Encryption for POCSAG and Flex Paging Systems

Document Number 100-1034. Revision B
© 2009-2014 Infostream Pty Ltd. All rights reserved

Authors

<table>
<thead>
<tr>
<th>Paul Schlusser</th>
<th>Technical Director</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slav Zinger</td>
<td>RF design and X5 product engineer</td>
</tr>
<tr>
<td>Rod Skinner</td>
<td>Cypher software development manager</td>
</tr>
</tbody>
</table>

Disclaimer

This paper constitutes the opinion of the author on its subject matter. Whilst all care has been taken in its preparation it is not intended as technical advice or guidance for any specific scenario or use. Readers should obtain technical advice and assistance before adopting or pursuing any solutions referred to in this paper. Neither the author nor Infostream Pty Limited will accept any liability of any nature for any claim or loss arising from or out of any reliance upon the contents of this paper. Neither the paper as a whole, nor any part of it, may be reproduced for commercial purposes in any form without the express consent of the author and Infostream Pty Limited.
1 Introduction
   1.1 Why encrypt? 3
   1.2 Effect on pager performance 4

2 Definition of the problem 5
   2.1 Assumptions 5
   2.2 Security threats addressed 5
   2.3 Security threats not addressed 5
   2.4 System diagram 6

3 Encryption algorithm 7
   3.1 Block ciphers 7
   3.2 Stream cipher 8
   3.3 Nonces 8

4 Cypher and X5 encryption 12
   4.1 Encrypted transmission 12
   4.2 Encrypted message reception 13
   4.3 Encryption keys 13
   4.4 Message encryption 14
   4.5 User interface security 15
   4.6 Limitations 15
   4.7 Security considerations 16

5 Facts and fallacies 19

6 Conclusions 20

7 Appendix 21
   7.1 References 21
   7.2 Glossary of terms 21
1 INTRODUCTION

This paper discusses the problem of transmitting messages securely over a paging network and how it can be solved by a correctly implemented encryption system. It also examines and discusses the encryption system used by Infostream products (Cypher 3 simulcast paging system and the X5 pager).

1.1 Why encrypt?

There are many good reasons to encrypt paging messages. The POCSAG and Flex protocols (the only two substantially deployed public paging protocols worldwide) are well known and there are several readily available off-air decoder software packages that enable the protocols to be decoded by anyone with a readily available commercial receiver and a personal computer and a minimal set of technical skills. Several websites exist in many countries that display in real-time, the paging traffic from various public and private paging systems, including systems utilized for public safety. Neither the Flex or POCSAG paging protocols themselves offer anything other than a minor inconvenience to someone wishing to eavesdrop on those networks, and no inconvenience at all where someone has already gone to the trouble of establishing a eavesdropping website.

1.1.1 Protecting operationally sensitive information

The more obvious or one of the first reasons that comes to mind regarding encryption is to protect operationally sensitive information. If unintended recipients can read paging data, that data can be used to prejudice the outcome of the operation. Criminals can evade detection and prosecution and member of the press and/or public can interfere with the sensitive emergency operations, possibly prejudicing the outcomes of those operations and public safety generally.

1.1.2 Disclosure of individuals’ confidential information

Less obviously, but very importantly, there are many cases where pagers are used in such a way as to transmit personal information to the intended recipient but which should not be received by anyone else. Routinely, ambulance officers utilizing paging would like to have information regarding the patients condition as part of the paging message. Transmitting such information however, seriously compromises the patient’s privacy. Indeed, even if the condition of the patient is not transmitted (which is a common strategy), of necessity, the location of the patient generally is transmitted. In today’s connected world, it is no effort at all to connect an address with an individual and discover something about that individual which ought not be made public. In many jurisdictions, there are legal requirements to ensure that patient confidentiality is maintained and these requirements are at odds with the use of unencrypted pagers.

1.1.3 Infostream encryption

The VIPER and Cypher 3 paging encryption system provides a unique and strong encryption system based on sound cryptographic principles and is extremely efficient on airtime and robust against corruption.

The encryption system does not require the transmission of cryptographic nonces, which would otherwise increase the length of the transmitted message and therefore reduce the network capacity. Each original character in the original (unencrypted) message requires only
a single character to be transmitted to air, maintaining network capacity, even after encryption is implemented.

Apart from selecting capcodes to be encrypted, the encryption system is completely transparent to both the operator of the VIPER system and the pager. Key selection is handled entirely automatically by the VIPER and Cypher system, as is the OTA programming of the keys and capcodes.

The encryption system is publicly disclosed so that the effectiveness of its security can be verified. The encryption of the messages does not rely on secrets about the encryption system, other than the secret keys.

