AES-VCM, AN AES-GCM CONSTRUCTION USING AN INTEGER-BASED UNIVERSAL HASH FUNCTION.

ED KNAPP

Abstract. We give a framework for construction and composition of universal hash functions. Using this framework, we propose to swap out AES-GCM's $\mathbb{F}_{2^{128}}$-based universal hash function for one based on VMAC, which uses integer arithmatic. For architectures having AES acceleration but where either $\mathbb{F}_{2^{128}}$ acceleration is absent or exists on the same execution unit as AES acceleration, an integer-based variant of AES-GCM may offer a performance advantage, while offering identical security.

1. Introduction

Construction of MACs based on universal hash functions was first proposed by Wegman and Carter [13]. Bernstein [1] provides a nice summary of the history universal-hash-based MACs.

We give some hash function definitions and notation.

Definition 1. *Let $\{h_k\}_{k\in K}$ be a collection of functions $h_k\colon A \to B$, where A, B, K are arbitrary. If*

$$\Pr[h_k(m) = h_k(m')]_{k\in K} \le \varepsilon,$$

for all m, $m' \in A$, then we say that the collection $\{h_k\}$ is ε almost universal (ε-AU). *When the collection $\{h_k\}$ is understood, we may simply say h_k is ε-*AU.

Definition 2. *Consider the group $\langle G, +\rangle$. Let $\{h_k\}_{k\in K}$ be a collection of functions $h_k\colon A \to G$, where A, K are arbitrary. If*

$$\Pr[h_k(m) - h_k(m') = \delta]_{k\in K} \le \varepsilon,$$

for all $\delta \in G$ and distinct m, $m' \in A$, then we say that the collection $\{h_k\}$ is ε almost delta universal (ε-AΔU) with respect to $+$. *When G and the collection $\{h_k\}$ are understood, we may simply say h_k is ε-*AΔU.

Definition 3. *Let $\{h_k\}_{k\in K}$ be a collection of functions $h_k\colon A \to B$, where A, B, K are arbitrary. If*

$$\Pr[h_k(a) = b]_{k\in K} \le \varepsilon,$$

*for all $a \in A$, $b \in B$, then we say that the collection $\{h_k\}$ is ε-*distributed. *When the collection $\{h_k\}$ is understood, we may simply say h_k is ε-*distributed.

From here on, $\mathbb{F}$ is understood to be a field, G is understood to be a group, and $\mathbb{Z}_n$ is the integers modulo n. For a set S and non-negative integers $a \le b$, define

$$S^{[a,b]} = \bigcup_{i=a}^{b} S^i.$$

Informally, we give a result regarding the use of a universal hash function to construct a secure MAC.

Lemma 1. [11, Theorem 1] *Let h be ε-AΔU. We can construct a "secure" counter-based mac as* $\mathrm{MAC}(c, m) = h_{k_1}(m) + \mathrm{AES}_{k_2}(c)$.

Construction of authenticated encryption schemes based on universal hash functions include AES-GCM [10] and a VMAC-based scheme [9]. Herein, we describe a MAC using a VHASH-like universal hash function. Also, we describe a cipher based on this universal hash function and modeled after AES-GCM, called AES-VCM.

2. universal hash function primatives

We discuss several constructions for universal hash functions which can be composed to build hash functions suitable for MAC constructions.

Winnograd intoduced a method for computing inner products using fewer multiplications [14]. This technique was adapted to universal hash functions by Halevi and Krawczyk [5], attibuting the result to unpublished work by Wegman and Carter. We present the result in only two coordinates and later give a generic way to generalize universal hash functions to larger domains.

Lemma 2 (Pseudo dot product hash)**.** [2, Theorem 4.2] *Let n be a positive integer. For $k \in \mathbb{Z}_n^2$, define $h_k \colon \mathbb{Z}_n^2 \to \mathbb{Z}_{n^2}$ as*

$$h_k(m) = (m_1 + k_1) \cdot (m_2 + k_2),$$

where addition is in $\mathbb{Z}_n$ and multiplication is lifted to $\mathbb{Z}_{n^2}$. Then h_k is n^{-1}-AΔU.

