Cryptography and the Internet

Steven M. Bellovin

AT&T Labs–Research, Florham Park, NJ, USA
smb@research.att.com
http://www.research.att.com/~smb

Abstract. After many years, cryptography is coming to the Internet. Some protocols are in common use; more are being developed and deployed. The major issue has been one of *cryptographic engineering*: turning academic papers into a secure, implementable specification. But there is missing science as well, especially when it comes to efficient implementation techniques.

1 Introduction

In early 1994, CERT announced[1] that widespread password monitoring was occuring on the Internet. In 1995, Joncheray published a paper explaining how an eavesdropper could hijack a TCP connection [Jon95]. In mid-1998, there is still very little use of cryptography. Finally, though, there is some reason for optimism.

A number of factors have combined to change people's behavior. First, of course, there is the rise of the Internet as a mass medium, and along with it the rise of Internet commerce. Consider the following quote from a popular Web site:

> How does ——.com protect my credit card if I order online?
> ——.com takes every precaution to protect the privacy of your credit card information. We utilize Secure Socket Layers (SSL), the most advanced security system available.
> All of your ordering information – including your name, address, and credit card number – is encrypted using a Secure Server for maximum security. Your credit card and billing information cannot be read as it travels to our ordering system. In addition, our ordering system is not connected to the Internet and is not accessible in any way.
> You can also use our Standard Server, pay by phone option, or fax us your order.

There are several noteworthy things here. First, of course, they advertise their use of encryption. Second, as evidenced by the phone payment option—a relatively common choice—there is still persistent public uneasiness about Internet security. Cryptography, of course, is part of the solution; thus, companies that wish to attract business are touting their use of encryption.

[1] CERT Advisory CA-94:01, 3 February 1994.

A second major driver for the adoption of cryptography has been the transformation of the Internet into *the* data network. More and more, corporations are using the Internet for general data transfer, in the same was as they have traditionally used the phone network for voice traffic. A branch office may have its own link to the Internet and communicate with the home office via this channel, instead of using a leased line. Telecommuters can use an ISP's modem pool to dial in to work. But both of these practices are risky without encryption.

Finally, the technology is ready. Computers are fast enough that the overhead for encryption is—barely, in some cases—tolerable. Standards exist for the much important types of encryption. And most—but not all—of the necessary science exists.

2 Current Uses of Cryptography on the Internet

Perhaps the most mature cryptographic technology in use on the Internet is secure email. Two different schemes—PGP and S/MIME—have achieved reasonably broad penetration. Both have been hampered, though, by lack of a widespread public-key infrastructure. While not strictly necessary, especially for use within comparatively small groups, a more broadly-based certificate graph is necessary for some uses. Imagine, for example, trying to solve the "spam" email problem by relying on PGP's Web of Trust.[2]

Another notable use of cryptography is SSL, the Secure Socket Layer. While in theory quite general, in practice SSL is used almost exclusively for communication between Web browsers and servers. Furthermore, in almost all cases authentication is at best one-way—servers have certificates; clients rarely do—and in practice is *unauthenticated*, since most users of the technology neither know nor care what a certificate is, nor who has signed it. For that matter, the popular browsers give very little guidance on whether or not certificates should be accepted, what the meaning of the signing certificate is, etc. We thus have the dual of the situation with secure email: the certificate authorities exist, and are used, but to little practical effect.

The third major area for cryptography in the Internet is the network-layer encryption and authentication protocol set, IPSEC.[3] There is also a key exchange protocol derived from Diffie et al's STS [DvW92] and Krawczyk's SKEME [Kra96].

IPSEC provides broad-spectrum protection of communications, below the application and transport layers. It is thus invisible to them, but protects them nevertheless.

[2] "Spam" is the common term for bulk, unsolicited commercial email.

[3] At this point, the new IPSEC RFCs have not yet been issued, though they are expected momentarily. The existing ones—RFCs 1825-1829—describe an obsolete version of the protocol. While they are useful as a general guide to IPSEC, there are a number of cryptographically significant differences between them and the newer standards.

IPSEC also provides the ability to trade cost for granularity of protection. A single IPSEC key can protect a user, a host, or an entire network. An organization can use a single gateway, to minimize its costs; conversely, it can add IPSEC to every host, thus guarding against certain attacks within its site.