1.2 Effect on pager performance

It is a common misconception that adding encryption to a paging system necessarily either:

- Increases the message size leading to lower efficiency and bandwidth or
- Increases the likelihood of receiving a corrupted message.

As with many things, there are some grains of truth behind both these misconceptions, but as this document will show, a properly designed and implemented encryption system will have no effect on either the length of the encrypted message or the number of errors in the received message.

Certainly there are commercially deployed examples of pager encryption that do lead to both these problems to varying degrees, but these implementations are flawed and sub-optimal. Increased message size and sensitivity to message corruption are definitely not inevitable outcomes of the encryption process but may be the result of a poorly designed one.
2 DEFINITION OF THE PROBLEM

2.1 Assumptions

The following assumptions are made regarding the encryption techniques and systems discussed in this whitepaper.

- An attacker may be able to read individual messages from a pager by looking at the screen.
- All message data transmitted to the pager is easy to intercept and decode at the paging protocol level.
- The implementation has access to the Flex/POCSAG encoder.
- The solution must be capable of being implemented on a wide range of paging receivers.
- The method of operation of the encryption system is publicly disclosed for scrutiny and relies solely on the secrecy of the encryption keys that are programmed by the system operator for its security.
- It should not be necessary to recall all pagers in the event that one or several pagers that share encryption keys are lost.
- A set of encryption keys can be securely programmed into the pager and they cannot be easily read back out.
- The pager’s clock can be synchronized to the paging network clock to an accuracy of less than one second.

2.2 Security threats addressed

The following threats/issues are discussed and solutions are proposed to address them through the encryption system described in this whitepaper.

- Paging message can be eavesdropped using standard receivers or another standard pager if they are not encrypted.
- Pager may be lost. It may take up to a week until lost / stolen pager is reported.
- Message can be jammed and then later replayed to the pager(s) to generate false message repeats at a later time.

2.3 Security threats not addressed

Any encryption system can only address a defined set of specific threats. Some threats are completely outside of the scope of the system to control. The following list indicates some of the obvious threats that are outside the scope of discussion in this paper.

- Message may be eavesdropped when displayed on the LCD of the pager.
- It is possible to send a fraudulent message to a pager from outside of the system through the normal message sending mechanisms. (No validation of message origin). Note however that many solutions already exist to solve this problem.
- Pager may be stolen or “temporarily stolen” to retrieve previously received messages.
• Pager may be unknowingly stolen that then can be used to eavesdrop group or personal messages.

• Pager terminals may be attacked to retrieve security keys and to eavesdrop on outgoing messages or to read messages before they are encrypted.

• Denial of Service (DoS) attacks (Jamming etc)

• Insider attacks. The attacker gains unauthorized access to the system or is granted access that is then used to launch an attack.

• Network attacks on paging system itself (Computer viruses etc)

• Sensitive operational information may be obtained by observing un-encrypted parts of the messages (e.g. address/capcode of the message and the time it was sent).

2.4 System diagram

The following system diagram illustrates the entire end-to-end message chain for an encrypted paging system. Although the encryption systems described in this whitepaper deal only with the paging terminal to pager communications, even simple web-based message entry can be made completely secure using readily available security protocols such as HTTPS.
3 ENCRYPTION ALGORITHM

There are two main types of encryption algorithms:

- **Block Ciphers** that encrypt/decrypt an entire block of data in a single action. Block ciphers are sensitive to bit errors in the cipher text and a bit error may result in corruption of a whole block of information. There are many commonly used block ciphers including DES, IDEA, RC5, AES and Blowfish. Some of these ciphers have been standardized and have been well tested by the cryptographic community.

- **Stream Ciphers** which encrypt/decrypt a stream of data “on the fly”. Stream Ciphers are not as sensitive to bit errors in transmission of cipher text and a single error will result in only a single symbol corruption. Stream ciphers are less common and one of the few that are standardized is the proprietary RC4 (as used in Wi-Fi and SSH).

3.1 Block ciphers

Block ciphers work by dividing the original message (plain text) into blocks (typically 128 bits) and applying an encryption process to the block. The encryption algorithm uses an encryption key known to both the sender and receiver of the message so that the receiver can reverse the encryption process performed on the block. In addition to the key, each encoded block uses an initialization vector that is unique to the block to prevent identical segments of plain text encoding to the same cipher text. The initialization vector is typically an incrementing counter, although more complex systems are also used. The method by which the initialization vector is generated for each block is known as the block cipher mode.

Block ciphers by themselves are not suitable for pager encryption because bit errors in the received cipher-text will destroy the entire block of plain text, dramatically amplifying the effect of the error. If a pager encryption system were deployed using block ciphers, it would have a devastating effect on the paging receiver performance.