Next, we prove that, additionally, the pseudo dot product is ε-distributed.

Lemma 3. *Let n be a positive integer. For $k \in \mathbb{Z}_n^2$, define $h_k \colon \mathbb{Z}_n^2 \to \mathbb{Z}_{n^2}$ as*

$$h_k(m) = (m_1 + k_1) \cdot (m_2 + k_2),$$

where addition is in $\mathbb{Z}_n$ and multiplication is lifted to $\mathbb{Z}_{n^2}$. Then h_k is n^{-1} distributed.

Proof. Fix $(m_1, m_2) \in \mathbb{Z}_n^2$ and $\delta \in \mathbb{Z}_{n^2}$. Note that since addition by m_i modulo n is a permutation, we have

$$\Pr\left[(m_1 + k_1)(m_2 + k_2) = \delta\right] = \Pr[k_1 k_2 = \delta].$$

Since k_1, $k_2 \in \mathbb{Z}_n$, the product $k_1 k_2 \in \mathbb{Z}_{n^2}$ can be taken to be over the integers, a unique factorization domain. For each k_1, there exists at most one k_2 such that $k_1 k_2 = \delta$ and so $\Pr[k_1 k_2 = \delta] \le n^{-1}$. □

Next, we present a polynomial-based universal hash function, where the message is encoded as coefficients in the polynomial. The result was independently proposed by den Boer [4]; Johansson, Kabatianskii, and Smeets [6]; and Taylor [12].

Lemma 4 (Polynomial hash)**.** [12, Theorem 1] *Let N be a positive integer. For $k \in \mathbb{F}$, define $h_k \colon \mathbb{F}^N \to \mathbb{F}$ as*

$$h_k(m) = \sum_{i=1}^{N} k^i \cdot m_i.$$

Then $\{h_k\}$ is a ε-AΔU, where $\varepsilon = N \cdot |\mathbb{F}|^{-1}$.

We state a well-known composition lemma.

Lemma 5 (Composition)**.** *Let $f \colon A \to B$ be ε_1-AU and $g \colon B \to C$ be ε_2-AΔU. Then $g \circ f$ is $(\varepsilon_1 + \varepsilon_2)$-A$\Delta$U.*

Proof. Fix m_1, $m_2 \in A$, and $\delta \in C$. Then

$$\begin{aligned}\Pr[g(f(m_1)) - g(f(m_2)) = \delta] &= \Pr[g(f(m_1)) - g(f(m_2)) = \delta \mid f(m_1) = f(m_2)] \\ &\quad \cdot \Pr[f(m_1) = f(m_2)] \\ &+ \Pr[g(f(m_1)) - g(f(m_2)) = \delta \mid f(m_1) \neq f(m_2)] \\ &\quad \cdot \Pr[f(m_1) \neq f(m_2)] \\ &\leq \Pr[0 = \delta](\varepsilon_1) + (\varepsilon_2) \cdot \Pr[f(m_1) \neq f(m_2)] \\ &\leq (1)(\varepsilon_1) + (\varepsilon_2)(1) = \varepsilon_1 + \varepsilon_2\end{aligned}$$

□

The VMAC paper [3] gives a result allowing us to truncate the outputs of certain universal hash functions.

Lemma 6 (Round)**.** *Let a, b, and n be positive integers such that $2^a \leq n \leq 2^b$. Let $f\colon A \to \mathbb{Z}_n$ be ε-AΔU and define $g\colon A \to \mathbb{Z}_{2^a} : m \mapsto (f(m) \mod 2^a)$. Then g is $2^{b-a}\varepsilon$-AΔU.*

Proof. This follows from Corollary 5 of the second VMAC paper [3]. □

It follows trivially that a ε-AΔU function is a ε-AU function. The next lemma shows that we can create a ε-AU function from a ε-AΔU function while extending the domain.

