Crypto Agility: Research, Industry, and Policy Implications

Steven M. Bellovin
https://www.cs.columbia.edu

Algorithms Age—Agility is Mandatory

1996	2016	2036
DES, 3DES	AES-128	AES-256? – but are there key schedule issues?
MD5, SHA-1	SHA-2-256	SHA-2-512? SHA-3-512?
RSA, DSA, DH	ECDSA, ECDH	Post-quantum algorithm

You can't flash-cut the Internet!

Cryptography is *Hard*

- Ordinary cryptographic negotiations are very hard to get right
- Agility adds new failure modes, such as downgrade attacks
  - Look at all of the TLS problems stemming from the need for exportable ciphers
- Most organizations *won't* get it right

Negotiation

- Even the IETF got hash function negotiation wrong
  - The IETF is probably better than most at cryptographic protocols
- Strong recommendation: most companies should *never* design their own crypto agility
  - Well, or any other part of the crypto
- Corollary: appropriate organizations should design standard negotiation toolkits, APIs, code, etc.
  - But—given that most companies shouldn't design their own protocols, does this matter?

But…

- There are a few situations that demand universal compatibility
  - DNSSEC, BGPSEC, etc.
- Must design for transition
- This is very hard—if nothing else, what are the desired semantics of dual-mode?
- How do we negotiate modes of operation? Protocol versions?

Embedded Systems

- Embedded systems have an upgrade lifetime of very few years
  - The vendor moves on to a newer, more capable system, and relies on the newer capabilities
- Some systems last longer than the embedded cryptographic algorithms
  - Example: cars versus hash functions
- If we can't update the system, how can we add new algorithms?

Upgrading Algorithms?

- Is it possible to have separate, longer-lived algorithm updates?
- What about devices without easy upgrade paths?
- What if the vendor disappears?
- An open API for crypto algorithm updates?
  - Note: implies the need for variable-length over-the-wire fields
  - Sometimes, there are semantic issues, e.g., authenticated encryption versus separate encryption and MAC
  - Dynamic downloads? Crypto via downloaded JavaScript? How is this protected?!

Parameterized Algorithms

- The lifetime of many algorithms can be extended if certain parameters can be changed or increased
- Example: a higher iteration count, perhaps different S-boxes, etc.
- Negotiate these instead?
- Downgrade attacks, correctness, etc., are still issues, but this approach might help agility for embedded systems
- Recommendation: new algorithms should be tunable this way—but are there bounds?
  - Still need variable-length fields

What Algorithms "Should" a Site Use?

- Try to prevent certain downgrade attacks
- Is it possible to have out-of-band knowledge of a site's "correct" offerings?
  - In the DNS?
  - In a local cache?
- How can a site change that? What if it's out of sync?
  - You never want the same data in two places
- What if a site needs to roll back to a previous system version?

Retrofits

- Out-of-band knowledge is especially important during transitions
- During such periods, older algorithms will still be common—but they should be avoided if possible, because they're probably too weak
- How are downgrade attacks prevented during this interval?

Ratcheting Up

- At some point, old algorithms should no longer be permitted
  - The same is true for too-small parameter values
- How should a system decide when the “ratchet point” is reached?
  - Not all systems will have knowledgeable administrators
- When it’s reached, weak choices must be permanently disabled—a site “ratchets up”
  - How are other sites told that?
- Must avoid using negotiation to weaken crypto

Undesirable Algorithms

- Some countries insist on use of certain algorithms, presumably because they're easy to crack
- Crypto agility makes this easier
- Backwards compatibility is sometimes bug-wards compatibility—and this is exploitable

Privacy Impacts

- Some solutions affect privacy
- Example: how does a client publish its ratchet point without a static identity?
- Example: If an embedded system dynamically downloads new algorithms, it reveals what it is doing
  - What if some porn movie requires 2718-bit RSA and 314-bit AES? Fingerprinting?

Conclusions

- We must have agility—it's a fundamental requirement
- New *algorithms* should be designed for agility
  - Iterations, field sizes, etc.
  - Need to understand (and *prove*) bounds on these values
  - Can we parameterize modes of operation?
- Modes of operation and protocols are much harder—long-term research at best
- Data structures need to accommodate this
- Agility negotiation must be protected
  - And must be privacy-preserving
- Research is needed on privacy-protecting, secure ratcheting