2.B Algorithm Specifications and Supporting Documentations

2.B.6 STATEMENT of the advantages and limitations of NaSHA hash algorithm

1 Advantages

NaSHA-(m, k, r) hash algorithm has several advantages:

1. Implementation of the algorithm in various environments, including 64-bit, 32-bit and 8-bit platforms.

2. Usage of known and well examined starting bijection, to exclude the fear of a trapdoor function.

3. Usage of the wide-pipe design of Lucks [2, 3] and Coron's [1] suggestions, to gain resistance to some known generic attacks.

4. The quasigroups used in every iteration of the compression function are different and depend on the processed message block. This is done by the special definition of the quasigroup operations by extended Feistel networks. Even in the same iteration, we are using two different quasigroups for $\mathcal{A}$ and $\mathcal{RA}$ transformations, and both of them depend on the processed message block. This feature will surely make harder the attacker's job.

5. 16 deployments of the transformation $LinTr_{256}$ and 32 deployments of the transformation $LinTr_{512}$ can be implemented in hardware by 64 LFSRs. Note that $LinTr_{512}$ is in fact the LFSR obtained from the primitive polynomial $x^{32} + x^{25} + x^{15} + x^7 + 1$ over the Galois field GF(2), applied in parallel 64 times, while $LinTr_{256}$ is obtained in the same way from the primitive polynomial $x^{16} + x^{10} + x^7 + x^4 + 1$.

6. Possibility to hash messages of length up to $2^{128} - 1$ bits.

7. The definition of NaSHA hash algorithm allows the bit length m of a message digest to be any number between 125 and 512. Namely, the output result of NaSHA-$(256, 2, 6)$ is given by

$$\text{NaSHA-}(256, 2, 6)(M) = A_4||A_8||A_{12}||A_{16} \ (\texttt{mod}\ 2^{256})$$

and it is enough to take NaSHA-$(256, 2, 6)(M) = A_4||A_8||A_{12}||A_{16}$ (mod 2^m), for $125 \leq m \leq 256$, without any other changes in the algorithm. Similarly, for NaSHA-$(512, 2, 6)$ it is enough to take NaSHA-$(512, 2, 6)(M) = A_4||A_8||\ldots||A_{32}$ (mod 2^m), for $257 \leq m \leq 512$.

2 Limitations

We can see thew following limitations of NaSHA-(m, k, r) hash algorithm:

1. The speed-security performances of NaSHA-(m, k, r) depends at most on the complexity k of $\mathcal{MT}$, since k defines the number of composite mappings $\mathcal{A}_l$ and $\rho(\mathcal{RA}_l)$. Larger k means more security and lower speed. By our experience, if we increase k to 4 we have decreasing of the speed performances by factor about 1.5 to 2, and for $k = 6$ the decreasing of speed performances is almost 2.5.

2. Inability of parallelism of the composite mappings $\mathcal{A}_l$ and $\rho(\mathcal{RA}_l)$ in NaSHA-(m, k, r) compression function, because each of them depends of the previous mappings in line. There is no place for parallelism of the operations inside one composite mapping either.

3. NaSHA-$(224, 2, 6)$ and NaSHA-$(256, 2, 6)$ are slower than SHA224 and SHA256, by factor ranging from 1.74 to 1.89 on 32-bit architectures and by factor 1.3 on 64-bit architectures. NaSHA-$(384, 2, 6)$ and NaSHA-$(512, 2, 6)$ are slower than SHA384 and SHA512, by factor 2 on 64-bit architectures.

References

[1] J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya, *Merkle-Damgård revisited: How to construct a hash function*, CRYPTO 2005, LNCS **3621**

[2] S. Lucks, *Design Principles for Iterated Hash Functions*, Cryptology ePrint Archive, Report 2004/253

[3] S. Lucks, *A Failure-Friendly Design Principle for Hash Functions*, ASIACRYPT 2005, LNCS **3788** (2005), pp. 474–494