Insecurity in Public-Safety Communications: APCO Project 25

Stephen Glass[†], Vallipuram Muthukkumarasamy[‡],
Marius Portmann[†]and Matthew Robert

[†]NICTA, Queensland Research Laboratory, Brisbane, Australia
[‡]Griffith University, Gold Coast, Australia
stephen.glass@nicta.com.au,marius.portmann@nicta.com.au
v.muthu@griffith.edu.au,matt.robert@gmail.com

Abstract. *APCO Project 25* (P25) radio networks are perhaps the most widely-deployed digital radio technology currently in use by emergency first-responders across the world. This paper presents the results of an investigation into the security aspects of the P25 communication protocol. The investigation uses a new software-defined radio approach to expose the vulnerabilities of the lowest layers of the protocol stack. We identify a number of serious security flaws which lead to practical attacks that can compromise the confidentiality, integrity and availability of P25 networks.

Key words: communications networks, wireless network security, security analysis

1 Introduction

Emergency and public-safety communications systems are increasingly making use of digital technologies such as *Terrestrial Trunked Radio* (TETRA) and *APCO Project 25* (P25). Compared to the analogue land mobile radio systems that preceded them these digital systems claim improved radio spectrum use, increased geographical coverage, centralized channel management (trunking) and support for both voice and data services. A key advantage to digital systems is that they enable secure operation that ensures the confidentiality of voice and data traffic using proven cryptographic ciphers. As a result these systems have a reputation for being much more secure than analogue systems. In this paper we present the results of a critical security analysis of the P25 protocols and identify a number of flaws that lead directly to practical and effective attacks. These attacks include bypassing the authentication and access control mechanism, disabling specific nodes at will and the passive recovery of the encryption keys for some of the standard ciphers. We also describe in detail a widely-used proprietary P25 cipher system and show the encryption key can be recovered with only a relatively small effort. To the best of our knowledge this is the first time this cipher has been described.

1.1 Structure of The Paper

The structure of this paper is as follows: section 2 provides a brief background to P25 and the structure of P25 network traffic. Section 3 describes the motivation for the software radio approach used and describes the software tools we constructed for the investigation. Section 4 outlines the flaws we identified in the protocol. Section 5 the most effective attacks which result from these flaws. In section 6 we discuss related work and conclude in section 7.

2 APCO Project 25

P25-based systems are used by first-responder emergency services across the US, Canada, Australia and New Zealand. P25 radio systems may be used in simplex mode (i.e. without any infrastructure) but are typically used in infrastructure-based networks and consist of both fixed and mobile equipment as shown in figure 1. The *mobile radios* (MRs) are either hand-portable or vehicle-mounted and paired with a *mobile data terminal* (MDT) for accessing data services. The *fixed station* (FS) fulfills the roles of base station, *key management facility* (KMF), trunking controller and repeater. The FS may also provide data services and gateways to the public switched telephone network, automatic branch exchanges and other radio systems.

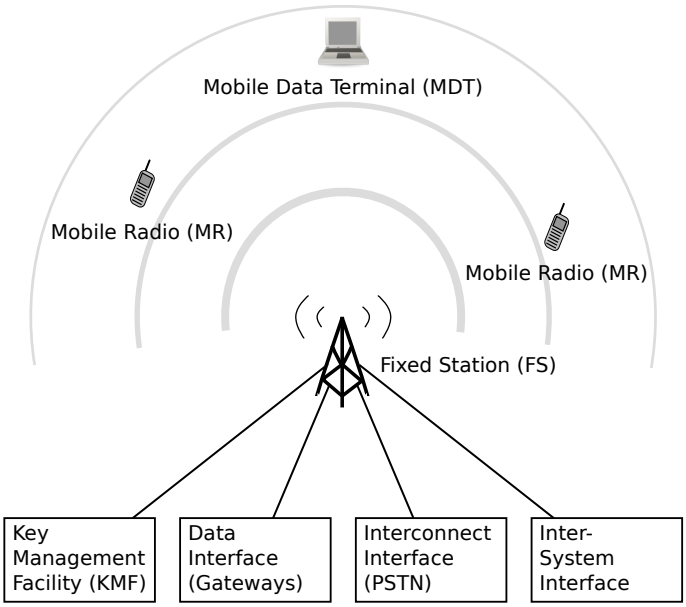

Fig. 1. P25 System Components

The P25 standard is jointly administered by the *Telecommunications Industry Association* (TIA) and the *American National Standards Institute* (ANSI) and ensures interoperability of equipment from different manufacturers. The P25 *Common Air Interface* (CAI) defines the modulation techniques, frame types and physical layer representations that must be implemented by all P25-compliant radios [1]. In the existing P25 standard CAI traffic is exchanged at 9600 bps using either 4-level *frequency-shift keying* (FSK) modulation in a 12.5 kHz half-duplex channel or $\frac{\pi}{4}$ *differential quadrature phase-shift keying* (DQPSK) modulation in a 6.125 kHz half-duplex channel. To accommodate the limited data rate, voice transmissions make use of the IMBE vocoder to encode voice traffic into compressed voice codewords; where each 88-bit codeword represents 20ms of uncompressed speech.