Lemma 7. *Let $f_k\colon A \to G$ be ε-AΔU. Define $g_k\colon A \times G \to G$ by $g_k(a, b) = f_k(a) + b$. Then g is ε-AU.*

Proof. Let (a, b), $(a', b') \in A \times G$ be distinct. If $a = a'$, then $b \neq b'$ and so for every key k, we have that $g_k(a, b) = f_k(a) + b \neq f_k(a) + b' = g_k(a', b')$ and so

$$\Pr[g_k(a, b) = g_k(a', b')] = 0 \leq \varepsilon.$$

If $a \neq a'$, then set $\delta = b' - b$ and so

$$\Pr[g_k(a, b) = g_k(a', b')] = \Pr[f_k(a) - f_k(a') = \delta] \leq \varepsilon.$$

□

Given universal hash functions h_1, h_2, we can construct functions $h(m_1, m_2) = h_1(m_1) + h_2(m_2)$, as well as $h(m) = \langle h_1(m), h_2(m)\rangle$. These constructions respectively expand the input length and decrease the collision proabability. We can compose the techniques, achieving both goals. The next lemma shows that we can compose the techniques while reusing some of the hash functions, reducing key lengths.

Lemma 8 (Stacked functions)**.** *Let n and s be positive integers. Let $f_k\colon M \to G$ be ε-AΔU, where $k \in K$. For $k' \in K^{s+n-1}$, define $h_{k'}\colon M^n \to G^s$ by $h_{k'}(m) = S \cdot \mathbf{1}_n$, where $S = [f_{k'_{i+j-1}}(m_j)]_{i,j}$ is the s by n matrix and $\mathbf{1}_n$ is the n-dimensional all-ones vector. Then $h_{k'}$ is ε^s-AΔU.*

Proof. Let m, $m' \in M^n$ be distinct and let $\delta \in G^s$. Let ℓ be the least integer such that $m_\ell \neq m'_\ell$. Let A_i be the statement "$[h_k(m)]_i - [h_k(m')]_i = \delta_i$". Then

$$\Pr[h_k(m) - h_k(m') = \delta] = \Pr\left[\bigwedge_{i=1}^{s} A_i\right] = \prod_{i=1}^{s} \Pr\left[A_i \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right].$$

Next, we restrict each probability space in the product to a single coordinate of $k \in K^{s+n-1}$, the $(i+\ell-1)$-th coordinate. We define $B_i(k_1^*, \ldots, k_{i+\ell-2}^*, k_{i+\ell}^*, \ldots, k_{s+n-1}^*)$ to be the statement "$k_j = k_j^*$ for all $j = 1, \ldots, s+n-1$, $j \neq i+\ell-1$". Then

$$\Pr\left[A_i \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right] = \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \wedge B_i(k^*) \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right]$$
$$= \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \,\middle|\, B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] \Pr\left[B_i(k^*) \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right].$$

We can rearrange the statement A_j as

$$A_j : \delta_j = [h_k(m)]_j - [h_k(m')]_j = \sum_{t=1}^{n} \left(f_{k_{j+t-1}}(m_t) - f_{k_{j+t-1}}(m'_t)\right)$$
$$= \sum_{t=\ell}^{n} \left(f_{k_{j+t-1}}(m_t) - f_{k_{j+t-1}}(m'_t)\right),$$

which shows that A_j depends only on k_r, for all $j+\ell-1 \leq r \leq j+n-1$. Notice that if $i < j$, then $i+\ell-1 < j+\ell-1$ and A_j does not depend on $k_{i+\ell-1}$. Therefore, for each $k^* \in K^{s+n-2}$,

$$\Pr\left[A_i \,\middle|\, B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] = \Pr\left[A_i \mid B_i(k^*)\right].$$

Next, notice that for each i, there exists δ_i^* independent of $k_{i+\ell-1}$ such that

$$[h_k(m)]_i - [h_k(m')]_i - \delta_i = f_{k_{i+\ell-1}}(m_\ell) - f_{k_{i+\ell-1}}(m'_\ell) - \delta_i^*$$

and therefore, for a fixed k^*, the statement A_i is equivalent to the statement "$f_{k_{i+\ell-1}}(m_\ell) - f_{k_{i+\ell-1}}(m'_\ell) = \delta_i^*$", which gives us

$$\Pr\left[A_i \mid B_i(k^*)\right] = \Pr\left[f_{k_{i+\ell-1}}(m_\ell) - f_{k_{i+\ell-1}}(m'_\ell) = \delta_i^* \,\middle|\, B_i(k^*)\right] \leq \varepsilon.$$