For this reason alone, block ciphers are not considered further in the context of pager encryption, except in the context of implementing a key-stream for a stream cipher.

Block cipher can be used to generate a stream cipher key stream and thus any of the standardized and highly secure block ciphers can be used to make an effective key-stream that is suitable for paging applications.
3.2 Stream cipher

Stream ciphers operate by combining a key stream with the plain text using a mathematical function such as modulo addition or exclusive “or” (for binary data). One useful characteristic of the stream cipher is that errors in the cipher text are not amplified in the recovered plain text. I.e. a single bit error in the received data will result in a single bit error in the decoded message.

Because of the way both Flex and POCSAG protocols are designed, errors only arise in the received message by alteration of the values of bits within a message and not by removal or addition of bits (except in the case of POCSAG where bits may be incorrectly added to or trimmed from the end – but only at the end). As a result, there are no synchronization problems during the message decoding. Each received bit will be correctly matched with the corresponding bit in the key-stream even for a message received with errors.

There are two common ways to generate a key-stream:

- Linear feedback shift register or
- Via a block cipher.

The linear feedback shift register approach is the simplest but there are few standardized algorithms available and fewer still that are considered very secure. The most common is known as RC4. RC4, while deployed in a number of other applications, has been proven to be somewhat unsafe when used with weak keys. Additionally, RC4 is a trademark of RSA and requires a license to use.

An alternative is the Sober-128 stream cipher developed by Qualcomm. Sober-128 is a relatively simple algorithm to implement that provides a high security level. Although there are known weaknesses in the Sober 128 system relating to message authentication, those weaknesses do not manifest in the application anticipated here.

3.3 Nonces

On the face of it, a good stream cipher and a strong encryption key would appear to make an effective encryption system for paging applications. However there are a number of weaknesses in this simple approach including replay attacks and key-stream extraction.

3.3.1 Replay attacks

A replay attack involves recording of the original message (over the air, which is simple to do), and then re-transmitting the same message to another pager sharing the encryption key or to the same pager at a later time. Since the recorded message was encoded using a valid key, providing the key is not changed in between, the pager would correctly decode the message. Although this attack cannot change the message content, multiple messages...
fragments might be combined so as produce new message content that coincidentally might appear to be a legitimate message.

In the paging context, if the attacker knew the structure of the original messages, and messages were basically similar in structure from message to message, this would be a feasible way of injecting false messages into a pager that could be assumed to be from a legitimate source.

3.3.2 Key-stream extraction

Another weakness of a simple stream cipher system is the ability to extract the key stream from a message if the plain text of the message is known. For example if the attacker were to observe (on the pager screen) the plaintext of a single message sent to a capcode and was also in possession of the cipher stream (received covertly over the air), the key stream can be obtained by performing a reversal of the encryption operation. What remains would be the original key-stream. If the key-stream doesn’t change between messages, all subsequent messages encrypted with the same key could then be decoded.

3.3.3 Application of a nonce

To eliminate the above two problems, the key-stream must be altered for each message by use of a nonce. A nonce is a cryptographic term that refers to a value that is used only once per message. In use, the nonce is combined with the encryption key to provide a unique seed value to the encryption algorithm. Because the nonce is unique to the message, key-stream extraction is of no use to in decoding different messages from the one that was used to extract the stream because the stream changes for each subsequent message. Also, if the pager only allows the use of a nonce once for the purpose of decoding the message, replay attacks are thwarted.

The nonce is only one part of the encryption seed (the key being the other part) and so knowing the nonce for a given message does not enable the attacker to learn the secret key, nor does the attacker gain the ability to create a new valid key-stream. Therefore the nonce can be publicly known for each message without weakening the security of the encryption scheme.

One commonly used approach is to transmit the nonce in clear text along with the message. This has the drawback (in the paging context) of lengthening the message and also making the message susceptible to received errors in the nonce. If the bits of the nonce become corrupted during message transmission and reception, the entire message would be lost.

Note: It is critical that the nonce be combined with the key to generate the cipher seed value for each message. If the nonce and key are used independently, knowing the nonce will allow the attacker to extract the key-stream.

At least one commercially available system incorrectly uses the nonce to generate the data input for the otherwise very strong encryption system and attempts to encrypt or obscure the nonce before transmission. However in that system, since the nonce is transmitted over the air and very weakly encoded, it is relatively easy to extract it, and thus unlock the encryption without having to learn the secret key thus rendering the otherwise strong encryption system nearly useless. Also, since the nonce is transmitted over the air, it can be corrupted and the entire message would be lost in that case. These features (having to transmit the nonce in addition to the message and having the message very susceptible to errors in the nonce) may also feed the false belief that encryption makes the paging messages longer and more likely to be corrupted. (Which is true for that flawed approach).
3.3.4 Generating the nonce

The only requirement for the nonce is that it is unique for each message and agreed between the sender and receiver. As discussed above, it can be transmitted in the clear (with the previously noted problems), or agreed between the sender and receiver by some other means.

