SHA3
WHERE WE'VE BEEN
WHERE WE'RE GOING

Bill Burr – May 1, 2013

updated version of John Kelsey's
RSA2013 presentation

Overview of Talk

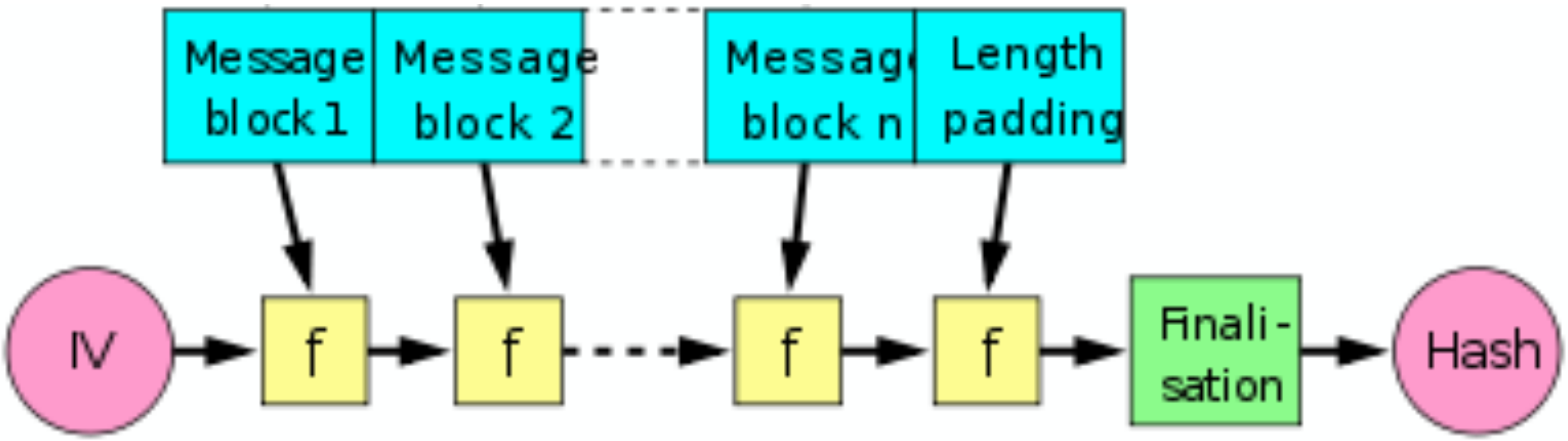
- Where We've Been:
  - Ancient history
  - 2004
- The Competition
- Where We're Going
  - What to standardize
  - Extras
  - Speculative plans

Ancient History (before 2004)

Origins

- Hash functions appeared as an important idea at the dawn of modern public crypto.
- Many ideas floating around to build hash functions from block ciphers (DES) or mathematical problems.
- Ways to build hash functions from compression functions
  - Merkle-Damgaard
- Ways to build compression functions from block ciphers
  - Davies-Meyer, MMO, etc.

Merkle-Damgaard



- Used in all widespread hash functions before 2004
  - MD4, MD5, RIPE-MD, RIPE-MD160, SHA0, SHA1, SHA2

Image from Wikipedia

The MD4 Family

- Rivest published MD4 in 1990
- 128-bit output
- Built on 32-bit word operations
- Add, Rotate, XOR, bitwise logical operations
- Fast
- First widely used dedicated hash function