Since IPSEC is just starting to be deployed, it is impossible to assess its usage patterns. Still, in at least two likely deployment patterns—firewall-to-firewall in a virtual private network (VPN) configuration, and remote employees-to-firewall—the certificate will be used for *authorization*. This suggests that certificates will be meaningful, but that a widespread PKI will not be needed; instead, each corporation will issue its own certificates.

If IPSEC is used in other than end-to-end mode, some intermediate points must be trusted. Furthermore, since the topology of the Internet is dynamic, there may not be a fixed set of trusted parties between two endpoints wishing to converse. In many cases, such delegations should be digitally signed by the ultimate endpoint. In other cases, such as a corporate firewall, the delegation is in fact in the reverse order. That is, the administrator for some zone `gigacorporation.com` could in fact specify the IPSEC proxies for all hosts within that domain. Regardless, the exact set of IPSEC gateways to be used must be discovered anew for each connection.

Mention should also be made of SET, a secure electronic payment protocol developed by the banking and credit card industry. It's especially interesting because it's a multiparty protocol: the consumer, the merchant, and the bank. It is worth noting the collision here between cryptographic theory and commercial reality: while one might think that a signature-based protocol would eliminate any need to transmit an actual credit card number, that turns out not to be the case; some merchants use credit card numbers as the look-up key for the customer databases, and are unwilling to lose the previous purchase history. Accordingly, the card number may still be sent to the merchant, though of course in encrypted form.

3 Planned Uses and Missing Pieces

There are a number of things we would like to do on the Internet; however, we don't know how to do them. I will focus on three: efficient cryptographic processing, routing, and multi-party cryptography.

The first is, of course, obvious: we need faster algorithms. While Moore's Law has helped, often the effect of a faster CPU is that system designers demand more of it. A modern tunnel server, for example, may handle hundreds of simultaneous connections. But if these sessions are cryptographically protected, more CPU power is needed. Worse yet, the public key operations to establish these connections are very expensive. If a server handling 500 remote users crashes and reboots, it can be a quite a while before all of the sessions are re-established: the necessary public-key operations are quite expensive.

A less obvious place where efficiency improvements are desperately needed is for authentication and integrity algorithms. During the development of IPSEC, we learned that encryption without integrity-checking is all but useless [Bel96]—it turned out to be practical to use the error propagation properties of CBC mode to construct all manner of useful but fraudulent packets. To give just one example, an attacker could splice together the body of a target packet with the header of a packet destined for his or her program.

The current choices for such integrity algorithms—IPSEC currently specifies HMAC [BCK96] with either MD5 [Riv92] or SHA-1 [NIS95]—are too slow in many cases, and are not particularly suited for hardware accelerators [Tou96]. Alternatively, an encryption mode or algorithm that prevented any tampering with the ciphertext might suffice. (Rivest's all-or-nothing encryption mode [Riv97] is too expensive.)

There is also a strong need for secure routing protocols. Internet routers exchange reachability and link cost information with their neighbors; from this, each router computes the optimal path to each destination network on the Internet.[4] There is no global knowledge of the actual topology.

If a router lies about the networks it can reach, its neighbors will be deceived. This in turn can result in traffic being diverted to paths controlled by an enemy. While traffic encryption should prevent eavesdropping, routing attacks represent at least a denial of service attack, and—in the absence of other encryption—more serious threats.

It is not obvious how to use cryptography to secure this structure. Protecting the routing exchanges between each pair of routers is straight-forward enough; the problem, however, is that each router knows only what its neighbors have said. They themselves could have been deceived. A solution will involve verifying the full path back to the owner of the network. And that in turn requires calculating and verifying many digital signatures, which is prohibitively expensive. While some work has been done [SK97, SMGLA97, HPT97, MB96], much more remains.

Another interesting research area is providing adequate security for multicast sessions. While a number of protocols have been proposed, it is not clear that they are suitable. There are a number of reasons for this; prominent among them is that there is no one model for what multicast is. It may be a television-style broadcast, with authentication of all messages and perhaps encryption so that only subscribers can watch. It may be a conversation between a small number of participants. It may be a combination of a broadcast and a question-and-answer session; while anyone can speak, the session is under control of a central site, which must have the ability to exclude disruptive participants.