3.3.4.1 Timestamp based nonce

The most obvious source of a nonce is a timestamp, which, provided the time cannot be altered on the pager side is unique for all time, and the same for both the sender and receiver (provided the clocks are synchronized).

Nonetheless, because of the need to round the real-time clock to some integer value to arrive at a nonce, even for closely synchronized clocks, there is a finite probability that the pager will select a different rounded value to the paging terminal that encrypted the message. To overcome this problem, both alternative nonces much be tried and an independent mechanism must then be used to determine which is the correct outcome. One possible mechanism is to include a short known value in the message that will only decode correctly in one case. This approach increases the message length (slightly). There are other better approaches however and the alternatives are used for the X5.

The above only presents a problem in the POCSAG case, because of its relatively simple message structure. In the case of Flex encryption, a message sequence number and checksum is always included in the encrypted message, and that can be used to disambiguate the nonce.

3.3.4.2 Rolling or incrementing value nonce

Another approach is to use a rolling or incrementing number for the nonce (or a hash of that number) that is incremented each time a nonce is required. This has the advantage over the timestamp-based nonce in that it doesn’t rely on synchronized clocks.

The rolling nonce approach has the problem that if the pager misses a message, the sender and the pager can get out of sync with each other requiring some kind of synchronization mechanism between them. A typical and practical way to do this is to compare the message decoded with the next few nonce values until a valid message is found. Providing the decoding is not too computationally intensive even a few dozen combinations can be tried to take account of potentially a few dozen missed messages since the last valid message. The additional step required to achieve this is to determine which is the valid message and that can be determined by examining the resulting decodes.

Many wireless remote access systems (key fobs for cars for example) use rolling codes to guard against replay attacks. Sometimes the remote control and the receiver can get out of sync and a manual synchronization process must be used to reset the synchronization. A similar approach can be used with the pager by either having the user enter a value that is generated from the paging message encryption system (say, as read out over the telephone) or having the encryption system send a special message to the pager to reset the sequence.

The advantage of a rolling nonce is that it does not rely on the time synchronization between the pager and the encoding terminal. Thus it is possible to encrypt the message at its origin rather than at the point of encoding. However this also requires that the message source is the ONLY source of messages for the encrypted capcode or else the pager will get out of sync with the encryption system. Furthermore, this approach means that, unless special steps are taken, the cipher-text will include unprintable control characters that may not transmit well over protocols used to send them to the paging terminal.
Finally, encoding at the message source requires that the message source have access to the encryption keys and a means to encode the message. Most paging entry programs do not have encryption capabilities.

### 3.3.4.3 Transmitting the nonce

The third approach to the nonce problem is to simply transmit it unencrypted along with the message. Since a typical pager may receive several hundred or thousand messages between being reprogrammed with new keys, the transmitted nonce must be sufficiently (numerically) large so that it is not reproduced twice during the lifetime of the keys. Two seven-bit printable ASCII characters provide about ten thousand possible nonce values (excluding control characters). However, as discussed above, the nonce is susceptible to errors that would cause the loss of the entire message. For robustness, an additional copy (or copies) of the nonce should be transmitted to ensure that it could be recovered even during bad reception conditions.

Transmitting the nonce, while simple and versatile and requiring no special support on the part of the paging terminal does make the message more susceptible to loss through receiving errors and does also increase the length of the message. For those two reasons, it is not the preferred approach.

---

It is important to understand that although the nonce is transmitted in the clear, knowledge of the nonce does not enable an attacker to recover the message without access to the secret key. Furthermore, even though the nonce is easily guessable (a fast computer can try all ten thousand possibilities in an instant), without the key, no decoding is possible. The nonce merely serves to ensure that knowing one message (by overlooking a pager screen) does not provide information about how to decrypt any other message using the same key.
4 CYPHER AND X5 ENCRYPTION

The following is an explanation of the system employed by the Infostream Cypher 3 encoder and X5 pager to implement a highly effective and secure encryption system based on the principles described above. With slight variation, the same basic encryption system is used for both the Flex and POCSAG versions of the X5 pager and Cypher 3 encoder.

The following diagram illustrates the basic approach employed by the Cypher 3 and X5 paging system. The description of the encrypted transmission and reception process follows.