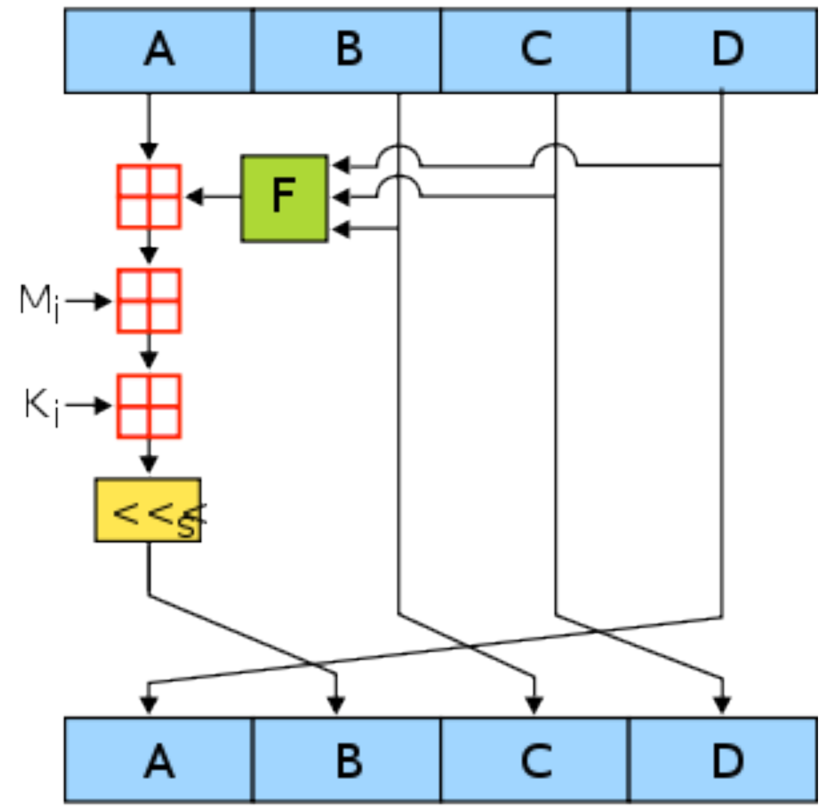


Image from Wikipedia MD4 Article

MD5

- Several researchers came up with attacks on weakened versions of MD4
- Rivest created stronger function in 1992
- Still very fast
- Same output size
- *Some attacks known*
  - *Den Boer/Bosselaers*
  - *Dobbertin*

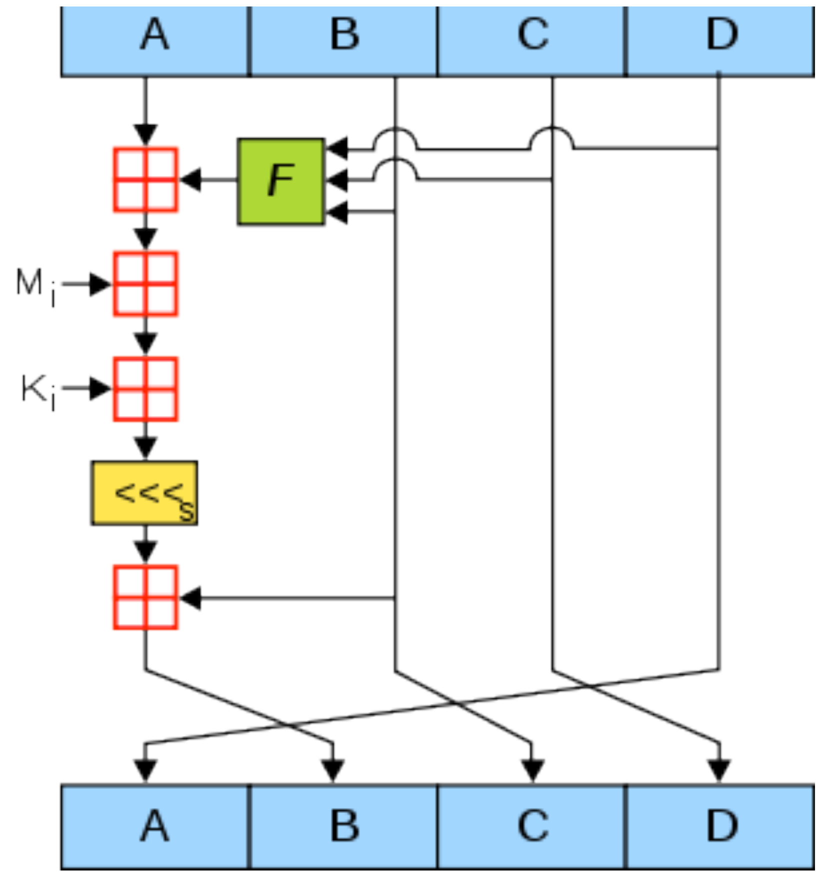


Image from Wikipedia MD5 Article

SHA0 and SHA1

- SHA0 published in 1993
- 160-bit output
  - (80 bit security)
- NSA design
- Revised in 1995 to SHA1
  - Round function (pictured) is same
  - Message schedule more complicated
- *Crypto '98 Chabaud/Joux attack on SHA0*

