Position Paper: Security and Simplicity

Steven M. Bellovin
Columbia University
https://www.cs.columbia.edu/~smb

1 Introduction

The last several months have raised public awareness of the extent of Internet monitoring by governments. The obvious defense is to use more cryptography, and in particular end-to-end encryption. This is done all too seldom, which in turn raises the question of why: why do so few sites, users, and protocols employ end-to-end encryption? Is there something that standards bodies such as the IETF and W3C can do?

At this point, the problem is probably not the basic protocol. Most standard protocols have encrypted variants, whether on a separate port (e.g., IMAP [7] on port 143 unencrypted or 993 encrypted) or via an explicit request to start encryption (e.g., STARTTLS for SMTP [6]). (Some vendor-proprietary protocols do not have encrypted variants, but there's not much a standards body can do about that.)

Sometimes, the issue is cost, whether in CPU resources or in user latency. TLS [14], for example, requires several extra round trips; if the server is located halfway around the world from the user, the laws of physics set a lower bound on connection startup time. (Increasing the speed of light is apparently out of scope for both the the IETF and the W3C.) At this point, however, this is likely not the real issue. Sites that are large enough that they have significant numbers of users on other continents are probably using distributed data centers anyway.

I assert that the underlying problem is protocol complexity; this in turn tends to manifest itself as implementation complexity. Quite simply, it is too hard for most site to configure encryption for most services because the system administrator doesn't understand the choices and/or because they're too hard to configure. This in turn suggests an appropriate response by the IETF and the W3C: *simplify*. Going forward, eliminate unneeded choices, frills, buttons, knobs, widget, and tweaks from security protocol designs; looking back, issue documents describing dead-simple default settings for important protocols like TLS and IPsec [10]. Vendors can then configure systems to match these defaults, resulting in a zero configuration experience for most sites. Yes, some sites will have different needs; these, though, tend to have more sophisticated personnel, and can spend the time and effort necessary to understand the options. In a few situations, new auxiliary protocols can be designed to simplify the design underlying protocols.

The experience with SSH deployment in the late 1990s following the initial software release [22] is instructive. SSH was "fire and forget"; the cryptography just worked, with no administrator changes needed. System administrators responded enthusiastically. They may not have taken advantage of all of the software features, but for the most basic functionality—enabling secure remote login—they needed to do nothing. It was even possible for vendors to ship systems where SSH worked out of the box. That deployment strategy has been somewhat problematic [4], but that was an implementation failure, not a failure of the basic design.

The remainder of this paper looks at several specific, important cases: IPsec and IKE, PKIX, and email. For lack of space, we omit discussion of DNSSEC, TLS, etc., but the same principles apply.

2 IPsec and IKE

IPsec [10] and IKE [17] are poster children for the complexity problem. Sites have far too many choices to make. Consider the following option descriptions for `racoon`, an open source implementation of IKE:

```
send_cert (on | off);
                    If you do not want to send a certificate, set this to
```

```
                    off.  The default is on.
send_cr (on | off);]
                    If you do not want to send a certificate request, set
                    this to off.  The default is on.
```

Why should a site not want to send a certificate? Why would a site not want to request one? The lack of any explanation is a problem with the racoon documentation, not the protocol, but the underlying fault is really in design of IKE: IKE can operate that with or without certificates. That in turn suggests that the real fix is to simplify the design, to insist on *one* mode of authentication based on certificates. Obviously, implementors could omit options they consider nonsensical, but if different implementors make different choices—and they have—interoperability will suffer.

Of course, different sites have different authentication requirements, especially for end-users. This can be dealt with via auxiliary protocols (and daemons) that implement these different styles, a suggestion made many years ago [2]. Having two protocols instead of one might seem more complex; in reality, most sites will need at most one of these auxiliaries. A site that uses only passwords can ignore the protocol and configuration for one-time tokens. A site that can provision certificates (see Section 3) directly need pay no attention to any of them.

Over-the-wire IPsec is also too complex. Sites have to decide on cipher suites, key lengths, AH or ESP, etc. Few sites have personnel sufficiently familiar with cryptographic arcana to make an informed decision. It is clear that the protocol must have algorithm agility; it is equally clear that very few sites should ever touch an intelligent default setting. The IETF's response, though, is simply a list of possible algorithms and their projected status [12]. A better solution would be an RFC saying "these are the parameters sites SHOULD use": 128-bit AES, 2048-bit RSA, or what have you.