**Overview of encryption system**

4.1 Encrypted transmission

The original message (and capcode) is converted into POCSAG blocks and BCH error correction is applied. Based on the capcode of the message, an encryption key is selected and a nonce is generated from the real-time clock to seed the AES block cipher. An incrementing counter is encrypted block-by-block and the resulting output is used as a key-stream to stream encipher the BCH encoded POCSAG code words (excluding the capcode) via an exclusive-or function. Finally, the encoded stream is formatted into the POCSAG stream along with the necessary preamble, sync and other code words that make up the POCSAG code structure.
Note that Cypher encryption system, the encryption takes place AFTER the BCH error correction is applied. This provides a mechanism for the receiver to detect a successful decipherment. It also avoids the problem of converting printable ASCII characters (that make up the message) into unprintable control characters that can interact with other protocols used to send messages into the paging terminal.

4.2 Encrypted message reception

The reception process starts with the recovery of the POCSAG data and detection of a capcode and message payload for the receiving pager. The non-error-corrected blocks are stored in the pager database, along with the timestamp of the message as it is received. That timestamp is later used to recreate the nonce.

If repeat message suppression is enabled, additional copies of the message are captured and stored in the pager database that are then used to detect which bits are different between the message copies and thus most likely to be in error. This provides “hinting” information to the BCH decoder to improve its error correction performance.

At the time of message display, the encryption key is determined from the capcode and it is combined with the time-stamp generated nonce to seed an identical AES block cipher that recreates the key-stream used by the encoder. If rounding of the timestamp is ambiguous (close to decision time for rounding), two alternative nonces are generated and the deciphering occurs twice. Only one version of the deciphering will result in valid BCH error correction – the other will appear like random noise and the BCH decoder will fail enabling the pager to determine the correct nonce and thus the correct message for display.

4.3 Encryption keys

The X5 pager is programmed with up to thirty-two (32) individual encryption keys of one hundred twenty eight (128) bits in length (32 hexadecimal numbers).

Each capcode/function code programmed into the pager can be assigned to utilize one of the available 32 keys at the time of pager programming. If more than thirty-two (32) capcodes are to be encrypted, the same key can be used across multiple capcodes.

Note that security is not enhanced by the use of multiple keys but providing multiple keys provides a mechanism for partitioning groups of pagers so that the compromise of an individual pager does not compromise all others.

A single capcode is assigned uniquely to the pager to enable re-programming of the encryption keys securely over the air. The capcode used for OTA programming should not be assigned for any other purpose.

Encryption keys are unique to the capcode. It is not possible to send encrypted messages to unencrypted capcodes, or vice versa in the POCSAG encryption system. The Flex system includes a secure message vector that is used to send encrypted messages. Thus in the Flex version of the X5, both encrypted and unencrypted messages can be sent to the same capcode.

4.3.1 Key security

All encryption keys, whether pre-programmed or received over the air are stored in the MPU integrated FLASH memory of the pager and cannot be read-out of the MPU without destroying it in the process owing to the built in security features of the MPU. Pagers cannot be intercepted, read out and returned undetected.
4.3.2 OTA programming of keys

Encryption keys can be reprogrammed over the air by sending an encrypted message containing an individual or a set of new keys to the pager on the capcode reserved for OTA programming. The programming of new keys is accomplished via an encrypted message using the encryption key unique to the pager. Thus, only a known secure key is used to program new keys over the air. For security reasons, OTA programming is never performed using a key shared with several pagers.

OTA key programming messages include a message checksum to verify the integrity of the OTA message. The OTA message is an “all or nothing” message, which ensures that if the user receives a notification of OTA programming, the pager is guaranteed to have received all new keys correctly.

4.3.3 Secure OTA group programming

If OTA keys are to be programmed to multiple pagers, a variation of the Infostream dynamic group call (a proprietary POCSAG extension feature) can be used. In the secure group call feature, an encrypted message header is transmitted that includes the dynamic group call vector (as per standard dynamic group call) plus a unique one-time key used to encrypt the message body (the set of new keys).

A dynamic group call header is sent to each pager requiring the new set of keys, each of which uses it’s own unique key to decrypt the header and thus the encryption key for the group-programming message. Only pagers that have successfully decoded the group call header will be able to decrypt the secure OTA key-programming message.

4.3.4 Viper key management integration

Where the Viper system is used to allocate and manage the encryption keys for the pagers, the process of key reprogramming in the event of a lost pager is handled entirely automatically. Viper will calculate the optimum OTA programming sequences required to reset all compromised keys based on the serial numbers of the compromised pagers.

4.4 Message encryption

An encrypted message is transmitted to the pager after being stream enciphered via the AES128 algorithm seeded with cryptographic hash generated from the encryption key specific to the capcode and a nonce. The hash chain provides an infinite set of one-time keys used to seed the block cipher. The key stream is generated by encoding an incrementing counter for successive blocks according to a method described as the block cipher CTR mode.