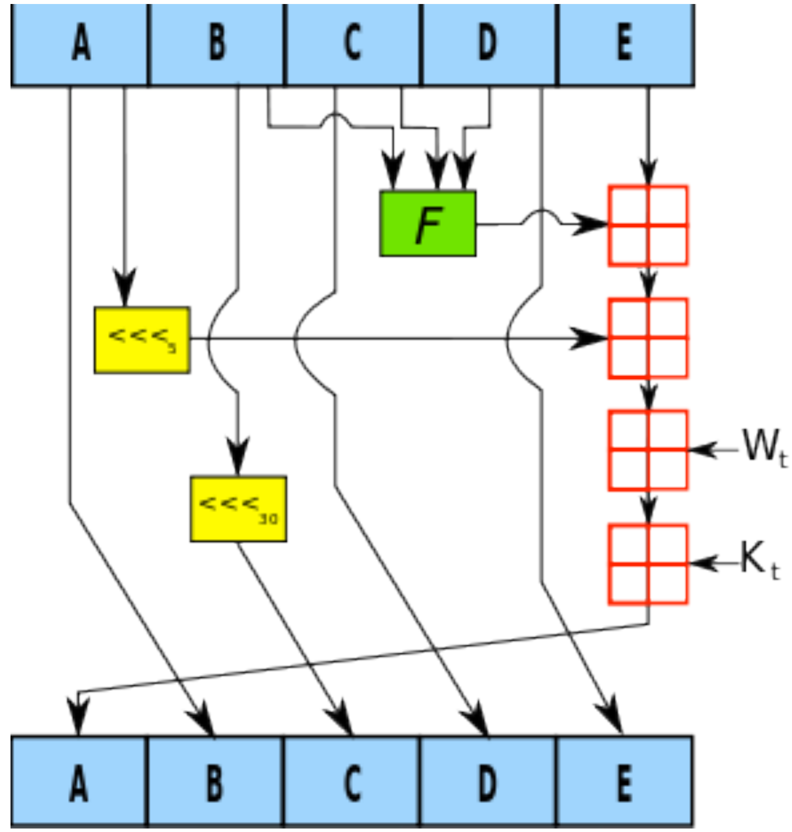


Image from Wikipedia SHA1 Article

SHA2

- Published 2001
- Three output sizes
  - 256, 384, 512
  - 224 added in 2004
- Very different design
- Complicated message schedule