A constraint on multicast security mechanisms is the trust model. Many proposed protocols assume that the key distribution graph is somehow related to the packet-forwarding graph. For some common uses of multicast technology, this is a bad assumption. Packet-forwarding is often configured by Internet Service Providers; ordinary users can and do create multicast sessions. A compromise

[4] The actual routing structure of the Internet is far more complex than is explained here.

position may be some way to identify some trustable subset of the forwarding graph; discovering this set—and deciding that it is trustable—is not trivial.

4 Trust and Policy Management

Many of the problems discussed earlier can be summed up in one question: who can be trusted to do what? More precisely, how can a user or a computer acting on that user's behalf know what certificates are acceptable for a given action? Furthermore, out of the set of potentially trustable parties, which are the right ones under some given set of circumstances.?

The problem shows up most clearly with IPSEC, where a machine may need to discover the identity of a security gateway for some connection. Even in that case, there can be considerably more complexity. For example, two hosts may wish to use end-to-end encryption. However, both sites are behind firewalls that wish to do their own encryption between them. Furthermore, one host may need to use authentication from it to the outbound firewall, to validate its right to send traffic out of the location.

A related issue is the specification of the desired policy. How can an administrator communicate to assorted hosts the identities, both cryptographic and network, of the various gateways that must be involved in a secure connection? More to the point, how is it communicated *securely*? Who is *authorized* to set such policies, and how do the endpoints know it?

With SSL and secure email, the trust question is made more complex because the answer must relate to the real world. If I request a secure connection to `www.wsj.com`, my Web browser warns me that the certificate was issued to `interactive.wsj.com`. Should these two be considered identical? The company name is Dow Jones; is that right? How should I know that, a priori? And domain names are often confusing; `nasa.com` bears no relation to `nasa.gov`. Will a user notice the distinction?

One can assert that no matter the cryptographic tricks, the user of a certificate is (and should be) responsible for validating its authenticity. Often, though, it is impossible for the user to do so. In particular, a conventional certificate does not indicate what roles the holder can fulfill. The company name in my certificate indicates correctly that I work for a telecommunications company; it does not say whether or not I am authorized to accept payment for phone bills.

Possibly, schemes such as PolicyMaker [BFL96, BFS98] or SDSI [RL96] will solve this problem. But enumerating all possible roles for a certificate is easy; enumerating the roles it may not fill is very hard. Furthermore, the distinction may be too subtle for a program. A server certificate valid for, say, accepting orders for books via a Web page may not be the proper certificate for software orders, even from the same Web server. But it may be the proper certificate for sending email to the customer care agent.

We must also be wary of techniques that work for humans, but not for programs. A person may be wary enough to note that my certificate contains that word "research", or that it says nothing about bill collection. But will a program check this? IPSEC may become the first large-scale use of certificates intended for checking by programs, not humans. Are our certificates adequate for the task?

5 Cryptography versus Cryptographic Engineering

Often, designing cryptosystems for use in the Internet is one of *cryptographic engineering*. Partly, it's a matter of translating abstract notions into concrete packet formats. That is relatively simple. It's harder to find a way to fit cryptography into a protocol that wasn't designed to accept it. But the hardest job is maintaining security in the face of actual network practice.

Consider, for example, the question of encrypting a message M. The academic paper on the subject would likely have said something like "Alice transmits $\{M\}_K$ to Bob". An implementation specification might say "Use CAST-128 in CBC mode, with key K, an IV as specified above, the whole preceded by a two-byte length field. The receiver's identity is specified in the previous message." But even that isn't sufficient. As we all know, ciphers can be broken. An implementable cryptographic protocol must have some way to indicate which cipher is being used. That in turn raises questions of what ciphers must be common to all implementations. Worse yet, the cipher to be used must be negotiated, and negotiated securely; an enemy who can force the use of DES instead of a more secure cipher may be able to do considerable damage.

Often, different security considerations produce contradictory constraints. In [Bel96], I showed that it was much more secure to use a separate key for each connection, as opposed to a single key for all connections between a pair of hosts. But in [Bel97], I showed that per-connection keying aided an enemy cryptanalyst. Which is right?