4.4.1 Nonce

The nonce is formed from the UTC second timestamp of the start of the message. Synchronization of the pager timestamp is performed automatically over the air to a resolution of less than 1 second using Infostream’s proprietary POCSAG OTA protocol. Flex includes a standard way of setting the pager time accurately.

Once the time is set for the pager, it is not possible to change the time (other than to track the time accurately to the network to account for drift) by resending false time signals to the pager. The pager will ignore any discontinuities in the time broadcast messages unless the user specifically resets the time synchronization system manually. Summer time (daylight saving) shifts do not affect the nonce, which is generated from a UTC timestamp and thus unaffected by time zone.
4.4.2 Nonce ambiguity resolution

4.4.2.1 POCSAG
The Infostream X5 POCSAG decoder stores the received messages in un-error corrected and un-decoded form. Ambiguity regarding the nonce arising from rounding errors to generate the nonce is resolved by examining the success of the BCH error correction for each alternative decode. The version of the message with the lowest reported bit errors is considered to be the correct version.

Note. Because of the way the stream cipher operates, it can be applied before BCH encoding at both the transmitting and receiving side. With 10 bits of parity data per codeword, for each additional codeword the likelihood of selecting the correct version of the decode increases at approximately one thousand (or slightly less if the message contains some bit errors).

4.4.2.2 Flex
In the Flex pager, the secure message vector used to send encrypted messages includes a checksum and message sequence number that is used to determine which version of the decoded message is the correct one.

4.5 User interface security
The X5 includes a PIN lock function that can be programmed to activate either on command (over the air) or automatically after a specified timeout. A pager that is lost and which then goes into PIN lock cannot be used for long to intercept encrypted messages. The screen lock and also be manually activated.

The user must enter a four-digit pin in order to unlock the pager and thus read any (encrypted) messages. Only 10 attempts in total are allowed before a successful unlocking or else the pager will enter a permanently locked state that requires reprogramming to unlock.

4.6 Limitations
There are technically feasible methods to read the encryption keys from the Flash memory of the X5 pager although the process is not trivial. The microprocessor could feasibly be removed from the pager and replaced with a duplicate (from another X5 pager) and then reprogrammed with the keys once they are extracted from the original pager.

The received messages in the pager can of course be read out from the pager using the user interface if the PIN lock is broken or has not been set. It is not possible to read the messages stored on the pager from the pager RAM because they are encrypted until they need to be
displayed. The keys must be extracted from the CPU in order to access the messages, significantly complicating the task.

4.7 Security considerations

There are several possible types of threats to security of any encryption system. The relevant ones are discussed here.

4.7.1 Public disclosure

It is important to note that a key principle of cryptography is that the encryption system itself is publicly disclosed. Other than the secret keys, it must be assumed that a potential attacker knows everything else about the encryption system. The X5 and Cypher 3 encryption system does not in any way rely on the secrecy of the algorithm for its security. It is assumed that all traffic sent over the paging system can be trivially decoded.

By making the security system public, it is subject to independent scrutiny and verification.

4.7.2 Security of encryption system (AES128)

The underlying encryption method employed by this system (AES128) is, as of the time of writing, regarded to be secure against any feasible attack. The security of this algorithm is covered elsewhere and will not be discussed further.

4.7.3 Loss of pagers

4.7.3.1 Using the lost pager to read messages

The loss of a pager is the most obvious and probably by far the most common security weakness. Once a pager is lost it could fall into the hands of a malicious agent and all security is lost until the loss of the pager is detected. The X5 pager features a PIN lock that can be used to at least partially guard against the security breach but in practice it is impossible to ensure that the pager is locked at the point at which it is lost.

Pagers can be remotely locked however, once the loss is detected to mitigate this threat.

For additional security, an intelligent charger (such as the Infostream X5 5 or 10 way gang charger) can be used to interrogate each pager on a regular basis. Using the ESN and secret key of the pager, the VIPER system can verify that all pagers are regularly detected and any missing pagers will be automatically flagged as a security threat.

4.7.3.2 Reading keys from the lost pager

Assuming the lost pager falls into the hands of a highly skilful and highly resourced malicious agency, the security of the system must still not be compromised.

All encryption keys are programmed into the integrated FLASH of the pager MPU. The MPU is itself protected via it’s own hardware systems against being read-out. The only feasible means of reading the keys out of the pager would be to (by some means) gain access to the internal connections of the MPU – a process that would by definition, destroy the MPU. Assuming that this could feasibly be done, it would require considerably time and resources and would be extremely difficult to do in such a way that the pager could be returned undetected to the original owner. (i.e. it would take considerable time).
4.7.4 System interconnections

Encryption of the messages occurs within the VIPER system itself. Therefore any traffic sent to VIPER, or access to VIPER screens that display messages would represent a breach of system security.

