

How Should We Evaluate Hash Submissions?

John Kelsey, NIST

5/25/2007

*Invited Talk to ECRYPT Hash Workshop 2007

Context: AHS

- NIST is running a hash competition
- Submissions probably due around August 2008
 - About one year from call for submissions
- We want to work out how to evaluate submissions
- SHA2 hashes to remain as standard

Context: Don't Break Stuff

Lots of standards already use hashes

- Often many implicit assumptions:
 - Damgaard-Merkle, 512-bit message block, 160-bit hash value, etc.
- Assume it just randomizes everything
 - This is imprecise and often wrong, but it's still what people have done
- We don't want new hash to break stuff that's not currently broken.

Context: What we have to do

- We want to:
 - Get a good hashing standard
 - Advance the state of the art
 - Get something that works with existing uses of hashes
- We have limited resources
 - Small staff of cryptographers
 - Lots of other stuff to do

How to Evaluate AHS?

- Performance
 - Desktop / smartcard / server / hardware / parallelism
 - How much weight should performance have?
- Formal/Informal Security Requirements
 - What do we ask from hash functions?
 - How does this affect submissions?
- Cryptanalysis
 - This is the scarce resource
 - “How do I know what’s an attack?”
- Scaling
 - How will this process scale with # of submissions?

Problem #1:

Requirements on Hash

- Many comments on this
 - Especially about “like a random oracle”
 - Great comments from IBM and MIT
- We’ve been looking at our standards and how hashes are used
 - This is the tip of the iceberg, but it gives us a minimal set of requirements

Requirements (2)

What we know we need

- Support existing digital signature stds
 - $2n$ bit hash has n bits of collision resistance
 - Can't rely on randomization for security
- HMAC as a PRF
 - This is used too widely for us to break it
- Avoiding dumb design flaws and “gotchas” (aka pitchforks, landmines)
 - No length-extension bug
 - Can truncate without breaking collision resistance

Requirements(3)

Formal definitions we need

- Some way to support HMAC
 - Maybe as a mode of operation
- Also need (maybe via HMAC)
 - PRF (key never known)
 - Randomized hashing (key known after hash)
 - “Computational universal hash”
 - Looks like collision resistance to me
 - Extractor (for KDFs, entropy distillation)
- What about side channels?

Requirements (4)

How do we get these things?

Require from each hash submission

- HMAC support
 - How to do HMAC
 - Argument, proof, other evidence that this gives good PRF
- Constructions (maybe HMAC) for
 - PRF, randomized hashing, MAC
- Arguments / constructions for
 - Extractor, universal hash, whatever else

Requirements (5)

The R word

- Okay, so we can't build a random oracle
 - And neither can any of the submitters
- How should we describe the property of “don't surprise me?”
 - Don't know anything about $F(x)$ till you compute it?
 - Some pseudorandomness/game sort of definition?
 - “I can't define it, but I know it when I see it?”

Problem: people use hashes like they're ROs

Cryptanalyzing Submissions and Security Proofs

- How should we weight flaws in submission documents?
 - Invalid or flawed proofs
 - Statements that can be falsified, but don't break the hash?
 - Anything else?

Requirements Wrapup

- Must support DSA/ECDSA and HMAC
- Must be able to give us
 - PRF, MAC, randomized hash
- Explain / argue why okay as
 - Extractor
- Must behave in a random way (yuck!)
- Mustn't break existing nonbroken uses

Performance (1)

- Platform
 - Software:
Desktop, server, smartcard, embedded
 - Hardware:
Speed (accelerators), gate count, power usage
- Parallelizability
 - Iterated structure imposes limits
 - Maybe a “modes of operation” issue?
- Submission:
 - Ask for any guidance on parallelizeability

Performance (2)

- We know we need:
 - Online (one-pass)
 - Speed not totally out of line with SHA256
- We don't know:
 - How to weight provably secure designs
 - How to measure/consider security margins

We don't want to encourage barely-secure designs!

