SPN-Hash: Improving the Provable Resistance Against Differential Collision Attacks

Jiali Choy[1], Huihui Yap[1], Khoongming Khoo[1], Jian Guo[2], Thomas Peyrin[3,*], Axel Poschmann[3,†], and Chik How Tan[4]

[1] DSO National Laboratories, 20 Science Park Drive, Singapore 118230.
{cjiali,yhuihui,kkhoongm}@dso.org.sg
[2] Institute for Infocomm Research, A*STAR, Singapore.
ntu.guo@gmail.com
[3] SPMS, Nanyang Technological University, Singapore.
{thomas.peyrin,aposchmann}@ntu.edu.sg
[4] Temasek Laboratories, National University of Singapore.
tsltch@nus.edu.sg

Abstract. Collision resistance is a fundamental property required for cryptographic hash functions. One way to ensure collision resistance is to use hash functions based on public key cryptography (PKC) which reduces collision resistance to a hard mathematical problem, but such primitives are usually slow. A more practical approach is to use symmetric-key design techniques which lead to faster schemes, but collision resistance can only be heuristically inferred from the best probability of a single differential characteristic path. We propose a new hash function design with variable hash output sizes of 128, 256, and 512 bits, that reduces this gap. Due to its inherent Substitution-Permutation Network (SPN) structure and JH mode of operation, we are able to compute its differential collision probability using the concept of differentials. Namely, for each possible input differences, we take into account all the differential paths leading to a collision and this enables us to prove that our hash function is secure against a differential collision attack using a single input difference. None of the SHA-3 finalists could prove such a resistance. At the same time, our hash function design is secure against pre-image, second pre-image and rebound attacks, and is faster than PKC-based hashes. Part of our design includes a generalization of the optimal diffusion used in the classical wide-trail SPN construction from Daemen and Rijmen, which leads to near-optimal differential bounds when applied to non-square byte arrays. We also found a novel way to use parallel copies of a serial matrix over the finite field $GF(2^4)$, so as to create lightweight and secure byte-based diffusion for our design. Overall, we obtain hash functions that are fast in software, very lightweight in hardware (about 4625 GE for the 256-bit hash output) and that provide much stronger security proofs regarding collision resistance than any of the SHA-3 finalists.

Keywords. SPN, wide-trail strategy, Hash Functions, collision resistance.

1 Introduction

For current hash function designs, there are mainly two approaches to obtain provable security. The first approach is to prove collision and/or preimage resistance in relation to *hard* problems. For instance, Contini et al.'s very smooth hash (VSH) [15] is a number-theoretic hash for which finding a collision can be proven to be equivalent to solving the VSSH problem of the same order of magnitude as integer factorization. Concerning preimage, an example is MQ-HASH [9] for which finding a preimage is proven to be as hard as solving a multivariate system of equations. For the SHA-3 candidate FSB [1], finding collisions or preimages imply solving syndrome decoding. The second approach is more practical and less rigorous, and aims at proving a good differential probability bound for a single characteristic path. However, collision resistance is only heuristically inferred from this bound.

The first approach accomplishes more than a proof of resistance to differential cryptanalyis. However, hash function schemes based on this design strategy often suffer significantly in terms of speed and performance. On the other hand, schemes using the second approach enjoy faster speeds but suffer from incomplete proof

*This author is supported by the Lee Kuan Yew Postdoctoral Fellowship 2011 and the Singapore National Research Foundation Fellowship 2012.

†This author was supported in part by Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

of collision resistance. In this paper, we seek to reduce the gap between these two approaches by providing a more powerful proof for collision resistance while maintaining similar speed as compared to the symmetric-key design hashes.

Here, we recall that a *differential characteristic* over a composed mapping consists of a sequence of difference patterns such that the output difference from one round corresponds to the input difference in the next round. On the other hand, a *differential* is the set of all differential characteristics with the same first-round input and last-round output differences. Most hash function designs only aim at showing that any single differential characteristic has sufficiently low probability and heuristically infer collision resistance from this. Examples of hash functions which adopt this approach include hashes such as `WHIRLPOOL` [26] and some SHA-3 finalists like `GRØSTL` [20] and `JH` [35]. In addition, this differential characteristic bound is hard to determine for Addition-Rotation-XOR (ARX) designs such as `BLAKE` [3] and `SKEIN` [19]. Therefore, the next step in collision resistance proof, as already done by the second-round `SHA-3` candidate `ECHO` [5], is to give a bound on the best differential probability instead of only the best differential characteristic probability. However, note that this security argument only takes into account attackers that limit themselves to a fixed colliding differential (i.e. with a fixed output difference of the internal permutation), while many exist.

Our proposal for a new hash function design is able to achieve a stronger differential collision resistance proof. For example, for our proposed 512-bit hash, we prove that the differential probability of 4 rounds of its internal permutation function, which has a 1024-bit state size, is upper bounded by 2^{-816}. We sum this upper bound over **all output differences that lead to a collision** (2^{512} candidates) in order to find that the differential collision probability of our proposed hash is then upper bounded by $2^{-304} < 2^{-256}$ after the final truncation. In contrast, for the SHA-3 semi-finalist `ECHO` [5], the maximal expected differential probability for four rounds of their 2048-bit `AES` extension, `ECHO.AES`, is 1.055×2^{-452}, but summing over all possible colliding output difference masks (at least 2^{1536} candidates) completely prevents such a collision-resistance argument. For SHA-3 finalist, `GRØSTL`, it is easy to compute the internal collision probability of its compression function f. However, its output transformation, involving a permutation P followed by a truncation, makes such a derivation much less straightforward for the external collision probability of the full `GRØSTL` hash function.

In addition, we have to consider that for some hash function constructions, it is necessary to prove low related-key differential probability instead of just low fixed-key differential probability. For example, consider the Davies-Meyer compression function instantiated with `AES`. The main `AES` cipher has very low differential characteristic probability which is bounded by 2^{-150} for every four rounds. However, in the Davies-Meyer mode, each input message block to the hash corresponds to the cipher key of the `AES`-based compression function. This makes the compression function vulnerable to the multicollision attack by Biryukov et al. [10], because AES does not have good resistance against related-key differential attack.

1.1 Our Contributions

In this paper, we propose a new hash function design, `SPN-Hash`, with variable output sizes of 128, 256, and 512 bits. It is specially constructed to circumvent the weaknesses in the proofs of differential collision resistance as well as to resist common attacks against hash functions.

Concerning the internal permutations, we use the Substitution-Permutation Network (SPN) structure as the building block for `SPN-Hash` to ensure that the maximum probability taken over all differentials (not only differential characteristics) will be low enough. In [30], Park et al. presented an upper bound for the maximum differential probability for two rounds of an SPN structure, where the linear transformation can have any value as its branch number. This bound is found to be low for SPN structures. For instance, the maximum differential probability for four rounds of `AES` is bounded by 1.144×2^{-111}. Based on Park's result, we deduce an upper bound for the differential collision probability of `SPN-Hash`. We use this bound to show that our hash functions are secure against a differential collision attack. Furthermore for our internal permutations, we need to consider non-square byte-arrays of size $m \times n$ where $m \neq n$. The designers of AES [17] gave a construction for $m \times n$ arrays where $m < n$ using optimal diffusion maps, but the differential bound is the same as that of an $m \times m$ array, which is sub-optimal for mn-byte block size. By their method, a 256-bit permutation would be constructed by a 4×8 byte-array that has the same differential bound 1.144×2^{-111} as a 4×4 byte-array. This is not close enough to $2^{-blocksize} = 2^{-256}$ for our security proof. **We generalize the optimal diffusion map of [17] to construct $m \times n$ byte-arrays where $m > n$, which can achieve near optimal differential bound close to $2^{-blocksize}$.**

We also analyzed the security of our internal permutations against the latest rebound-like attacks [32]. More precisely, we present distinguishing attacks on three versions of the internal permutation P for 8 out of

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

10 rounds. For the 256-bit permutation P, the 8-round attack requires time 2^{56} and memory 2^{16}. For the 512-bit permutation P, the 8-round attack requires time 2^{56} and memory 2^8, while for the 1024-bit permutation P, the 8-round attack requires time 2^{88} and memory 2^{16}.

Concerning the operating mode, we use the JH mode of operation [35], a variant of the Sponge construction [6]. In this design, assuming a block size of $2x$ bits, each x-bit input message block is XORed with the first half of the state. A permutation function P is applied, and the same message block is XORed with the second half of P's output. For this construction, the message blocks are mapped directly into the main permutation block structure instead of via a key schedule. **This eliminates the need to consider related-key differentials when analyzing protection against collision attacks**. Furthermore, the JH mode of operation is able to provide second preimage resistance of up to 2^x bits for an x-bit hash as compared to only $2^{x/2}$ for the Sponge construction with the same capacity.

To summarize, our SPN-Hash functions use AES-based internal permutations with fixed-key and a generalized optimal diffusion to ensure low and provable maximum differential probability. Then our JH-based operating mode allows us to apply directly our security reasoning and obtain a bound on the maximum probability of an attacker looking for collisions using a fixed input difference. To the best of our knowledge, **this is the only known function so far that provides such a security argument**.