- *Still looks strong*

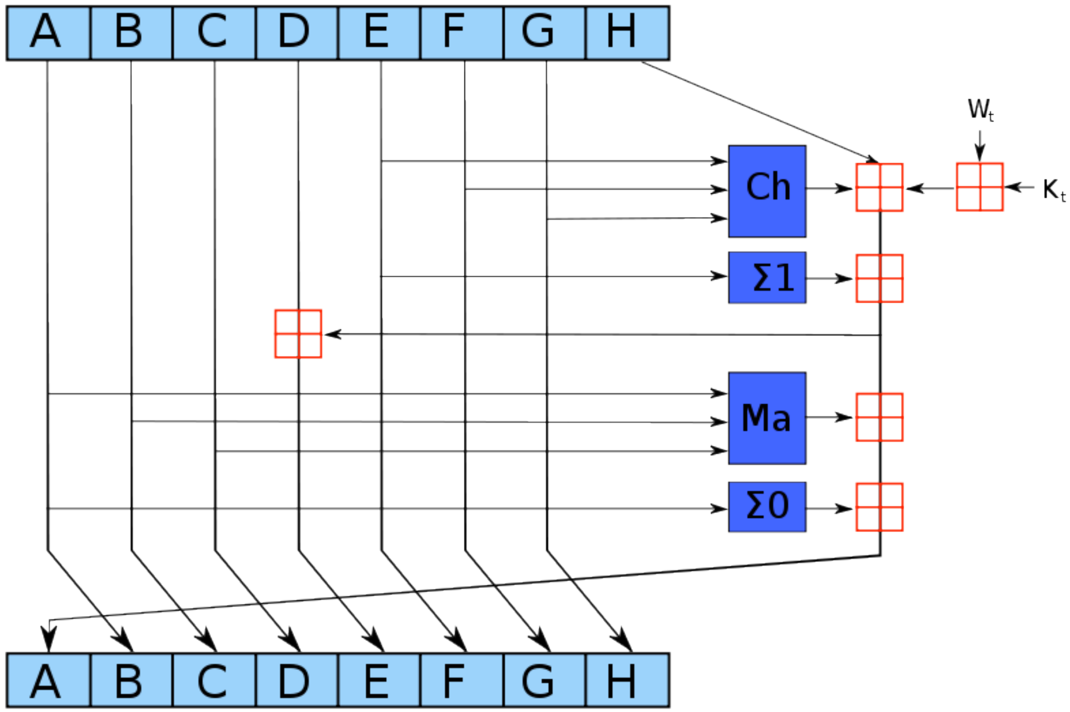


Image from Wikipedia SHA2 Article

As of 2004, we thought we knew what we were doing.

- MD4 was known to be broken by Dobbertin, but still saw occasional use
- MD5 was known to have theoretical weaknesses from Den Boer/Bosselaers and Dobbertin, but still in wide use.
- SHA0 was known to have weaknesses and wasn't used.
- SHA1 was thought to be very strong.
- SHA2 looked like the future, with security up to 256 bits
- Merkle-Damgaard was normal way to build hashes

2004: The Sky Falls

Crypto 2004: The Sky Falls

Conference:

- Joux shows a surprising property in Merkle-Damgaard hashes
  - Multicollisions
  - Cascaded hashes don’t help security much
- Biham/Chen attack SHA0 (neutral bits)

Rump Session:

- Joux shows attack on SHA0
- Wang shows attacks on MD4, MD5, RIPEMD, some Haval variants, and SHA0
  - Much better techniques used for these attacks

Aftermath: What We Learned

- *We found out we didn’t understand hashes as well as we thought.*
- Wang’s techniques quickly extended
  - Better attacks on MD5
  - Claimed attacks on SHA1 (2005)

- Joux’s multicollisions extended and applied widely
  - Second preimages and herding
  - Multicollisions even for multiple passes of hash
  - Much more

What to do next?

- All widely used hash functions were called into question
  - MD5 and SHA1 were very widespread
  - SHA2 and RIPE-MD160, neither one attacked, were not widely used.
- At same time, NIST was pushing to move from 80- to 112-bit security level
  - Required switching from SHA1 to SHA2
- Questions about the existing crop of hash functions
  - SHA1 was attacked, why not SHA2?

Preparing for the Competition

Pressure for a Competition

- We started hearing from people who wanted a hash competition
- AES competition had happened a few years earlier, and had been a big success
- This would give us:
  - Lots of public research on hash functions
  - A new hash standard from the public crypto community
  - Everything done out in the open

Hash Workshops

- Gaithersburg 2005
- UCSB 2006

- In these workshops, we got feedback on what a competition should focus on, what requirements should be, etc.
- Lots of encouragement to have a hash competition

2007: Call for proposals

- We spent a lot of time getting call for proposals nailed down:
  - Algorithm spec
  - Security arguments or proofs
  - Preliminary analysis
  - Tunable security parameter(s)

Security Requirements

- Drop-in replacement
  - Must provide 224, 256, 384, and 512 bit output sizes
  - Must play well with HMAC, KDFs, and other existing hash uses
- N bit output:
  - N/2 bit collision resistance
  - *N bit preimage resistance*
  - N-K bit second preimage resistance
    - K = lg(target message length)
- Eliminate length-extension property!
- Tunable parameter to trade off between security and performance.