A third issue, more one of implementation than specification, is topology. Realistically, there are only two common topologies: mobile device phone home and site interconnection. Complex configurations with IP addresses and netmasks are painful to set up. Implementations should offer a choice of these two modes, with a dead-simple way to specify the gateway or gateways involved.

We've combined many of these ideas and more in an IPsec configuration compiler [20]. There is exactly one policy decision: should mobile nodes route all traffic through the central site or not? There is one simple way to specify the topology. Finally, the site can specify the IPsec implementation for each node; the code will automatically generate configuration files, certificates and public keys, etc.

Recommendation: The IETF should define a very few recommended profiles for IPsec and IKE. As needed, auxiliary protocols can be defined. Vendors should follow these recommendations for their default configurations. The implementations' topology knobs should be as simple as possible.

3 PKIX

PKIX's complexity stems from two sources: the IETF's specification of certificates and the nature of PKI. Both are addressable by profiles and implementations.

Some of the issues are the same as for IPsec: end-sites are generally not competent to decide on algorithms and key lengths. There should be one RFC saying what most places should use.

There are about $\aleph_0$ different RFCs describing $\aleph_1$ different options; site administrators are forced to understand each of these if only to conclude that they don't need to know it. Most sites do not need, say, their logos [9] in their IPsec certificates; why force people to worry about the option to specify them? Similarly, there are really only a few types of certificates: CAs, web and email servers, and end-users. It is tempting to imagine a PKIX file system:

```
$ mkca /home/sysadmin/ipsec
$ cd /home/sysadmin/ipsec
$ mkserver --type IKE gwcert
$ mkuser --type IKE "Steven M. Bellovin"
$ ls "Steven M. Bellovin".*
Steven M. Bellovin.cert      Steven M. Bellovin.privkey
```

Those two files could be copied to the proper locations. Type "IKE" refers to an IETF-specified profile of what IKE certificates should contain.

Another source of the complexity is the nature of PKI, and in particular the reliance on commercial CA vendors. While arguably they're necessary for web use (but see below), they're not necessary for things like internal IPsec use. Yes, the computer that generates certificates is security-sensitive, but no more so than the computer that adds user logins. Again, running such CAs is much simpler if there is one simple specification for each common use case. Matters like key sizes are best left to experts.

The entire issue of the trust footprint of the Web's PKI has recently come under scrutiny. While I will not address the (obvious) security risks posed by every browser trusting hundreds of CAs, there is a complexity issue: navigating the twisty requirements to obtain "official" certificates can itself be daunting. Many vendors offer different levels of certificates, but the value proposition of these is unclear at best. A simple solution based on DANE [18] could ease this, especially for small sites. (This assumes, of course, that DNSSEC is simple to use. While I won't discuss it here, it also merits a close examination for unnecessary complexity.) DANE has been criticized as relying on the security policies of DNS registrars; for the most part, these companies do not have much experience with cryptographic security. This is likely true, and a full solution to the Web PKI problem must take this into account. Nevertheless, as a bootstrapping mechanism DANE certificates are far better than nothing, and nothing is precisely what most small sites offer when it comes to cryptographic security. If it were clearly understood what a basic web certificate should look like—a standard set of algorithms, the host names of example.com and www.example.com, and an expiration period of a few years—web server packages could generate the certificates and the DNS RRset at installation time.

Recommendation: The IETF should prepare a small set of PKIX profiles for common uses: web server, IKE gateway, email user, etc. Ideally, there should be no site-specific options, and an absolute minimum of fields to be filled in; these should be very obvious (e.g., username, web site hostname, etc.) or extremely clearly defined. Particular attention should be paid to client-side authentication certificates.

Recommendation: The W3C should encourage web server vendors to provide applications that easily generate and install simple site certificates.

Recommendation: The IAB should work with ICANN to facilitate deployment and use of DNSSEC and DANE.