Operational considerations produce their own conflicts. The Domain Name System (DNS) relies on caches, timeouts, and hierarchies of servers to reduce the load on the network. The design, originally specified in 1983 [Moc83], requires that the record's time-to-live be in the original response to a query, that it be decremented by servers that cache the response, and that this modified value be passed along to any machines that receive their response from the caching server. But that conflicted badly with a later desire to add digital signatures to DNS records [EK97]. Not only would recalculating the signature each time be prohibitively expensive, the caching server does not (and should not) possess the necessary signing key. Thus, the modified time-to-live field cannot be passed along in a secure fashion. Perhaps the lifetime should have been expressed as an absolute expiration time (though that has problems of its own). But Secure DNS is constrained to live with the earlier structure.

Secure DNS has run into other complications as well. The format of the signed records was designed to permit the signing operation to be done offline, to safeguard the private key. However, this operational requirement conflicts with the DNS Dynamic Update protocol [BRTV97]. It has also resulted in a situation where the mechanism to indicate that a record does not exist—signed front- and back-pointers—can be used by an enemy to learn all of the names in a domain, which conflicts with other security requirements. But safeguarding the signing key is critical; not only does it act as a certificate-signing key, fraudulent DNS records can be used to perpetrate a wide variety of attacks on Internet systems [Bel95].

IPSEC often conflicts with firewalls. A firewall cannot examine, and hence pass or reject, an encrypted packet. Should end-to-end encryption be permitted through firewalls? The fact that a packet is encrypted and authenticated does not mean that it is harmless; an attacker may have penetrated an external system that is authorized to make incoming calls. Even outgoing calls can be used to launch attack. Suppose that a firewall is configured to permit all outbound calls. Naturally, the reply packets must be allowed back in. However, if the port numbers are encrypted the firewall cannot distinguish between a reply packet and a packet attacking a different port on the inside host.

There are no good solutions for this problem. Presumably, some sort of key-sharing with the firewall must take place. Again, that demands strong verification of the firewall's right to the information. It may also demand multiparty key negotiation, or perhaps proxy cryptography [BBS98].

6 Protocol Verification

It should come as no surprise that the cryptographic protocols and mechanisms used in the Internet are in need of verification. They are complex, and as we all know, it is very easy to make mistakes in designing even simple cryptographic protocols. But the analysis here is harder, because it must contend with the complexities of real systems and real operational environments.

Several examples of this can be found in [Bel96]. In one class of attacks, I showed how replays could be used to trick the host operating system into decrypting messages. There were a number of variants of this attack; the simplest involved waiting until the target program had ended, then binding to its "port" and reinjecting the messages. The key remains secure, but the plaintext is revealed.

A more subtle flaw is exploited by Wagner's short-block guessing attack. The attacker attempts to guess at the contents of packets containing a single byte of application data. It requires a modest (and practical) amount of chosen plaintext (2^8 blocks) and a simple (2^8 packet) active attack. If the injected packet contains an erroneous guess of the data byte, the receiving machine will silently discard the packet. If the guess is correct, the network-level checksum will also be correct and the receiving machine will acknowledge the packet. (The ACK messages can be seen as a side channel, similar to those exploited by Kocher in his timing and power consumption attacks.)

To my knowledge, existing formal techniques cannot detect attacks such as these. At the very least, the formalism would have to include a complete description of the networking environment, and probably more besides.

There has already been some useful input from the theory community. IPSEC originally used keyed hash functions as MACs; Preneel and van Oorschot's attacks on these [Pv95, Pv96] caused us to adopt HMAC [BCK96], an algorithm that was proven to be secure, instead. Unfortunately, the help has not always been appreciated. The resentment has come not because of "interference" but because of its timing. In an environment where the phrase "sometimes it's time to shoot the engineers and ship the product" can be uttered, a complaint late in the design cycle is often rejected, simply because it's too late.

7 What Cryptography Can't Do

Cryptography is not a panacea for the security problems of the Internet. By my count, no more than 15% of the CERT advisories over the last 10 years describe vulnerabilities that would be irrelevant in a world with ubiquitous cryptography. Most of the other advisories concerned buggy programs, a failing that cryptography cannot address. Indeed, there were a number of reports of flaws in assorted encryption and authentication programs.

