.
- Sometimes, the phone lost the connection with our base station and became unresponsive. Stopping our base station and starting it again a few seconds later solved the problem. Restarting the phone or any other kind of interaction with the phone was never required, so this would only cause a delay in the execution of the attack, but not render it impossible. We are not entirely sure yet, whether this is a problem on the phone DECT stack or on our side.
- Sometimes, we received a few bits in the B-field incorrectly, resulting in some bits of the keystream recovered incorrectly. However, this only creates minor variations in the recovered audio stream, which usually will not be noticed.

  If this method would be used to recover a data-call (for example wireless internet access over DECT), this might be a problem. One could execute the attack multiple times to spot and correct variations in the keystreams recovered.

We also tested the attack successfully against a *Siemens Gigaset 4000 Classic* and a *T-Sinus 501* phone. Another test against an *AVM FritzFon* failed, due to genereal problems with our DECT stack, not related to the attack.

4.1 Performance

We were able to establish a connection with the phone about once every 14 seconds. A single connection lasted for 5 seconds, until the phones LCE.01 timer expired. (In fact our fake base station implements a 5 seconds LCE.01 timer, so our base station closes the connection just before the phone would have closed it. Disabling this timer on our base station makes the phone close the connection a few moments later, because it also implements the timer.) As a result, we were able to recover about 500 key streams in 14 seconds. The exact attack speed depends on the timing parameters chosen for the implementation and on the attacked phone.

As we did not send any call related messages on the C-channel, the phone did never ring or play any other sounds. Also, the content of the phone display did not change during the attack. Only when we had to restart our base station, the phone display indicated that it has lost the link to its base station and was now scanning for the base station. This makes it very unlikely that the attack is detected by a user by looking at the phone. Of course a user with a DECT sniffer could detect the presence of an attacker.

5. VARIATIONS AND FUTURE WORK

The attack is also applicable if the start of the call we would like to decrypt has not been recorded or has been recorded incorrectly. If so, the attacker does not know the random numbers used at the beginning of the call, and cannot replay them to the phone. But the attacker would still be able to send the `CIPHER-REQUEST` command to the phone, and as long as no subsequent call has been made on the phone, execute the attack anyway.

At the moment, we are just able to hold the connection to the phone for about 5 seconds, until the phones LCE.01 timer has expired. Setting up the connection again takes some time and slows down the attack from a theoretical maximal speed of 100 key streams per second to about 33 key streams per second. If one could find a way to send messages to the phone while the link is open to prevent the phones timer from expiring, this could speed up the attack. At the moment, we are not able to send C-channel messages over the link, because we cannot properly encrypt such messages, because we do not have a way to generate or recover the needed key streams.

The attack only recovers the parts of the key streams used by the phone to encrypt the B-field of frames, which are transmitted from the phone to the base station, more precisely bits 400 to 719 of the output of DSC (cs400 . . . cs719). If one could recover other parts of the output of DSC, one could decrypt parts of the data sent from the base station to the phone (bits 0 to 359 are used here), or parts of the C-channel traffic send from the phone to the base station (bits 360 to 399). Besides the full frame (named P32 in the DECT standard), the DECT standard also defines the *half frame format* (named P08j). Here, only 80 instead of 320 bits are used for the B-field. So bits 160 to 199 are used to encrypt the B-field send from the phone to the base station. If one would request the usage of half slots from the phone, one could recover bits 160 to 199 of the output of DSC and use these bits to decrypt parts of the voice-data send from the base station to the phone in the original call.

5.1 TETRA and GSM

TETRA (Terrestrial Trunked Radio)[2] is a digital radio protocol mostly used by government agencies to replace analog walkie-talkies. GSM (Global System for Mobile Communications)[3] is a radio protocol for digital cell phones). TETRA and GSM use a similar radio protocol. We think that the attack techniques used here might be applicable to these protocols too. A similar protocol level attack against GSM is known[1], but no attack on GSM or TETRA using our method has been published so far.