It is assumed that measure exist to prevent eavesdropping on the VIPER screens or on the message input links that are necessarily un-encrypted (although the links can operated over secure channels).

VIPER stores a complete set of encryption keys for each pager in the VIPER database. The keys are themselves encrypted using a password-generated master key that must be entered to enable the encryption system. Access to the master password would compromise the keys.

4.7.5 Access to servers

With respect to access to servers, there are two possible methods of attack that are broadly speaking, malicious operation of the system and authorized access to secure data. Again, it is assumed that access to the algorithms of the VIPER system, including the source code is available already to a malicious agent.

4.7.5.1 Malicious operation

Malicious operation involves an operator performing operations on the VIPER system in such a way that does not fundamentally compromise the system but which is not a proper use of the system. Examples include sending messages to encrypted capcodes.

4.7.5.1.1 Sending fake messages

A malicious agent with access to the Viper system naturally has access to all sent messages and the ability to send new encrypted messages. Good operational process is required including the use of strong passwords, proper installation security etc.

4.7.5.1.2 Real-time clock manipulation

It is important that no operator be able to adjust the time of the VIPER system. If the real time clock (RTC) of the VIPER system could be changed by a malicious agent, earlier or future hash values could be generated by the system that might be used to intercept future or past messages.

Proper administration policy will ensure that access to the RTC functions of the VIPER operation system is a highly secure operation. Security of this process should be treated with the same importance and security of the encryption keys themselves.

4.7.5.2 Access to data

Because VIPER automatically (and silently) generates all of the encryption keys, there is no user interface way of reading out the encryption keys.

Keys are stored in the VIPER system in an encrypted fashion, and decoded on the fly by a master security key. For added security, this key can be maintained in system RAM and would therefore have to be entered on system power up. Without this RAM key, the keys stored in the database would not be decipherable. However it does require that the security password must be known by at least one available operator at system boot up time.
This can be handled using a tamper evident envelope, which is opened in such a situation. As soon as practicable, the password would be replaced and the keys automatically re encoded with the new password and a new envelope filled and sealed.

Reading out the encryption keys from the database, without knowledge of the master encryption keys would not compromise message security.
### 5 FACTS AND FALLACIES

The following table addresses some of the common assumptions and fallacies that the authors are aware of in the paging industry in relation to encryption. They are summarized and answered below for easy reference.

<table>
<thead>
<tr>
<th>FACTS AND FALLACIES</th>
<th>ANSWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption increases the size of the transmitted message.</td>
<td>False. A stream cipher requires no padding to operate and simply transposes one bit value for another encrypted value. Provided a nonce is generated on a pre-agreed basis between sender and receiver (time-stamp, rolling value etc), no additional message characters are added to the encrypted messages.</td>
</tr>
<tr>
<td>Encryption makes the message more likely to be received corrupted.</td>
<td>False. Stream ciphers work by bit alteration – one bit for one bit. The received message is no more or less likely to be corrupted whether it is encrypted or unencrypted. Note however that if a transmitted nonce is used, corruption of the nonce will cause the entire message to be lost. For this reason, Infostream does not prefer or use this approach. Block ciphers are very susceptible to bit corruption and are thus not suitable for paging.</td>
</tr>
<tr>
<td>A strong encryption algorithm ensures secure communication.</td>
<td>Not always. Many encryption systems are broken not by breaking the encryption algorithm, but by exploiting some weakness in implementation or operation. A badly implemented system with strong encryption algorithms and keys can be easily broken.</td>
</tr>
<tr>
<td>Encryption requires powerful CPUs and large memories.</td>
<td>False. Many ciphers in use today are specifically optimized to enable rapid decoding with low memory requirements. Although pagers with small memory and CPUs may not be able to store many encryption keys, they can still implement effective encryption.</td>
</tr>
<tr>
<td>The method or algorithm of encryption must be a secret.</td>
<td>False. Be very suspicious of systems where the method of operation is not fully disclosed. Secrets do not last long, especially if they are not yours. If a vendor of the encryption system won’t tell you how it works in technical detail, how do you or anyone know it is secure? Good encryption relies on published and well-analysed algorithms and exclusively on the security of the encryption keys that you provide and not on secrecy about how the system works.</td>
</tr>
<tr>
<td>Changing keys often is necessary for security.</td>
<td>Not necessarily. Keys need to be changed only if there is reason to believe that the key is compromised. Regularly changing keys otherwise does not improve security. Key security is however less likely as time progresses, unless all devices can be accounted for.</td>
</tr>
<tr>
<td>Changing keys requires returning pagers for reprogramming.</td>
<td>False. If a pager is compromised and shared keys must be changed, it can be done with complete security over the air, using only keys that are unique to non-compromised pagers.</td>
</tr>
<tr>
<td>Messages must be encrypted at point of origin.</td>
<td>Not necessarily. Although snooping over the air is the easiest way to intercept paging messages, it is still possible to intercept messages between the source and paging terminal where they are encrypted. However, protocols are available to ensure secure communications between the message sender and the paging terminal thus making end-to-end security possible.</td>
</tr>
</tbody>
</table>
6 CONCLUSIONS

This whitepaper presents some basic principles of cryptography as applied to paging systems. We have examined the pros and cons of both block and stream ciphers as they apply to paging.