A second problematic area is the existence—dare I say the prevalence?—of bad cryptography. While part of the problem is lack of science—we're all familiar with new attacks on old algorithms and protocols—more of the trouble is a lack of education. About the time I was writing this note, it was disclosed that a major vendor's network encryption product inadvertently used DES with a 48-bit key size. That was bad enough, though forgiveable and fixable. But the same product used ECB mode, an egregious error described as a deliberate design choice. Other vendors misuse stream ciphers [SM98] or invent their own flimsy algorithms—and then rely on obscurity for protection.

Finally, the user interface to cryptographic functions is often lacking. I will give just one example, an encrypting mail program based on a symmetric cryptosystem. To avoid the need for the recipient to have any particular applications software, this program packages up everything into a self-extracting executable that prompts the recipient for the shared secret key. It is adding insult to injury that the keylength employed is a magnanimous 32 bits...

References

[BBS98] Matt Blaze, G. Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of Eurocrypt '98*, 1998. to appear.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology: Proceedings of CRYPTO '96*, pages 1–15. Springer-Verlag, 1996.

[Bel95] Steven M. Bellovin. Using the domain name system for system break-ins. In *Proceedings of the Fifth Usenix* UNIX *Security Symposium*, pages 199–208, Salt Lake City, UT, June 1995.

[Bel96] Steven M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix* UNIX *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.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[BFS98] Matt Blaze, Joan Feigenbaum, and Martin Strauss. Compliance checking in the PolicyMaker trust management system. In *Proceedings of the 2nd Financial Crypto Conference*, 1998. to appear.

[BRTV97] J. Bound, Y. Rekhter, S. Thomson, and P. Vixie. Dynamic updates in the domain name system (DNS UPDATE). Request for Comments (Proposed Standard) 2136, Internet Engineering Task Force, April 1997. (Obsoletes RFC1035).

[DvW92] W. Diffie, P.C. van Oorschot, and M.J. Wiener. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, page 107, 1992.

[EK97] D. Eastlake and C. Kaufman. Domain name system security extensions. Request for Comments (Proposed Standard) 2065, Internet Engineering Task Force, January 1997. (Obsoletes RFC1034).

[HPT97] Ralf Hauser, Tony Przgienda, and Gene Tsudik. Reducing the cost of security in link-state routing. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 93–99, 1997.

[Jon95] Laurent Joncheray. A simple active attack against TCP. In *Proceedings of the Fifth Usenix* UNIX *Security Symposium*, Salt Lake City, UT, 1995.

[Kra96] Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 114–127, February 1996.

[MB96] S.L. Murphy and M.R. Badger. Digital signature protection of the OSPf routing protocol. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 93–102, 1996.

[Moc83] P. Mockapetris. Domain names: Concepts and facilities. RFC 882, Internet Engineering Task Force, November 1983. (Obsoleted by RFC1034); (Updated by RFC973).

[NIS95] NIST. Secure hash standard (SHS), April 1995. Federal Information Processing Standards Publication 180-1.

[Pv95] B. Preneel and Paul C. van Oorschot. MDx-MAC and building fast MACs from hash functions. In *Proceedings of CRYPTO '95*, pages 1–14, 1995.

[Pv96] B. Preneel and Paul C. van Oorschot. On the security of two mac algorithms. In *Proceedings of Eurocrypt '96*, pages 19–32, 1996.

[Riv92] R. Rivest. The MD5 message-digest algorithm. Request for Comments (Informational) 1321, Internet Engineering Task Force, April 1992.

[Riv97] Ronald Rivest. All-or-nothing encryption and the package transform. In *Proceedings of the Fast Software Encryption Conference*, 1997. To appear.

[RL96] Ronald Rivest and Butler Lampson, 1996. Several papers can be found at `http://theory.lcs.mit.edu/~cis/sdsi.html`.

[SK97] K.E. Sirois and S.T. Kent. Securing the nimrod routing architecture. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 74–84, 1997.

[SM98] Bruce Schneier and P. Mudge. Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP), November 1998. 5th ACM Conference on Computer and Communications Security, to appear.

[SMGLA97] B.R. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 85–92, 1997.

[Tou96] Joseph D. Touch. Performance analysis of MD5. In *Proceedings of ACM SIGCOMM '95*, pages 77–86, 1996.