[2] `http://www.etsi.org/website/Technologies/TETRA.aspx`

[3] `http://www.etsi.org/WebSite/technologies/gsm.aspx`

6. COUNTERMEASURES

We think the attack is hard to counter. The attack uses no implementation specific problems and just relies on the standard. However, we discuss countermeasures in this section:

6.1 Forbidding the reuse of random numbers

In our attack, we negotiate the same key again as in the original call. A phone could keep a record of the last used keys and/or random numbers in the key exchange messages to prevent renegotiation of an old key. First of all, the standard does not prohibit the reuse of random numbers, so this could lead to compatibility problems with base stations, which tend to reuse keys. In addition, this will be hard to implement on low-cost consumer phones, with very limited memory. One could store the list of old keys in RAM, which would be erased when the phone is turned off. Storing the list in the EEPROM would also be unwise, because the EEPROM can only do a limited amount of write-cycles, before the memory gets permanently damaged. The attacker could also negotiate a lot of random keys with the phone first, so that the key the attacker is targeting for would be erased from that list.

In addition, cipher keys should be erased from the phone memory, after a link using that cipher key has been terminated.

6.2 Requiring authentication of the base station first

In contrast to GSM, DECT also supports authentication of base stations. So far, we have not seen a phone in the real world, that uses this feature[6]. This feature is defined in the standard and could easily be implemented. A phone could request that a base station authenticates itself first, before being allowed to enable encryption on a link or to request authentication. However, a skilled attacker could relay these requests to the original base station, impersonating the phone. As a result, the attack would still be possible, however the base station would be able to report the abnormal high rate of authentication requests. Because this authentication procedure does not affect the cipher key, the base station and the phone cannot alter the cipher key by choosing different random numbers for every authentication.

6.3 Checking for changes in the multiframe number broadcasted by the base station

Setting the multiframe number used by our base station back to a value used in the original call is vital to our attack. In practice, a base station should never change their multiframe number, except for a base station, which has just been started or restarted. Frequent changes in the multiframe number of a base station can be detected by a phone and should be seen as in indicator of an attack. A phone could, for example, put a base station that changes their multiframe number frequently on a temporary blacklist. This could be used to render the attack less effective, but not impossible, because a base station has to be allowed to reset its multiframe number when it has been restarted.

6.4 Rekeying

To counter key recovery attacks on the DSC, a new update of the DECT GAP (Generic Access Profile) Standard[2] was published in 2010[4], which allows rekeying during a call. Because current attacks on DSC need a lot of keystreams and a single cipher key is only used for 60 seconds and thus generates 6000 keystreams at most, making key recovery attacks impractical. For rekeying, a new `AUTHENTICATION-REQUEST` message is being sent during an encrypted call, negotiating a new key. Our attack relies on the fact that we can see the random numbers embedded in that message and replay them later. As a result, we would only be able to decipher the first 60 seconds of a call, until the first rekeying event. The `AUTHENTICATION-REQUEST` message cannot be decrypted using our method because it is sent from the base station to the phone, but not in the reverse direction. Also, we would be able to decipher the last seconds of a call, if no subsequent call has been made, and the key is still present in the phone memory.

To attack such phones, attempts to change the multiframe number during a call could be made, so that the `AUTHENTICATION-REQUEST` message sent from the base station in its encrypted form can be replayed. However, we are at the moment not in possession of a phone, which implements this updated DECT GAP standard, and therefore we cannot perform tests on such a phone.

6.5 Disabling B-field encryption, when unused

Another countermeasure would be to send the B-field unencrypted, when it is not used to transport payload. Alternatively, the B-field could be filled with random data instead.

7. CONCLUSION

We have shown that calls made with DECT phones, which do not allow unencrypted calls and have no other implementation flaws, can be decrypted. This works even for very short calls, where a key recovery attack against DSC is not (currently) possible. Weak spots are the missing replay protection and that the cipher key is only derived from the UAK and random numbers chosen only by the base station, but not by the phone. The attack is practical and can be executed at very low cost.

