NIST Hash Competition

Bill Burr

NIST

June, 2008

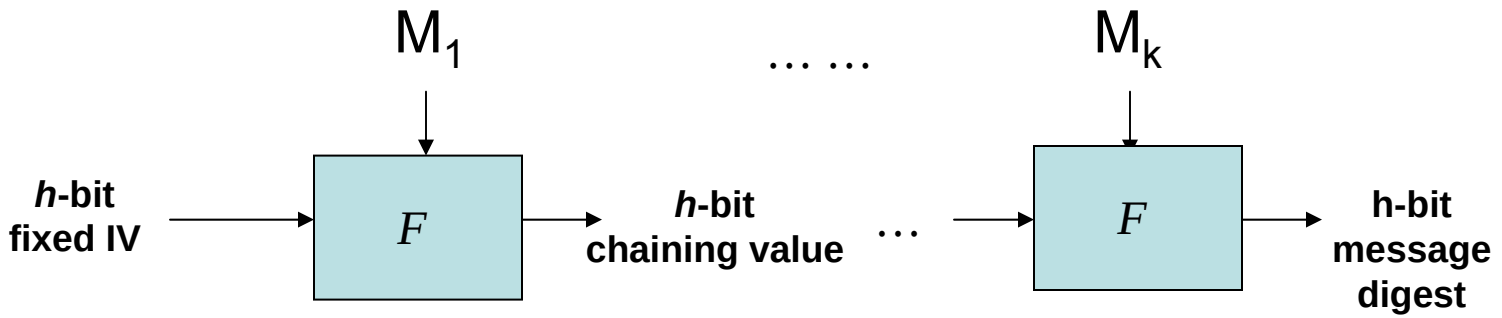
National Institute of Standards and Technology

What is a Hash Function?

- Hash functions take a variable-length message and reduce it to a shorter fixed *message digest*
- Core requirement: Use ***hash(x)*** as a stand-in for ***x*** in digital signatures, MACs, file comparisons, etc.
- Many applications: “Swiss army knives” of crypto:
  - Digital signatures (with public key algorithms)
  - Random number generation
  - Key update and derivation
  - One way function
  - Message authentication codes & user authentication (with a secret key)
  - Code recognition (list the hashes of good programs or malware)
  - Commitment schemes

Merkle-Damgard Hashing

- Most widely used hashes use MD
- Break message into blocks
  - M_1, M_2, M_3…M_k (pad out last block)
- "Compression function," *F*, mixes each block successively into *h*-bit state
- Last output of compression function is the *h*-bit message digest.



Properties of Hash Functions

- Goal: use ***hash(x)*** as a stand-in for ***x***.
- Preimage Resistance
  - Given ***y***, can't find ***x*** to satisfy ***y = hash(x)***
  - Roughly means the hash is "one-way."
- Second Preimage Resistance
  - Given ***x***, can't find ***x**** so that ***hash(x)=hash(x*)***
  - can't match the hash of some given string
- Collision Resistance
  - You can't find ***x,x**** to satisfy ***hash(x)=hash(x*)***
  - Can't find two strings with same hash

Work Factors

- Preimage Resistance for any *n*-bit hash
  - Given ***y***, find ***x*** to satisfy ***y = hash(x)***
  - Need to hash about 2^n messages
- Second Preimage Resistance
  - Given ***x***, find ***x**** to satisfy ***hash(x)=hash(x*)***
  - Need to hash about 2^n different messages
- Collision Resistance
  - Can't find ***x,x**** to satisfy ***hash(x)=hash(x*)***
  - Need to hash about $2^{n/2}$ different messsages

Results in the last 4 years

- Lots of new results on finding collisions
  - Down in the bits of the hash function
  - MD4, MD5, some Haval variants, RIPE-MD, SHA0, SHA1, Tiger
  - Potentially a lot of practical impact
- Generic second preimage and “herding” attacks
  - Attacking the Damgard-Merkle structure
  - Researchers have started to combine with collision attacks and turn them into practical attacks
  - prediction of winner of US 2008 presidential elections