By using a combination of a strong block cipher to generate a key-stream, a stream cipher system is described that provides the following benefits:

- Highly secure end-to-end encryption of paging messages
- No transmission overhead associated with encryption
- No loss of message integrity arising from the encryption process
- Immunity to replay and key extraction attacks through the use of a time dependent nonce.

Infostream has applied sound cryptographic principles to implement a complete system of pager encryption delivering the above benefits. When combined with the Viper fleet management software, OTA management of encryption keys is completely automatic and secure.

There are of course threats to security of a paging system that lay outside the scope of the encryption technology and these include theft or loss of physical pagers, jamming, attacks on the paging infrastructure itself and poor operational practice. No technical solution exists to address these external threats.

Although the system presented and implemented in the Infostream products relies on specific operation of the paging encoder, similar systems can be devised along the same principles to enable encryption to be implemented on alternative systems.
7 APPENDIX

7.1 References

- *Pager Network Encryption* – High level Overview, (Infostream Internal Document)

7.2 Glossary of terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain text</td>
<td>In cryptographic terms, plain text refers to the readable version of the message that is to be exchanged between the parties. In the paging context, it is the message that is sent by the operator and read on the screen of the pager.</td>
</tr>
<tr>
<td>Cipher text</td>
<td>The cipher text is the encrypted and therefore unreadable form of the message.</td>
</tr>
<tr>
<td>Key-stream</td>
<td>A key-stream is a pseudo-random sequence of (binary) numbers that is mathematically combined in a reversible fashion with the plain text to produce the cipher text. The key-stream must appear to be random and unbiased to prevent decryption of the messages.</td>
</tr>
<tr>
<td>Block cipher</td>
<td>A block cipher is an encryption method that operates on fixed blocks of data (typically 64, 128 or 256 bits) and modifies the input data according to the block cipher algorithm and a secret key in such a way that the input data cannot be deduced from the output data without knowledge of the secret key. A good block cipher will be computationally infeasible to break by any method including brute force.</td>
</tr>
<tr>
<td>Brute force attack</td>
<td>A brute force attack is a method of cryptographic attack where every possible combination of an unknown key is tried in order to find a key that results in a successful decoding of the message.</td>
</tr>
<tr>
<td>Stream cipher</td>
<td>A stream cipher is an encryption technique that involves mathematically combining a key stream with the plain text to arrive at the cipher text. Most stream ciphers use reversible operations such that applying the same transformation using the same key stream a second time will revert the cipher text to the plain text.</td>
</tr>
<tr>
<td>Nonce</td>
<td>A nonce is a value that is used once and once only for each encryption so that decoding one message encrypted with a stream cipher will not enable an attacker to decode any other message. The nonce will also prevent replay attacks from working because the nonce cannot be used for a second message (or a second copy of the same message).</td>
</tr>
<tr>
<td>Replay attack</td>
<td>A replay attack is a method of foiling an encryption system by sending an intercepted message (without needing to decode it) at a later time so as to fool the recipient into believing that another legitimate message was sent.</td>
</tr>
<tr>
<td>Self-synchronising</td>
<td>A self-synchronising cipher is one that can correctly align itself based on the recent history of data received. Self-synchronising encryption is not required for paging</td>
</tr>
<tr>
<td><strong>Encryption key</strong></td>
<td>An encryption key is a secret number (shared between the sender and receiver of a message and not known outside) that is used to seed an encryption algorithm to produce a unique encryption for that key. Without knowledge of the key, it should be impossible to decode an encrypted message, even if full knowledge of how the encryption is done is available.</td>
</tr>
<tr>
<td><strong>Hash function</strong></td>
<td>A hash function is an algorithm that generates an output of fixed length from an input of variable length that is highly likely to be different for each possible input sequence.</td>
</tr>
<tr>
<td><strong>Hash sequence</strong></td>
<td>A hash sequence is a technique used to generate an infinite set of unique keys (required to seed a cipher) from a single fixed key and a nonce.</td>
</tr>
</tbody>
</table>