SHA-3 for Internet Protocols

Quynh Dang & Tim Polk
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology

Overview

- SHA-3 Competition background and status
- SHA-3 (and SHA-2)
  - Security, Functionality, and Performance
- Useful Data Points For Protocol Analysis
- Review of Specific Protocols

SHA-3 Competition: Background

- 2004-2005 Wave of new cryptanalysis
  - Wang, Biham, Joux, Kelsey all published significant papers….
  - Cast doubt on existing hash standards and the traditional Merkle-Damgård construction
- 2005, 2006 NIST Hash Function Workshops
  - Industry and academia encouraged NIST to run a competition and contribute to planning
- 2007 NIST organized SHA-3 competition
  - 64 candidates submitted 31 Oct. 2008

Requirements for SHA-3

- Plug-compatible with SHA-2 in current apps
- Support digital signatures, hash-based MACs, PRFs, RNGs, KDFs, etc.
- Required security properties
  - Collision resistance of approximately n/2 bits,
  - Preimage resistance of approximately n bits,
  - Second-preimage resistance of approximately n-k bits for any message shorter than 2^k bits,
  - Resistance to length-extension attacks.

SHA-3 Competition: Current Status

- Five Finalists identified late in 2010.
  - Blake, Grøstl, JH, Keccak, Skein
- Final tweaks submitted January 2011.
- Final Workshop held last week (March 2012) in Washington DC
- NIST will announce the winning candidate late in 2012

http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

SHA-3: Security

- Confidence in the security of SHA-3 candidates is very high
- SHA-3 candidates are based on new constructions
  - not vulnerable to well known attacks on Merkle-Damgård construction (e.g., length extension attack)
- However, cryptanalysis since 2005 has actually eased our concerns about SHA-2

SHA-3: Performance (1 of 3)

- There is no free lunch this time: collision resistance takes a lot of computation
  - We will not replicate the across-the-board performance increase we got with AES (as compared with Triple DES)

SHA-3: Performance (2 of 3)

- SHA-2 is surprisingly efficient and could be faster (or smaller) in some environments than SHA-3, no matter which of the 5 candidates is selected.
- SHA-256 is competitive in low-end SW platforms and "constrained" HW
- Depending on the selected algorithm, SHA-3 could be faster than SHA-2 in some environments
  - Skein & Blake faster in software on high end computing platforms
  - Keccak is fast in hardware in general

SHA-3: Performance (3 of 3)

- However, SHA-3 *potentially* offers significantly better performance for one important function: hash-based MACs on short messages
  - Only requires a single pass since we aren't worried about length extension attacks
- In fact, a single pass keyed hash for some candidate algorithms *would be faster than a SHA-1 HMAC for short messages*

The (New) Real Question

- Which Candidate best complements SHA-2?
  - SHA-2 is not apparently broken
    - SHA-2 collision resistance seems fine but SHA-3 candidates have greater security margins
  - All candidates have much higher multicollision resistance than SHA-2 and fix the other generic limitations of Merkle-Damgård
  - No candidate has dramatically better performance across the board
    - Some candidates have much better performance for some kinds of implementations
    - All will support a single pass keyed MAC
  - Some candidates offer extras
    - Wide block cipher, authenticated encryption

Questions for Protocol Developers

- Should NIST emphasize high-end or low-end HW or SW or any combination of these in our selection process?
- For low-end HW, should NIST emphasize energy consumption, throughput to area ratio, minimum size, or any combination of these in our selection process?
- Should NIST specify a single algorithm for all digest sizes, or two algorithms as in SHA-2 (SHA-256 and SHA-512)?

Data Points For Protocol Analysis (1/2)

- How are hash algorithms used?
  - Hash, Signature, HMAC, KDF, PRF?
- Is SHA-2 already specified for that protocol?
- Is the protocol agile in practice?
  - Is cipher suite negotiation supported?
  - Do all end points need to be updated before use?

Data Points for Protocol Analysis, (2/2)

- Is SHA-2 widely implemented? used?
  - If so, SHA-3 would be a straightforward and suitable backup
- Would SHA-3 present specific practical advantages?
  - E.g., single pass MACs, tree mode?
  - If so, consider SHA-3 for the Mandatory To Implement cipher suite

Initial Thoughts on Specific IETF Protocols and SHA-3

- Reviewed about a dozen protocol families
- High-level analysis for a few
  - TCP-AO
  - Constrained Devices (core, roll, 6lowpan)
  - Mature X.509 protocols (PKIX, S/MIME, TLS)
- Defer more detailed analysis once the selection has been made

Quick Example: TCP-AO

- TCP-AO uses hash functions to derive keys and generate HMACs
  - SHA-1 HMACs are truncated to 96 bits
  - No support for SHA-2
- Has algorithm agility, so it would be easy to specify
- Assuming use of a single pass MAC, SHA-3 would provide a better alternative than SHA-256
  - Performance would be comparable to current SHA-1 based implementations
- No strong justification for mandatory to implement
  - No real security motivation (96 bit tag!)

Constrained Devices

- Core, roll, and 6lowpan
  - Devices may have limited resources (e.g., processing power, memory, or battery power)
  - Common “solution” in this market sector is weak security
- Single pass hmacs consumes less power
  - SHA-3 candidates which provides additional functionality (e.g., authenticated encryption) would enable memory/area savings

PKIX

- Hashes primarily support digital signatures on “messages” ranging from 1kbyte to tens of megabytes, but …
- Review of ASN.1 in RFC 5912 shows hash algs also used in nearly every plausible mode
- SHA-2 support well specified, but migration process has been painful
- For general purpose certs, infrastructure can’t issue until “all” end systems are ready

S/MIME

- Hashes primarily support digital signatures on "messages" ranging from a few kbytes to tens of megabytes and certificate handling
- SHA-2 support well specified in RFC 5754, and starting to be deployed, but not widely used
- Security of SHA-1 (which is widely used) is inadequate
- *SMIMECapabilities* conveys sender's set of supported crypto algorithms, but we still have a bootstrap issue

TLS

- Uses hash algorithms extensively
  - e.g., PRF, hmac, and certificate handling
- SHA-256 well specified in most recent protocol specs
- Messages can be relatively large, so the incremental performance improvement from single pass MAC may be limited
- Ciphersuites are negotiated so incremental deployment might be practical

Questions?