The performances of SPN-Hash are good since the internal permutation is very similar to the one used in the SHA-3 finalist GRØSTL. We propose a **novel construction to use parallel copies of the** PHOTON 8×8 **serialized MDS matrix over** $GF(2^4)$ **from [22], to create a secure and very lightweight byte-based diffusion for our design in hardware.**[5] Moreover, the area of SPN-Hash is also lowered by the relatively small internal memory required by the JH mode of operation. Hardware implementations require 4625 GE for 256-bit hash output, while current best results for the SHA-3 competition finalists require 10000 GE or more. Overall, **our proposal achieves both excellent software speed and compact lightweight implementations**.

Our paper is organized as follows: We state some necessary preliminaries concerning differential cryptanalysis in Section 2. Then we describe our proposed SPN-Hash design and give instantiations of 128-, 256-, and 512-bit SPN-Hash in Section 3 before proceeding to provable security against differential collision attacks in Section 4. Section 5 is devoted to analysing the security of our hash function against preimage, second preimage, and rebound attacks and finally in Section 6, we show some performance comparisons.

2 Preliminaries

Substitution Permutation Network. One round of an SPN structure consists of three layers: key addition, substitution, and linear transformation. In the key addition layer, a round subkey is XORed with the input state. The substitution layer is made up of small non-linear substitutions called S-boxes implemented in parallel. The linear transformation layer is used to provide a good spreading effect of the cryptographic characteristics in the substitution layer. As such, the SPN structure has good confusion and diffusion properties [33]. One round of the SPN structure is shown in Figure 1 in Appendix A.

Maximum differential probability of an S-Box. In this paper, we follow the standard definitions related to differential cryptanalysis, such as those in [17]. We take all S-boxes to be bijections from $GF(2^s)$ to itself. Consider an SPN structure with an st-bit round function. Let each S-box S_i be an s-bit to s-bit bijective function $S_i : GF(2^s) \rightarrow GF(2^s)$, $(1 \leq i \leq t)$. So the S-box layer consists of t s-bit S-boxes in parallel.

Definition 1. *For any given* $\Delta x, \Delta y \in GF(2^s)$*, the* differential probability of each S_i *is defined as*

$$DP^{S_i}(\Delta x, \Delta y) = \frac{\#\{x \in GF(2^s) \mid S_i(x) \oplus S_i(x \oplus \Delta x) = \Delta y\}}{2^s},$$

where we consider Δx *to be the input difference and* Δy *the output difference.*

Definition 2. *The* maximal differential probability of S_i *is defined as*

$$DP((S_i)_{max}) = \underset{\Delta x \neq 0, \Delta y}{max} DP^{S_i}(\Delta x, \Delta y).$$

[5] Note that the approach of [22] to do an exhaustive search for serialized MDS matrix over $GF(2^8)$ by MAGMA is only feasible for $n \times n$ matrix up to size $n = 6$. Therefore we need our current approach to construct serailized 8×8 matrix over $GF(2^8)$.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

Definition 3. *The* maximal value of $DP((S_i)_{max})$ *for* $1 \leq i \leq t$ *is defined as*

$$p = \underset{1 \leq i \leq t}{max} (DP(S_i)_{max}).$$

An S-Box S_i is strong against differential cryptanalysis if $DP((S_i)_{max})$ is low enough, while a substitution layer is strong if p is low enough.

A *differentially active* S-box is an S-box having a non-zero input difference. A differentially active S-box always has a non-zero output difference and vice versa. In order to evaluate security against differential cryptanalysis, other than the differential probabilities of the S-box or S-box layer, one also has to consider the number of active S-boxes whose value is determined by the linear transformation layer.

Substitution-Diffusion-Substitution function. In order to ease the analysis of the SPN structure, we define an SDS (Substitution-Diffusion-Substitution) function as shown in Figure 2. Let the linear transformation layer of the SDS function be defined by L, its input difference by $\Delta x = x \oplus x^*$, its output difference by $\Delta y = y \oplus y^* = L(x) \oplus L(x^*)$. If L is linear, we have $\Delta y = L(\Delta x)$. The number of differentially active S-boxes on the input/output of the SDS function is given by the branch number of the linear transformation layer.

Definition 4. *The* branch number *of a linear transformation layer* L *is defined as*

$$\beta_d = \min_{v \neq 0} \{wt(v) + wt(L(v))\},$$

where the $wt(x)$ *is the number of non-zero* s*-bit characters in* x*.*

If we want to find the number of active S-boxes in two consective rounds of the SPN structure, we only need to consider the SDS function. β_d gives a lower bound on the number of active S-boxes in two consecutive rounds of a differential characteristic approximation.

Definition 5. *A linear transformation layer on* t *elements is* maximal distance separable (MDS) *if* $\beta_d = t+1$.

Maximum differential probability of an SPN. The differential probability, which is the sum of all differential characteristic probabilities with the same input and output difference, gives a more accurate estimate of resistance against differential cryptanalysis (than that of a single characteristic path). In [30], Park et al. proved an upper bound for the maximum differential probability for 2 rounds of the SPN structure.

Theorem 1 *[30, Theorem 1] Let* L *be the linear transformation of an SPN structure and* β_d *be the branch number of* L *from the viewpoint of differential cryptanalysis. Then the maximum differential probability for 2 rounds of the SPN structure is bounded by*

$$\max \left\{ \max_{1 \leq i \leq t} \max_{1 \leq u \leq 2^s - 1} \sum_{j=1}^{2^s-1} \{DP^{S_i}(u,j)\}^{\beta_d}, \max_{1 \leq i \leq t} \max_{1 \leq u \leq 2^s - 1} \sum_{j=1}^{2^s-1} \{DP^{S_i}(j,u)\}^{\beta_d} \right\}.$$

As a consequence, we get the following theorem.

Theorem 2 *[30, Corollary 1] Let* L *be the linear transformation of an SPN structure and* β_d *be the branch number of* L *from the viewpoint of differential cryptanalysis. Then the maximum differential probability for 2 rounds of the SPN structure is bounded by* $p^{\beta_d - 1}$*, where* p *is the maximal value of* $DP((S_i)_{max})$ *for* $1 \leq i \leq t$.

3 The SPN-Hash functions

In this section we describe our proposed hash function design, SPN-Hash , with variable hash output sizes of 128, 256, and 512 bits. We adopt the JH mode of operation [35], a variant of the Sponge construction [6], operating on a state of $b = r + c$ bits. b is called the width, r the rate, and c the capacity. Our design is a simple iterated construction based on a fixed-length unkeyed permutation P, where $r = c$. The internal state of P can be represented by an $n \times m$ matrix of 8-bit cells, where n is the number of bytes in a bundle, and m is the number of bundles. Thus, P operates on a width of $b = 8nm$ bits, the rate and capacity are $4nm$-bit each, and the output is a $4nm$-bit hash value.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

Firstly, the input message x of length N bits is padded and divided into blocks of $r = 4nm$ bits each. The padding function produces the padded message, x', of length a multiple of $4nm$. It follows "Padding Method 2" in [29, Algorithm 9.30]: first append the bit '1' to x, followed by a sequence of $z = (-N - 2nm - 1 \mod 4nm)$ '0' bits. Finally, append the $2nm$-bit representation of $l = (N + z + 2nm + 1)/4nm$. The integer l represents the number of message blocks in the padded message x'. The maximum message length for $4nm$-bit `SPN-Hash` is thus set as $4nm \cdot (2^{2nm} - 1) - 2nm - 1$.

Then, all the bits of the state are initialized to the value of an Initialization Vector (IV). The IV of $4nm$-bit `SPN-Hash` is taken to be the $8nm$-bit binary representation of $4nm$. That is, in big-endian notation, the IVs are `0x00...0080` for 128-bit `SPN-Hash` , `0x00...0100` for 256-bit `SPN-Hash`, and `0x00...0200` for 512-bit `SPN-Hash` .

For each padded message block, the `JH` mode of operation iteratively XORs the incoming $4nm$-bit input message block M_i into the left half of the state, applies the permutation $P : GF(2)^{8nm} \rightarrow GF(2)^{8nm}$ to the internal state and XORs M_i into its right half. After all the message blocks have been processed, the right half of the last internal state value is the final message digest and therefore our construction produces a $4nm$-bit hash. It is summarized as follows:

$$
\begin{aligned}
&Padded\ Input = M_0, M_1, \ldots, M_{N-1} \\
&(H_{0,L}, H_{0,R}) = IV \\
&\text{For } i = 0 \text{ to } N-1\text{:} \\
&\quad (H_{i+1,L}, H_{i+1,R}) = P((M_i \oplus H_{i,L}, H_{i,R})) \oplus (0, M_i) \\
&Hash = H_{N,R}
\end{aligned}
$$

where $M_i \in GF(2)^{4nm}, (H_{i,L}, H_{i,R}) \in GF(2)^{8nm}$ and N is the total number of padded message blocks. A diagram of our `JH` mode of operation is shown in Figure 3 in Appendix A.