The Competition

Hash Competition Timetable

Date	Event	Candidates Left
11/2/2007	Call for Proposals published, competition began	
10/31/2008	SHA3 submission deadline	***64***
12/10/2008	*First-round candidates announced*	*51*
2/25/2009	First SHA3 workshop in Leuven, Belgium	51
7/24/2009	*Second-round candidates announced*	*14*
8/23/2010	Second SHA3 workshop in Santa Barbara, CA	14
12/9/2010	*SHA3 finalists announced*	*5*
3/22/2012	Third SHA3 workshop in Washington, DC	5
10/2/2012	*Keccak announced as the SHA3 winner*	***1***

Initial submissions

- We started with 64 submissions (10/08)
- 51 were complete and fit our guidelines
- We published those 51 on December 2008

- Huge diversity of designs
- 51 hash functions were too many to analyze well
- There was a *lot* of cryptanalysis early on, many hash functions were broken

Narrowing the field down to 14

BLAKE BMW Cubehash Echo Fugue ***Grostl*** Hamsi
JH ***Keccak*** Luffa SHABAL SHAVite SIMD ***Skein***

- Many of the first 51 submissions were broken or seriously dented in the first year of the competition.
- Others had unappealing performance properties or other problems.
- AES competition had 15 submissions; we took a year to get down to 14.
- Published our selections in July 2009

Choosing 5 finalists

BLAKE Grostl JH Keccak Skein

- Published selection in Dec 2010
- Much harder decisions
  - Cryptanalytic results were harder to interpret
  - Often distinguishers of no apparent relevance
- All five finalists made tweaks for third round
  - BLAKE and JH increased number of rounds
  - Grostl changed internals of Q permutation
  - Keccak changed padding rules
  - Skein changed key schedule constant

Choosing a Winner: Security

- Nobody was knocked out by cryptanalysis
- Different algorithms got different depth of cryptanalysis
  - Grostl, BLAKE, Skein, Keccak, JH
- Keccak and Blake had best security margins
- Domain extenders (aka chaining modes) all had security proofs
- Grostl had a very big tweak, Skein a significant one
- ARX vs non-ARX designs

Keccak looks very strong, and seems to have been analyzed in sufficient depth to give us confidence.

Choosing a Winner: Performance

- All five finalists have acceptable performance
- ARX designs (BLAKE and Skein) are excellent on high-end software implementations
- JH and Grostl fairly slow in software
  - Slower than SHA2
- Keccak is very hardware friendly
  - High throughput per area

Keccak performs well everywhere, and very well in hardware.

Complementing SHA2

- SHA3 will be deployed into a world full of SHA2 implementations
- SHA2 still looks strong
- We expect the standards to coexist.
- SHA3 should *complement* SHA2.
  - Good in different environments
  - Susceptible to different analytical insights

Keccak is fundamentally different from SHA2. Its performance properties and implementation tradeoffs have little in common with SHA2.

Wrapup on Selecting a Winner

- Keccak won because of:
  - High security margin
  - Fairly high quality, in-depth analysis
  - Elegant, clean design
  - Excellent hardware performance
  - Good overall performance
  - Flexability: rate is readily adjustable
  - Design diversity from SHA2

How Did It Work Out?

- The competition brought forth a huge amount of effort by people outside NIST
- The cryptographic community did the overwhelming majority of the work:
  - Submissions
  - Analysis
  - Proofs
  - Reviews of papers for conferences/journals
- NIST's main job was to understand that work and make decisions based on it.

SHA3: What Function Will We Standardize?

Keccak as SHA3: Goals

- Play well with existing applications
  - DRBGs, KDFs, HMAC, signatures
- Drop-in replacements
  - SHA-224, -256, -384, -512, and even SHA1 and MD5
- Fast and efficient everywhere
- Benefit from tree hashing
- Benefit from Keccak extras
  - Variable output, efficient PRF, authenticated encryption, DRBG

Variable output length

- Keccak is equipped to provide variable-length output from a hash.
- This is endlessly useful
  - Protocols roll their own version of this all the time
  - OAEP
  - Key derivation functions
  - DSA Vaudenay attack fix
- SHA3 standard will support variable output sizes

Hash Function Security Notions

- Collision Resistance
  - Needed so that Hash can be a proxy for message in a digital signature and other commitment schemes
  - Infeasible to find two messages, $M_1 \neq M_2$ such the H(M_1) = H(M_1)
  - "Birthday paradox:" a collision can be found for any n-bit hash in about $2^{n/2}$ hash operations. Can't do better than this.
- Preimage Resistance
  - Needed for hash based message authentication codes, and other keyed hash function applications.
  - Given only an n–bit hash output, x, it should infeasible to find a message, M, such that H(M) = x
  - We expect to find a M by brute force in about 2^{n-1} operations