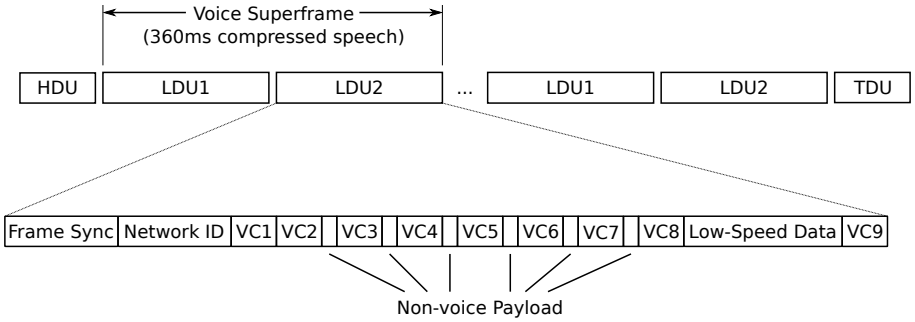

Fig. 2. P25 Voice Transmission Frame Structure

All P25 voice and data traffic is transported by data-link layer frames which are known as *data units* (DUs). Data traffic uses variable-length *packet data units* (PDUs) whereas voice transmissions use a variety of fixed-size frames that occur in a fixed structure. Figure 2 shows the structure of a voice transmission. Each voice transmission begins with a *header data unit* (HDU), followed by a number of voice superframes which carry the compressed voice traffic. That is followed by a *terminator data unit* (TDU). Each superframe is composed of alternating *logical data unit 1* (LDU1) and *logical data unit 2* (LDU2) frames; each of which contains nine IMBE compressed voice codewords and differ only in the meaning attached to the non-voice payload of each frame.

3 Approach to Security Analysis

The purpose of the security analysis is to identify any security flaws present in the protocol. The adversary model we presume is that of an external attacker who has complete access to the message transmissions but who has no knowledge of the encryption keys in use. We begin by studying the standard to identify possible vulnerabilities, and progress to study the traffic in a real

test-bed network. Unfortunately, in commercially-available equipment, low-level access to the protocol stack is not usually available. Re-purposing commercial P25 equipment is difficult because device programming specifications are either unavailable or available only when entering into non-disclosure agreements. This approach is further complicated because P25 equipment often employs a degree of security-through-obscurity and tamper-proofing measures. These problems motivated the investigation to adopt the use of a novel *software-defined radio* (SDR) or software radio approach. An SDR is one in which the majority of the signal processing is done in software as opposed to purpose-designed electronic circuits. This approach enables us to examine and manipulate message traffic at the physical and data-link layers of the protocol stack. The principal advantage of this technique is that the SDR is not constrained to the behaviours expected from commercial equipment and can be used to expose flaws, implement attacks and prototype countermeasures. The software radio approach can also assist in reverse-engineering protocols, which are undocumented and would otherwise not be available for analysis (an example of this is discussed in Section 5.3).

3.1 Software-Defined Radio Implementation

To facilitate the investigation we have developed software tools that allow us to create, transmit, receive and analyse P25 message traffic. The software tools are built using the GNU Radio framework [2]. This is a free software framework for writing software radios in C++ and Python and is available under the terms of the GNU Public License. The GNU Radio framework provides a large collection of signal-processing blocks which transform their input signal(s) into output signal(s) in a well-defined way. Blocks can even be combined into a functioning software radio using a graphical, direct-manipulation editor called the *GNU Radio Companion* (GRC). Using the GNU Radio framework provides a robust signal-processing framework and ensures hardware independence because GNU Radio can make use of common abstractions to communicate with a wide variety of signal sources and sinks.

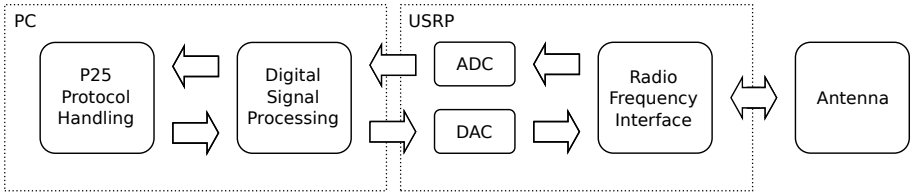

Fig. 3. Block Diagram View of Software Radio

Figure 3 shows a block diagram of the software radio. A host PC is responsible for the P25 data-link layer and the digital signal processing needed for the physical layer whilst an SDR provides the means of converting digital samples to and from radio frequency signals. The actual SDR is a *Universal Software Radio*

Peripheral (USRP) [3]; this is a low-cost device that has been purpose-designed to work with the GNU Radio framework. The USRP provides for high-speed analogue/digital and digital/analogue conversion and implements a radio frequency interface that is responsible for amplification and frequency conversion from the baseband to the appropriate part of the radio spectrum. This is achieved using a daughterboard which, in this instance, provides receive and transmit capabilities for the UHF frequency bands used in public-safety communications. The digital samples are passed between the host PC and USRP via either a USB2 or a Gigabit Ethernet interface.

3.2 P25 Receiver

The USRP can process approximately 6 MHz of the radio spectrum at one time, allowing hundreds of P25 signals to be processed simultaneously. This makes it a useful tool for both security analysis and network monitoring because network traffic can be captured, monitored and analysed to assist in identifying security flaws or diagnosing operational problems. The P25 Receiver itself is a hybrid Python/C++ program that allows for message traffic to be captured off the air for storage and analysis. The P25 Receiver can be thought of as comprising four main stages:

1. Filtering to select a particular channel of interest and performing frequency translation to produce a series of baseband samples.
2. Demodulation transforms the baseband samples into a stream of dibit symbols using a *4-level frequency shift keying* (4FSK) demodulator.
3. Decoding recovers the P25 data-link frames from the symbol stream. This requires a custom signal-processing block to identify frame boundaries and re-construct the frame bodies, de-interleaving and applying forward error-correction.
4. Distribution of the resulting traffic for further processing. CAI traffic can be distributed via the Internet or intranet using a CAI-in-UDP encapsulation[1]. This traffic can be multicast or unicast to listeners that perform tasks such as audio decoding, message logging, re-transmission or detailed traffic analysis.