Using appropriate parameters m and n such that m is even and m divides n, we will be able to construct a wide range of hash functions of different output sizes:

128-bit `SPN-Hash` : $m = 4, n = 8$
256-bit `SPN-Hash` : $m = 8, n = 8$
512-bit `SPN-Hash` : $m = 8, n = 16$

3.1 The Internal Permutation P

The $8nm$-bit permutation P iterates a round function for 10 rounds. Its internal state can be represented by an $n \times m$ matrix of 8-bit cells, where n is the number of bytes in a bundle, and m is the number of bundles. Here, each column can be viewed as a bundle consisting of n bytes. In each round, there is a substitution layer, followed by an MDS layer, a generalized optimal diffusion layer, and lastly, an XOR with a round constant. Thus, the linear transformation layer of the SPN structure introduced in Section 2 is actually a composition of the MDS layer and the generalized optimal diffusion layer while the "round keys" of the SPN structure are taken to be the round dependant constants. A diagram of the permutation function P is shown in Figure 4 in Appendix A.

The substitution layer σ takes in a $8nm$-bit input and splits it into nm bytes. It then applies the `AES` 8-bit S-box [17] to each of these bytes in parallel. This is chosen due to its low maximum differential and linear approximation probabilities of 2^{-6}, which strengthens resistance against differential and linear attacks. In hardware, it is possible to achieve a very compact implementation of the `AES` S-box using "tower-field" arithmetic, as proposed in [13]. In software, one could use the Intel AES-NI instruction set [16] for efficient implementation.

The MDS layer θ combines consecutive n bytes into bundles and applies on each of these m bundles an MDS transformation described in Section 3.3.

The generalized optimal diffusion layer π is a permutation of bytes that achieves good spreading effect. It is an instantiation of the generalized optimal diffusion which we define in Section 3.2. We write this layer

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

π as $(\pi_1, \pi_2, \ldots, \pi_n)$, where $0 \leq \pi_i \leq m-1$. This notation indicates that row i is rotated by π_i positions to the left:

128-bit SPN-Hash: $\pi = (0, 0, 1, 1, 2, 2, 3, 3)$
256-bit SPN-Hash: $\pi = (0, 1, 2, 3, 4, 5, 6, 7)$
512-bit SPN-Hash: $\pi = (0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7)$

These byte permutations are indeed generalized optimal diffusions since exactly n/m bytes from each column is sent to each of the m columns.

The round constant $RCon_i$ that is XOR-ed with the state is different for every round i. This is to defend against slide attacks [11, 12] and to prevent fixed points present over reduced rounds from being propagated to the entire permutation P. Each $RCon_i$ can be viewed as an $n \times m$ matrix A, where $A_{x,y}$ ($0 \leq x < n$, $0 \leq y < m$) denotes the entry in row x and column y. Then for $RCon_i$ used in round i,

$$A_{x,y} = \begin{cases} y \oplus i & \text{if } x = 0 \\ 0 & \text{otherwise,} \end{cases}$$

where i is the round number viewed as an 8-bit value. These values of round constants are chosen as they are light in hardware.

3.2 Generalized Optimal Diffusion

Definition 6. Generalized Optimal Diffusion*: Let m divide n. We consider a concatenation of n bytes as a* bundle *and we consider a concatenation of m bundles as a* block. *A linear transform, π, mapping a block of m bundles to m bundles is called a* (m, n)-generalized optimal diffusion *if for each input bundle of a block, n/m bytes of that input bundle is mapped to each of the m output bundles.*

Our (m, n)-generalized optimal diffusion is a generalization of the optimal diffusion layer used in the wide-trail strategy of Rijmen and Daemen [17]. The latter corresponds to the case $m = n$ and the ShiftRows function in AES is a particular instantiation of it. For our hash function design, m must be even and m must divide n.

The following two results compute the maximum differential probability of SPN-Hash. Their proofs can be found in Appendix B.

Theorem 3 *Let $\theta : [GF(2^8)^n]^m \to [GF(2^8)^n]^m$ be an MDS layer formed by concatenating m $n \times n$ MDS transforms over $GF(2^8)$. Let $\pi : [GF(2^8)^n]^m \to [GF(2^8)^n]^m$ be a (m, n)-generalized optimal diffusion mapping m bundles to m bundles. Then $\pi \circ \theta \circ \pi$ is a $m \times m$ MDS transform over $GF(2^{8n})$.*

Theorem 4 *The probability of any non-zero input-output differential for the internal permutation P described in Section 3.1 is upper bounded by $\left(126 \times (2^{-7})^{n+1} + (2^{-6})^{n+1}\right)^m$.*

3.3 MDS Layer

The MDS layer provides an independent linear mixing of each column. In the following, we describe the mixing function of each column and show that it is indeed an MDS transform.

128- and 256-bit SPN-Hash. In [22], the authors proposed a method for generating the 8×8 MDS transform over $GF(2^4)$ in a serial way that is very compact. However, it is difficult to find an 8×8 serialized MDS matrix over $GF(2^8)$ using the exhaustive search method of [22]. Thus, we show here a way to construct such a matrix using two parallel copies of the PHOTON 8×8 serialized MDS matrix[6] over $GF(2^4)$ [22, Appendix C]. This method of construction, similar to the one used for the MDS layer of ECHO [5], produces an MDS transform that is very lightweight as compared to, for example, the 8×8 matrices[7] over $GF(2^8)$ used in WHIRLPOOL [4] or GRØSTL [20].

[6] We use PHOTON 's serialized matrix as we verified that it has the lowest binary weight over $GF(2^4)$.
[7] A comparison of their hardware estimations can be found in Section 6.2.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

In what follows, we describe this MDS transform for 128- and 256-bit SPN-Hash. Label the 8 bytes in each column as $a_1, a_2, \ldots, a_8$. We may write each byte as a concatenation of two 4-bit values, $a_i = a_i^L \parallel a_i^R$. Let $a^L = (a_1^L, a_2^L, \ldots, a_8^L)$ and $a^R = (a_1^R, a_2^R, \ldots, a_8^R)$. Let Q be the 8×8 MDS matrix over $GF(2^4)$ used in the PHOTON [22, Appendix C] hash function, i.e.

$$Q = (A_{256})^8 = \begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 2&4&2&11&2&8&5&6 \end{pmatrix}^8 = \begin{pmatrix} 2&4&2&11&2&8&5&6 \\ 12&9&8&13&7&7&5&2 \\ 4&4&13&13&9&4&13&9 \\ 1&6&5&1&12&13&15&14 \\ 15&12&9&13&14&5&14&13 \\ 9&14&5&15&4&12&9&6 \\ 12&2&2&10&3&1&1&14 \\ 15&1&13&10&5&10&2&3 \end{pmatrix}$$

The matrix Q is chosen as it can be implemented with a very low area footprint in hardware. This is due to the shifting property of A which simply updates the last cell of the column vector with a linear combination of all the vector components, and then rotates the vector by one position towards the top. The MDS layer is thus composed of 8 applications of A to the input column vector. This allows reuse of existing memory without need for temporary storage or additional control logic. Furthermore, the hash function using Q can be implemented efficiently in software using precomputed tables that combine the S-box and matrix coefficients.

We compute $b^L = Q \cdot a^L = (b_1^L, b_2^L, \ldots, b_8^L)$ and $b^R = Q \cdot a^R = (b_1^R, b_2^R, \ldots, b_8^R)$. For field multiplication over $GF(2^4)$, the irreducible polynomial $x^4 + x + 1$ is chosen with compactness as the main criterion. Then the output of the local diffusion layer is taken to be $(b_1, b_2, \ldots, b_8)$, where each b_i is a concatenation of the two 4-bit values, $b_i = b_i^L \parallel b_i^R$.

It can be shown that this transform is indeed MDS over $GF(2^8)$. Suppose the input a is non-zero. Then at least one of a^L or a^R is non-zero. Without loss of generality, suppose a^L is non-zero. Since Q is MDS, this means that the number of non-zero 4-bit values in (a^L, b^L) is at least 9. Hence, the number of non-zero bytes in (a, b) is at least 9.

512-bit SPN-Hash. The choice of matrix for the 16×16 MDS is left open to the reader. One possibility is to use the matrix proposed by Nakahara *et al.* in [25]. Note that Nakahara *et al.*'s matrix may not be lightweight due to its large size necessitating a large number of primitive operations. However, this is not an issue since it is unlikely that a 512-bit hash function will be used for lightweight purposes.

4 Provable Security of SPN-Hash Against Differential Collision Attack

Suppose an adversary wants a nonlinear component such as an S-box to have a specified input-output differential, $(\Delta_{input}, \Delta_{output})$, in a differential attack on a block cipher. He can always choose a pair of input messages, (x, x'), which satisfies the input difference $x \oplus x' = \Delta_{input}$. However, he cannot ensure this will result in the output difference $S(x \oplus k) \oplus S(x' \oplus k) = \Delta_{output}$ since he does not know the secret key k. Therefore the differential path is satisfied only with a certain probability.

However, there is no key in a hash function. Thus the adversary can choose a pair of input (x, x') corresponding to the message bits such that $x \oplus x' = \Delta_{input}$ and $S(x) \oplus S(x') = \Delta_{output}$ with probability 1 (as long as a solution exists). He can easily perform this process to bypass as many S-boxes as he can, all with probability 1, as long as the message bits corresponding to the input differentials of these S-boxes are independent of each other. Usually, this simple freedom degrees fixing method can be carried on to only a few rounds of the hash function (the *controlled rounds*), depending on the quality of diffusion of the internal components. Indeed, this process to bypass S-boxes for free becomes harder or impossible to execute in later rounds of the hash function since it is not easy to control message bits to satisfy several nonlinear equations simultaneously. As a consequence, after a few controlled rounds, the attacker has to let the S-box differentials happen probabilistically as in a block cipher differential attack for the remaining rounds of the function (the *uncontrolled rounds*).

Therefore, the optimal setting for a classical collision attack against a hash function is to consider differential characteristics for the controlled rounds and differentials for the uncontrolled rounds. As explained above, the differential characteristic is usually satisfied with low complexity in the controlled rounds where the attacker will be able to exploit freedom degrees (i.e. the number of independent binary variables he has to

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

determine). On the other hand, he considers differentials in the uncontrolled rounds which are fulfilled only probabilistically since the freedom degrees have already been fixed in the controlled rounds.

While some hash functions such as ECHO [5] do provide upper bounds on the maximum differential probability for a certain number of rounds, one has to note that *an attacker could leverage the fact that several output differences may lead to a collision.* The attacker then maximizes its probability to find a collision by considering differential characteristics for the controlled rounds and *all differentials that lead to a collision* for the uncontrolled rounds. As a consequence, for a sharper estimation of the collision resistance of a hash function, one has to sum the maximum differential probability bound over all the possible colliding output differences. To the best of our knowledge, no known hash function has yet provided such a collision resistance proof.

Looking at the SHA-3 competition candidates, such a proof seems hard to get. First, for most of the ARX (Addition-Rotation-XOR) functions submitted such as BLAKE [3] and SKEIN [19], obtaining a good maximum differential probability upper bound is really hard and remains an open problem. Even some SPN functions such as JH [35] could not provide any interesting bound regarding this criteria, due to its highly nested structure. On the other hand, ECHO [5] designers could prove that the maximum differential probability for four rounds of its internal permutation is upper bounded by 1.055×2^{-452}. However, for any of the ECHO variants, the number of colliding output differences is at least 2^{1536}, rendering such a proof impossible to apply.

The GRØSTL [20] hash function processes l-bit message blocks, $m_1, \ldots, m_t$, as $h_i \leftarrow f(h_{i-1}, m_i)$ for $i = 1, \ldots, t$, where $h_0 = IV$. The compression function f is given by $f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h$, where P and Q are two different l-bit permutations. Thus, f maps two l-bit inputs to an l-bit output. After the last message block has been processed, an output transformation Ω is applied, where Ω is given by $\Omega(x) = trunc_n(P(x) \oplus x)$ for an n-bit hash. Note that l, the size of the chaining value, shall always be at least twice the size of the hash output. It is easy to check that the internal differential collision probability of f is less than the birthday bound. However, the output transformation, involving a permutation followed by truncation of the last block, makes such a derivation much less straightforward for the external collision probability of the full hash function. Till now, there have been no published results on the differential collision probability of GRØSTL.

We show in the following that SPN-Hash can provide good maximum differential probability upper bounds for 4 rounds of its internal permutation and that its operating mode allows us to go further to prove that the sum of all colliding differential probabilities is still much lower than what an attacker would get with a generic birthday collision attack. We note that among the PHOTON [22] family of ciphers, our proof can only apply to the variant which uses the 288-bit internal permutation P_{288} involving 8-bit S-boxes. The other 4 members of the family use 4-bit S-boxes in their internal permutations which do not give low enough differential bounds after truncation.

4.1 Collision Resistance Analysis of the General SPN-Hash Construction

For our SPN-Hash construction, we make the extremely conservative estimate that the number of uncontrolled rounds is at least 4, which means that the adversary will run out of freedom degrees after six rounds or less, and he will not be able to control intermediate differentials in subsequent rounds. Note that the currently best known freedom degree utilizations for AES-like primitives, the rebound attack [28] and its variants [27, 21, 32] allow us to control three rounds only.

Let $\Delta Input$ denote the input differential and $\Delta Output$ be the output differential of the 4 last rounds of the SPN-Hash internal permutation. A collision for the hash function can occur either by an internal collision (a collision on the full $8nm$-bit internal state) or by an external collision during the last iteration (a collision on the right side of the $8nm$-bit internal state, the left side being truncated before outputing the hash value).

In the case of an external collision, this corresponds to P having an output differential of the form $(\Delta x, \Delta M) \in GF(2)^{4nm} \times GF(2)^{4nm}$, where XOR with the message difference ΔM in the right half will give a zero difference. The left half is truncated, so it can take any difference Δx. Thus the differential external collision probability in the last 4 rounds is given by:

$$Pr(\Delta Output = (\Delta x, \Delta M)) = \sum_{\Delta x \in GF(2)^{4nm}} Pr(\Delta Input \xrightarrow{4R} (\Delta x, \Delta M)).$$

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

We apply Theorem 4 to bound $Pr(\Delta Input \xrightarrow{4R} (\Delta x, \Delta M))$ in the following computation:

$$\begin{aligned}
Pr(External\ Collision) &= \sum_{\Delta x \in GF(2)^{4nm}} Pr(\Delta Input \xrightarrow{4R} (\Delta x, \Delta M)) \\
&\leq 2^{4nm} \times \max_{\Delta x \neq 0} Pr(\Delta Input \xrightarrow{4R} (\Delta x, \Delta M)) \\
&< 2^{4nm} \times \left(126 \times (2^{-7})^{n+1} + (2^{-6})^{n+1}\right)^m \\
&< 2^{4nm} \times \left((2^{-7})^n + (2^{-6})^{n+1}\right)^m \\
&= 2^{4nm} \times \left[(2^{-6n})(2^{-n} + 2^{-6})\right]^m \\
&< 2^{4nm} \times 2^{-6nm} = 2^{-2nm}
\end{aligned}$$

where the complexity of a generic birthday attack is $2^{4nm/2} = 2^{2nm}$.

In the case of an internal collision on the $8nm$-bit permutation P, since there is no truncation, the differential probability is given by $Pr(\Delta Input \xrightarrow{4R} (0, \Delta M_i))$ for all possible message differences ΔM_i. By Theorem 4 and in the same way as the computation above, we can show that this is lower than 2^{-2nm}, the complexity of a generic birthday attack for the hash function.

Assuming the adversary can bypass half of the rounds by exploiting freedom degrees (which is far from being possible with currently known cryptanalysis techniques), the above computations show that the adversary cannot gain any advantage by a differential collision attack (internal or external) since the `SPN-Hash` internal permutation has 10 rounds. Even if the adversary can find some clever way to extend the differential attack, we still have several rounds as buffer for protection.

Therefore, to summarize, the probability of success of a differential collision attack on $4nm$-bit `SPN-Hash` is less than 2^{-2nm} when the attacker can control at most all but 4 rounds, and this attack has worse complexity than a generic birthday technique.

4.2 Application to 128-, 256-, and 512-bit `SPN-Hash`

In this section, we give bounds for the differential collision probability of the 128-, 256-, and 512-bit `SPN-Hash` described in Section 3.3. In Section 5, we go on further to analyze the security of these hashes against preimage, second preimage, rebound attacks and its latest variants.

128-bit SPN hash. For 4 uncontrolled rounds, the maximum differential probability is upper bounded by:

$$\left(126 \times (2^{-7})^9 + (2^{-6})^9\right)^4 = 2^{-214.730}.$$

We sum this upper bound over all differential $\Delta x \in GF(2)^{128}$ (external collision) or $\Delta M_i \in GF(2)^{128}$ (internal collision) to bound the differential collision probability by

$$2^{128} \times 2^{-214.730} = 2^{-86.730} < 2^{-64}.$$

and this shows that a differential collision attack with at least 4 uncontrolled rounds will not perform better than a generic birthday attack.

256-bit SPN hash. For 4 uncontrolled rounds, the maximum differential probability is upper bounded by:

$$\left(126 \times (2^{-7})^9 + (2^{-6})^9\right)^8 = 2^{-429.461}.$$

We sum this upper bound over all differential $\Delta x \in GF(2)^{256}$ (external collision) or $\Delta M_i \in GF(2)^{256}$ (internal collision) to bound the differential collision probability by

$$2^{256} \times 2^{-429.461} = 2^{-173.461} < 2^{-128}.$$

and this shows that a differential collision attack with at least 4 uncontrolled rounds will not perform better than a generic birthday attack.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

512-bit SPN hash. Similarily, we show that for 4 uncontrolled rounds, the maximum differential probability is upper bounded by:

$$\left(126 \times (2^{-7})^{17} + (2^{-6})^{17}\right)^8 = 2^{-815.989}.$$

We sum this upper bound over all differential $\Delta x \in GF(2)^{512}$ (external collision) or $\Delta M_i \in GF(2)^{512}$ (internal collision) to bound the differential collision probability by

$$2^{512} \times 2^{-815.989} = 2^{-303.989} < 2^{-256}.$$

and this shows that a differential collision attack with at least 4 uncontrolled rounds will not perform better than a generic birthday attack.

The JH mode of operation is very handy for this type of proof as the amount of colliding differentials always remains acceptable compared to the internal permutation size. This is true even in the case of internal collisions because of the partial feedforward of the message block. The proof would be even easier to apply if we reduced the amount of message bits inserted during each iterations, but we maximize the bit-rate by incorporating message blocks of half the size of the internal permutation (for bigger message blocks, one could not guarantee ideal collision resistance anymore).

5 Analysis of Other Attacks on SPN-Hash

5.1 (Second)-Preimage Attack

The JH mode of operation has a similar structure to the Sponge construction [7], which can ensure differential collision resistance by proving good differential bounds for the underlying permutation. However, the Sponge construction provides preimage resistance up to $2^{c/2} = 2^{2nm}$, which is much lower than the expected 2^{4nm}. This is due to the fact that the permutation can be inverted easily, and one can compute a preimage in a MITM way. I.e., given a message with two blocks $M_0||M_1$, one can try 2^{2nm} different M_0' and 2^{2nm} truncations $H_{2,L}$, to meet at the state H_1. Note the left half of the state can be matched immediately once a proper M_1' is chosen, hence the time complexity is essentially a meet-in-the-middle for $4nm$ bits. In the JH mode of operation, there is an XOR of the message in the right half at the end of the permutation to make the meet-in-the-middle (MITM) attacks invalid. With the XOR, this MITM attack does not work since once M_1' varies, the state value H_2 changes as well and one cannot fix H_2 and vary M_1' simultaneously. This attack on the Sponge construction can easily be modified into a second pre-image attack, which is also defeated by the feedforward XOR in the JH mode of operation.

In [8], Bhattacharyya et al. provide the indifferentiability proof and a preimage attack against the JH-512 hash function with both time and memory complexity 2^{507}, using multi-collision techniques. This attack is a theoretical improvement because the complexity is still of the same magnitude as brute force search. Moreover, a generic time-memory trade-off (TMTO) attack with trade-off curve $T \cdot M^2 = N^2$ (attack complexity T, memory M and search space N), will perform much better. After a pre-computation with 2^{512} complexity (which is equivalent to performing Bhattacharyya's attack 32 times). Using 2^{507} memory to store the result will allow each subsequent online attack to take just $T = N^2/M^2 = 2^{10}$ time complexity.

5.2 Rebound Attack - Distinguishing Attack on Permutation P

In this section, we describe an attack on the permutation P to distinguish the permutation P from an ideal primitive on the same domain. Our overall strategy is to use the non-full-active Super S-box cryptanalysis technique as detailed in [32]. The technique in [32] is an enhancement of the original Super-Sbox analysis in [21, 26].

Central to the previous rebound [28] or start-from-the-middle [27] attacks, we attempt to find a pair of internal state values in the middle of an optimally chosen truncated differential path such that the path is verified for as many rounds as possible in both the backward and forward directions. This part is called the *controlled rounds* or *inbound* and the rest of the path fulfilled probabilistically are called the *uncontrolled rounds* or *outbound*. The main idea of the non-full-active Super S-box analysis is to use a differential path in which its Super S-boxes in the controlled rounds comprise only non-full-active states. For non-active bytes, the differential transition 0 to 0 always holds. Thus attackers can freely choose the value without breaking the path. This gives the attackers the freedom degrees to adjust other bytes inside the Super-Sbox to be connected efficiently. In the remaining of this section, we consider the 256-, 512- and 1024-bit permutations P of the 128-, 256-, and 512-bit SPN-Hash respectively.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

8-round differential paths. We view the 256-bit, 512-bit and 1024-bit internal state of the permutation P as a 8×4 matrix of bytes, a 8×8 matrix of bytes and a 16×8 matrix of bytes respectively. Figures 5, 6 and 7 of Appendix A depict 8-round differential paths for 256-bit P, 512-bit P and 1024-bit P respectively. A gray cell denotes an active byte while a white cell denotes a passive byte.

The controlled rounds. With reference to Figures 5, 6 and 7, the beginning of round 3 until the end of round 5 are the controlled rounds, indicated by dashed arrows in the figures. In round 4, we interchange the generalized optimal diffusion layer with the XOR operation followed by the substitution layer. This enables us to view the composition of substitution layer, MDS layer, XOR with round constant in round 4 and substitution layer of round 5 as an overall application of eight Super-Sboxes in parallel. Then by applying non-full-active Super S-box technique in [32],

- 256-bit P: we can obtain 2^{16} starting points, which are solutions of the controlled rounds with time 2^{16} and memory 2^{16};
- 512-bit P: we can obtain 2^8 starting points, which are solutions of the controlled rounds with time 2^8 and memory 2^8;
- 1024-bit P: we can obtain 2^{16} starting points, which are solutions of the controlled rounds with time 2^{16} and memory 2^{16}.

Hence we obtain a starting point with time 1 on average. As the computation procedure of the Super-Sboxes involves much technicalities, we refer interested readers to Section 4.2 of [32] for a detailed explanation.

The uncontrolled rounds. We obtained valid candidates from the beginning of round 3 until the end of round 5 but we have no control on the difference values prior to round 3 and beyond round 5. That is, the rest of the path is fulfilled probabilistically, denoted by solid arrows in Figures 5, 6 and 7.

- 256-bit P: The path is fulfilled with probability approximately 2^{-24} in round 2 and approximately 2^{-32} in round 6. For the rest of the rounds in the uncontrolled rounds, the path fulfilled with probability approximately 1. Therefore the differential characteristic probability of the 8-round differential path depicted in Figure 5 is approximately 2^{-56}.
- 512-bit P: The path is fulfilled with probability approximately 2^{-24} in round 2, approximately 2^{-24} in round 6 and approximately 1 in round 8. For the remaining of the uncontrolled rounds, the path is deterministic. Therefore the differential characteristic probability of the 8-round differential path depicted in Figure 6 is approximately 2^{-48}.
- 1024-bit P: Similarly, the differential characteristic probability of the 8-round differential path depicted in Figure 7 is approximately $2^{-24} \times 2^{-64} = 2^{-88}$.

The amount of freedom degrees. Before analyzing the complexity of the distinguishing attack, we first clarify the amount of freedom degrees available to the attacker. We want to ensure that there are enough solutions for the controlled rounds such that it is highly probable that at least one of them will fulfill the entire differential characteristic.

- 256-bit P: As calculated, we need 2^{56} starting points. Since we can choose 2^{80} differences at the start of the controlled rounds (4-byte difference at the beginning of round 3 and 6-byte difference at the end of round 5), we have enough freedom degrees to find a pair of values following the path.
- 512-bit P: As calculated, we need 2^{48} starting points. Since we can choose 2^{72} differences at the start of the controlled rounds (5-byte difference at the beginning of round 3 and 4-byte difference at the end of round 5), we have enough freedom degrees to find a pair of values following the path.
- 1024-bit P: Since we need 2^{88} starting points and we can choose $2^{8\times4} \times 2^{8\times9} = 2^{104}$ differences at the start of the controlled rounds, we have enough freedom degrees to find a pair of values following the path.

Distinguishers for P. We consider the attack complexity in the ideal case. More generally, we study the problem of mapping a i-bit zero difference mask to a j-bit zero difference mask through a t-bit ideal permutation. Following the argument of [21], if $j \leq 2(t-i)$, then $2^{\frac{j}{2}}$ input values from one single structure are sufficient to achieve a collision on the j target positions. The attack complexity is about $2^{\frac{j}{2}}$.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

– 256-bit P: By substituting $t = 256$, $i = 64$ and $j = 128$, the attack complexity in the ideal case is 2^{64}.
– 512-bit P: By substituting $t = 512$, $i = 192$ and $j = 192$, the attack complexity in the ideal case is 2^{96}.
– 1024-bit P: By substituting $t = 1024$, $i = 128$ and $j = 512$, the attack complexity in the ideal case is 2^{256}.

In contrast, the complexity of finding a valid pair for the whole 8-round differential path using the non-full-active Super S-box technique is as follows:

– 256-bit P: 2^{56} operations and 2^{16} memory,
– 512-bit P: 2^{48} operations and 2^{8} memory, and
– 1024-bit P: 2^{88} computations and 2^{16} memory.

We thus obtain a distinguisher for the 8-round reduced 256/512/1024-bit permutation P. To the best of our knowledge, the differential paths presented are among the longest paths with the least complexities. Since P comprises ten round functions, the distinguishing attacks do not threaten the security of the hash function.

5.3 Exploiting the MDS layer structure for 128- and 256-bit SPN-Hash

An attacker could try to exploit the special structure used to build the MDS layer. Indeed, in the case of 128- and 256-bit SPN-Hash, instead of using a diffusion matrix over $GF(2^8)$ (like for AES, WHIRLPOOL or GRØSTL), the layer is built by applying two times independently a diffusion matrix over $GF(2^4)$. Note that since the process is still equivalent to a MDS matrix, the proof on the number of active S-boxes is not impacted. However, in the case of truncated differential paths, an attacker could try to make some differential transitions to happen with better probability than expected by forcing at some stage that the differences remain on the left or on the right side of the bytes processed.

For example, a truncated differential transition verified probabilistically from 8 active bytes to 1 active byte should cost $2^{7\times 8} = 2^{56}$ tries, but if the all truncated differences are forced to be on the left or on the right side, then the complexity becomes $2^{7\times 4} = 2^{28}$ tries. This observation was one of the main properties used for the first attacks [31] on the ECHO hash function.

However, we believe such a strategy would very likely fail because this right/left property would be destroyed by the application of the AES S-box. Alternatively, forcing this property to be maintained for each active byte through the S-box layer would imply a big cost for the attacker, bigger than the gain from the truncated differential transitions. This has been confirmed by experiments, as there is no strong bias through the AES S-box in order to reach an all-right or an all-left difference (forward or backward). Note that in the case of ECHO this is not true since the 128-bit ECHO S-box is implemented by two AES rounds, for which forcing good truncated differential paths at no cost is easy.

6 Implementation

6.1 Software Performance

Due to its similarity with GRØSTL concerning the construction of the internal permutation, it is interesting to analyze SPN-Hash's software speed in the light of this SHA-3 candidate. The internal permutation of 256-bit SPN-Hash is comparable with GRØSTL-256 since their (individual) internal permutation are of the same size, and their amount of message bits per call to the internal permutation is the same (256-bit SPN-Hash compression function processes 256 bits message in 10 round operations, compared to 512-bit message in 20 very similar round operations by GRØSTL-256). The round function of these two hash functions should take about similar amount of time due to: 1) The equal number of substitution operations using the same AES sbox; 2) The speed of MDS multiplication is independent of the MDS coefficients in most table-based implementations; 3) The ShiftByte in GRØSTL is done together with step 2 in table-based implementations; 4) The round constants are a bit simpler than those in GRØSTL; and 5) There are three times $\oplus$ for 512 bits in GRØSTL and twice 256-bit $\oplus$ in 256-bit SPN hash. We did a simple and unoptimized implementation based on table lookups, which turns to be 34 cycles per byte on a Intel(R) Xeon(R) CPU E5640 clocked at 2.67GHz. We believe that there remains an important room for improvements by implementing SPN-Hash with optimized assembly instructions. Similar comparison argument applies when one considers implementations with the AES new instruction set, while GRØSTL-256 runs at 12 cycles per byte with internal parallelization of the two permutations P and Q, we expect SPN-Hash-256 to run at 12 to 24 cycles per byte mostly due to the fact that similiar parallelization is not possible. Note that the 128-bit SPN-Hash shall run as fast as the 256-bit version, since its compression function takes half the message bits, and uses roughly half the amount of operations. Test vectors are provided in Appendix Table 2.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

6.2 Hardware Performance

We have implemented 128-bit and 256-bit SPN hash in VHDL and used *Synopsys DesignCompiler A-2007.12-SP1* to synthesize it to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3*, which is based on the *UMC L180 0.18µm 1P6M* logic process with a typical voltage of 1.8 V. We used *Synopsys Power Compiler* version *A-2007.12-SP1* to estimate the power consumption of our ASIC implementations. For synthesis and for power estimation we advised the compiler to keep the hierarchy and use a clock frequency of 100 KHz.

Figure 8 in Appendix A shows the datapath of the hardware architecture. As can be seen, our serialized design is very similar to the design proposed in [22] and consists of six modules: `MDS`, `State`, `IO`, `AC`, `SC`, and `Controller`.

`IO` allows to 1) initialize our implementation with an all '0' vector; 2) input the IV, 3) absorb message chunks; 4) store the whole message block M_i until the end of the permutation; and 5) forward the output of the `AC` module to the `SC` module without further modification. It requires 638 GE and 1 236 GE, the vast majority of which is used for storing M_i.

`State` comprises a nm array of flip-flop cells storing 8 bits each. Every row constitutes a shift-register using the output of the last stage, *i.e.* column 0, as the input to the first stage (column $m-1$) of the same row and the next row. Using this feedback functionality Generalized Optimal Diffusion (GOD) can be performed in $m-1$ clock cycles with no additional hardware costs. Further, since GOD is performed on column 0, also a vertical shifting direction is required for this column. Consequently, columns 0 and $m-1$ consist of flip-flop cells with two inputs (6 GE), while columns 1 to $m-2$ consist of flip-flop cells with only one input (4.67 GE). Also the final addition of the message block to the right half is included in this module. The overall gate count for this module is $8 \cdot 2.67 + 8 \cdot n \cdot ((m-2) \cdot 4.67 + 2 \cdot 6)$ GE, that is 1 365 GE and 2 588 GE for 128-bit and 256-bit SPN-hash respectively.

`MDS` uses two instances of of A_{256} and calculates their last row in one clock cycle. The result is stored in the `State` module, that is in the last row of column 0, which has been shifted upwards at the same time. Consequently, after n clock cycles the MixColumnsSerial operation is applied to an entire column. Then the whole state array is rotated by one position to the left and the next column is processed. In total $m \cdot (n+1)$ clock cycles and 290 GE are required to perform MDS. In comparison, back-of-the-envelope estimations of the implementation of a single row of the 8×8 MDS matrices of `WHIRLPOOL` and `GRØSTL` using our technology indicate area requirements of around 256 GE and 350 GE, respectively. Adding minimal storage requirements of around 260 GE to store seven temporary bytes,[8] one can see that our MDS layer is indeed very light.

`AC` XORs $RCon_i$ to the output of the `State` module when a control signal indicates that the first row is processed. It requires 24 GE.

`SC` performs the S-box layer and comprises a single instantiation of Canright's representation of the AES S-box [13] which requires 233 GE. It takes nm clock cycles to perform AddConstant and SubCells on the whole state.

`Controller` uses a Finite State Machine (FSM) to generate all control signals required. Furthermore, also the round constants and the internal constants are generated within this module, as their values are used for the transition conditions of the FSM. The FSM consists of one idle state, one state for the combined execution of AC and SC, $m-1$ states for GOD and two states for MDS (one for processing one column and another one to rotate the whole state to the left). Naturally, its gate count varies depending on m: 200 GE for $m = 4$ and 254 GE for $m = 8$ respectively.

Table 1 in Appendix A compares our implementations of SPN hash with the remaining five `SHA-3` candidates with regards to area, latency and a FOM proposed by [2]. In order to have a fair comparison, we only include figures for fully-autonomous low-area ASIC implementations and omit figures for implementations that are optimized for high throughput. Among the `SHA-3` candidates `BLAKE`, `GRØSTL`, and `SKEIN`, 256-bit `SPN-Hash` is by far the most compact proposal. Though it has only the second highest FOM, our estimates for a 64-bit datapath implementation indicate that it can achieve the highest FOM, while still being 35% smaller than the most compact `SHA-3` candidate.

References

1. D. Augot, M. Finiasz, P. Gaborit, S. Manuel, and N. Sendrier. SHA-3 Proposal: FSB. Submission to NIST, 2008.

[8] The alternative would be to implement all eight rows in parallel, thereby increasing the area eight-fold.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

2. J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia. Quark: A Lightweight Hash. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *LNCS*, pages 1–15. Springer, 2010. `http://131002.net/quark/`.
3. J.-P Aumasson, L. Henzen, W. Meier, and R.C.-W. Phan. SHA-3 Proposal BLAKE. Candidate to the NIST Hash Competition 2008. Available at `http://131002.net/blake/`.
4. P. Barreto and V. Rijmen. The Whirlpool Hashing Function. Available at: `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html`.
5. R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw, and Y. Seurin. SHA-3 Proposal: ECHO. Submission to NIST (updated). 2009.
6. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge Functions. ECRYPT Hash Workshop, 2007.
7. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.
8. R. Bhattacharyya, A. Mandal, and M. Nandi. Security Analysis of the Mode of JH Hash Function. In Hong and Iwata [24], pages 168–191.
9. O. Billet, M.J.B. Robshaw, and T. Peyrin. On Building Hash Functions from Multivariate Quadratic Equations. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *LNCS*, pages 82–95. Springer, 2007.
10. A. Biryukov, D. Khovratovich, and I. Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.
11. A. Biryukov and D. Wagner. Slide Attacks. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 245–259. Springer, 1999.
12. A. Biryukov and D. Wagner. Advanced Slide Attacks. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *LNCS*, pages 589–606. Springer, 2000.
13. D. Canright. A Very Compact S-Box for AES. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005. The HDL specification is available at the author's official webpage `http://faculty.nps.edu/drcanrig/pub/index.html`.
14. J. Choy and K.M. Khoo. New Applications of Differential Bounds of the SDS Structure. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 367–384. Springer, 2008.
15. S. Contini, A.K. Lenstra, and R. Steinfeld. VSH, an Efficient and Provable Collision-Resistant Hash Function. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *LNCS*, pages 165–182. Springer, 2006.
16. Intel Corporation. Advanced Encryption Standard (AES) Instruction Set. Available at: `http://softwarecommunity.intel.com/articles/eng/3788.htm`, 2008/10/30.
17. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
18. Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.
19. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker. The Skein Hash Function Family. Submission to NIST (Round 2), 2009.
20. P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schlaffer, and S.S. Thomsen. Grøstl addendum. Submission to NIST (updated). 2009.
21. H. Gilbert and T. Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Hong and Iwata [24], pages 365–383.
22. J. Guo, T. Peyrin, and A. Poschmann. The PHOTON Family of Lightweight Hash Functions. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 222–239. Springer, 2011.
23. L. Henzen, J.-P. Aumasson, W. Meier, and R.C.W. Phan. VLSI Characterization of the Cryptographic Hash Function BLAKE. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, (99):1–9.
24. S. Hong and T. Iwata, editors. *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *LNCS*. Springer, 2010.
25. J. Nakahara Jr. and É. Abrahão. A New Involutary MDS Matrix for AES. *International Journal of Network Security*, 9(2):109–116, 2009.
26. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 126–143. Springer, 2009.
27. F. Mendel, T. Peyrin, C. Rechberger, and M. Schläffer. Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Jr. Jacobson, M.J., V. Rijmen, and R. Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 16–35. Springer-Verlag, 2009.
28. F. Mendel, C. Rechberger, M. Schläffer, and S.S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In O. Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 260–276. Springer-Verlag, 2009.
29. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
30. S. Park, S.H. Sung, S. Lee, and J. Lim. Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In Thomas Johansson, editor, *FSE*, volume 2887 of *LNCS*, pages 247–260. Springer, 2003.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

31. T. Peyrin. Improved Differential Attacks for ECHO and Grøstl. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 370–392. Springer, 2010.
32. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox Analysis: Applications to ECHO and Grøstl. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 38–55. Springer, 2010.
33. C. Shannon. Communication Theory of Secrecy System. *Bell System Technical Journal*, 28:656–715, October 1949.
34. Tillich, S. and Feldhofer, M. and Issovits, W. and Kern, T. and Kureck, H. and Mühlberghuber, M. and Neubauer, G. and Reiter, A. and Köfler, A. and Mayrhofer, M. Compact Hardware Implementations of the SHA-3 Candidates Arirang, Blake, Grøstl, and Skein. *IACR ePrint archive, Report 2009/349*, 2009.
35. H.J. Wu. The Hash Function JH. Submission to NIST (updated), September 2009. Available `http://ehash.iaik.tugraz.at/uploads/1/1d/Jh20090915.pdf`.

A Tables and Figures

Table 1. Comparison of Low-Area Hardware implementations of SPN hash and a selection of `SHA-3` finalists.

Digest size	Alg.	Ref.	Msg. size	Technology	Area [GE]	Latency [clk]	T'put@100KHz [kbps]	FOM [nbps/GE^2]
128	SPN-Hash-128		256	UMC 0.18	2 777	710	36.1	2 338
	SPN-Hash-128		256	*estimate*	4 600	230	55.7	2 627
256	SPN-Hash-256		512	UMC 0.18	4 625	1 430	35.8	837
	SPN-Hash-256		512	*estimate*	8 500	230	111.3	1 541
	BLAKE-32	[23]	512	UMC 0.18	13 575	816	62.8	340
	GRØSTL-224/256	[34]	512	AMS 0.35	14 622	196	261.2	1 222
	SKEIN-256-256	[34]	256	AMS 0.35	12 890	1 034	24.8	149