Putting this all together, we have

$$\begin{aligned}
&\Pr[h_k(m) - h_k(m') = \delta] \\
&\quad = \prod_{i=1}^{s} \Pr\left[A_i \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad = \prod_{i=1}^{s} \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \,\middle|\, B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] \Pr\left[B_i(k^*) \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad = \prod_{i=1}^{s} \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \mid B_i(k^*)\right] \Pr\left[B_i(k^*) \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad \leq \prod_{i=1}^{s} \varepsilon \sum_{k^* \in K^{s+n-2}} \Pr\left[B_i(k^*) \,\middle|\, \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad = \varepsilon^s,
\end{aligned}$$

which proves the lemma. □

The next lemma shows that if we start with an f_k that is also ε-distributed, then we can extend the domain of the result, making is easier to construct variable-length universal hash functions.

Lemma 9. *Let n and s be positive integers. Let $f_k\colon M \to G$ be ε-AΔU and ε-distributed, where $k \in K$. For $k' \in K^{s+n-1}$, define $h_{k'}\colon M^{[1,n]} \to G^s$ by $h_{k'}(m) = S_\ell \cdot \mathbf{1}_\ell$ for $m \in M^\ell$, where $S_\ell = [f_{k'_{i+j-1}}(m_j)]_{i,j}$ is the s by ℓ matrix and $\mathbf{1}_\ell$ is the ℓ-dimension all-ones vector. Then $h_{k'}$ is ε^s-AΔU.*

Proof. Let $m \in M^a$, $m' \in M^b$ be distinct, where a, b are integers in $[1, \ldots, n]$. Let $\delta \in G$. By symmetry, take $a \leq b$. If there exists an index ℓ such that $m_\ell \neq m'_\ell$, then the proof follows exactly as in Lemma 8 and we're done.

Assume that $a < b$ and $m_1 = m'_1$, …, $m_a = m'_a$. Set $\ell = a + 1$. Let A_i be the statement "$[h_k(m')]_i = \delta_i$" and B_i be as in Lemma 8. Observe that

$$\begin{aligned}
&\Pr[h_k(m) - h_k(m') = \delta] \\
&\quad = \Pr\left[\bigwedge_{i=1}^{s} A_i\right] = \prod_{i=1}^{s} \Pr\left[A_i \;\middle|\; \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad = \prod_{i=1}^{s} \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \;\middle|\; B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] \Pr\left[B_i(k^*) \;\middle|\; \bigwedge_{j=i+1}^{s} A_j\right]
\end{aligned}$$

Again, we can follow the proof in Lemma 8 by observing that A_i is independent of k_{i+j-1} for all $j < \ell$ and that there exists δ_i^* independent of $k_{i+\ell-1}$ such that A_i is equivalent to "$[f_{k_{i+\ell-1}}(m'_\ell)]_i = \delta_i^*$", which gives us, for each k^*, that

$$\Pr\left[A_i \;\middle|\; B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] = \Pr\left[A_i \mid B_i(k^*)\right] \leq \varepsilon.$$

Finally,

$$\begin{aligned}
&\Pr[h_k(m) - h_k(m') = \delta] \\
&\quad = \prod_{i=1}^{s} \sum_{k^* \in K^{s+n-2}} \Pr\left[A_i \;\middle|\; B_i(k^*) \wedge \bigwedge_{j=i+1}^{s} A_j\right] \Pr\left[B_i(k^*) \;\middle|\; \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad \leq \prod_{i=1}^{s} \varepsilon \sum_{k^* \in K^{s+n-2}} \Pr\left[B_i(k^*) \;\middle|\; \bigwedge_{j=i+1}^{s} A_j\right] \\
&\quad \leq \varepsilon^s,
\end{aligned}$$

proving the Lemma. □

3. Higher-order universal hash functions

In this section, we compose our functions from Section 2 to construct a universal hash function capable of taking more-arbitrary inputs and mapping to a hash-output-sized range. Defining a function similar to $\{0,1\}^* \to \mathbb{Z}_{2^{128}}$ seems like a nice goal. For our cipher construction, we would like to be able to hash two independent bytestrings, the AAD and the ciphertext. To this end, our goal for this section will be a function of the form $\{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_{2^{128}}$.

