One-Way Cryptography

Sandy Clark Travis Goodspeed Perry Metzger Zachary Wasserman Kevin Xu
Matt Blaze
University of Pennsylvania

14 April 2011 – Preprint, Cambridge Security Protocols Workshop 2011

1 A (Perhaps Typical) One-Way Protocol

In a forthcoming paper[2], we examine the security of the *APCO Project 25* ("P25")[3] two-way digital voice radio system. P25 is a suite of digital protocols and standards designed for use in narrowband short-range (VHF and UHF) land-mobile wireless two-way communications systems. The system is used by law enforcement, national security, public safety, and other government users in the United States and several other countries.

Because two-way radio traffic is easily intercepted, P25 includes a number of security features, including encryption of voice and data under a variety of cipher algorithms and keying schemes. It is regarded as being sufficiently secure to carry highly sensitive traffic, including confidential law enforcement criminal surveillance operations and to support classified national security investigations, and is extensively used for these purpose by the various U.S. federal agencies that conduct such activities.

A full review of the P25 protocols and security model is well beyond our scope (see our forthcoming paper). However, several protocol features will be relevant to us here:

- Radios can be loaded with one or more symmetric traffic keys that can be used to encrypt outgoing transmissions and decrypt incoming transmissions.
- Traffic keys are loaded into radios either manually (with special hardware) or upon request over the air (via a key distribution server). Keys can be set to expire or self-destruct under certain conditions.
- There are no "sessions"; each transmission is designed to be self-contained and decoded by any receiver (assuming it has the the correct key material if encrypted).
- Received cleartext voice transmissions are always decoded and played over the receiver speaker.
- Received encrypted voice transmissions are decoded if the correct key is present (a key identifier is included in the transmission).
- There is no authentication – the security model is designed to assure confidentiality only.
- The sender's radio unilaterally makes all security decisions about its transmissions. Radios can be configured to always send in the clear, always send encrypted, or to encrypt based on a user-selectable switch. In practice, most radios use the latter configuration.

We might observe that while P25 is used in “two-way” radio systems, the security protocol is, in fact, entirely *one-way*. That is, unlike most of the security protocols used today in communication systems, there is no “negotiation” with the receiver or, indeed, any round-trip between sender and receiver before the transmission commences. The sender effectively broadcasts blindly, in the hope that the desired receiver(s) have the key required to correctly decrypt and decode the transmission.

Our study found that “secure” P25 systems are, in practice, actually frightfully insecure. We found a number of fundamental weaknesses in the protocols that are beyond the scope of this paper, the most serious of which render P25 systems almost uniquely vulnerable to highly efficient and difficult to detect denial of service attacks. But protocol weaknesses aside, we also found evidence of an even more serious, and widespread, practical problem that we examine here: the “encrypted” radios are often actually transmitting in the clear, with their users none-the-wiser.

In particular, we found that unintended transmission of cleartext commonly occurs in practice, even among trained users engaging in sensitive communication. We sampled over-the-air P25 traffic from the secure two-way radio systems used by federal law enforcement agencies in several US metropolitan areas over a two year period and found, to our great surprise, that a significant fraction of the most sensitive “encrypted” voice transmissions sent over the air are actually sent in the clear, without apparent detection by the users. We typically captured 20-30 minutes per day per city of unintended cleartext on the federal law enforcement frequencies we monitored.

We saw no evidence that these unintended clear transmissions were caused by low-level protocol weaknesses being exploited by malicious active attacks (although we also found that the protocols are, in fact, vulnerable to such attacks). Instead, they were caused by the protocol’s susceptibility to particular (and difficult to detect) user and configuration errors that, in combination, make it very difficult to ensure that the “secure” mode is actually being used and that make cleartext a de-facto default mode of operation even in systems that have encryption enabled.

Sensitive traffic was sent in the clear under three different scenarios:

- *Individual Error:* One or more users in the clear, but other users encrypted. In this scenario, all users clearly shared a common cryptographic key, since communication was able to occur unimpeded. But the users transmitting in the clear apparently accidentally switched their radios to transmit in the clear mode. Because the offending users still received the other users’ encrypted traffic and because those users had no way to reliably tell that they were sometimes getting clear traffic, this situation typically remained undetected.

- *Group Error:* All users operated in the clear, but gave an indication that they believed they were operating in encrypted mode. In some cases, this involved one user explaining to another how to set the radio to encrypted mode, but actually described the procedure for setting it to clear mode. In other cases, the users would simply announce that they had just rekeyed their radios to operate in encrypted mode (but were actually in the clear).

- *Keying Failure:* One or more users did not have the correct key, is unable to receive encrypted transmissions, and asks (in the clear) that everyone switch to clear mode for the duration of an operation so that all group members are able to participate.

Across all the agencies we monitored, the unintended cleartext we intercepted was roughly evenly split among the Individual and Group Error and the Keying Failure categories. In general, even when users knew they were operating in the clear (because they expressly switched to clear mode due to keying failure) and were engaged in sensitive operations, they made little effort to conceal the nature of their activity in their transmissions. Every system we monitored had P25 encryption capability, and, indeed, most of the traffic was apparently successfully encrypted most of the time. Yet we nevertheless were able to intercept literally hundreds of hours of very sensitive traffic sent in the clear over the course of the last two years.

We emphasize that we were analyzing "live" traffic of real users doing their day-to-day work, not synthetic user-behavior in a controlled laboratory environment. This approach has both limitations and benefits. On the one hand, we did not perform a controlled usability study such as the one seminally performed by Whitten and Tygar with PGP [4]. Such a study might allow us to isolate variables or compare the efficacy of different protocols or user interfaces with one another. On the other hand, we captured "real world" conditions – especially the motivation of the users to maintain security while getting their work done – which gives us a better representation of the system's true usability under field conditions [1].