To assist with the inspection and analysis of P25 network traffic a plug-in module for the WireShark protocol analyzer has been contributed by Michael Ossman. This plug-in allows P25 traffic to be analyzed and filtered using WiresShark. A detailed description of an earlier version of the P25 Receiver can be found in [4].

3.3 P25 Transmitter

The P25 transmitter is also a hybrid Python/C++ program which accepts P25 data-link frames as its input and produces a P25 radio signal as its output. These frames are read from file and encoded into a symbol stream that is modulated and amplified before being sent to the USRP. The use of pre-prepared files for

[1] IANA has registered UDP port 8062 for use by the CAI-in-UDP encapsulation.

the input was chosen because message traffic can be prepared in advance and offers precise control over how and when the message traffic is to be injected. The format of the P25 input file is chosen to be the same as that used by the WireShark protocol analyzer. This means that traffic captured by the P25 receiver can be re-injected with little effort. A block diagram for the transmitter is shown in figure 4 that shows how the signal-processing blocks are connected together.

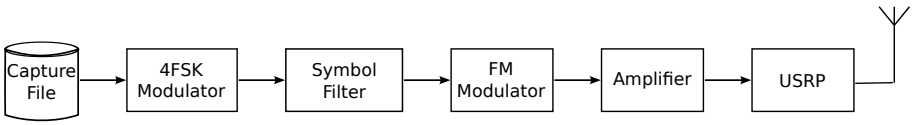

Fig. 4. Block Diagram view of P25 4-Level FSK Transmitter

4 Security Flaws in P25

Although P25 supports the use of cryptographically sound ciphers such as AES and 3DES the use of such ciphers alone is not sufficient to ensure secure operation. In this section we summarize the security flaws that we have identified in the P25 specifications.

4.1 Optional Encryption

Possibly the most important shortcoming of P25 is that the use of secure communications is optional. For *mobile radios* (MRs) an additional hardware module or firmware upgrade is usually required before encryption can be used. As we discuss later (§4.6), the security protocols of P25 do not provide an effective authentication mechanism and cannot establish the authenticity of a message. Although it is an inter-operability advantage to be able to fall back to un-encrypted or even analogue modes of operation one of the most severe consequences is that all radios must process messages that are sent in the clear. Therefore, an adversary can inject messages into the network which are in the clear and rely on network devices and infrastructure handling them as though they are legitimate. This exposes radios to the risk of "fuzzing" attacks by adversaries who can create illegal traffic that is intended to crash or otherwise compromise the integrity of radios.

4.2 Flawed Authentication and Access Control Mechanism

Authentication and access control seek to restrict access to the network to users who are suitably authorized. The original P25 standard did not mandate an authentication and access control mechanism but has been amended to include

an optional authentication mechanism [5]. This a relatively recent development and has not yet been widely implemented. The result is that the vast majority of P25 systems do not have *any* means by which to prevent unauthorized access. The authentication mechanism defined by P25 uses a cryptographic challenge/response protocol to authenticate the *Mobile Radios* (MRs) and the *Fixed Station* (FS). This provides for one-way authentication (MR to FS) and mutual authentication (MR to FS and FS to MR).

The one-way *Radio Authentication* (RA) protocol is shown in protocol 1. In this protocol a unique secret key K_{MR} is shared between the MR and an *Authentication Centre* (AuC). Authentication begins in response to the MR sending a registration request to the FS. In step 1 the AuC generates an 80 bit random seed RS and 128 bit authentication key K_S and sends them to the FS. The authentication key K_S is derived from RS using the $AM1$ procedure (which zero-pads RS to the AES block size and encrypts it under K_{MR}). In step 2 the FS generates a 40 bit random challenge $RAND1$ which is sent with RS to the MR. At this point the MR can compute $RES1$ by using RS to derive K_S and encrypting $RAND1$ using the $AM2$ procedure (which zero-pads $RAND1$ to the AES block size and encrypts it under K_S). At this point the FS can compare $RES1$ against the value it has computed. If, the two values match the MR is considered to be authenticated and registered successfully; otherwise the registration attempt is rejected. An extension of this protocol allows for *Mutual Authentication* (MA).

Protocol 1 Radio Authentication Protocol

1 $AuC \rightarrow FS$: $\mathrm{K}_S = AM1_{K_{MR}}(RS), RS$
2 $FS \rightarrow MS$: RAND1, RS
3 $MR \rightarrow FS$: RES1=$\mathrm{AM2}_{K_S}(RAND1)$

There is, however, a serious security flaw present in this authentication and access control mechanism. The authentication process decouples authentication and key agreement — successful authentication does *not* establish a session key but instead merely changes the state of the association to the authenticated state. This is a consequence of the the optional status of both the authentication and encryption services which can be used completely independently of each other. An adversary can monitor the channel and learn the identity of MRs that have already registered successfully and then assume those identities. The assumption that an adversary cannot discover the identities of registered stations or easily change their identity maybe valid for typical commercial systems but for a software radio it is trivial to monitor traffic and assume the identity of registered stations. As a result this mechanism provides absolutely no defence against an intruder.