Nostradamus Attack

- A good illustration of what can be done with collision attacks
  - Collisions on meaningful message
- Nov. 30, 2007: Stevens, Lenstra & Weger posted 10 predictions that each of the following would win the US 2008 Presidential Election
  - John Edwards, Fred Thompson, Ralph Nader, John McCain, Mitt Romney, Jeb Bush, Al Gore, Barack Obama, Oprah Winfrey, Paris Hilton & 2 mystery winners
- All predictions are PDFs with the same MD5 hash!
- Used Sony Playstations to mount the attack
  - Actually a very powerful (if inexpensive) machine
- Combines Wang style differential collision attack, Joux multicollisions and Kelsey's Nostradamus attack.
  - http://www.win.tue.nl/hashclash/Nostradamus/

Hashes: the problem

- MD4, MD5, SHA-0, SHA-1,SHA-256, SHA-512
  - All related Merkle-Damgard hashes roughly descended in this order
- SHA-1, SHA-256 & SHA-512 are FIPS
- MD4, MD5, SHA-0, SHA-1 now broken
- SHA-256 & SHA-512 design never fully explained
  - Are the SHA2's next?

The Impact of Collisions

- Collisions have a big impact when:
  - Attacker chooses messages for target to sign & Target hashes and signs
- Damaging Collision attacks are harder if:
  - Same party creates message & signs it
    - Nonrepudiation is the issue
  - Target appends an unpredictable prefix to message to be signed
- *Current SHA-1 collision attacks don't seem to affect HMAC.*

NIST Hash Function Policy

- Federal Users may use SHA-2 family hash functions (SHA-224, SHA-256, SHA-384, & SHA-512) for all hash function applications.
- For digital signatures and other apps that require collision resistance, Federal users:
  - Should convert to SHA-2 as soon as practical, but
  - Must stop using SHA-1 for these apps by end of 2010
- Federal users may use SHA-1 after 2010 for:
  - HMAC
  - Key derivation
  - Random number generation
  - To verify old signatures (signed before 2011)

SHA-3 Hash Competition

- Motivated by collision attacks on commonly used hash algorithms, particularly MD5 & SHA-1
  - No actual collisions yet announced on SHA-1
    - We think SHA-1 collision work factor ≈ 2^{60} operations
- Held 2 hash function workshops
- Jan 2007 proposed criteria for new hash function comment period
- Many comments received
- “SHA-3” Competition announced Nov. 2, 2007

SHA-3 Competition Timeline

- 1Q07 draft submission criteria published
- 11/2/07 Federal Register Announcement
- 8/31/08 Preliminary submissions:
  - NIST will review for completeness by 9/30/08
- 10/31/08 Final submissions due
- 2Q09 First Candidate Conference
- 2Q10 Second Candidate Conference
- 3Q10 Announce Finalist Candidates
- 4Q10 Final Tweaks of Candidates
- 1Q12 Last Candidate Conference
- 2Q12 Announce Winner
- 4Q 12 FIPS package to Secretary of Commerce

SHA-3 Criteria: these didn't make it

- Many comments to separate compression function from the iterated structure.
  - NIST doesn't understand how to make this work, without assuming a great deal about the iterated structure in advance
- Number theoretic hash function
  - Not ruled out but "drop in" makes it hard for "number theoretic" hashes

SHA-3 Criteria

- Want an “On-line” hash function
  - Some cryptographers would like an “in-memory” alternative
    - Doubtful an in-memory hash would be widely used – big latencies
- Security is most important criterion
  - Problem is limited cryptanalysis resources
- “Drop in” compatibility required for
  - Current digital signature standards
  - HMAC
  - NIST RNGs & key derivation functions
- Diversity is a good thing
  - Don’t want the same attack to fell SHA-2 & SHA-3
- Requirements about “generic” attacks
  - Joux multicollisions, etc.
  - Resistance to attacks > $2^{n/2}$ not required, but get extra credit

SHA-3 Cryptanalysis

- Cryptanalysis of candidates
  - Long pole in the tent.
  - Very labor intensive
  - Few people can do it
- NIST will depend heavily on crypto community for cryptanalysis
  - We expect to have John Kelsey and 2 or 3 well qualified guest scientists for SHA-3 cryptanalysis
  - In AES competition NIST had no in-house cryptanalysts