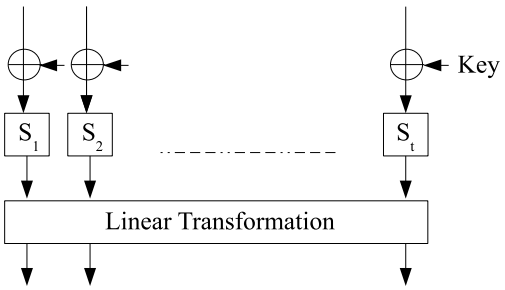

Fig. 1. One round of a SPN structure

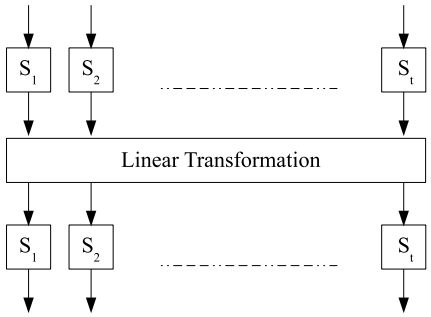

Fig. 2. The SDS function

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

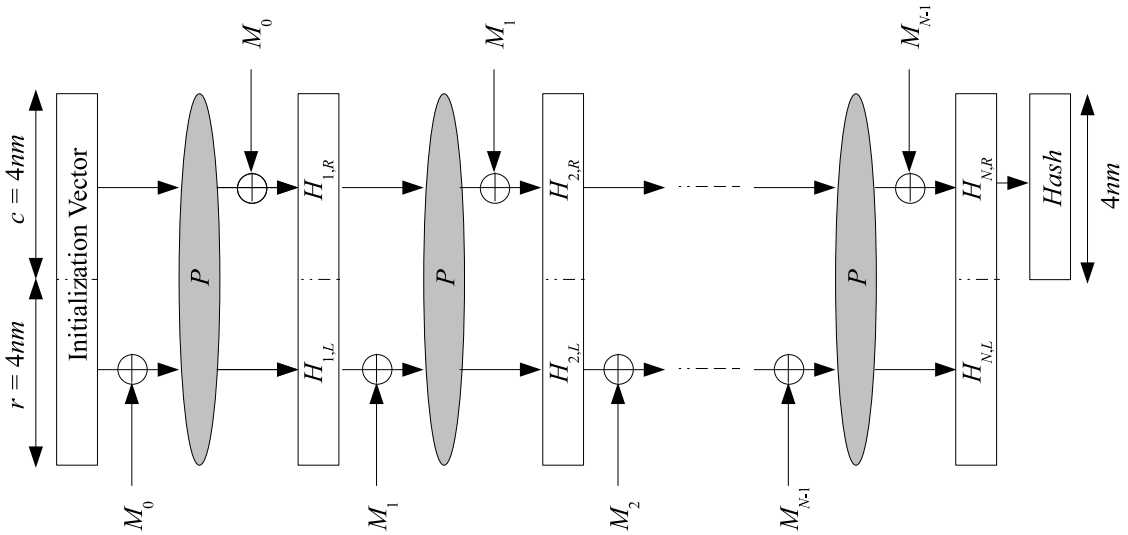

Fig. 3. The `JH` mode of operation

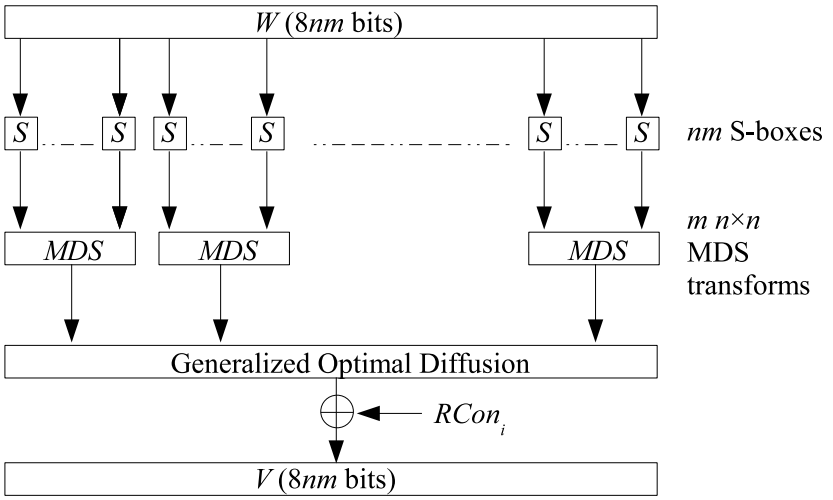

Fig. 4. The round function in permutation P

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

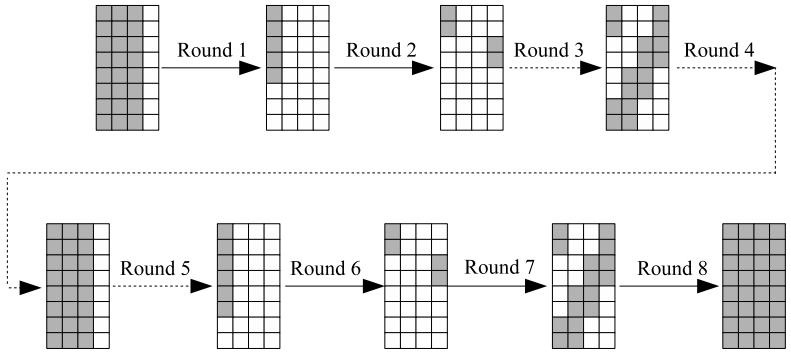

Fig. 5. 8-round differential path for a 256-bit permutation P

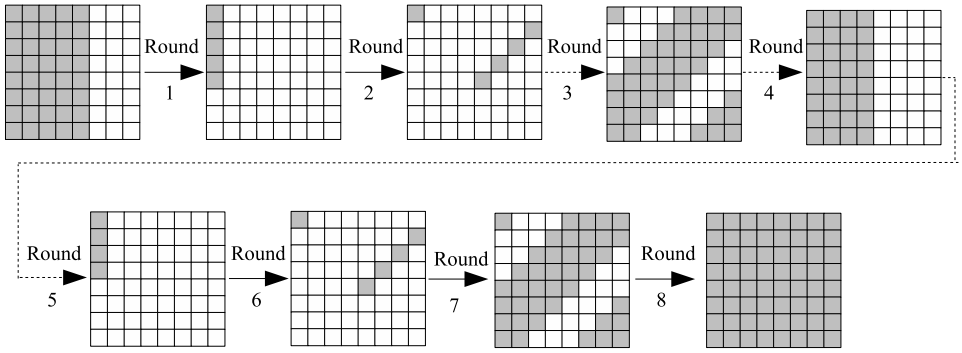

Fig. 6. 8-round differential path for a 512-bit permutation P

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

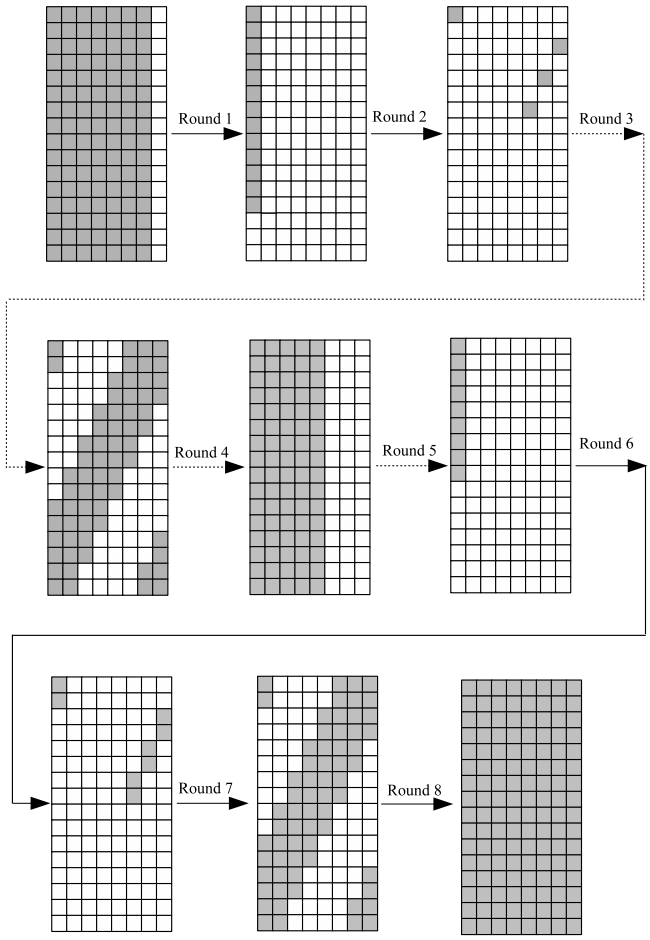

Fig. 7. 8-round differential path for a 1024-bit permutation P

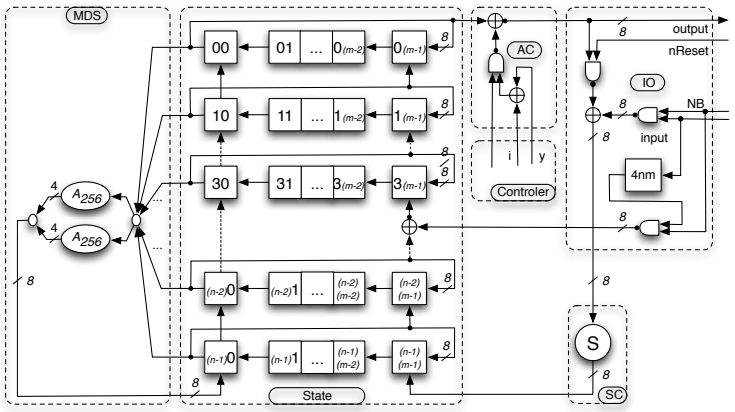

Fig. 8. Serial hardware architecture of 128-bit and 256-bit SPN hash. 512-bit SPN hash can use the very same architecture, except for the `MDS` component.

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012

B Proofs

B.1 Proof of Theorem 3

Proof. Firstly, note that since π is a (m, n)-generalized optimal diffusion, by definition, so is π^{-1}. Suppose we have one active bundle in the layer θ. Since the transforms in that layer are all MDS, there are $n+1$ differentially active input/output bytes in that bundle. These originate from or get distributed to at least $\lceil (n+1)/(n/m) \rceil = m+1$ bundles. Clearly, the remaining case where more than one bundle in θ is active will result in at least $m+1$ active input/output bundles to $\pi \circ \theta \circ \pi$, making it a $m \times m$ MDS transform over $GF(2^{8n})$.

B.2 Proof of Theorem 4

Proof. Here we base much of the proof on the wide trail strategy (see [18, Theorem 3], [17, Theorem 9.5.1]). Since σ and π are byte-based operations, we may interchange their order. After this admissible re-arrangement of operations, we can view four rounds of permutation P as a successive alternation between $\sigma \circ \theta \circ \sigma$ and $\pi \circ \theta \circ \pi$.

Each bundle over a $\sigma \circ \theta \circ \sigma$ transformation is a $8n$-bit SDS structure consisting of the linear map θ sandwiched between two layers of n 8-bit S-boxes. Since the MDS layer θ has branch number $n+1$ and the S-boxes are all affine equivalent to the inversion function over $GF(2^8)$, we can apply Theorem 1 with $s = 8$ and $\beta_d = n+1$ for the `AES` S-Box. The maximum differential probability of $\sigma \circ \theta \circ \sigma$ is thus bounded by $DP_{\sigma\circ\theta\circ\sigma}$ which is at most

$$\max \left\{ \max_{1 \le u \le 255} \sum_{j=1}^{255} \{DP^S(u,j)\}^{n+1}, \max_{1 \le u \le 255} \sum_{j=1}^{255} \{DP^S(j,u)\}^{n+1} \right\}.$$

We need to find the difference distribution $DP^S(u,j)$ where $u \in GF(2^8)$ is fixed and j varies over all of $GF(2^8)$. This is given by the difference distribution for each row of the difference table. Similarly, we need to find the difference distribution of the columns. Since all the S-boxes used are affine transforms of the inversion function, they have the same difference distribution. Furthermore, as explained in [14, Section 4], the difference distribution of the columns of the inversion function is the same as that for the rows. Thus, we only need to consider the rows in the difference table. This simplifies the upper bound in Theorem 1 to $\sum_{j=1}^{255} \{DP^S(u,j)\}^{n+1}$, $u \neq 0$.

The difference distribution for each row of the inversion mapping is the following: 129 differences with no occurence, 126 differences with two occurences and 1 difference with four occurences. Hence, $DP_{\sigma\circ\theta\circ\sigma} \le 126 \times (2^{-7})^{n+1} + (2^{-6})^{n+1}$.

By Theorem 3, $\pi \circ \theta \circ \pi$ is a $m \times m$ MDS transform over $GF(2^{8n})$, i.e. it has a branch number $m+1$ acting on bundles of n bytes. Thus, we can view 4 rounds of P as $\pi \circ \theta \circ \pi$ sandwiched between two layers of m bundles, where each bundle has differential probability at most $126 \times (2^{-7})^{n+1} + (2^{-6})^{n+1}$. Then by Theorem 2, the maximum differential probability of 4 rounds of P is bounded by

$$(DP_{\sigma\circ\theta\circ\sigma})^m < \left(126 \times (2^{-7})^{n+1} + (2^{-6})^{n+1}\right)^m.$$

C Test Vector

We hash the message "SPN-Hash: Improving the Provable Resistance Against Differential Collision Attacks" with three variants of the `SPN-Hash` family, and the following are digests generated by our reference implementation.

`SPN-Hash-128`	`2b021df78220afd2a41fa3592dc7d284`
`SPN-Hash-256`	`eabd18110d48e81d0663a7034b265462bf93f8019ca292e58ec1d830f90d67c5`
`SPN-Hash-512`	`f3e4a3dcc44acb2cf4d6f5f67bd8ce50ef030f55e0189a322136b5fc46af3cf5` `e071f1ee9bf1851bbd854540da1ccc496d679b43090f8e24f486d6866092ac02`

Table 2. Test vectors for three variants of `SPN-Hash` family

Appeared in S. Vaudenay (Ed.): AFRICACRYPT 2012, LNCS 7374, pp. 270–286, 2012