Batteries Included

Features and Modes for Next Generation Hash Functions

Stefan Lucks, David McGrew, Doug Whiting

Bauhaus-Universität Weimar, Germany, Cisco Systems, USA, Exar Corporation, USA

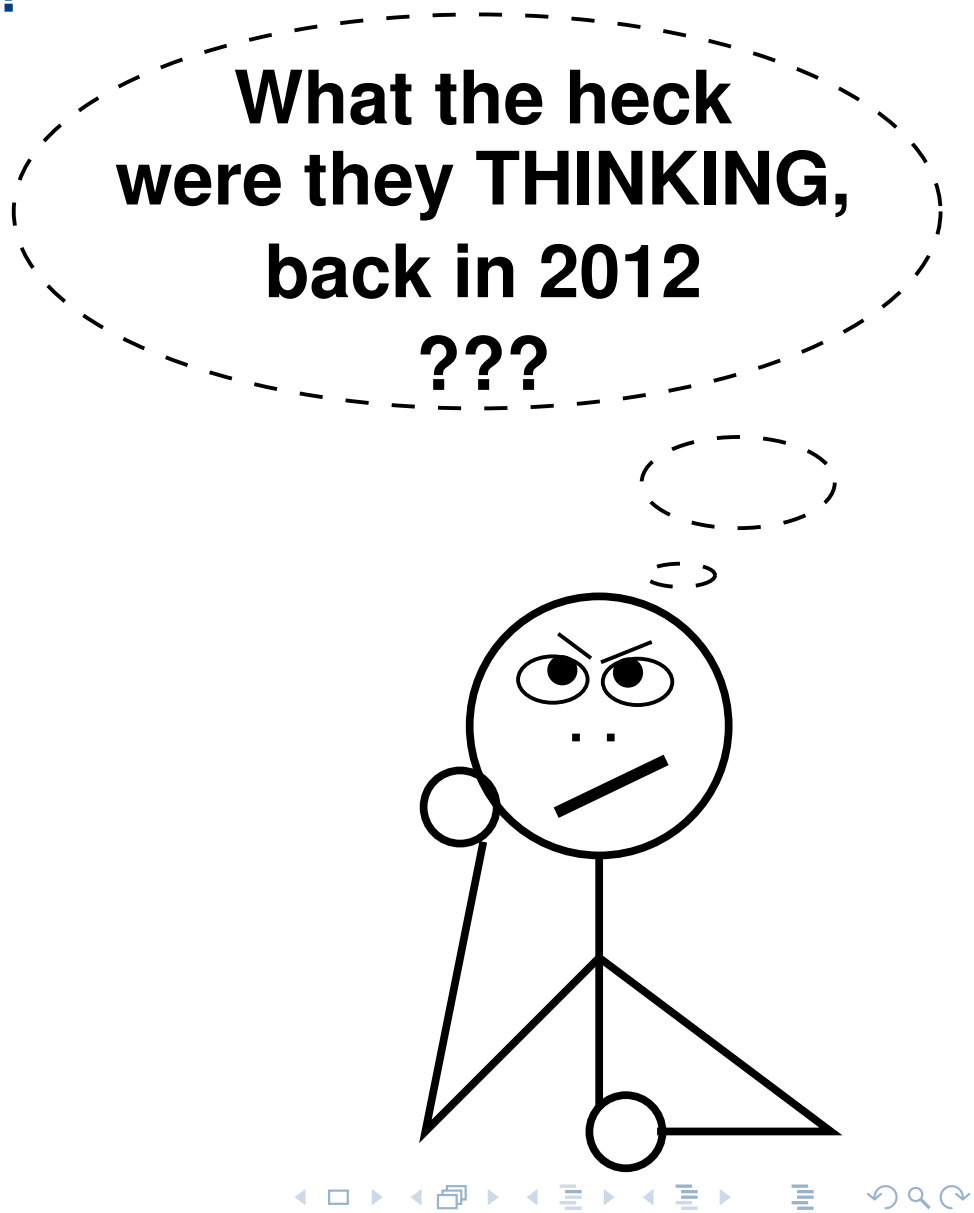
March, 2012

Skipping Forward to 2022

Why could a SHA-3 user be unhappy?

The fool doth think he is wise, but the wise man knows himself to be a fool.

– Wiliam Shakespare,
(As You Like It)



First Generation Hash Functions

MD5, SHA-1, SHA-2, . . . share

1. a design principle (Merkle-Damgård)
2. a set of structural weaknesses (length extension)
3. a usage model
   - message of arbitrary size is read sequentially;
   - eventually an **n**-bit hash value is emitted,
   - where **n** is the single output size supported
4. a simple security definition, depending on **n**:
   - no collisions at $\ll \mathbf{2^{n/2}}$ units of time
   - no preimages at $\ll \mathbf{2^{n}}$ units of time

First Generation Hash Functions Were Extremely Successful

Used in many applications not foreseen by the original hash function designers:

- keyed hashing (i.e., turn the hash functions into a MAC)
- processing the plaintext in public-key encryption schemes, (e.g., OAEP)
- key derivation,
- entropy extraction from non-uniform random sources,
- etc.

As of 2012, we can forsee these and some more applications for SHA-3. We should be prepared!

Hash Function Based MACs

- keyed hashing is the second most common cryptographic algorithm cited in Internet standards
- HMAC outnumbers AES by 2:1
- First generation:
  - $MAC_K(M) = H(K||M)$ is insecure, due to M-D principle
  - Remedy:

$$\text{HMAC}(K, M) = H((K \oplus \text{opad})||H((K \oplus \text{ipad})||M))$$

    provably secure (under reasonable assumptions)
    but inefficient for short messages
- **SHA-3 should support more efficient message authentication (=keyed hashing) than HMAC-SHA-3!**

Different Output Sizes

- Sometimes, the user of a **n**-bit hash function just happens to need **s** bits of output.
- Any choice is possible: $\mathbf{s} < \mathbf{n}$, $\mathbf{s} = \mathbf{n}$ and $\mathbf{s} > \mathbf{n}$.
- First generation:
  - ad-hoc methods to truncate or extend the hash
  - not always in a compatible way
- SHA-2: SHA-256 and SHA-224, SHA-512 and SHA-384, recently SHA-512/*t*
- **SHA-3 should provide a single hash function supporting any output size you want!**

Smaller Output Sizes ($\mathbf{s} < \mathbf{n}$)

- easy to realize, e.g.

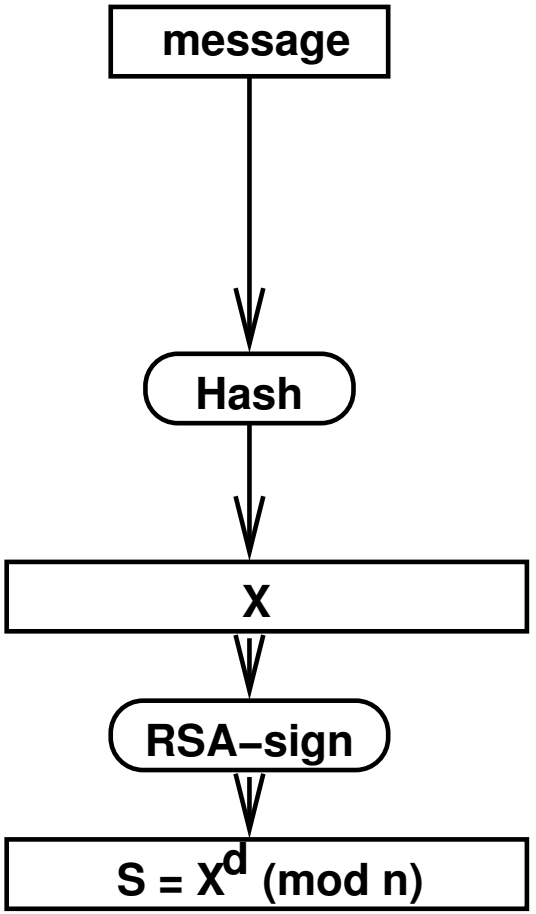
$$H_{\mathbf{s}}(M) = \text{truncate}_{\mathbf{s}} H(H(\mathbf{s})||M)$$

- "for free" if one uses the same **s** all the time
- people will use this anyway – but without standardization probably in incompatible ways
- take care about trivial near-collisions between $H_{\mathbf{s}}$ and H, and trivial collisions between $H_{\mathbf{n}}$ and H
- If H doesn't natively support different output lengths: **better make $H_{\mathbf{s}}$ the standard, rather than H!**

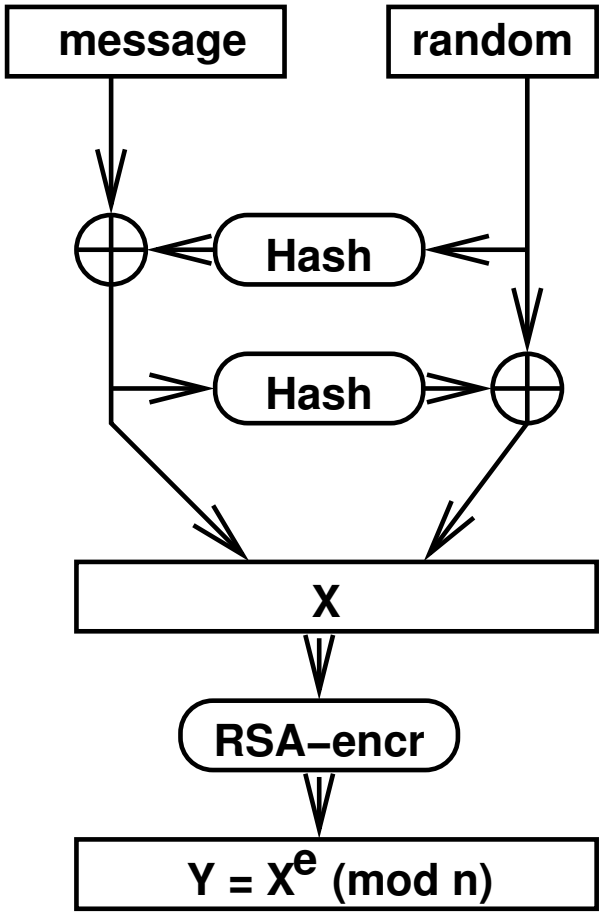
Two Example Applications with s > n

Currently using ad-hoc techniques to generate s output bits

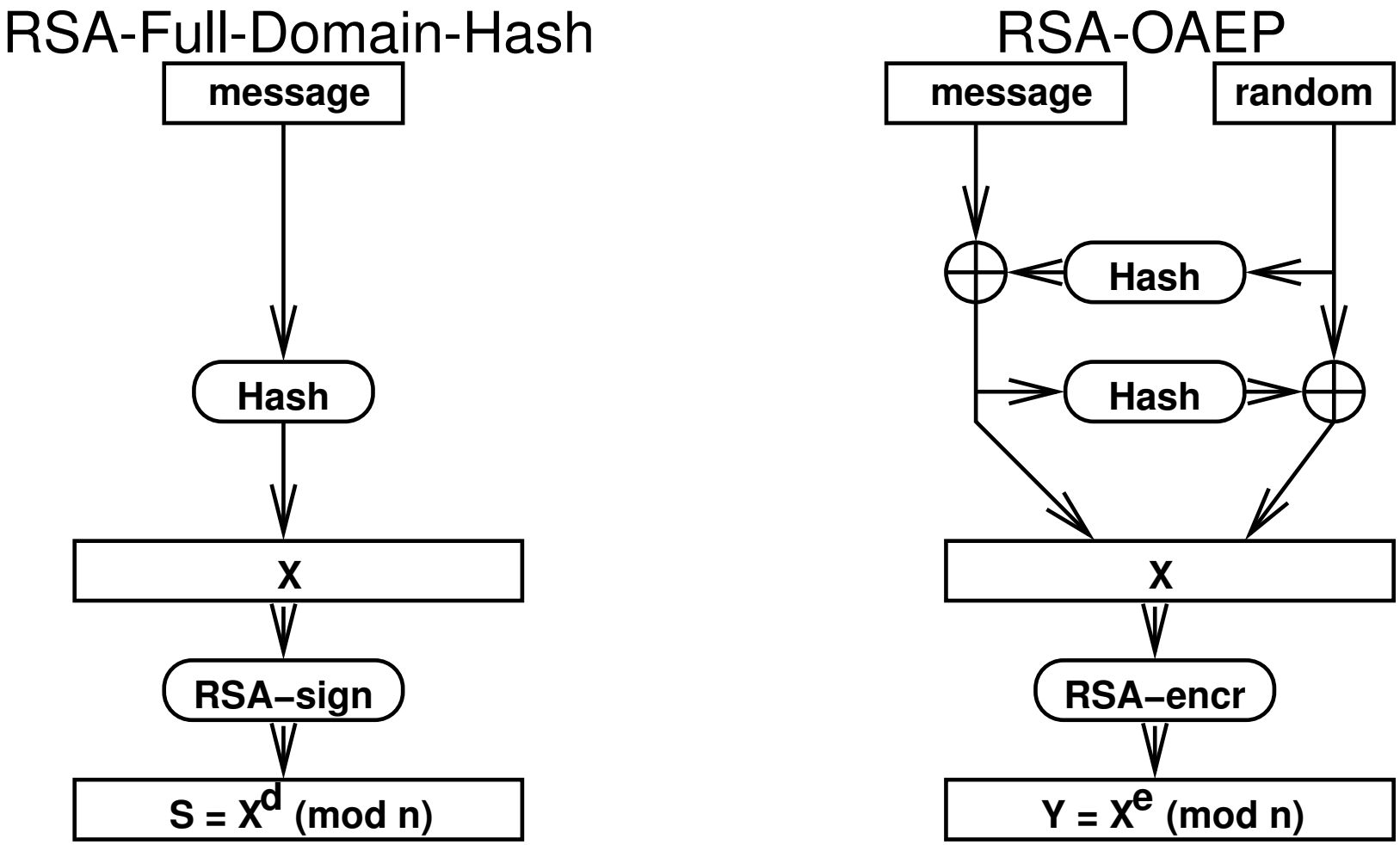
RSA-Full-Domain-Hash



RSA-OAEP



SHA-3 should not require ad-hoc methods to extend the hash size for applications such as OAEP.



Tree Hashing

- first proposed to support hash-based digital signatures to handle many one-time signatures
- but can also used to support an arbitrary level of parallelism for hashing large messages
- following the current trend in computer architecture, parellelized tree hashing may become important
- easy to implement for any secure hash function (cf. Bertoni et al, 2009)
- but take care about the possibility of generating trivial collisions between the sequential and the tree hash:

$$M \neq M^t \quad \text{with} \quad H^t(M) = H(M^t)$$

- and take care to maintain compatibility between different tree hashing applications
- **Anticipate a demand for (SHA-3-based) tree-hashing.**

Hash-Based Signatures

- long history (Lamport, Diffie, Winternitz, Merkle, . . .)
- extremely compact implementations
- no unproven security conjecture, beyond the hash function's own security (no factorization or discrete log assumption)
- number one candidate for post-quantum secure signatures
- well suited for
  - applications requiring tamper resistance, such as protection against hardware trojans,
  - and for low-end systems with short messages, e.g., a single measured value in a sensor network

Implementations of Hash-Based Signatures would benefit from

- a *keyed hashing mode* for key derivation.
- a *shortened hash* (**s** $<$ **n**) in the one-time signature component, to save signature size without reducing security, and
- a *tree signature scheme* for the tree hash component.

A SHA-3 standard that incorporated these modes would encourage and support hash-based signatures.

Other Modes that would be nice to have

Personalized Hashing:

- hash functions H_P, depending on "personalization information" P
- useful to define "independent" hash functions, e.g., for OAEP
- useful to hinder *key substitution attacks* and *domain parameter substitution attacks*
- ...

Support for Encryption and Authenticated Encryption:

- a constrained device can perform both hashing and encryption using one single primitive
- a common motivation to study block cipher based hashing

Remark

- Two of us are members of the Skein team, and Skein actually adresses the issues we raised here.
- Still, this talk is not about advertising Skein!
  Any finalist can be made to address these issues.
  But some would need to be tweaked/patched!
- Which of the following choices would you prefer?
  1. Address the issues now, before SHA-3 has been fixed.
  2. Fix SHA-3 now, and later “patch” these issues by filing additional standards.
  3. Don’t address these issues. Allow the proliferation of different incompatible and sometimes insecure ad-hoc solutions.

Summary

For the next generation of hash functions, the raw hash function isn't enough. It is important that the next generation of hash functions supports

- hash function based MACs,
- hashing for different output sizes **s**, both $\mathbf{s} < \mathbf{n}$ and $\mathbf{s} > \mathbf{n}$,
- and parallelized tree hashing.

Among other benefits, the support for these features would aid the usage of hash-based signatures.

Further useful options would be a way to personalize SHA-3, and to use SHA-3 for encryption purposes. It should be possible to combine these features, and to formally prove them secure.

Skipping Forward to 2022 (2)

Can we make this guy happy?

- We **cannot** forsee all future usage patterns for SHA-3.
- But we **can** forsee some.
- And SHA-3, the first one of a new generation of hash function, **can** support these usage patterns, where SHA-1 and SHA-2 had failed.