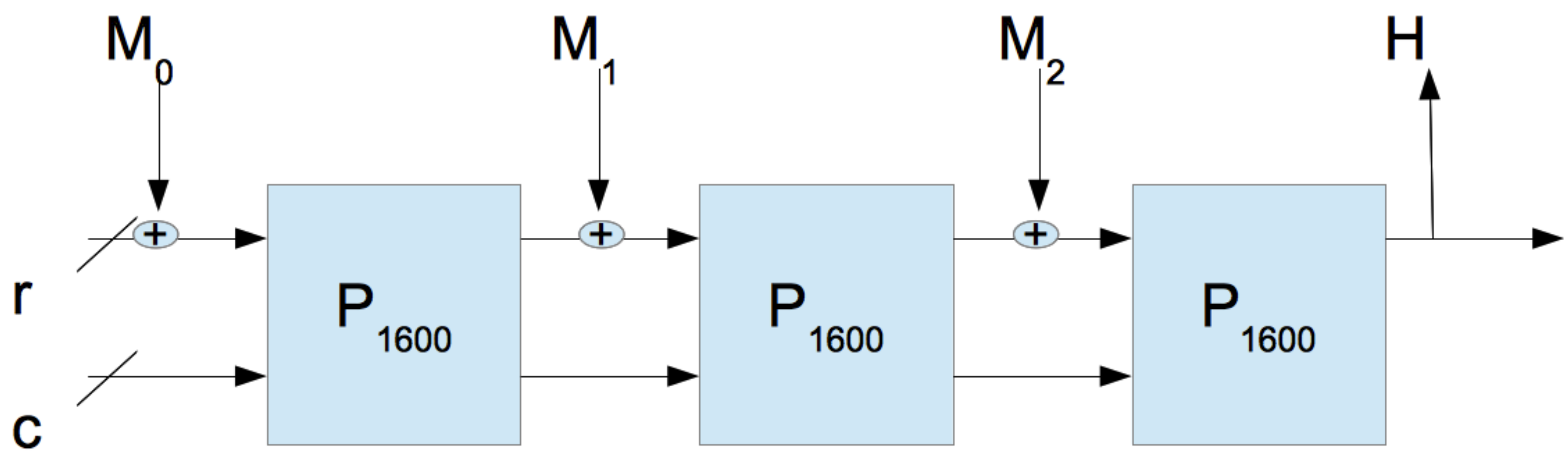
Security and Output Size

- Traditionally, hash functions' security level is linked to their output size
  - SHA256: 128 bit security against collisions, 256 against preimage
  - Best possible security for hash with 256-bit output.
- Keccak has variable output length, which breaks this link
  - Need a notion of security level separate from output size
- Keccak is a sponge
  - Security level is determined by *capacity*
  - Tunable parameter for performance/security tradeoff

Capacity and Security

- Keccak's security level is based on its capacity
  - Adjustable parameter: more security = less performance
  - C = 2*security level
  - C/2 bits of security *against both preimages and collisions*

.



Security Levels and Hashing

- SHA256 has a security level of 128 bits
  - Determined by collision resistance
  - Used with public key and symmetric algorithms of comparable security level –
  - Is 256 bits of security against preimages necessary?

- We propose changing this
  - Hash function that supports k bit security level should require only k bits of preimage resistance.
  - Question: Is there any practical weakness introduced by this decision?

Smaller capacity, faster hash

- Keccak's SHA3 submissions paid a substantial performance cost to get these high preimage resistance numbers.
  - Keccak-512 has 1024-bit capacity
  - Keccak-256 has 512-bit capacity

- Our proposal:
  - Security of k means k bits of security needed for all attacks.
  - This will make SHA3 considerably faster everywhere.

Too Many Capacities!

