Grøstl update

How to get assurance in a crypto primitive?

- Cryptanalysis, Cryptanalysis, Cryptanalysis
- Proofs save cryptanalysis time
- Security margin

Grøstl delivers this

Reassuring proofs at many levels

- High-level proofs assuming ideal permutations
  - Hash function (indifferentiability)
  - Compression function (collision and preimage security)
- Proofs against differential attacks

Cryptanalysis results

Designed for simple analysis: Rebound attack

- Hash function collision: up to 6 rounds
- Compression function collision: up to 7 rounds

Also externally, by

Ideguchi, Tischhauser, Preneel

Sasaki, Li, Wang, Sakiyama, Ohta

Peyrin

Leurent

Peyrin and Gilbert

...

Cryptanalysis results

Designed for simple analysis: Rebound attack

- Hash function collision: up to 6 rounds
- Compression function collision: up to 7 rounds

By a small change of constants...

we reduce impact of the new observations
virtually no impact on implementations

Updated cryptanalysis, attacks remain valid:

Hash function collision: up to 3 rounds

Compression function collision: up to 6 rounds

Security margin

- Many proofs save cryptanalysis time
- We are halfway through the competition, yet most candidates haven‘t received a lot of analysis yet
- Grøstl has
- Grøstl comes with additional claims to cement assurance (collision/preimage security of CF)
  - Gives huge additional freedom for attacker (because of wide-pipe)
  - Still no way known to violate such a claim
  - Suggests a large security margin

Implementation aspects that stand out

- Fastest among the AES-like candidates
- Performance improvement using AES-NI
- Almost all large hardware surveys independently arrived at the conclusions that Grøstl is in the top 3

Conclusions

- Peaceful retirement for Bill,
  no MD5/SHA-1 surprise again!
- Assurance is key
  - Cryptanalysis + Proofs at many levels help there
- Grøstl delivers that
- Future-proof: balanced implementation characteristics, e.g.
  - Grøstl is efficient in hardware
  - Can use AES instructions, table-based, bit-sliced, byte-sliced implementation approaches