4.3 Flawed Key Hierarchy

Serious security flaws are present in the design of the key hierarchy used by P25. Most importantly, the standards do not mandate a key hierarchy that ensures that individual associations have their own unique encryption key. Instead a single *traffic encryption key* (TEK) is shared by a number of MRs that are known as a cryptogroup. A single MR may belong to several cryptogroups and so a number of these TEKs can be programmed into a MR. This allows for some radios to be programmed with keys not present in others and preserves operational security between unrelated groups. The transmitter identifies which encryption key is in use by means of a sixteen-bit *key id* (KID) that is transmitted in the plain as part of the *header data unit* (HDU) and repeated as part of the *logical data unit 2* (LDU2) non-voice payload. The second level of keys in the key hierarchy are the *Key Encryption Keys* (KEKs) used by the KMF to perform an *over-the-air re-keying* (OTAR) operation in which an MR's encryption keys can be remotely re-programmed [6]. The KEKs are used by the KMF only for the distribution of encryption keys and encryption of other OTAR messages. The initial KEKs are bootstrapped into the MRs using a hardware device known as a keyloader whereas further TEKs/KEKs programmed using OTAR messages.

The use of a single TEK for many different transmissions/users means that all traffic encrypted under that key can be decrypted as the result of a successful key-recovery attack. This effect is compounded because the same traffic encryption key is likely to remain in use for an extended period of time. This is because key management can be a problem when there are many devices and key changes must be co-ordinated across many different groups. The difficulty of this task means that it tends to be performed infrequently. Australian emergency responders, for example, do not use OTAR and usually change their TEKs on an annual basis. The combined effect is that an adversary has a significant incentive to recover an encryption key because a successful key recovery will reveal the contents of a large amount of traffic. They also have a large amount of time in which to do so and, once the encryption key is discovered, have complete access to traffic in real-time.

Another serious problem with using a single key for an entire cryptogroup means that any station can masquerade as any other within the same cryptogroup. Although this is principally an insider attack it presents problems when an MR is stolen. The key plus the transmitter's station identity is assumed to be sufficient to authenticate a station. When a MR is stolen it is quite possible for an adversary to change the device's identity whilst preserving encryption keys. The theft of an MR can be mitigated in several ways. Firstly, OTAR allows for the TEKs of legitimate stations to be changed in response to a reported theft; if the stolen radio is within radio range and remains powered then OTAR permits the keys present in the stolen radio to be erased remotely. The second line of defence are the physical security and anti-tampering measures of the MR itself. It is typical for an MR to employ tamper-proofing measures that erase the encryption keys to prevent their recovery.

4.4 Weak Encryption

P25 allows for the use of several optional cipher systems including DES, 3DES and AES. Some of these cipher systems employ weak cryptographic ciphers that are subject to key-recovery attacks. At the time of writing the cipher in most widespread use is DES in OFB mode[7]. Although it is marked for "backward compatibility" by TIA several factors favour DES-OFB and militate against the use of the other, more secure, ciphers such as 3DES and AES:

- DES-OFB is the *only* cipher that manufacturers *must* implement in order to comply with the specification,
- users frequently encounter interoperability problems when using optional features of equipment from different manufacturers and
- the export of certain US-manufactured devices (such as AES-capable keyloaders) requires an export license for shipping outside of the US and Canada.

These factors entrench the use of the DES-OFB cipher system which remains in widespread use. DES has remained largely resistant to cryptographic attacks but, because of its limited key size, exhaustive key search attacks have proven to be very effective. Using specialized hardware such attacks can now be conducted very quickly using only modest resources.

4.5 No Guarantee of Message Freshness

P25 is also vulnerable to message replay attacks. The adversary can record messages and re-inject them into the system at a later time. To protect against replay attacks requires an authenticated nonce, sequence number or timing information to be included in the messages so that replayed messages can be detected by the receiver and ignored. Data frames can optionally meet these requirements using a monotonically-increasing *message number* (MN) and a MAC computed across the MN and message payload. Unfortunately, the optional nature of these protections permits an adversary to construct traffic which misrepresents its identity and indicates that no MN is present.

4.6 Flawed Message Authenticity and Integrity Mechanism

We have already alluded to the fact that there is no explicit guarantee of authenticity for voice traffic; this is a direct consequence of the optional status of the encryption protocol. Data messages are usually protected only with *cyclic redundancy checks* (CRCs) computed over the ciphertext of the frame and sent in the plain. These offer no protection against message modification and replay attacks. As an option for data traffic, P25 allows for the use of DES/CBC to compute *message authentication codes* (MACs) to authenticate the values of some data and control frames. The CBC/MAC protocol is, however, optional and can easily be bypassed (see §5.2).

5 Security Attacks and Defences in P25

The security flaws present in the P25 protocol introduce the threat of attacks and this section identifies the threats and, where possible, the proposed countermeasures. The experimental method consists of identifying security threats in the behavior of the P25 network using a simple test-bed that consists of a single P25 transceiver used together with the SDR P25 implementation. A repeater was not available for testing and so a simplex UHF radio channel is used to empirically verify the attacks.