We start by applying lemmas 2 and 3 with $n = 2^{64}$ to obtain a 2^{-64}-AΔU and 2^{-64}-distributed function,

$$\mathbb{Z}_{2^{64}}^2 \to \mathbb{Z}_{2^{128}}. \tag{1}$$

Using Lemma 8 with $n = 8$, $s = 2$, and Equation (1) as f_k, we obtain a 2^{-128}-AΔU function

$$(\mathbb{Z}_{2^{64}}^2)^8 \to (\mathbb{Z}_{2^{128}})^2. \tag{2}$$

Using Lemma 9 with $n = 8$, $s = 2$, and Equation (1) as f_k, we obtain a 2^{-128}-AΔU function

$$(\mathbb{Z}_{2^{64}}^2)^{[1,8]} \to (\mathbb{Z}_{2^{128}})^2. \tag{3}$$

Applying Lemma 6 to equations (2), (3) with $a = 126$ and $b = 128$, we obtain 2^{124}-AΔU functions

$$(\mathbb{Z}_{2^{64}}^2)^8 \to (\mathbb{Z}_{2^{126}})^2, \tag{4}$$

$$(\mathbb{Z}_{2^{64}}^2)^{[1,8]} \to (\mathbb{Z}_{2^{126}})^2. \tag{5}$$

We extend the domain of Equation (5) using Lemma 7, yielding a 2^{124}-AU function

$$(\mathbb{Z}_{2^{64}}^2)^{[1,8]} \times (\mathbb{Z}_{2^{126}})^2 \to (\mathbb{Z}_{2^{126}})^2. \tag{6}$$

Let f_4, f_5, and f_6 denote equations (4), (5), and (6) respectively. Let $m \in (\mathbb{Z}_{2^{64}}^2)^*$ have ℓ coordinates (each in $\mathbb{Z}_{2^{64}}^2$). Set $\ell' = \lceil \ell/8 \rceil - 1$, define $m'_i = (m_{8i+1}, \ldots, m_{8i+8}) \in (\mathbb{Z}_{2^{64}}^2)^8$ for $i = 1, \ldots, \ell'$, and define $\tilde{m} = (m_{8\ell'+1}, \ldots m_\ell) \in (\mathbb{Z}_{2^{64}}^2)^{\ell - 8\ell'} \subseteq (\mathbb{Z}_{2^{64}}^2)^{[1,8]}$. We define $\mathbb{Z}_{2^{124}}^2$ functions NH and NH′ as

$$\mathrm{NH}\colon (\mathbb{Z}_{2^{64}}^2)^\ell \to \mathbb{Z}_{2^{126}}^{2\ell'+2} \colon m \mapsto \langle f_4(m'_1), \ldots, f_4(m'_{\ell'}), f_5(\tilde{m}) \rangle, \tag{7}$$

$$\mathrm{NH}'\colon (\mathbb{Z}_{2^{64}}^2)^\ell \times \mathbb{Z}_{2^{126}}^2 \to \mathbb{Z}_{2^{126}}^{2\ell'+2} \colon (m, a) \mapsto \langle f_4(m'_1), \ldots, f_4(m'_{\ell'}), f_6(\tilde{m}, a) \rangle. \tag{8}$$

Equations (7) and (8) give us functions capable of encoding unbounded (but not arbitrary) bitstrings. To handle arbitrary (pairs of) bitstrings, we encode the length(s) into $\mathbb{Z}_{2^{126}}^2$. Let $m_1, m_2 \in \{0,1\}^*$. Let ℓ_1 be the bitlength of m_1 modulo 128 and ℓ_2 be the bitlength of m_2. Let $\tilde{m}_1, \tilde{m}_2$ be m_1, m_2 zero-padded to a bitlength that is a multiple of 128. We construct

$$\begin{gathered}\{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_{2^{126}}^{2\lceil \ell_1/128 \rceil + 2\lceil \ell_2/128 \rceil} \colon \\ (m_1, m_2) \mapsto \langle f_7(\tilde{m}_1), f_8(\tilde{m}_2, \ell_1 \cdot 2^{64} + \ell_2, \ell_1 \cdot 2^{64} + \ell_2) \rangle,\end{gathered} \tag{9}$$

which is a 2^{124}-AU.

