Vortex: A New Family of One-way Hash Functions Based on AES Rounds and Carry-less Multiplication

Shay Gueron[2, 3, 4] and Michael E. Kounavis[1]

[1] Corresponding author, Corporate Technology Group, Intel Corporation, Hillsboro, OR, USA
[2]Department of Mathematics, University of Haifa, Haifa, ISRAEL,
[3]Applied Security Research Group, Center for Computational Mathematics and Scientific Computations, University of Haifa, Haifa, ISRAEL
[4] Mobility Group, Intel Corporation, IDC, Haifa, ISRAEL

Abstract. We present Vortex a new family of one way hash functions that can produce message digests of 256 bits. The main idea behind the design of these hash functions is that we use well known algorithms that can support very fast diffusion in a small number of steps. We also balance the cryptographic strength that comes from iterating block cipher rounds with SBox substitution and diffusion (like Whirlpool) against the need to have a lightweight implementation with as small number of rounds as possible. We use only 3 AES rounds as opposed to 10 since our goal is not to protect a secret symmetric key but to support perfect mixing of the bits of the input into the hash value. Three AES rounds are followed by our variant of Galois Field multiplication. This achieves cross-mixing between 128-bit sets. We present a set of qualitative arguments why we believe Vortex supports collision resistance and first pre-image resistance.

1 Introduction

Guaranteeing message and code integrity is very important for the security of applications, operating systems and the network infrastructure of the future Internet. Protection against intentional alteration of data is typically supported using one way hash functions. A one way hash function is a mathematical construct that accepts as input a message of some length and returns a digest of much smaller length. One way hash functions are designed in such a way that it is computationally infeasible to find the input message by knowing the digest only. One way hash functions which have been in use today include algorithms like MD-5 and SHA1. One way hash functions which are likely to be used in the future include SHA256, SHA384 and SHA512 [10].The problem with using these algorithms is that they are time consuming when implemented in software. One way hash functions typically involve multiple shifts, XOR and ADD operations which they combine in multiple rounds in order to produce message digests. Because of this reason, one way hash functions consume a substantial number of processor clocks when executing which limits their applicability to high speed secure network applications (e.g., 10 Gbps e-commerce transactions), or protection against malware (e.g., hashed code execution).

In this document we describe an alternative approach where a family of one way hash function is built from other security algorithms used as building blocks, which help with achieving fast mixing across a large number of input bits. Using the Merkle-Damgård construction [6, 7]

as a framework we construct a compression function from AES rounds and a novel merging technique based on Galois Field (GF(2)) multiplication. Using three successive AES rounds we provide mixing across 128 bits. Using a merging function based on Galois Field (GF(2)) multiplication we provide mixing across sets of 128 bits. Perfect mixing is accomplished through combinations of AES rounds and our merging function.

We have conducted 2^{20} experiments computing the collision resistance and the pseudorandom oracle preserving property of our family. Whereas our work is in progress our first results indicate that there is no experimental evidence that Vortex is inferior in terms of its collision resistance and pseudorandom oracle preserving property when compared to SHA256. Performance-wise, however, the difference is substantial. SHA256 operates at 21 cycles per byte on a Core 2 Duo processor. Vortex is expected to operate at a speed of 1.5 cycles per byte in future CPUs with instruction set support for AES round computation and Galois Field (GF(2)) multiplication, which is the current trend in processor industry. We believe that the design of the Vortex family is important because it represents a scalable on-the-CPU solution for message and code integrity and can be used for supporting both high speed secure networking and protection against malware in next generation computing systems.

The document is structured as follows: In Section 2 we describe the design methodology of the Vortex family. In Section 3 we describe the algorithm. In Section 4 we describe our experiments and present qualitative arguments why we believe the Vortex family is secure. Finally in Section 5 we provide some concluding remarks.

2 Design Methodology of the Vortex Family

Vortex represents a new family of one way hash functions that can produce message digests of 256 bits. The main idea behind the design of these hash functions is that well known algorithms can support very fast diffusion in a small number of steps. Our intent is to allow each bit of an input block to affect all bits of a hash after a small number of computations.