Evaluation and Performance

- Big potential problem:
 - Once all attacked hashes out of competition, only hard numbers are for performance
 - How to keep these from dominating?
- Ideas?

Cryptanalysis

This is the scarce resource for the competition!

- Do we know what we're doing attacking hash functions?
 - State of the art is not stable!
 - Good representations, tools, building on others' analysis
- Can we clearly define what qualifies as an attack?
 - Models

Cryptanalysis (2)

- Judging from AES history:
- Low-hanging fruit
 - Cookbook application of attacks
 - Very simple attacks
 - Designers didn't understand attacks or made some kind of error
- Good submissions
 - Each attack is a one-off by a talented craftsman
 - Not uncommon to spend a year or more working on an attack

Cryptanalysis: How do we decide what is an attack?

- Models
 - Random Oracle (too hard)
 - Sponge (generalization of iterated hash)
 - Collision resistance/preimage resistance
 - “Don’t surprise me” (too informal)
- We don’t need to specify everything that will count as an attack up front
 - But important that cryptanalysts know what is and is not meaningful--no 2^{95} attacks on DES, please!

Cryptanalysis: How Secure is Enough?

- We need n bit hash to require $2^{n/2}$ to collide
- What about preimage and 2nd preimage?
- Long message 2nd preimage attack?
- Bounds on PRF security?

When $n=256$, does a 2^{240} second preimage attack matter?

Cryptanalysis: Automated Testing

- Good news is it's cheap
 - One-time cost to code up tests
 - Needs a programmer, not a cryptanalyst
- Bad news is, it's pretty limited
 - Statistical tests ought to catch disasters
 - Maybe other ideas can help
- Could use as metric of sorts
 - How many rounds till pass stat tests
 - Not clear how to interpret this, though
 - Very unclear how to weight security vs. performance

Cryptanalysis: Other Thoughts

- One big cost to cryptanalyze something is understanding it
 - Takes time even for simple algorithms (Rijndael)
 - Much worse for complex algorithms (MARS)
- Sometimes need tools to even start analysis (SHA)
- Good explanation of algorithm, pictures, etc. is very valuable!
 - How do we encourage submitters here?
 - What should we (NIST, crypto community) do?

Compression Function vs. Chaining Mode

- Maybe not same people good at designing these
 - Chaining modes--security proof people
 - Compression function--analysis and design
 - Examples mostly support this!
- Resource limits: No way we do two!
- Some compression functions adapted to mode
 - HAIFA, RadioGatun, Salsa20

My thought: New chaining mode w/o reduction proof will have a hard life....

Scaling of Competition

- How Many Submissions Will We Get?
 - In range of 15-20, not too bad
 - If we got 50, how would we cope?
- AES as an example
 - 15 submissions
 - Several eliminated right away
 - Others had performance issues
 - Maybe half might have made it
- Low hanging fruit
 - Time spent killing weak submissions isn't spent analyzing the strong ones.

Discussion: Requirements

- Digital signature compatibility
 - DSA/ECDSA need n -bit hash with $2^{n/2}$ collisions
- HMAC compatibility
 - How much flexibility for submission?
 - Must they all be iterated? (Sponge)
 - PRF / MAC / universal hash
- Other requirements?
 - Randomization? Extractors?
- Require proof or argument?

Discussion: Evaluation

- Driven by requirements
 - How does flawed proof/argument in submission affect submission?
- Define a model to attack
 - Clear explanation
 - Attacker and submitter must be able to agree on what an attack looks like
- Cryptanalysis is the scarce resource
 - How do we get more?
 - How do we encourage it?

Discussion: Other Criteria

- Modes of operation vs. compression function
- Performance requirements
 - Speed
 - Platforms
 - Parallelism

Discussion: Scaling Problems

- Question: How many submissions are we likely to get?
 - Process very different for 10 submissions than 100
- Large number of submissions
 - Need filtering of bad submissions early
 - Need more barriers to entry
- Small number of submissions
 - Can spend more time on each submission