
From: hash-forum@nist.gov on behalf of Martin Schl affer [martin.schlaeffer@iaik.tugraz.at]
Sent: Tuesday, November 30, 2010 11:40 AM
To: Multiple recipients of list
Subject: OFFICIAL COMMENT: Gr ostl implementation update

Dear all,

recently, new Gr ostl implementations have been developed and added for benchmarking in eBash. In addition to the table-lookup based approach, there are at least 3 other (cache-timing resistant) methods to implement Gr ostl:

- * cache-timing resistant Gr ostl implementation using the Intel AES-NI instructions running at 13.8 c/b (~1.5x faster than table-based [1])
- * cache-timing resistant bitsliced Gr ostl implementation *without* the need of parallel message blocks (only ~1.5x slower than table-based on Core 2 Duo [2])
- * cache-timing resistant vperm based Gr ostl implementation (only ~1.2x slower than table-based on Core i5 [3])

Furthermore, the Gr ostl implementation of the portable C library sphlib is now running correctly in eBash. The sphlib submitted to eBash did not compile on all machines until a month ago. Thanks to Thomas Pornin for fixing this bug.

This gives improved performance results on many machines benchmarked recently (e.g. [1], supercop version 20101111 or newer), and renders some comparisons made more than a month ago invalid. Additionally, we are currently working on improved Gr ostl implementations for 32-bit platforms.

Also, an efficient 8-bit implementation for ATmega163 running at about 450 c/b is available from the Gr ostl website [4]. Furthermore, Gr ostl-256 and Gr ostl-512 can be combined with small overhead in hardware since the main transformations SubBytes and MixBytes are exactly the same.

The new results confirm the balanced implementation characteristics of Gr ostl throughout the whole spectrum from 8 to 128 bits, and show some further optimization potential.

Note that also tweaked constants (as announced at the Second SHA-3 Candidate Conference) will affect these performance numbers only in a negligible way.

Kind regards,
the Gr ostl team

- [1] <http://bench.cr.yp.to/xweb-hash/long-groestl256.html>
- [2] written by Stefan Tillich and benchmarked in eBash
- [3] posted by Cagdas Calik on 2010/11/10
- [4] <http://groestl.info/implementations.html>

From: hash-forum@nist.gov on behalf of Kelsey, John M. [john.kelsey@nist.gov]
Sent: Tuesday, November 30, 2010 1:35 PM
To: Multiple recipients of list
Subject: RE: OFFICIAL COMMENT: Grøstl implementation update

Martin,

Can you point out a reference to the new constants?

--John

De: hash-forum@nist.gov [hash-forum@nist.gov] En nombre de Martin Schläffer
[martin.schlaeffler@iaik.tugraz.at]
Enviado el: martes, 30 de noviembre de 2010 11:40 a.m.
Para: Multiple recipients of list
Asunto: OFFICIAL COMMENT: Grøstl implementation update

Dear all,

recently, new Grøstl implementations have been developed and added for benchmarking in eBash. In addition to the table-lookup based approach, there are at least 3 other (cache-timing resistant) methods to implement Grøstl:

- * cache-timing resistant Grøstl implementation using the Intel AES-NI instructions running at 13.8 c/b (~1.5x faster than table-based [1])
- * cache-timing resistant bitsliced Grøstl implementation *without* the need of parallel message blocks (only ~1.5x slower than table-based on Core 2 Duo [2])
- * cache-timing resistant vperm based Grøstl implementation (only ~1.2x slower than table-based on Core i5 [3])

Furthermore, the Grøstl implementation of the portable C library sphlib is now running correctly in eBash. The sphlib submitted to eBash did not compile on all machines until a month ago. Thanks to Thomas Pornin for fixing this bug.

This gives improved performance results on many machines benchmarked recently (e.g. [1], supercop version 20101111 or newer), and renders some comparisons made more than a month ago invalid. Additionally, we are currently working on improved Grøstl implementations for 32-bit platforms.

Also, an efficient 8-bit implementation for ATmega163 running at about 450 c/b is available from the Grøstl website [4]. Furthermore, Grøstl-256 and Grøstl-512 can be combined with small overhead in hardware since the main transformations SubBytes and MixBytes are exactly the same.

The new results confirm the balanced implementation characteristics of Grøstl throughout the whole spectrum from 8 to 128 bits, and show some further optimization potential.

Note that also tweaked constants (as announced at the Second SHA-3 Candidate Conference) will affect these performance numbers only in a negligible way.

Kind regards,
the Grøstl team

[1] <http://bench.cr.yp.to/xweb-hash/long-groestl256.html>

[2] written by Stefan Tillich and benchmarked in eBash [3] posted by Cagdas Calik on

2010/11/10 [4] <http://groestl.info/implementations.html>

From: hash-forum@nist.gov on behalf of Lars Ramkilde Knudsen [Lars.R.Knudsen@mat.dtu.dk]
Sent: Thursday, December 02, 2010 8:12 AM
To: Multiple recipients of list
Subject: RE: OFFICIAL COMMENT: Grøstl implementation update

John,

we are currently preparing a document describing the tweak in full. For the time being, we offer the following explanation.

One of the main design criteria of Grøstl was its simple design strategy to facilitate cryptanalysis and encourage external analysis. Indeed, this attracted lots of attention and led to the invention of rebound attacks [1], its extensions [2,3,4] and to the internal differential attacks [5,6].

None of these results pose any threat to our security claims of both the compression function and hash function of Grøstl. However, by simply changing the constants we can increase the security margin with negligible performance penalties.

If Grøstl gets to the next round we will:

- * significantly increase the size of the round constants to make the internal differential attack and its extensions impossible
- * and use different rotation constants in Q to make P and Q more different which further increases the security margin by one round.

All previous cryptanalytic work on Grøstl can still be used and applied immediately to the tweaked variant. Using Grøstl-256 (10 rounds) as an example, we get the following improved security margins:

- * the best attack on the reduced-round Grøstl-256 hash function decreases from 6 rounds to 3 rounds
- * the best collision attack on the reduced-round Grøstl-256 compression function reduces from 7 rounds to 6 rounds.
- * the approach of [6] to construct non-random properties of the compression function and permutation is thwarted (even though they are irrelevant for the security of the Grøstl hash function)

To summarize, the performance and storage impact will be small, while we increase the security margin.

Kind regards,
the Grøstl team

- [1] Florian Mendel, Christian Rechberger, Martin Schläffer, Søren S. Thomsen, "The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl", FSE 2009
- [2] Florian Mendel, Christian Rechberger, Martin Schläffer, Søren S. Thomsen, "Rebound Attacks on the Reduced Grøstl Hash Function", CT-RSA 2010
- [3] Henri Gilbert, Thomas Peyrin, "Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations", FSE 2010 [4] Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, Kazuo Ohta, "Non-Full-Active Super-Sbox Analysis: Applications to ECHO and Grøstl", ASIACRYPT 2010
- [5] Kota Ideguchi, Elmar Tischhauser, Bart Preneel, "Improved Collision

Attacks on the Reduced-Round Grøstl Hash Function", ISC 2010 [6] Thomas Peyrin,
"Improved Differential Attacks for ECHO and Grøstl",
CRYPTO 2010