The algorithms we use in our design are:

- The AES round due to its capability to perform very fast mixing across 32-bits as a stand-alone operation and 128-bits if combined with at least one more round; and:
- A variant of Galois Field (GF(2)) multiplication due to its capability to cross mix bits of different sets (i.e., the input operands) in a manner that is cryptographically stronger than the simpler Feistel reordering proposed in modes like MDC-2 [3].

We also balance the cryptographic strength that comes from iterating block cipher rounds with SBox substitution and diffusion (like Whirlpool) against the need to have a lightweight implementation with as small number of rounds as possible. We use only 3 AES rounds as opposed to 10 since our goal is not to protect a secret symmetric key but to support perfect mixing of the bits of the input into the hash value. The design choice of 3 comes from the fact that 2 rounds is the bare minimum number needed for 128-bit wide mixing. Our design, however, is open for introducing more rounds if this is proven necessary in the future. Three AES rounds are followed by our variant of Galois Field Multiplication. This achieves cross-mixing between 128-bit sets. Our transformation is not simple carry-less multiplication but is combined with some bit reordering and combination of XORs and additions with carries. In this way our variant of Galois Field Multiplication:

- achieves better cross-mixing than the straightforward carry-less multiplication between the 128-bit inputs
- is a non commutative operation protecting against attacks based on swapping the order of the chaining variables in the processing of a message.

Our family of one way hash functions uses the AES round as specified in the standard FIPS-197 as a building block but also introduces new a merging functions for combining the outputs of AES round transformations into 256-bit digests as explained in detail below.

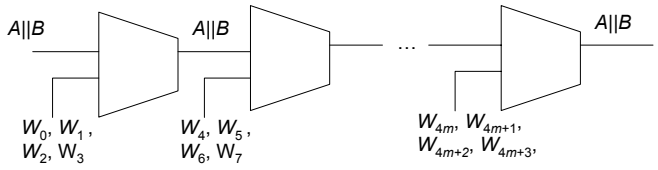

Figure 1: Vortex as a Merkle-Damgård construction

3 Algorithm Description

Vortex processes an input stream as a sequence of 512-bit blocks. The stream is padded with a '1'. If the length of the stream is not a multiple of 512 minus 96, then the stream is padded with zeros following the '1'. The last 96 bits indicate the configuration of the hash (32 bits) and the length of the stream (64 bits). Each block is divided into two sub-blocks of 256 bits each and each sub-block is divided into words W_0 and W_1 of 128 bits each.

Vortex operates on 2 128-bit variables A and B initialized to some constant values. It processes each block using AES rounds modifying the values of A and B. In the end it returns the concatenation of A and B $A\|B$.The algorithm for processing a sub-block is the following:

Vortex sub-block(A, B, W_0, W_1)
{
; W_0, W_1 be the words of the current sub-block to be processed
$A \leftarrow \widetilde{A}_{W_0}(A)$
$B \leftarrow \widetilde{A}_{W_1}(B)$
$A\|B \leftarrow V_M^{(A)}(A,B)$
return(A, B)
}

The algorithm for processing a block is the following:

Vortex block(A, B, W_0, W_1, W_2, W_3)
{
$(A,B) \leftarrow (A,B) \oplus$ Vortex sub-block(A, B, W_0, W_1) ; uses W_0 and W_1
$(A,B) \leftarrow (A,B) \oplus$ Vortex sub-block(A, B, W_2, W_3) ; uses W_2 and W_3
}

As one can see the Vortex block is essentially a Merkle-Damgård construction. It accepts a chaining variable $A\|B$ and four input words W_0, W_1, W_2, W_3 and returns an updated value of the chaining variable $A\|B$. Such construction in illustrated in Figure 1.

The other aspect that can be observed is that Vortex block incorporates a Davies-Meyer structure around the Vortex sub-block in order to make the transformation non-reversible. Such structure is repeated twice as shown in Figure 2:

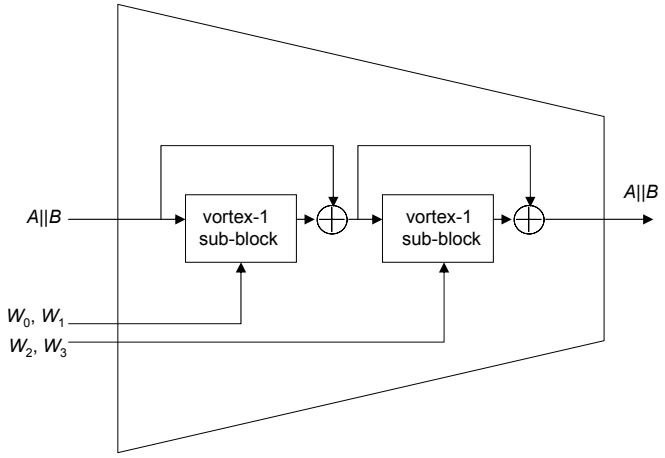

Figure 2: Davies-Meyer structure of the Vortex block

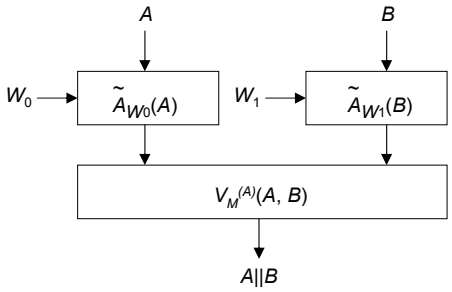

Figure 3: Vortex sub-block

The Vortex sub-block is built upon two mathematical functions: The transformation $\widetilde{A}_K(x)$ which is a lightweight block cipher and the merging function $V_M^{(A)}(A,B)$. There are two instances of the transformation $\widetilde{A}_K(x)$ in the Vortex sub-block. Each instance processes a different chaining variable among A, B. Each instance of the transformation $\widetilde{A}_K(x)$ treats its input chaining variable as a plaintext and its input word, which is one from W_0, W_1, W_2, W_3, as

a key as it is the norm in the Davies-Meyer structure. The merging function $V_M^{(A)}(A, B)$ combines the outputs of the two instances of $\widetilde{A}_K(x)$ into the new value of the concatenation $A\|B$ of the chaining variables A, B. The structure of the Vortex sub-block is shown in Figure 3.

The transformation $\widetilde{A}_K(x)$ is a lightweight block cipher based on an AES round that encrypts x, which is 128 bits long, using the key K. $\widetilde{A}_K(x)$ uses three AES rounds as specified in the standard FIPS-197 [1]. Each AES round consists of a round key addition in GF(2), followed by an SBox substitution phase, followed by the ShiftRows transformation, followed by the MixColumns transformation. The key schedule algorithm used by $\widetilde{A}_K(x)$ is different from that of AES. $\widetilde{A}_K(x)$ uses three 128-bit wide *Rcon* values RC_1, RC_2 and RC_3 to derive three round keys RK_1, RK_2 and RK_3 as follows:

$RK_1 \leftarrow \text{SBox}(K \boxplus RC_1)$
$RK_2 \leftarrow \text{SBox}(RK_1 \boxplus RC_2)$
$RK_3 \leftarrow \text{SBox}(RK_2 \boxplus RC_3)$

where by '$\boxplus$' we mean addition modulo 2^{128}. As explained before, a single AES round performs diffusion across 32 bits. This is accomplished through the combination of the S-Box and Mix Columns transformations. Two AES rounds diffuse across 128 bits. This is accomplished through the combination of the subsequent Shift Rows and Mix Columns transformations. Three rounds further strengthen the diffusion performed. The *Rcon* values can be considered fixed or deriving from A and B. The exact relationship between *Rcon*, A, B is yet to be determined.

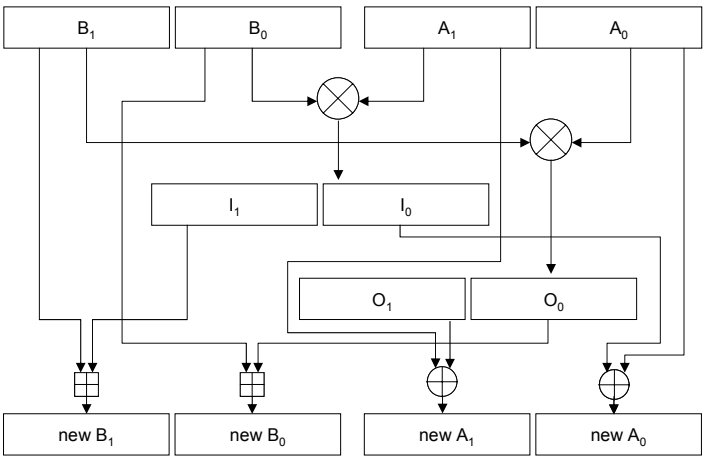

Figure 4: The Merging Function of Vortex

The merging function, shown in Figure 4, $V_M^{(A)}(A, B)$ operates as follows:

$V_M^{(A)}(A, B)$
{

let $A = [A_1, A_0]$
let $B = [B_1, B_0]$
$O \leftarrow A_0 \otimes B_1$
$I \leftarrow A_1 \otimes B_0$
let $I = [I_1, I_0]$
let $O = [O_1, O_0]$
return $[B_1 \boxplus I_1, B_0 \boxplus O_0, A_1 \oplus O_1, A_0 \oplus I_0]$
}

where by '$\boxplus$' we mean addition modulo 2^{64}, and '$\otimes$' we mean carry-less multiplication.

The merging g function is based on carry-less multiplication. This is stronger than Feistel reordering which is a simple bit permutation. Our merging function makes sure that the bits of A impact the bits of B and vice versa. In fact, each bit of one variable affects a significant number of the bits of the other variable in a non-linear manner. This makes our design better than a straightforward XOR or other simple mathematical operation. One can also observe that even though our merging function is strong cryptographically, it does not accomplish perfect mixing by itself. This is because each bit of A or B affects a large number of bits of the other variable but not all of them. Perfect mixing is accomplished by the 3 AES rounds that follow our merging function. So, for a pair of input words W_0, W_1 perfect mixing is accomplished after a sequence of 3 AES rounds (mix across 128 bits), merging using Galois Field multiplication (cross-mix across 128 bit sets but not perfect mixing) and another set of 3 AES rounds as part of the sub-block processing to follow. For this reason whereas a regular Vortex sub-block is processed using 3 AES rounds and a merging function, the last Vortex sub-block is processed using 3 AES rounds, merging, yet another 3 AES rounds and subsequent merging again.

4 Security and Performance of Vortex

The security of the Vortex family was investigated experimentally by conducting a large number of experiments (2^{20}) hashing the Vortex specification document with random perturbations, which were superimposed on it. For these experiments we computed the probability of collision and the distribution of the output differentials. Subsequently we compared the numbers we got from Vortex with numbers we got from SHA256. No collision occurred in our experiments. Our first results indicated that there is no experimental evidence that Vortex is inferior in terms of its collision resistance and pseudorandom oracle preserving property when compared to SHA256.

In fact our results can be interpreted that there is strong indication that the Vortex family is at least as secure as SHA256 even though it uses a small number of block cipher rounds. There are several reasons for this. First AES round is a good mixing function. The key used is completely data dependent and hence our scheme does not suffer from known attacks on compression functions that use a small set of keys [8]. The key schedule transformation of Vortex is stronger than AES due to the fact that the SBox transformation is applied across each 128 bit round key as opposed to 32 bits only and that round constants are added using integer addition modulo 2^{128} as opposed to XOR. It is the combination of two independent sources of non-linearities in the key schedule, i.e., addition with carries and inversion in GF(256) that makes the key schedule transformation provides potentially effective against differential attacks -even though the number of AES rounds is reduced for achieving better performance.

The merging function of Vortex combines linear (XORs) and non-linear (adds with carries) transformations with 64-bit carry-less multiplication building blocks. This operation is non-

commutative and when combined with previous and subsequent AES rounds and Galois Field multiplication achieves perfect mixing across 256 bits. By designing the merging function to be non-commutative we destroy any symmetry in the computation of the Vortex sub-block that could be a potential source of collision. If Vortex was designed such that its merging function is commutative, then an attacker could easily create a collision by generating a message that swaps the position of chaining variables *A* and *B* as compared to another given message.