5.1 Denial of Service - the Inhibit Attack

In all radio systems there are denial-of-service risks at the physical layer through collision jamming and other attacks. In digital and trunked systems such as P25 there are new threats from attacks directed at the network control protocol. The Inhibit attack makes use of the anti-theft measures of the P25 standard to disable legitimate nodes. The P25 protocol contains an anti-theft mechanism which is intended to prevent a stolen radio being used by an adversary. This feature is known colloquially as "stun" and is implemented by sending a special abbreviated PDU known as a *Trunk Signaling Data Unit* (TSDU) to the device. The payload of the TSDU is an "inhibit" *Extended Function Command* (XFC) the structure of which is shown in figure 5. Once a radio has been stunned by the receipt of an inhibit command the standard requires that it remains in-operational and unresponsive to the operator console or device programming interface until it receives an "uninhibit" XFC on the frequency it received the inhibit. The attack exploits the lack of any guarantee of authenticity for the frame Inhibit/Uninhibit types. The adversary simply directs "inhibit" commands towards legitimate stations causing them to become disabled without any explanation. The format of the XFC is shown in figure 5. Note that the XFC message payload may be sent either encrypted (P=1) or un-encrypted (P=0) and that there is no explicit means of authenticating the inhibit command.

The inhibit function presents a serious threat to availability and does not provide a satisfactory anti-theft measure because a thief can uninibit the radio themselves. For this reason some manufacturers allow for radios to be configured to ignore inhibit commands. This is often a configuration option that can be set for each MR using the equipment's programming interface. Allowing inhibit to be disabled is intended to mitigate the threat of DoS attacks but does so at the cost of negating the anti-theft measure.

5.2 Message-Modification Attack

The weak authenticity and integrity provisions of P25 expose it to threat of message modification attacks. A message modification attack can detect the presence of MAC-protected frames, remove the MAC and substitute a CRC in its place. The receiver will accept such frames as legitimate even though they do

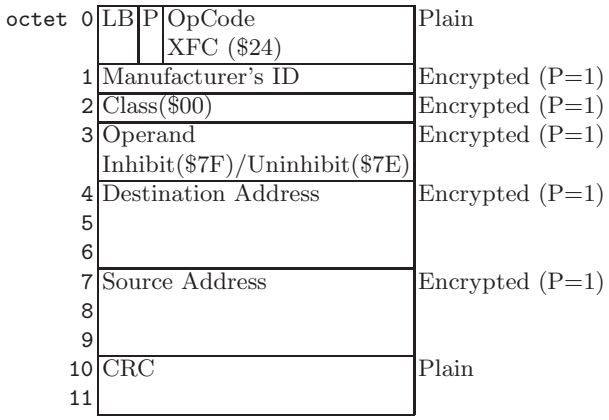

Fig. 5. Extended Function Command

not possess the MAC because they conform to the specification. A two-bit field is present in the frame header and indicates whether the frame has no checksum, a CRC or a cryptographic MAC. Although this field is encrypted the adversary can detect the presence or absence of the MAC based on frame size. The use of stream ciphers means that an adversary can perform a simple XOR operation to change the state of these bits. Thus messages can be modified by an adversary to remove the MAC without the receiver's knowledge and without possessing the encryption key. The only remedial measure is to make the use of MACs mandatory as the strongest authenticity and integrity mechanism available in the standard and ignore all traffic which is not suitably protected. Unfortunately such a move would still fail to protect voice traffic and would not be compatible with existing equipment.

5.3 Key Recovery by Exhaustive Key Search

The use of weak ciphers by P25 equipment makes it possible to recover the encryption key using an exhaustive search. DES is no longer regarded as secure because an exhaustive key search can be mounted to recover the encryption key. Motorola's proprietary *Advanced Digital Privacy* (ADP) cipher, which is described here for what we believe to be the first time, uses a 40 bit key and is considerably less secure than even DES/OFB. In this section we will describe how this attack can be conducted to recover encryption keys.

Known-Plaintext The exhaustive key search presented here exploits the fact that voice messages contain a known-plaintext that occurs at known locations in the message. These arise because, when a voice message is finished but there are unused voice codewords in the current frame, the transmitter is required to complete the LDU with silence codewords [1, §8.2.3]. A similar process, known as audio muting, occurs at the beginning of voice transmissions and results in the first few voice codewords being encoded as silence Our observations have shown

that audio muting provides 4 silence silence voice codewords at the beginning of a transmission. If an adversary can correctly identify a silence codeword then they can reveal 11 consecutive octets of keystream. An adversary monitoring a voice transmission can identify the first and/or last frame in a transmission and those codewords which have the highest probability of being silence. Exhaustively searching for the key which generates the appropriate keystream is possible when the key space is small enough and allows the adversary to discover the encryption key.

DES/OFB The DES/OFB cipher system is the only cipher system which the standard declares to be a mandatory option. That is, equipment suppliers must be able to offer DES/OFB as an option on their equipment in order to pass the compliance testing. DES has a 56 bit key which means a key space of approximately 7×10^{16} unique keys. On average an adversary would search half of the key space before discovering the key. Exhaustively searching this key space is computationally intensive but modern hardware makes such a strategy possible. To conduct an exhaustive key search against DES/OFB the known-plaintext must be chosen carefully so that 16 sequential octets of keystream are revealed; these must be aligned on a 64 bit boundary and represent the input and output of a single DES/OFB encryption operation. Given these two blocks an exhaustive key search simply encrypts the input value using DES/ECB under every possible key until the actual output value is found.