We are confident, that these problems and all other known problems in the DECT standard will be fixed by ETSI in a new release of the DECT standard with a new version of the security algorithms namely DSAA and DSC and the corresponding changes in the DECT protocol. We have contacted DECT device manufacturers so that they can evaluate our results and implement countermeasures. Today, DECT is still a great standard for cordless phones, which outperforms WIFI-phones when it comes to implementation costs and energy efficiency, but cannot provide the same level of security as an WPA2 protected wireless network.

We would like to thank the DECT Forum and ETSI. We notified both of them in advance, so that they can start to incorporate countermeasures against the attack in the next release of the DECT standard.

8. REFERENCES

[1] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. *Advances in Cryptology-CRYPTO 2003*, pages 600–616, 2003.

[2] European Telecommunications Standards Institute. ETSI EN 300 444 V2.1.1: Digital Enhanced Cordless Telecommunications (DECT); Generic Access Profile (GAP), Oct 2008.

[3] European Telecommunications Standards Institute. Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 1: Overview, June 2010.

[4] European Telecommunications Standards Institute. ETSI EN 300 444 V2.2.1: Digital Enhanced Cordless Telecommunications (DECT); Generic Access Profile (GAP), June 2010.

[5] S. Lucks, A. Schuler, E. Tews, R. Weinmann, and M. Wenzel. Attacks on the DECT authentication mechanisms. *Topics in Cryptology–CT-RSA 2009*, pages 48–65.

[6] A. Mengele. Digital Enhanced Cordless Telecommunication (DECT) devices for residential use. Diploma thesis, Technische Universität Darmstadt, 2009.

[7] H. Molter, K. Ogata, E. Tews, and R. Weinmann. An Efficient FPGA Implementation for an DECT Brute-Force Attacking Scenario. In *2009 Fifth International Conference on Wireless and Mobile Communications*, pages 82–86. IEEE, 2009.

[8] K. Nohl, E. Tews, and R. Weinmann. Cryptanalysis of the DECT standard cipher. In *Proceedings of the 17th international conference on Fast software encryption*, pages 1–18. Springer-Verlag, 2010.

[9] M. Weiner, E. Tews, B. Heinz, and J. Heyszl. Fpga implementation of an improved attack against the dect standard cipher. In *ICISC 2010 - 13th Annual International Conference on Information Security and Cryptology*, LNCS. Springer, Nov 2010.

APPENDIX

A. ACRONYMS

Name	Description	Section
A11	Algorithm in DSAA	2.2
A12	Algorithm in DSAA	2.2
A21	Algorithm in DSAA	2.2
A22	Algorithm in DSAA	2.2
AUTHENTICATION-REQUEST	Message to request authentication	2.2
AUTHENTICATION-RESPONSE	Response to an authentication request	2.2
C-Channel	A channel for control traffic	2.1
CIPHER-REQUEST	Message to request start of ciphering	2.3
CK	Cipher Key	2.2
DSAA	DECT Standard Authentication Algorithm	1, 2.2
DSC	DECT Standard Cipher	1, 2.3
FP	DECT base station	1
frame	Time period of 10 ms	2.1
LCE.01	Timer which starts when a connection is not required anymore	2.1
LCE-PAGE-REQUEST	Message to request a phone to start a connection	2.1
multiframe number	Most significant bits of a frame number	2.1
P00	Packet format with no B-field	2.1
P08j	Packet format with 80 bits B-field	5
P32	Standard packet format with 320 bits B-field	2.1
packet	A single data burst	2.1
PP	DECT phone	1
RAND_F	Random number chosen by a base station	2.2
RAND_P	Random number chosen by a phone	2.2
RES1	Response to a challenge during authentication of a phone	2.2
RES2	Response to a challenge during authentication of a base station	2.2
RS	Random number chosen by a DECT network	2.2
UAK	User Authentication Key (shared by phone and base station)	2.2