Deploying New Hash Functions

Steven M. Bellovin
smb@cs.columbia.edu

Eric K. Rescorla
ekr@networkresonance.com
Deploying Hashes

The Problem

• We have to deploy new hash functions — if not today, at some point soon

• We try for algorithm-agility in our protocols — but certificates are a special case

• Certificates rely on hashes

• Goal: maintain security while new code is deployed

• Did we get it right?

• No...
Gradual Conversions

- We cannot upgrade all systems at once
- Support for new hash (and signature) algorithms will appear gradually
- Newer systems need to be able to “switch-hit” — use new algorithms when talking to other newer systems, but fall back to old algorithms when talking to legacy systems
- This requires some sort of signaling
- The signaling has to be secure, to prevent downgrade attacks
Protocols Analyzed

- We looked at S/MIME, TLS, and IPsec/IKE/IKEv2; we have preliminary results for DNSSEC

- None of them got it right

- Note: for brevity, this talk will not discuss hash functions use with HMAC or as PRFs; see the paper for details
The Root of the Problem

- We had MD5 in 1992, and SHA-1 in 1995. In other words, for the entire commercial life of the Internet we have had the same two algorithms
- Everyone supported both; there was no need for signaling
- Unused protocol paths are just as bad as unused code paths
S/MIME

- If the sender has more than one certificate, which should be used for signing email?
- If you don’t have the receiver certificate, you have to use old algorithms (but \textit{never} use MD5 for signing)
- Eventually, switch to the newer algorithm as the default; users can resend if needed (mail clients should cache such information)
- Multiple signatures are defined in the spec, but many implementations won’t handle this case properly
- If you have the receiver’s certificate(s), use the newest algorithms possible
- There is a proposed SMIMECapabilities certificate extension, but it’s not yet standardized, let alone implemented
TLS

- TLS server certificates are the most important case for upgrades
- Need TLS extension (or, possibly), overloaded ciphersuite for client signaling to server
- Similarly, the server should be able to signal what client certificates it can accept (though client-side certificates are rare)
- Other situations: RSA digital signatures in TLS use MD5 concatenated with SHA-1. Best option: have newer implementations use the hash algorithm from the signer’s certificate
- Similar considerations for the TLS Finished message
IPsec and IKE

- IKEv2 and IKE Main Mode have negotiation messages at the right time, but there is no negotiation of certificate hash function or certificate signature algorithms.
- It is possible to overload the meaning one option to select hash function.
- IKE Aggressive Mode (which has four different variants) uses hash functions before any negotiation.
- In some situations, heuristics based on certificates can be used.
- Possible practical solution: IPsec is used primarily in closed environments.
Preliminary Analysis of DNSSEC

- Difficult, because no possibility of negotiation; server must send out all possible signatures
- To guard against downgrade, the over-the-wire protocol is probably sufficient
- The DS message should be overloaded to indicate which algorithms should be expected
- This is a change in interpretation, and hence requires a new RFC and code changes
- This will increase DNS message size
Signature Algorithms

- Most of our analysis applies to signature algorithms, too
- Note that DSA can only be used with SHA-1
- Adapting to new signature algorithms is harder than new hash functions, since the heuristics we sometimes suggest won’t work
Deploying Hashes

Estimated Conversion Timeline

1 year  Design of new protocol features by the IETF
1-2 years  Design, code, and test of new features by vendors
2-5 years  Deployment by the user community — note that many machines are never upgraded, merely replaced

Standardization of a new hash function can proceed in parallel with protocol redesigns. If a new hash technique requires a different API, it may lengthen the design/code/test time.

Given the modest threat posed by collision attacks (except, of course, for signed email), the speed of the upgrades may be driven by support for ECC.
Deploying Hashes

Recommendations

S/MIME  Support multiple signatures properly; stop using MD5; add SMIMECapabilities certificate extension

TLS    Add signaling for server and client certificates; change digitally-signed element and Finished message definition

IPsec  Add hash function signaling in the initial SA exchange

DSA    Define DSA-2 or way to use DSA with other hashes

Vendors Add policy and preference knobs, for users and administrators
Conclusions

• Agility is hard to get right unless you actually try deploying a new algorithm

• All of the protocols we looked at need more work. Other protocols — , SECSH, OpenPGP, and more — should be examined by the appropriate WGs.

☞ Most protocols need either an updated version or a BCP describing how to manage the transition.

• Implementors need to think about it, too

• Most of our analysis applies to new signature algorithms