When using the beginning of a captured transmission for an exhaustive key search we can use the fact that the VC1 and VC2 voice codewords at the start of the LDU 1 are silent to reveal such blocks 4 and 5 of the DES/OFB keystream. The presence of silence in the voice codewords at the start of the transmission make this the preferred choice. The situation is slightly more complex when using the codewords at the end of the LDU1 or LDU2 because they will be silent only with a given probability distribution. The uses of these latter codewords is further complicated because of the presence of two octets of non-voice payload which are unknown to the adversary. The example for an LDU1 is shown in figure 6 and a similar situation exists at the end of the LDU2. This doesn't pose a serious problem because the exhaustive key search will produce 2^{16} candidate blocks which can be verified simply by repeating the encryption and matching the resulting block to the ciphertext revealed by the known plaintext.

A commodity 2.5GHz Intel Core i7 processor can easily compute one million DES keys per second in software using the OpenSSL library. This is, however, optimized for the case of encrypting the key with large volumes of traffic and not key searching. A bit-sliced implementation carefully optimized for key searching can reach in excess of twenty-eight million keys/second. Even so, DES is not trivially defeated. Even at one hundred million keys per second it will take almost twenty-three years to search the whole key space. It is possible to achieve much better performance using dedicated hardware and many processors running in parallel. In 1998 the EFF constructed an ASIC-based device that could search the DES keyspace within 9 days at a cost of 250,000 US$ [8]. Since then the cost of computation has fallen and efficient DES cores have been developed such as

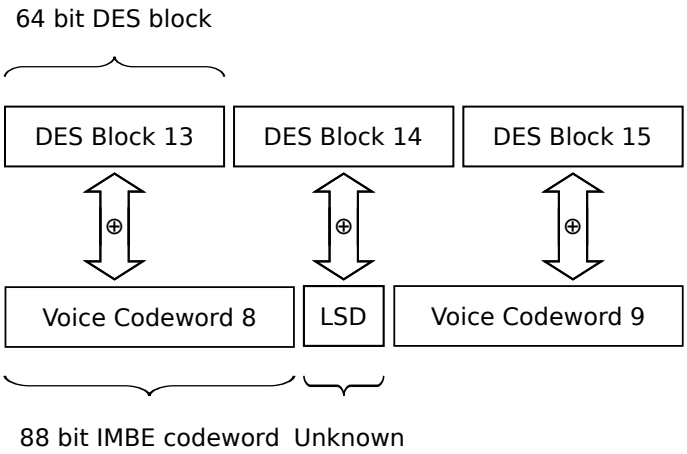

Fig. 6. Presence of Non-Voice Data in Encryption Schedule

the core developed by the UCL Crypto Group at the University of Lovain-la-Neuve which is optimized for such key searches [9, 10]. This core has been used in COPACABANA — a recent FPGA-based device that can exhaustively search the DES key space within nine days at a cost of just 10,000 US$ [11].

ADP *Advanced Digital Privacy* (ADP) is a proprietary cipher system that is available on some Motorola equipment as a firmware upgrade. There is no publicly available documentation describing the ADP and so we have reverse-engineered the cipher to discover how it operates. We know from the user interface of the radio management software that the ADP cipher has a 40 bit key. This appears to have been chosen to meet the now-defunct US export restrictions for cryptographic products. The size of the keyspace is much too small to protect traffic from an exhaustive keyspace search.

We conducted our investigation using traffic captured from a Motorola XTS 5000 hand-held radio with the ADP cipher option enabled. A transmission was made that consisted of audio silence and was sent without encryption. Inspection of the first transmission showed that the radio was correctly transmitting the silence codeword values as required by both the CAI and the IMBE vocoder specification [12]. A second transmission also of audio silence was made using ADP under a known encryption key. ADP is rumoured to make use of the RC4 cipher and so we subjected the encrypted message to a simple analysis in which different combinations of the key and IV are used to generate 2048 octets of keystream. The resulting keystream is compared with the presumed keystream from an encrypted frame and the result scored on the number of mismatches to the expected silence plaintext.

We confirmed that the cipher used by ADP is RC4 in which 40 bit secret key is combined with the 64 bit IV to form a 104 bit encryption key. The RC4 cipher is used produce 484 octets of keystream which is used to encrypt/decrypt the

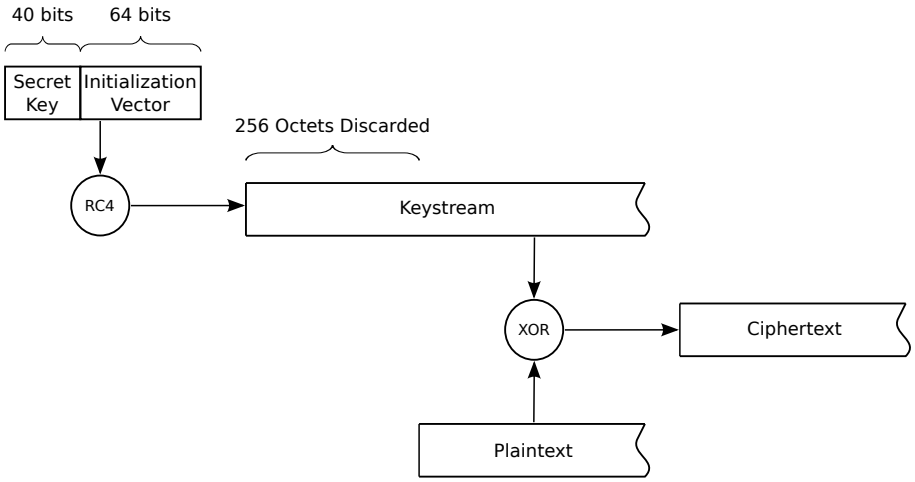

Fig. 7. ADP Cipher Encryption

payload of the voice superframe. The operation of the ADP cipher is outlined in figure 7. ADP appears to makes use of RC4 in a secure fashion and:

- ADP*appends* the IV to the secret key to make the encryption key making it difficult for an observer to identify frames encoded under weak keys — one of the key flaws common to many RC4 implementations.
- ADP discards the initial 256 octets of the keystream which have been shown to be correlated with the encryption key. In this ADP has followed the advice on the correct use of the RC4 cipher.

Exhaustive key search for ADP consists of using a silence codeword to recover the probable keystream and then using the IV for the message to search every one of the possible 2^{40} ($\approx 1 \times 10^{12}$) secret keys to find one which generates that keystream. Searching a keyspace of this size in software is well within the capabilities of ordinary commodity processors. Table 1 shows several processors and the number of millions of keystream/s that each processor core is capable of searching.

Table 1. Performance of ADP exhaustive key search

Processor	Clock Speed GHz	Cores/CPU	VC1 keys/s $\times 10^6$
Intel Core 2 Duo	1.2	2	.270
Intel Core 2 Duo	2.2	2	.475
Intel Core i7	2.6	2	.632
AMD Opteron 252	2.6	2	.375

Each of the processors identified uses an optimized RC4 implementation and generates sufficient keystream to decode the VC1 of the initial LDU1 frame. On

a single core of a dual-core Intel i7 processor the search will take, on average, 10.6 days. The search time is inversely proportional to the total number of CPU cores used to conduct the search. An alternative approach is to make use of the computational capability of commonly-available GPUs which use a *single instruction/multiple data* (SIMD) architecture and can process many threads in parallel. We have investigated RC4 on GPUs and the best results improve on the performance of CPU implementations by a factor of between 3 and 5 which, allowing for hardware differences, are in general agreement with those of Li *et al.* [13]. Although this is a significant improvement these performance figures represent an extremely disappointing result and fall a long way short of the capability of the hardware. The problems in performance are explained principally by the very low occupation of the GPU by the RC4 implementation. The implementation is making use of just 6% of the available computational resources but is constrained by the memory limitations of the GPU device. RC4 implementations that are able to make more effective use of the GPU's computational resources have the potential to be much faster. Alternatively, an FPGA implementation of the RC4 cipher running on the Cube FPGA cluster can search the entire 40 bit key space in just three minutes [14]. This implementation is approximately four times faster per FPGA core than the same search running on a single CPU core.

Operational Responses to Exhaustive Key Search A response to the threat of exhaustive key search adopted by some operators is to change the encryption keys relatively frequently. This reduces the time available for the adversary to search the keyspace, increases the amount of searching they must do and limits the amount of traffic that may be disclosed once the key is compromised. The P25 Over-The-Air-Rekeying (OTAR) protocol simplifies the key management process and allows MR equipment to be rapidly re-keyed.

Unfortunately, when using a weak cipher such as DES frequent re-keying does not significantly increase the work for the adversary. Even if an adversary can search only a small percentage of the keyspace they are likely to discover the key within a reasonable time as long as they re-start the search every time the key is changed because there is a uniform probability of picking a key within the adversary's search space. If $P(e)$ be the probability of choosing a key outside of the search space of the adversary then the probability $P(d)$ of picking a key within the adversary's search space after n re-keying attempts is given by:

$$P(d) = 1 - P(e)^n \tag{1}$$

The problem for the adversary is that searching in this way is not guaranteed to discover the key whereas searching the whole key space is. This suggests the adversary is better off storing all rekeying messages and decrypting them in turn once the original key is discovered. Unfortunately, enough of the rekeying packet is sent in the plain to allow them to be identified and stored — permitting complete decryption once the original key has been found.

A final warning relates to the use of OTAR with weak encryption keys. An adversary that can store OTAR frames can use the subsequent discovery of a

TEK to provide a known plaintext and then repeat the search to recover the KEK. Once in possession of the KEK they will be able not just to monitor all traffic but to re-program the encryption keys used throughout the network.

6 Related Work

Clark *et al.* have also conducted an analysis of the security weaknesses present in the P25 protocol [15]. They identify the lack of authentication on voice and most other types of data traffic as being a significant problem, propose a novel attack against location privacy that can be used to locate a radio even when its user is not actively using the radio and discuss a physical layer jamming technique that can be used to perform denial-of-service attacks. These are all significant problems and largely complimentary in nature to those discussed here. The investigation of Clark *et al.* also makes use of the same SDR software that is presented in this paper as the basis of their investigation. The independent application of the SDR software demonstrates the utility of the approach when applied to the critical security analysis of wireless networks.

Another closely-related body of work is that of Project 54 conducted by the University of New Hampshire [16]. The focus of Project 54 is on in-car human computer interaction to provide police cruisers with an integrated environment in which communications Project 54 has implemented a P25 base station by pairing a PC with a conventional radio transceiver [17]. Ramsey *et al.* implemented this data transmitter for P25 using a conventional radio transceiver. The baseband signal is captured from the radio transceiver using the PC soundcard and the remaining signal processing stages are performed in software [18].

The RC4 cipher as used in ADP is also the basis for the flawed *Wired Equivalent Privacy* (WEP) used in IEEE 802.11. WEP does not correctly use the RC4 cipher and is subject to the weak-key attack of Fluhrer *et al.* [19] and Mantin [20]. The contrast with ADP is quite marked because ADP avoids the mistakes in the use of RC4 that were made by the designers of WEP. In other respects the P25 security protocol has similar weaknesses to the WEP flaws described by Borisov *et al.* [21]: the access control and authentication mechanism that is trivially by-passed, there are no guarantees of message freshness and the integrity controls are insufficient to protect against deliberate damage.