A more thorough analytical study on the security of the Vortex family is yet to be conducted. As past of future work we plan to develop a methodology for computing the collision resistance and the first pre-image resistance of our construction based on the divide-and-conquer approach that was first developed in the study of the MDC-2 mode by Steinberger [3]. Such approach helps with reasoning about the collision and pre-image resistance of specific components of hash functions. Components of hash functions include adders, shifters, XORs, S-Boxes, linear diffusers, bit permutations etc. Whereas our merging function is more complex than the MDC-2 mode of operation we believe that it can be potentially analyzed due to the fact that it combines relatively simple building blocks (i.e., multipliers adders and XORs). In addition multipliers are carry-less accepting small size input operands (i.e., 64 bits). These facts make the collision and pre-image resistance of our construction potentially easier to compute than MDC-2.

This is work in progress. We believe that future designs of the Vortex family may be different than what is described in this paper. As part of future work we want to investigate the optimal relationship between the *Rcon* constants of the Vortex key schedule and the chaining variables *A*, *B*. We would like to also investigate whether the presence of simple carry-less multiplication is sufficient in the merging function or not. Any non-zero operand multiplied with zero results in zero. Such fact can increase the collision probability associated with merging function of Vortex. If this is proven to be a design deficiency it can be potentially corrected with simple modifications to the algorithm. For example, a single carry-less multiplication can be replaced by two multiplications where in one of the two operands are XOR-ed with a correcting constant and the results of the multiplications are XOR-ed with each other.

We estimate that the Vortex family of algorithms can have substantial performance gain when implemented in software in future processors with support for AES round computation and Galois Field multiplication. This is the current trend in the industry. Expected performance is at 1.5 cycles per byte which is approximately 14X gain as compared to SHA256.

5 Concluding Remarks

We presented Vortex a new family of one way hash functions that can produce message digests of 256 bits. The main idea behind the design of these hash functions is that we use well known algorithms that can support very fast diffusion in a small number of steps. We presented a set of qualitative arguments why we believe Vortex supports collision resistance and first pre-image resistance and described a setoff experiments that gave us confidence that the Vortex design is not inferior to SHA256 in terms of its security properties. Performance-wise the expected difference between Vortex and earlier work is substantial. SHA256 operates at 21 cycles per byte on a Core 2 Duo processor. Vortex is expected to operate at a speed of 1.5 cycles per byte in future CPUs with instruction set support for AES round computation and Galois Field (GF(2)) multiplication. We believe that the design of the Vortex family is important because it represents a scalable on-the-CPU solution for message and code integrity and can be used for supporting both high speed secure networking and protection against malware in next generation computing systems.

References

1. "Advanced Encryption Standard", Federal Information Processing Standards Publication 197, available at: http://csrc.nist.gov/publication/fips
2. J. Daemen and V. Rijman, "The Wide Trail Design Strategy", *B. Honary (Ed.): Cryptography and Coding 2001,* LNCS 2260, pp. 222-238, Springer Verlag 2001.
3. J. P. Steinberger, "The Collision Intractability of MDC-2 in the Ideal Cipher Model", *Advances in Cryptology - EUROCRYPT 2007*, LNCS 4515, pp. 35-41, 2007
4. L. Knudsen, X. Lai and B. Preneel, "Attacks on Fast Double Block Length Hash Functions", *Journal of Cryptology*, No. 11, pp. 59-72, International Association for Cryptologic Research, 1998.
5. S. Lucks, "Design Principles for Iterated Hash Functions", *Cryptology ePrint Archive*, Report 2004/253, 2004. Available at: http://eprint.iacr.org
6. I. Damgård, "A Design Principle for Hash Functions", *Advances in Cryptology – CRYPTO 1989*, LNCS 435, pp. 416-427, 1989.
7. R. Merkle, "One Way Hash Functions and DES", *Advances in Cryptology – CRYPTO 1989*, LNCS 435, pp. 428-446, 1989.
8. J. Black, M. Cochran and T. Shrimpton, "On the Impossibility of Highly Efficient Block Cipher-based Hash Functions", *Advances in Cryptology – EUROCRYPT 2005*, LNCS 3494, pp. 526-541, 2005.
9. M. Bellare and T. Ristenpart, "Multi-Property-Preserving Hash Domain Extension and the EMD Transform", *Advances in Cryptology – ASIACRYPT 2006*, LNCS 4284, pp. 299-314, 2006.
10. "Secure Hash Standard", Federal Information Processing Standards Publication 180-2, available at: http://csrc.nist.gov/publication/fips