- Keccak specified four different capacities
  - 448, 512, 768 ,1024
- But four seems needlessly complex
  - 224 not on a 64-bit boundary
  - What do we gain for this added complexity?
- Our plan would drop those to
  - 256, 512
- However, the 4 different capacities in the Keccak submission did provide domain separation for the 4 “drop in replacement” variants of SHA3

Drop-in replacements

- We need drop-in replacements for SHA-224, -256, -384, and -512.
  - Replace one with the other in protocols and apps
- Then with the variable length outputs we get something like the following SHA-3 variants:
  - SHA3-Dropin-224(message) (c=256)
  - SHA3-Dropin-256(message) (c=256)
  - SHA3-Dropin-384(message) (c=512)
  - SHA3-Dropin-512(message) (c=512)
  - SHA3-Fast(message, output length) (c=256)
  - SHA3-Strong(message, output length) (c=512)

Drop-in replacements

- SHA-384 uses the same compression function as SHA-512, and truncates the output to 224-bits, but starts with a different IV. SHA-224 and SHA-256 are similar.
- Don’t want the unexpected property in SHA-3 that:
  - SHA3-Dropin-256(message) = abcdefgh
    and,
  - SHA3-Dropin-224(message) = abcdefg
    or,
  - SHA3-Dropin-512(message) = ABCDEFGH
    and,
  - SHA3-Dropin-384(message) = ABCEDF
- SHA2 does not have this property

Message Padding Scheme

- Keccak designers have proposed a padding scheme that will (among other things) distinguish the drop in replacements from each other – A paper is coming
  - If we change message padding we can incorporate other information
  - Tree structure/location
  - Alternative message encodings
  - Anything else?

Summing Up SHA3

- Variable-length output
- Extended message padding scheme
- Only two capacities
  - Requires encoding variable output length in message padding of SHA-2 drop-in replacements.
- Security decision: Preimages need only be as hard to find as collisions.

What comes next?

Keccak offers a lot of extras

- Our first job is to write a SHA3 FIPS
  - Write standard to allow later standards to build up these extras
  - Question: What should we call this? Keccak? SHA3?
- PRF
- Tree hashing
  - Not part of Keccak spec, but used with it
- Authenticated encryption
- Random number generation
- Key derivation

PRF

- Keccak defines a more efficient PRF
- Can we specify this as a drop-in replacement for HMAC?
  - Note: HMAC-Keccak is also fine, just inefficient
- Question: Are there uses of HMAC that wouldn't work right with the Keccak PRF?
- Question: Can we use PRF for randomized hashing?

Tree Hashing

- NIST has committed to doing a standard for generic tree hashing, using any approved hash function
- Planning to incorporate some support for tree hashing in message padding rules for SHA3.
- Approach #1: Full hash tree
  - Specify leaf size, fan-out, maximum height
- Approach #2: Interleave mode
  - N hashes done in parallel, until end when they're all hashed together.

Tree Hashing, Cont’d

- Our current plan is to specify general mechanisms, and recommend some parameters
- Example: parallel interleaved mode with N=16
- Example: tree mode with leaves of 8 message blocks and fan-out of 8.

- Question: Would we be better off allowing only small set of parameters?
- Comments or suggestions very much appreciated here
  - This effort is just beginning now.

Authenticated Encryption

- Keccak designers defined "duplex mode" which can be used to build authenticated encryption mechanism
- Authentication is as secure as hash function
- Encryption is secure if hash function behaves randomly in some sense.
  - See Duplex Mode paper from Keccak team for details

- Our Plan: after SHA3 is published, we will strongly consider writing a standard for authenticated encryption with Keccak.

Random Number Generation

- Keccak in duplex mode can also be used to build a deterministic random number generator
- SP 800-90A has several DRBGs specified
- After the SHA3 standard is published, NIST will strongly consider adding a new DRBG based on Keccak in Duplex mode

Speculative: Smaller Permutations

- Keccak specifies several smaller permutations
  - Full SHA3 is built on 1600-bit permutation
  - Smaller permutations are closely related
- We may specify hashes based on these smaller permutations at some point.
  - Useful for constrained devices
  - This depends on building up confidence in those small permutations
  - So far, they have seen little analysis.
  - NIST would love to see more analysis

Speculative: Alternative Modes

- The Keccak designers have proposed alternatives for more efficient authenticated encryption or message authentication
- Different modes
- Smaller permutations
- Fewer rounds
- NIST might eventually consider these for standardization, if we become confident in their security.

Wrapup and Questions

Questions for Community

- Is there a problem reducing preimage resistance to security level?
  - What application will be broken with preimage resistance of 256 bits?
- Tree hashing: Flexibility vs simplicity of standards?
  - What are important tree hashing applications?
- What should we call it?
- What are your questions?

NIST