7 Conclusions

P25 radio systems are more secure than conventional analogue radio systems but not nearly as secure as the term "encrypted" would imply. The most serious security flaw in P25 is the optional nature of the security protocol, however even when the security protocol is used several serious security flaws present the design of P25 cryptographic protections, remain:

– Weak encryption permits an attacker to recover the encryption key, and frequent re-keying is not an effective defence.

– There is no effective authentication and access control mechanism.
– The lack of a key hierarchy means that a single key is used to encrypt traffic between many users over many sessions.
– The integrity, authenticity and freshness of traffic cannot be ensured even when the security protocol is in use.
– Serious denial-of-service threats against individual stations are possible.

The contribution of this paper is in several parts: firstly, we have applied the techniques of software-defined radio to enable the study and network security analysis. This approach has the potential to expose network traffic at all layers of the protocol stack. Secondly we have identified a number of serious security flaws that are present in the P25 protocol and described attacks which exploit them.

8 Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. *Project 25 FDMA Common Air Interface Description*. Number TIA-102.BAAA-A. Telecommunications Industry Association, 2500 Wilson Boulevard, Arlington, VA 22201, USA, September 2003.
2. GNU Radio. Project website. `http://www.gnuradio.org`.
3. Ettus research llc,. Company website `http://www.ettus.com`.
4. Stephen Glass, Vallipuram Muthukkumarasamy, and Marius Portmann. A software-defined radio receiver for APCO Project 25 signals. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 67–72, New York, NY, USA, June 2009. ACM.
5. *Project 25 — Digital Land Mobile Radio — Link Layer Authentication*. Number TIA-102.AACE. Telecommunications Industry Association, 2500 Wilson Boulevard, Arlington, VA 22201, USA, December 2005.
6. *Project 25 Over-The-Air-Rekeying(OTAR) Operational Description*. Number TIA-102.AACB. Telecommunications Industry Association, 2500 Wilson Boulevard, Arlington, VA 22201, USA, January 2002.
7. *Project 25 DES Encryption Protocol*. Number TIA/EIA-102.AAAA-A. Telecommunications Industry Association, 2500 Wilson Boulevard, Arlington, VA 22201, USA, 2001.
8. Mike Loukides and John Gilmore. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998. Available online at `http://cryptome.org/cracking-des/cracking-des.htm`.

9. Gaël Rouvroy, François-Xavier Standaert, Jean-Jacques Quisquater, and Jean-Didier Legat. Design strategies and modified descriptions to optimize cipher FPGA implementations: Fast and compact results for DES and Triple-DES. In Peter Y. K. Cheung and George Constantinides, editors, *Field Programmable Logic and Application*, volume 2778 of *Lecture Notes in Computer Science*, pages 181–193. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-45234-8_19.
10. Gäel. Rouvroy, François-Xavier Standaert, Jean-Jacques Quisquater, and Jean-Didier Legat. Efficient uses of FPGAs for implementations of DES and its experimental linear cryptanalysis. *IEEE Transactions on Computers*, 52(4):473–482, April 2003.
11. Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking ciphers with COPACOBANA — a cost-optimized parallel code breaker. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, LNCS, pages 101–118. Springer Berlin/Heidelberg, October 2006.
12. *Project 25 Vocoder Description.* Number ANSI/TIA/EIA-102.BABA-1998. Telecommunications Industry Association, 2500 Wilson Boulevard, Arlington, VA 22201, USA, May 1998.
13. Changxin Li, Hongwei Wu, Shifeng Chen, Xiaochao Li, and Donghui Guo. Efficient implementation for MD5-RC4 encryption using GPU with CUDA. In *3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication (ASID 2009)*, pages 167–170, August 2009.
14. Oskar Mencer, Kuen Hung Tsoi, Stephen Craimer, Timothy Todman, Wayne Luk, Ming Yee Wong, and Philip Heng Wai Leong. Cube: A 512-FPGA cluster. In *5th Southern Conference on Programmable Logic*, (SPL 2009), pages 51–57, April 2009.
15. Sandy Clark, Perry Metzger, Zachary Wasserman, Kevin Xu, and Matthew A. Blaze. Security weaknesses in the APCO Project 25 two-way radio system. Technical Report MS-CIS-10-34, University of Pennsylvania, 2010. http://repository.upenn.edu/cis_reports/944.
16. Project 54. Project website `http://project54.unh.edu`.
17. Andrew L. Kun, W. Thomas Miller III, and William H. Lenharth. Computers in police cruisers. *IEEE Pervasive Computing*, 3(4):34–41, October–December 2004.
18. Eric R. Ramsey, W. Thomas Miller III, and Andrew L. Kun. A software-based implementation of an APCO Project 25 compliant packet data transmitter. In *2008 IEEE International Conference on Technologies for Homeland Security*, Boston, MA, 12–13 May 2008. Institute of Electrical and Electronics Engineers.
19. Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 1–24, London, UK, 2001. Springer-Verlag.
20. Itsik Mantin. A practical attack on the fixed RC4 in the WEP mode. In *ASIACRYPT*, pages 395–411, 2005.
21. Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the 7th Annual International Mobile Computing and Networking Conference*, pages 180–189, New York, NY, USA, 2001. ACM SIGMOBILE, ACM Press.