Selecting SHA-3 Finalists

- Might get 30 or more SHA-3 submissions
- Need to cut this down to about 5 finalists
  - May have 10 or 15 pretty darn good candidates
  - Want to be fair and thorough, but
    - Must focus cryptanalysis resources on a few candidates
    - A fairly small blemish may kill an initial candidate
      - Bad English writing skills may hurt some candidates
  - Our in-house cryptanalysts should help here
- Some performance data is fairly easy to collect
  - But good hardware data may take longer
  - Selection of finalists may depend heavily on performance on "Wintel/Mactel" desktop computers
    - Arguably not the critical platform
- Some good algorithms won't make the initial cut
  - Some folks are almost bound to be unhappy

Other SHA-3 Thoughts

- Balance between submission requirements and resources required
  - Want academics to be able to play
  - A team of 2 academics designed Rijendael (AES)
- "Sponge Model"
  - Interesting generalization of hash functions, but maybe a bit much to impose for the competition
  - Will probably influence our thinking
- Tried to allow design flexibility
  - Some cryptographers probably want more specifics
  - We want them to make design choices and argue why their choice is right.

John Kelsey's slides on SHA-3 selection issues

June 2008

NIST
National Institute of Standards and Technology

Timetable

- **Submissions due Oct 31, 2008**
  - **I hope you've all started by now!**
- **First SHA3 workshop after FSE2009**
  - **Leuven Feb. 25-27, 2009**
- **Call for comments on submissions**
- Try to narrow down to 5 or so finalists by third quarter of 2010.
  - Allow finalist tweaks
- Select winner in 2nd quarter of 2012

Our Problem: How to Choose?

- Requirements set the minimum bar
  - But we expect to get many minimally acceptable hashes!
- Many different things to weigh
  - Security
  - Performance
  - Implementation Issues
- Not so clear how to weigh them
  - Security

Our Problem: How to Choose?

- Can't break existing stuff
  - HMAC has to work
  - DSA/ECDSA has to work
  - KDFs and PRNGs have to work
- Acceptable performance / implementation
  - Fast enough everywhere
  - Low power/low gate count
  - Workable on smartcards and embedded processors
- Secure (otherwise, why bother?)

Our Constraints

- We have limited resources
  - Counting on community to help
- Goal is to get a good hash function
  - Not important to get the best in any one category
  - Very important to get something that's acceptable in all categories and very likely to be secure

What's Important?

- Performance
  - Speed of different implementations
  - Resource requirements
  - Implementability (can if fit?)
- Analysis
  - Automated analysis
  - Side channel issues
  - Proofs and properties
  - Cryptanalysis and design

Performance

- Speed
  - Many different platforms
  - Parallelizability
- Implementation Issues
  - Can it be implemented?
  - Gate count/power consumption
  - RAM, ROM, and other resources

Performance: Speed

- Many Platforms
  - Desktop, embedded, smartcard, low-end hardware, high-end hardware
  - Good everywhere > superfast one place
  - Existing stuff is easy to measure
  - Should benefit from future advances
  - Both bulk speed and short msg speed

Easy to measure, easy to overemphasize.

Some questions

- How important is speed on:
  - Desktops
  - Multicore desktop machines
  - High-end HW implementations
  - Low-end HW implementations
  - Embedded processors
- Are there platforms we don't care about?
  - As long as it can run….

Parallelism

- Most hashing modes are sequential
  - Damgaard-Merkle: Process blocks in order
  - Can have parallelism in compression fn.
- Simple variants possible
  - Interleave message blocks for 32-way parallelism
  - Makes short messages ugly
- How important is this?

Simple 32-way mode

- Let interleave(message,j) be the message formed from every 32nd message block, starting with j.
  - Thus, interleave(M,10) =

    M[10,42,74,106,…]
- PH(M) =
  - Hash'(interleave(M,0)||interleave(M,1)||…)
- Hash' = Hash starting with different IV
  - Example: IV' = IV xor 0xf0f0f0…f0

Implementability

Performance is nice, but it's really important to be able to get the algorithm to run.

- RAM requirements
- Minimal gate count
- Special stuff (multipliers, barrel-shifters)

Is it okay to be slow but possible on low-end processors, low-end hardware, etc?

What if the algorithm just won't fit on a smartcard or without 16KB of RAM?

Benefiting from the Future

- Future computers will probably be multicore 64-bit machines
  - Algorithms that can do well in that environment have an advantage
- What else can we say about future machines?
  - Do we care? Moore's law says we'll be using machines at 1000x current speed in 15 years.