2 One-Way Encryption is Hard

What's going on here? How can such a simple solution to such a seemingly simple security problem go so wrong in practice?

Part of the answer may be that it isn't actually as simple a problem as it seems. As we note above, the P25 protocols do not fit the negotiated communication model under which most security protocols are designed (and to which our community devotes most of its attention). One-way protocols in which there is no negotiation or exchange between the transmitter and the receiver are actually rather unusual, and relatively little is known (or written in the literature) about robust designs principles for them.

Let us make a few observations that may serve as a basis for progress here.

2.1 One-way protocols reverse the "safe" assumption

The first goal for any communications system or protocol – secure or not – is usually that it enable the authorized parties to communicate, not that it prevent unauthorized interception, which is at best only a secondary requirement as far as its users are concerned. But as misplaced as these priorities may seem to us security specialists, it does not matter much in practice. We can still usually accomplish both reliable communication and reasonably reliable security.

But the reality that security is often only a secondary requirement has far more dangerous implications in one-way protocols than in conventional two-way protocols. The reason is that in a two-way protocol, the senders and receivers can *negotiate downward* from the *most* secure configuration, whereas in a one-way protocol, the sender can only guess about the receivers' capabilities.

That is, a reasonable (and indeed, standard) way to design a conventional two-way security protocol is for the initiator to ask the responder what its capabilities are, match those against its own, calculate the most "secure" set of overlapping capabilities, and proceed from there if the result is sufficiently secure to conform with policies of both. Done properly and carefully (something that is admittedly non-trivial), the parties communicate using the most secure techniques available to them, and either party has the opportunity to halt the exchange if there result is not sufficiently secure.

But we can do nothing of the sort in a one-way protocol. The sender aims to communicate with one or more remote receivers, the current capabilities and keys of which it can only assume. If the sender is very conservative and places a higher value on security than communication, it might choose to assume that the receiver has a fully secure configuration with properly matching keys. (Whether such senders exist in real systems is open to debate). But if the sender is chiefly motivated by the desire to get its message through, as most probably are, it must assume the *least* secure probable configuration by the receivers and transmit accordingly.

In the case of P25 systems, where receivers' keys can be frequently updated and out of sync with one another, this often means that the "safest" configuration for the transmitter is to use the clear, since that is guaranteed to work.

2.2 Error detection and recovery is important

A frequent unintended cleartext scenario we encountered was a single user transmitting in the clear as part of a group that was otherwise successfully using encryption. That is, the cleartext sender must have the current key being used with the rest of the group; if it did not, it would be unable to hear there. The only reason for the user to be transmitting in the clear is a simple error – its encryption switch being in the wrong position.

This is a potentially dangerous error, since the damage of transmitting sensitive information is done as soon as the push-to-talk button is pressed, but we might expect the cleartext transmissions to be quickly noticed by the receiving users, who could inform the offending user. And yet in the two years of transmissions we analyzed, this never once actually happened. The lone cleartext users would continue on sending sensitive data in transmission after transmission, despite sharing a key with other users who had their encryption functions correctly enabled.

The problem appears to be not only insufficient alerting of the sender that it is operating in the clear, but of the receivers that something they are hearing was sent in the clear.

We can imagine user interfaces that might do a better job of alerting users to insecure configurations and settings. For example, radios in encrypted mode might ignore clear received cleartext althogether, or might beep loudly when receiving a clear signal. But there are tradeoffs to consider when the system is to be used in a critical public safety application. How do we also ensure, for example, that emergency messages get through, even if they were sent in the clear?

3 Where have we seen this before?

Although most security protocols are two-way, P25 radio systems are not unique. In fact, the problems in P25 remind us of perhaps the most "classic" application of encrypted communication, one which also shares the one-way property: electronic mail.

We hope we can be forgiven for observing that e-mail has not exactly been a shining example of our community's success in deploying encrypted communications systems. Despite PGP, GPG, and S-MIME's long availability, almost no one – much less your authors – actually uses e-mail encryption of any form routinely.

However, perhaps we can apply some of what we've learned from email to other one-way systems. Recall, for example, the call, from very early on, for a key distribution infrastructure. In some respects, a PKI or key distribution infrastructure can be thought if as making a one-way protocol partly into a two-way protocol, in which the infrastructure serves as a proxy for the receiver. We can say what we might about the problems with PKI, the presence of an online infrastructure does appear to have some benefits here.

Acknowledgements

We are grateful to Peter Sullivan for many helpful discussions on the practical requirements for public safety radio systems. Partial support for this work was provided by a grant from the National Science Foundation, CNS-0905434.

References

[1] Sacha Brostoff and M. Angela Sasse. Safe and sound: a safety-critical approach to security. In *Proceedings of the 2001 workshop on New security paradigms*, NSPW '01, pages 41–50, New York, NY, USA, 2001. ACM.

[2] Sandy Clark and Travis Goodspeed and Perry Metzger and Zachary Wasserman and Kevin Xu and Matt Blaze. Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-Way Radio System. In *Proc. USENIX Security Symposium*, 2011.

[3] Telecommunications Industry Association. Project 25-DataOverview-NewTechStandards. Technical Report TIA-102.BAEA-A.

[4] Alma Whitten and J. D. Tygar. Why Johnny Can't Encrypt. In *Proceedings of the 8th USENIX Security Symposium*, 1999.