Thinking Security

Stopping Next Year's Hackers

Steven M. Bellovin

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Thinking Security

Table 6.3: Recommended IETF Cryptographic Protocols

Protocol	*Uses*
IPsec	General network-layer encryption [S. T. Kent and Seo 2005]
TLS	Simple encrypted circuits [Dierks and Rescorla 2008]; use Version 1.2 or newer
HMAC/SHA-2	Message authentication [NIST 2015b; Weinrib and Postel 1996]
Security/Multipart	Secure email and similarly formatted text [Ramsdell and Turner 2010]
XMLDSIG	Signed XML [Eastlake, Reagle, and Solo 2002]
CMS	Secure objects [Housley 2009]

designed algorithm Skipjack [Biham, Biryukov, and Shamir 1999]. I marveled aloud about it to a very knowledgeable friend, noting that taking away just one round made such a difference. His reply: "You call it a weakness; I call it good engineering." Maybe the NSA really has that deep an understanding of cipher design—or maybe the safety margin is low. We'll know for sure in a few decades; for now, I do recommend AES's use.

6.7 Analysis

As long as we stay with today's standard models of interaction—client/server, for simple transmission or object security—I don't expect significant changes in protocols. The important cases are handled reasonably well; the difficult problems, such as preserving access to the key used to encrypt a backup tape, are inherent in the problem statement. One can't rule out breakthroughs, of course—prior to Diffie and Hellman's work [1976], no one in the civilian world had even conceived of the concept of public key cryptography—but such insights occur once in a generation at most.

An interesting question is what will happen with different interaction models, such as inherently 3-way or 4-way sessions with no one universally trusted party. (No, I don't know what sort of really popular application would require that; if I did, I might find myself a venture capitalist.) The key management protocol might be a bit tricky, but lots of other things get complicated with more than two parties.

My very strong statement that people should stick with well-understood, standardized protocols is extremely likely to stand. Remember the quote from Needham and Schroeder discussed earlier. It goes down as one of the more prescient statements in a technical

paper; indeed, it took 18 years for what in retrospect was an obvious flaw in one of their own schemes to be discovered [Lowe 1996].

A place I hope we'll see improvements is the usability of cryptographic technology. A certain amount of over-the-wire flexibility is mandatory, if only to permit migration to different algorithms over time. Unfortunately, this protocol flexibility generally manifests itself as more buttons, knobs, and sliders for the poor, benighted users, who neither know nor care about, say, the rationale for using what is essentially SHA-2-256 truncated to 224-bit output [Housley 2004] to better match triple-DES; all they know is that they're presented with yet another incomprehensible option. Couple that with the inherent issues of trust management—who *really* owns a given key?—and you end up with applications that very few people can actually use successfully [Clark et al. 2011; S. L. Garfinkel and R. C. Miller 2005; Whitten and Tygar 1999]. Thus far, cryptography has succeeded where users weren't given any decisions to make; if it's all hidden away under the hood, people accept it and feel better for having used it.

Given the concerns I expressed in the last section, do I think that the algorithm recommendations in the previous section will change soon? It's important to note the assumptions behind my suggestions. First, I'm assuming no drastic improvements in the price/performance of hardware. Even if I'm off by a factor of 100—about 7 bits of keylength—it probably doesn't matter. If Moore's Law runs into a brick wall within the desired secrecy lifetime (it will at some point; it seems extremely unlikely that we can produce gates smaller than an atom), the situation is better yet for the defender. I'm also assuming no unforeseen cryptanalytic results. The algorithms and protocols discussed are all well studied, but breakthroughs happen. Again, it is unlikely that a modern algorithm will suddenly shatter, so there will still most likely be a large work factor required—but that's a prediction, not a promise.

Large-scale quantum computers, should they ever become real, will change things significantly. In particular, there are efficient quantum factoring algorithms [Shor 1994]; that will probably rule out all of today's public key algorithms. However, it's still unclear whether such computers are possible.

Finally, remember Shamir's advice in 1995: "Don't use cryptographic overkill. Even bad crypto is usually the strong part of the system."[6]

Given all that, where should cryptography be used? The security advantages of universal encryption are clear enough that I won't bother reviewing them; the disadvantages are not always obvious beyond the problem of "what do I do if lose my key?" That latter is especially serious (albeit obvious) for object encryption, enough so to merit a blanket statement: do not encrypt stored objects unless the risk is *very* great; if you do, take

6. "Notes on 'Cryptography—Myths and Realities,' a talk by Adi Shamir, CRYPTO '95," http://www.ieee-security.org/Cipher/ConfReports/conf-rep-Crypto95.html.

adequate precautions to preserve keys at least as carefully. If you're encrypting backup media, stage regular practice drills in retrieving files (a good idea in any event, if only to test the quality of the backups and your operational procedures for using them) and keys.

There are other disadvantages as well. The difficulty in network operations has long been recognized: it's impossible to see the contents of an encrypted message, even if you need to understand why it's causing trouble. This can also cause problems for security folk; network intrusion detection systems can't peer inside, either. It would be nice if we had some sort of magic cryptanalysis box that could only look at packets with the evil bit set [Bellovin 2003]; thus far, no one has developed a suitable one.