Security Issues

- General issues with evaluation
- Automated analysis
- Side channel issues
- Proofs and properties
- Cryptanalysis and design

General Evaluation Issues

- How we evaluate depends on how many submissions we get
  - All “complete and proper” submissions will be given time at SHA-3 workshop
- Submissions break into three categories
  - Obviously flawed or unacceptable
  - Marginal
  - Apparently acceptable
    - (This is where finalists come from!)

Analysis time is scarce resource

- Most of resource is outside NIST
- Large number of serious submissions dilutes analysis time
  - Low hanging fruit targeted first
  - Analyzing a really new design can take a long time
- Very different picture with 50 submissions or with 20 submissions!

Obviously Flawed

- Obviously flawed algorithms
  - Successful cryptanalysis
  - Horrible performance
  - Inability to meet basic requirements
- Quickly excluded from most analysis
  - Often known unofficially at first, as with algorithms with published attacks, or awful performance.
- After we verify flaws, we can be sure these won't be finalists.

Marginal

- Distinguishing acceptable from marginal submissions is problem for 1st round
  - Incomplete analysis in submission
  - Noticeable performance issues in some platforms
  - Problems that don't amount to an attack
  - Usually takes some close reading/analysis

Not obvious to community! We may all have different opinions about which algorithms are marginal!

Acceptable

- These are algorithms that we might select as finalists
  - Hope to get most analysis concentrated on them
  - Harder to break stronger algorithms
  - …but has bigger impact.
- Ideally, community has idea of these
  - This was probably true for AES.

Major Goal: Sort Quickly

- Early analysis that sorts candidates into these bins makes everything else work well.
- If we have 40 acceptable submissions, very hard to select 5 finalists!
- Performance comparisons likely to dominate early, because they're easy to do.

Automated Analysis

- Some automated tools exist for looking at hash functions
  - Main advantage: Fast and cheap in terms of analyst time
- Statistical tests:
  - AES process: almost useless
  - Ecrypt stream ciphers: very useful
- Other tools?
  - Important to be able to understand what results mean!

Automated Analysis (2)

- Statistical tests on parts of hash
  - AES tests not so useful because looked at whole cipher
  - Tests of how many rounds passed more useful, but harder to interpret.
- Similar issues with other tools
  - If we find bad properties of whole hash, it's probably "Obviously Flawed."
  - Results on parts of cipher, or ambiguous measures, not so easy to interpret

Side channel issues

- Some algorithms have inherent side-channel issues
  - S-boxes and cache timing attacks
  - Multipliers
  - Modular exponentiation/etc with variable execution paths
- How important is this?
  - Keyed hashes, hashing secrets?

May be easy to evaluate hashes, but not so easy to decide what to do with evaluations!

Properties and Proofs

- Some designs may come with security proofs
  - How critical is a flaw in a proof?
  - How much weight should proof get?
- Some may claim better properties of iteration scheme
  - Long-message second preimage attacks
  - Indifferentiability, property-preserving, etc.
  - Not clear how to weigh these--probably not decisive
  - Maybe small advantage among finalists?

What properties are important?

- Not much apparent consensus
- Mostly can point to bad behavior, not define desired behavior
- What observed/demonstrated properties "break" or call into question a hash function?
  - Malleability, ability to ignore inputs in computation, ability to control some outputs in computation, etc.

Cryptanalysis

- Most important question is “can someone break this hash function?”
  - Some question about definition of “break.”
- But many kinds of break well known
  - Collision, preimage, second preimage
  - Breaking pseudorandomness in HMAC
  - Showing unacceptability for existing apps.

Cryptanalysis (2)

- Strongly dependent on right tools
  - For MDx,SHA-{0,1} family, we have these tools
  - Maybe have some for SHA256, RIPEMD
  - Maybe have some for Snefru/Tiger
- Radically new designs may take years to develop good attack tools!
  - How to weigh that in selection of finalists?
  - Think of HPC, MARS, DFC

Cryptanalysis (3)

- Major input here is time of skilled analysts
  - This is the most scarce resource
  - This is why it's important to narrow field to most likely finalists quickly
- Think of history of MD4/5 and SHA0/1.

Conclusions

- We have a big job ahead of us
- The number of submissions will matter
  - 100 very different from 20
  - More important: how many are not obviously flawed?
- We are counting on community for most of the work of evaluation!
  - We'll do what we can, but that's very limited!

Links

- Hash competition: http://www.nist.gov/hash-function