4 Email

Email encryption has two main aspects: SMTP encryption and end-to-end encryption via S/MIME or PGP. SMTP encryption in turn divides into submission [19] and site-to-site email [15]. The three problems are very different.

Submission encryption—providing an encrypted channel for mail user agents to talk to their organization's or ISP's mail server—requires a server certificate and MUA configuration. Certificate generation is per Section 3. MUA configuration is trickier, though DANE-based solutions are appealing. Sites also have to decide on and specify acceptable cipher suites, given constraints of both security and implementation reality—but per the discussion of IKE, implementation interoperability is also hindered by endless choice. (The issues for secure IMAP and POP3 are essentially identical.)

Site-to-site SMTP encryption is a more interesting case, and one that requires some IETF work. It is of necessity opportunistic; flash-cutting every MTA to offer and require encryption is clearly impossible. This in turn requires very different encryption protocols. For example, an email submission service might very rationally decline to offer RC4, since it seems to be cryptographically weak. An MTA, however, should not do that; even weak encryption is far stronger than no encryption. There thus needs to be a different IETF-developed profile for MTA-to-MTA TLS configuration.

There is a also a thorny question about what an SMTP PKI should look like. Should SMTP servers use commercially-signed certificates? What are the proper semantics of certificate validation in the presence of MX records? Does Server Name Indication suffice [8]? What if the actual destination site uses DNSSEC but the target of an MX does not? These questions require a new RFC.

Finally, and perhaps hardest of all, we need to improve the availability of end-to-end encryption via S/MIME [16] or PGP [13]. There are many different issues here, including usability [21, 3]. The IETF's contribution here is severalfold. First, and most obvious, it should define an email end-user certificate profile. Second, it it should specify an LDAP profile for *external* certificate discovery. That is, if Pat wants to send email to chris@example.com,

Pat's MUA should have a simple way to ask example.com for Chris' certificate. LDAP [11] may suffice, but many companies are rightly concerned about about the confidentiality of their employee directories. A complementary design would entail attaching an extra MIME part to all outbound emails; that MIME part would contain the sender's certificate, even on otherwise unprotected email. That sort of viral certificate spread would help.

The most difficult problem, though, is certificate issuance. Corporations could easily issue certificates to their employees; an informational RFC (perhaps describing suitable tools) would help. Certificates for individuals are more problematic. ISPs could perhaps have automated certificate issuance servers linked to email addresses; they could then run the same sort of certificate server as corporations would do. Support for self-generated certificates, perhaps with some key continuity mechanism, are another possibility

Ad-supported services such as Gmail are trickier. It might be possible to use some variant of encrypted search for this [1]. That said, there are privacy problems; the same technology that a server might use to select ads could also be used by intelligence agencies to find persons of interest.

Finally, many users read their email via web browsers. It is not possible to do this securely for encrypted email; users should *not* trust downloaded JavaScript with their private key. A browser extension to decrypt selected sections of HTML would be necessary. [5] may be a useful starting point.

Recommendation: The IETF should define profiles for SMTP encryption, including the semantics for MX-redirected email.

Recommendation: The IETF should develop informational RFCs on enterprise use of encrypted email. In addition, it should develop (or develop a profile for) certificate distribution protocols.

Recommendation: The IRTF should foster development of advanced cryptographic techniques for encrypted search, both for users and and for ad-supported services.

Recommendation: The W3C should develop standards for encrypted HTML elements, and should encourage web server and browser vendors to support them.

5 Conclusion

Many IETF cryptographic protocols have too many options. (The same is true of many non-cryptographic IETF protocols, with similar deleterious effects on their usability and implementability.) This makes them hard to implement, hard to configure, and hard to use. It is too late to repair most older protocols, but suitably simple profiles can be defined. Going forward, much more attention should be paid to design simplicity.

References

[1] Adam J Aviv et al. "Ssares: Secure searchable automated remote email storage". In: *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. IEEE. 2007, pp. 129–139.

[2] Steven M. Bellovin and Robert G. Moskowitz. *Client Certificate and Key Retrieval for IKE*. Obsolete Internet draft. Nov. 2000. URL: https://www.cs.columbia.edu/~smb/papers/draft-ietf-ipsra-getcert-00.txt.