There's a more subtle issue, though. Obviously, protecting cryptographic keys is extremely vital. A key that doesn't exist on a machine can't be stolen from it; a key that is there but itself strongly encrypted is likewise effectively immune to compromise. This poses a dilemma: the best way to keep a key safe is to avoid using it. Of course, if we never use it, it's rather pointless to have it. Nevertheless, we can draw an important conclusion: high-value keys should be employed as sparingly as possible and removed from machines when they're no longer necessary. Given the rate of host compromise, a long-term key in constant use—say, for routinely signing all outbound email—is at great risk; a recipient should therefore attach a lot less value to the signature than one produced by a key that is rarely used and well protected at other times. In the absence of strong key storage (and general-purpose hosts rarely have such facilities), strong overall security therefore requires different keys for different sensitivity levels, suitable (and suitably usable) software to let users manage such complexity, plus a lot of user education and training on how to behave.

Chapter 7

Passwords and Authentication

> "I haven't told him about you, but I have told him to trust absolutely whoever has the key word. You remember?"
>
> "Yes, of course. *Meshuggah*. What does it mean?"
>
> "Never mind." Abrams grinned.
>
> *Ensign Flandry*
> —POUL ANDERSON

7.1 Authentication Principles

Authentication is generally considered to be one of the most basic security principles. Absent bugs—admittedly a very large assumption—authentication effectively controls what system objects someone can use. In other words, it's important to get authentication right.

Most discussions of authentication start by describing the three basic forms: something you know (e.g., a password); something you have, such as a token or a particular mobile phone; and something you are, that is, some form of biometric. While this categorization is indeed useful, it understates the *systems* nature of authentication. The total environment—who will use it, how you deal with lost credentials, what the consequences are of lack of access or access by the wrong person, and more—is at least as important. The most important question of all is how people will actually use the authentication technology in the real world.

Another important thing to remember: we authenticate in many more situations today than we did in the not very distant past. Once upon a time, we would log in to a work machine or two. Now, we log in to many different web sites, mail systems, devices, doors, and even cars. The challenges, and hence the solutions, can differ.

7.2 Passwords

This book is about demythologizing security. Few areas are in sorer need of that than passwords. The trouble started with a classic—and still correct—paper by Morris and Thompson [1979], which among other things showed why guessable passwords were bad. However, that result is often misapplied today; in particular, insufficient attention is paid to the threat model (what assets you are protecting, and against whom) and to the tension between security and usability. ([Singer, W. Anderson, and Farrow 2013] gives a good presentation of the history of how we got to where we are and of the many mistakes and unjustified assumptions made along the way.)

The problem of password strength is easy to explain. Simple experiments using the classic Unix password-hashing algorithm show that given a hashed password, an early 2009 laptop—by no means a state-of-the-art computer—can try more than 150,000 password guesses per second. If the enemy has 1,000 such computers—trivial for any self-respecting botnet owner—all possible passwords of up to eight lowercase letters can be tried in less than half an hour. Even if digits are included in the mix, the guessing time is still only about 5¼ hours. The attacker's problem is often even simpler than that; people don't pick truly random strings like "gisegpoc" or "A*9kV#2jeCKQ"; they prefer words or names, or simple variants of these. A recent study based on the RockYou dataset, a list of passwords posted by some hackers, showed that 19 of the top 20 passwords found fit this model [Weir et al. 2010]; the list of most frequent choices included "`abc123`", "`princess`", and the ever-popular "`password`" (#4 on the list, preceded only by "`123456`", "`12345`", and "`123456789`").

It seems simple enough to solve—just choose strong passwords!—but it's very problematic in practice. Users today don't have just one or two passwords to remember, they have many dozen passwords—at the moment, I personally have well over 100—with importance ranging from online financial accounts down to stores, for-pay news sites, social networking sites, and assorted random places that simply want you to register. I cannot possibly remember that many different passwords, let alone that many strong ones. Besides, each site has different rules for what a password should be like. Some sites insist on punctuation; others ban it. Some want long passwords; others have length limits. Some insist on mixed case; others don't check that, but are case sensitive; still others are case

insensitive. The most restrictive set of rules I've seen comes from a US Customs and Border Protection site used by the public.

This is the actual text, copied and pasted from their web site:

- Minimum Length : 8
- Maximum Length : 12
- Maximum Repeated Characters : 2
- Minimum Alphabetic Characters Required : 1
- Minimum Numeric Characters Required : 1
- Starts with a Numeric Character
- No User Name
- No past passwords
- At least one character must be ~ ! @ # $ % ^ & * ( ) - _ + ! + = { } [ ] \ | ; : / ? . , < > " ' ` !

Good luck remembering your password for it, especially since you'll use that account about once a year.

Suppose you do forget your password. What then? *Every* real-world system has to have some provision for password recovery or reset. This is very much a trade-off between cost and security; except for high-value sites (banks, employers, etc.), there are rarely secure solutions. This issue is discussed in more detail in Section 7.5.

Another important issue is the change in threat model. When Morris and Thompson wrote their paper, the primary danger was theft of the password file, followed by an offline guessing attack. Remember that in those days, /etc/passwd was world readable; anyone with unprivileged access to the machine could grab a copy. They certainly realized that the host or its login command could be subverted, in which case it was game over, but that wasn't the threat model their solution was intended to deal with. Unfortunately, today that is one of the most serious problems. The attackers aren't stupid; phishing attacks, compromising servers, and compromising client hosts are the easiest ways to grab passwords. But if the attacker has accomplished any of these, a strong password is no defense at all; a keystroke logger doesn't care about the number of special characters you've chosen. Is password strength obsolete [D. Florêncio, Herley, and Coskun 2007]?

Well, not necessarily. As is frequently the case, the correct answer is "it depends." In this case, it depends on the kinds of attacks you're trying to defend against and on your total system design: