Cryptographic Modes of Operation for the Internet

Steven M. Bellovin
smb@research.att.com
AT&T Labs Research

Matt Blaze
mab@research.att.com
AT&T Labs Research

1 Introduction

Modes that may be appropriate and secure in one application or environment sometimes fail badly in others. This is especially true of stream modes where, e.g., re-use of the same segment of keystream to protect different plaintext renders the cipher insecure. The circumstances that can render a mode insecure are not always obvious, nor are the relevant characteristics of a particular application always apparent.

Application and protocol designers, even those with experience and training in cryptography, cannot be expected to always identify accurately the requirements that must be met for a mode to be used securely or the conditions that apply to the application at hand.

We strongly urge that, for each adopted mode, the standard include a clear statement of the requirements and assumptions that must be met in order for the mode to be used securely and what security properties the mode can be assumed to have and not have. Furthermore, we urge that detailed examples of acceptable and unacceptable application for each mode be provided as well.

In this draft, we discuss some of the security properties, and pitfalls, of several proposed stream modes, and we note several ways in which these modes would be difficult to use securely in the context of Internet Network-, Transport- and Application-layer protocols

2 Characteristics of Internet Protocols

The Internet has several characteristics that complicate the choice of a mode of operation. Many of them rest on one of its fundamental design principles: the Internet is based on unreliable datagrams. Even though most applications see a reliable stream, the underlying datagram nature still complicates the cryptography.

2.1 Properties of Internet Datagrams

Every datagram on the Internet is treated independently. Any given datagram may be dropped, duplicated, damaged, or delivered out of order [Pax97]. Cryptographic mechanisms at the network- (and often the link-) layer must account for this property.

An obvious conclusion is that any form of stream or message encryption cannot extend beyond the limits of a single datagram. Similarly, if CBC mode is used, the IV used to encrypt each packet must be present within the packet or derivable from other information in that packet. It cannot be chained implicitly from the previous packet in that “connection”; that packet may be have been dropped, damaged, delayed, etc.

Some stream ciphers, such as RC4, are even more problematic. RC4 depends on a reliable underlying sequence of bytes; such does not exist at the lower layers of the Internet.

There are ways around these difficulties, but they sometimes lead to other problems. Thus, the designers of the WEP (Wired Equivalent Privacy) feature for 802.11 networks, which uses RC4, devised a scheme that employs a different RC4 key for each packet. A portion of the actual key is included in each packet as a form of IV. Ultimately, this strategy has failed, not just because of unforseeable attacks on RC4 [FMS01] but because of fundamental limitations of this cipher scheme when applied to 802.11 end nodes. In particular, every datagram *must* use a different “IV”, but there was no way to retain state about which IVs were used. (This was coupled with some other poor choices, such as a too-small IV field and non-random starting points. And in fact, the mechanism for combining the IV with the base key contributed to the attacks described in [FMS01], though that was only partially predictable.)

2.2 Properties of Internet Streams

Most applications on the Internet use TCP [Pos81], which implements a reliable transport mechanism. Stream ciphers can indeed be used above TCP. However, some pitfalls remain.

For one thing, there are no standard error recovery mechanisms above TCP. TCP is assumed to provide reliable transport — and in particular, to cope with the datagram properties of the lower layer — which frees the application of any necessity perform any such function. As a consequence, there is no session layer or equivalent to recover from problems not handled by TCP. In a security setting, however, this can prove fatal. Suppose that a fraudulent packet is injected, one crafted to pass TCP’s (minimal, non-cryptographic) checks. It will be accepted by TCP, and passed to the upper layers. A cryptographic upper layer can detect the insertion, but it has no mechanism for telling TCP about this. In fact, TCP itself is unlikely to recover from such a malicious packet [Jon95]; there is little choice but to abandon the connection. In other words, using a stream cipher on top of TCP makes systems more vulnerable to easy denial-of-service attacks.

By contrast, a cryptographic mechanism that operates at lower layers, such as IPsec [KA98], can take advantage of TCP’s built-in retransmission capabilities. A fraudulent packet will be detected and dropped before TCP sees the packet. TCP will treat this as a packet dropped by the network, and will retransmit and recover without interfering with the application.

There is a more subtle operational weakness to encryption above the TCP layer. It *is* above the TCP layer, and thus cannot easily be processed by custom hardware.

Unless TCP itself is moved outboard — an approach tried in the past, but largely abandoned today because the complexity of the host-to-front end protocol approached that of TCP — the receiving computer must do all of the TCP processing. Trying to use a cryptographic co-processor for decryption at that point requires a lot of overhead for device setup, as well as a large increase in the memory bandwidth required for packet reception. (By contrast, NIC cards that include IPsec are available today from several different vendors.)

2.3 Known and Probable Plaintext

Many cryptanalytic attacks rely on large quantities of known plaintext. As such, these attacks are often considered "certificational weaknesses". While these are taken seriously during evaluations, they are often dismissed as impractical in practice.

We disagree. Unless the cryptographic protocols being deployed are specifically designed to counter such attacks, the attacks are often quite feasible. As shown in [Bel96], there are many ways in which an attacker can generate large amounts of known plaintext. For example, if the target user is reading email via a VPN, the attacker can send a large, tailored file. The signature of such a large download is easily detectable, thus providing the attacker with a "crib".

Similarly, [Bel97] shows that the TCP and IP headers contain large amounts of probable plaintext. These, too, can be employed in cryptographic attacks.

The point of these assertions is not to claim that it is impossible to encrypt Internet traffic, or even that doing so takes special skill. We do, however, claim that the form of encryption used, and in particular the modes of operation, must be matched to the peculiar characteristics of the Internet.

2.4 Active Attacks

Active attacks are often considered even less likely. Again, these are feasible on the Internet. Descriptions not only exist in the public literature [Jon95], there are off-the-shelf hacking tools[1] that implement man-in-the-middle attacks. Some of these tools are even targeted at cryptographic protocols!

2.5 Rekeying

Many of these attacks are avoidable if sessions are rekeyed frequently enough. Unfortunately, this is not always possible. One obvious reason is the expense. Key management protocols generally employ public key cryptography, and with it large, modular exponentiations; these consume a great deal of CPU time and battery power on smaller or hand-held devices.

In other cases, users sometimes rely on static keys. This may be due to the lack of a suitable PKI, or it may be due to lack of protocol support. In at least one case, 802.11's WEP, there is no key exchange protocol defined. We are not endorsing or even defending such behavior; we do note that it exists, and must be taken into account.

[1]See, for example, `http://www.monkey.org/˜dugsong/dsniff`.

3 Requirements and Pitfalls for Various Modes

Keystream generation modes, such as counter mode and OFB, produce a stream of bits that must be combined with the plaintext. The dangers of key stream reuse are well-known. Unfortunately, they are often difficult to avoid.

The attack on WEP [BGW01] is one such example. In this case, there were a number of contributing factors, including use of a stream cipher, static keying, and a too-small "IV". Attacks on another design for stream ciphers with IPsec are described in [Bel96].

A second class of weakness of such modes of operation is the ability to make predictable changes to the plaintext. (This is also exploited in one of the attacks described against WEP.) The existence of toosl for active attacks makes even stronger the message in [Bel96]: "It is quite clear that encryption without integrity checking is all but useless." The primary justification for counter mode is the need for high-speed, easily parallized, encryptors. But without designs for equally fast integrity-checkers, such a mode is extremely dangerous.

Other modes of operation have their own pitfalls. The self-healing property of CBC and CFB modes have often been exploited in attacks [Win84, MW88, Bel96]. Again, integrity-checking is needed.

3.1 IV management

Guidelines on IV selection for feedback modes have been known for many years. We suggest that such guidelines be made part of the formal standard defining the modes of operation. For example, the IPsec specification for use of CBC mode [MD98] requires that IVs be random. It goes on to say, in an Implementation Note:

> Common practice is to use random data for the first IV and the last 8 octets of encrypted data from an encryption process as the IV for the next encryption process; this logically extends the CBC across the packets. It also has the advantage of limiting the leakage of information from the random number genrator. No matter which mechnism is used, the receiver MUST NOT assume any meaning for this value, other than that it is an IV.
>
> To avoid ECB encryption of very similar plaintext blocks in different packets, implementations MUST NOT use a counter or other low-Hamming distance source for IVs.

Other guidance should be provided as well. [VK83] gives one set. For counter mode, a large-enough counter is clearly needed. Better yet, an IV composed of several different pieces offers more protection. The proposed replacement standard for modes of operation suggests in Section B.2 that the counter for a message be composed of a per-message nonce and an ordinary counter. Kent made similar suggestions at at IETF presentation. An implementation suggestion similar to that given above might suffice: use the last block (or a portion of the last block) of ciphertext from one message as the initial counter (or as part of the initial counter) for encryption of the next message. compromised.

A stronger requirement would be to ban use of counter mode unless an automated key exchange mechanism were used. The greatest risk of confidentiality compromise arises when a keystream is reused; this dnager, in turn, is most likely to arise when the counter is reinitialized. If that can only happen in conjunction with a new key, the danger is averted.

Multiple-sender situations pose a particular challenge. Consider a multicast video conference where all parties share a (negotiated) key. Counter mode would seem to be a natural choice for such a situation, but if two senders used the same counter, the confidentiality of the session would be

4 Conclusions

We have pointed out a number of weaknesses in various standard and proposed modes of operation. These are especially serious for counter mode. We suggest that the formal specification include operational constraints and warnings. For counter mode, these should include:

- It *must not* be used with static or manual key management.
- It *must not* be used in multiple-sender, shared key environments unless strong measures are taken to prevent counter collision.
- It *must* be used with cryptographically strong integrity checks.

All of these address situations found today on the Internet.

References

[Bel96] Steven M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix U*NIX SECURITY SYMPOSIUM, pages 205–214, July 1996.

[Bel97] Steven M. Bellovin. Probable plaintext cryptanalysis of the IP security protocols. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 155–160, 1997.

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of MOBICOM 2001*, 2001.

[FMS01] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorith of RC4, 2001.

[Jon95] Laurent Joncheray. A simple active attack against TCP. In *Proceedings of the Fifth Usenix U*NIX SECURITY SYMPOSIUM, Salt Lake City, UT, 1995.

[KA98] S. Kent and R. Atkinson. Security architecture for the internet protocol. Request for Comments 2401, Internet Engineering Task Force, November 1998.

[MD98] C. Madson and N. Doraswamy. The ESP DES-CBC cipher algorithm with explicit IV. Request for Comments 2405, Internet Engineering Task Force, November 1998.

[MW88] Chris Mitchell and Michael Walker. Solutions to the multidestination secure electronic mail problem. *Computers & Security*, 7(5):483–488, 1988.

[Pax97] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, 1997.

[Pos81] J. Postel. Transmission control protocol. Request for Comments 793, Internet Engineering Task Force, September 1981.

[VK83] V. L. Voydock and S. T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15(2):135–171, June 1983.

[Win84] Robert S. Winternitz. Producing a one-way hash function from DES. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 203–207. Plenum Press, 1984.