[3] Simson L. Garfinkel and Robert C. Miller. "Johnny 2: a user test of key continuity management with S/MIME and Outlook Express". In: *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*. Pittsburgh, Pennsylvania: ACM, 2005, pp. 13–24. ISBN: 1-59593-178-3. DOI: http://doi.acm.org/10.1145/1073001.1073003.

[4] Nadia Heninger et al. "Mining your Ps and Qs: Detection of Widespread Weak Keys in Network Devices". In: *Proceedings of the 21st USENIX Security Symposium*. 2012. URL: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final228.pdf.

[5] E. Rescorla and A. Schiffman. *Security Extensions For HTML*. RFC 2659. RFC Editor, Aug. 1999, pp. 1–4. URL: http://www.rfc-editor.org/rfc/rfc2659.txt.

[6] P. Hoffman. *SMTP Service Extension for Secure SMTP over Transport Layer Security*. RFC 3207. RFC Editor, Feb. 2002, pp. 1–9. URL: http://www.rfc-editor.org/rfc/rfc3207.txt.

[7] M. Crispin. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. RFC 3501. RFC Editor, Mar. 2003, pp. 1–108. URL: http://www.rfc-editor.org/rfc/rfc3501.txt.

[8] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. RFC 3546. RFC Editor, June 2003, pp. 1–29. URL: http://www.rfc-editor.org/rfc/rfc3546.txt.

[9] S. Santesson, R. Housley, and T. Freeman. *Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates*. RFC 3709. RFC Editor, Feb. 2004, pp. 1–21. URL: http://www.rfc-editor.org/rfc/rfc3709.txt.

[10] S. Kent and K. Seo. *Security Architecture for the Internet Protocol*. RFC 4301. RFC Editor, Dec. 2005, pp. 1–101. URL: http://www.rfc-editor.org/rfc/rfc4301.txt.

[11] *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*. RFC 4510. RFC Editor, June 2006, pp. 1–7. URL: http://www.rfc-editor.org/rfc/rfc4510.txt.

[12] V. Manral. *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 4835. RFC Editor, Apr. 2007, pp. 1–10. URL: http://www.rfc-editor.org/rfc/rfc4835.txt.

[13] J. Callas et al. *OpenPGP Message Format*. RFC 4880. RFC Editor, Nov. 2007, pp. 1–90. URL: http://www.rfc-editor.org/rfc/rfc4880.txt.

[14] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, Aug. 2008, pp. 1–104. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[15] *Internet Message Format*. RFC 5322. RFC Editor, Oct. 2008, pp. 1–57. URL: http://www.rfc-editor.org/rfc/rfc5322.txt.

[16] B. Ramsdell and S. Turner. *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification*. RFC 5751. RFC Editor, Jan. 2010, pp. 1–45. URL: http://www.rfc-editor.org/rfc/rfc5751.txt.

[17] C. Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 5996. RFC Editor, Sept. 2010, pp. 1–138. URL: http://www.rfc-editor.org/rfc/rfc5996.txt.

[18] R. Barnes. *Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)*. RFC 6394. RFC Editor, Oct. 2011, pp. 1–12. URL: http://www.rfc-editor.org/rfc/rfc6394.txt.

[19] R. Gellens and J. Klensin. *Message Submission for Mail*. RFC 6409. RFC Editor, Nov. 2011, pp. 1–20. URL: http://www.rfc-editor.org/rfc/rfc6409.txt.

[20] Shreyas Srivatsan, Maritza Johnson, and Steven M. Bellovin. *Simple-VPN: Simple IPsec Configuration*. Tech. rep. CUCS-020-10. Department of Computer Science, Columbia University, July 2010. URL: https://mice.cs.columbia.edu/getTechreport.php?techreportID=1433.

[21] Alma Whitten and J.D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0". In: *Proceedings of Usenix Security Symposium*. 1999. URL: http://db.usenix.org/publications/library/proceedings/sec99/whitten.html.

[22] Tatu Ylonen. "SSH – Secure Login Connections over the Internet". In: *Proceedings of the Sixth Usenix Unix Security Symposium*. July 1996, pp. 37–42. URL: http://www.usenix.org/publications/library/proceedings/sec96/ylonen.html.