Equation (9) takes as input two arbitrary bitstrings but produces an output whose length is a fraction of the input. The next step is to produce a fixed-length output. We instantiate Lemma 4 with $p = 2^{127} - 1$ and $N = 2^{32} + 1$. We invoke Lemma 8 with the resulting function, $n = 2$, an $s = 1$ to produce a $(2^{32} + 1)p^{-1}$-AΔU function

$$\mathbb{Z}_p^N \times \mathbb{Z}_p^N \to \mathbb{Z}_p. \tag{10}$$

We desire to build a function with a variable-length domain out of Equation (10). We define the following injection

$$\mathbb{Z}_p^{[0,N-1]} \to \mathbb{Z}_p^N \colon (m_1, \ldots, m_t) \mapsto (0, \ldots, 0, 1, m_t, \ldots, m_1). \tag{11}$$

Put more simply, we postpend $1 \in \mathbb{Z}_p$, followed by zero-padding, and reverse the order of the result. Composing the injection from Equation (11) with Equation (10), we obtain a function

$$\mathbb{Z}_p^{[0,N-1]} \times \mathbb{Z}_p^{[0,N-1]} \to \mathbb{Z}_p. \tag{12}$$

Note that substituting a positive integer $N' < N$ for N in the previous construction gives an identical construction to restricting the domain $\mathbb{Z}_p^{[0,N-1]}$ to $\mathbb{Z}_p^{[0,N'-1]}$ in Equation (12). This implies that we can obtain a stronger hash function, a $N'p^{-1}$-AΔU function, if we restrict the domain of the $N = 2^{32}$ instantitation.

Let f_9 denote Equation (9) and f_{12} denote Equation (12). Given a message $(m_1, m_2) \in \{0,1\}^* \times \{0,1\}^*$, let $\tilde{m}_1, \ldots, \tilde{m}_{2t}$ be such that

$$f_9(m_1, m_2) = \langle \tilde{m}_1, \ldots, \tilde{m}_{2t} \rangle \in \mathbb{Z}_{2^{126}}^{2t}.$$

Define m_1^*, $m_2^* \in \mathbb{Z}_p^t$, dividing up the even and odd $\tilde{m}_i$ terms as follows

$$m_1^* = \langle \tilde{m}_1, \tilde{m}_3, \ldots, \tilde{m}_{2t-1} \rangle$$
$$m_2^* = \langle \tilde{m}_2, \tilde{m}_4, \ldots, \tilde{m}_{2t} \rangle$$

and define

$$\{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_p \colon (m_1, m_2) \mapsto f_{12}(m_1^*, m_2^*). \tag{13}$$

For messages m_1, m_2 with bitlengths bounded by ℓ_1, ℓ_2 respecitvely, set $\ell^* = 2 \cdot \lceil \ell_1/1024 \rceil + 2 \cdot \lceil \ell_2/1024 \rceil$. By Lemma 5, the function in Equation (13) is ε-AΔU where $(2^{-124} + (\ell^* + 1) \cdot p^{-1})$. We can bound ℓ^* by $(\ell_1 + \ell_2)2^{-9} + 2$ and p^{-1} by 2^{-126} giving us $\varepsilon \le (\ell_1 + \ell_2)2^{-135} + 2^{-123}$.

4. Comparison with other VHASH functions

We compare simplified versions of four VHASH-like functions, the two functions defined in the VMAC papers [3, 8], the actual VMAC implementation [7], and our new function. In the following comparisons, we ignore the need for hashing variable-length messages, since it follows the technique outlined with Section 3.

Let $p = 2^{127} - 1$ and $q = 2^{64} - 2^8 - 1$. The VMAC functions are composed principally of an NH hash, $(\mathbb{Z}_{2^{64}}^2)^8 \to \mathbb{Z}_{2^{128}}^2$, followed by a polynomial hash $\mathbb{Z}_p^*$. The result of the NH step is rounded down to 126 bits so that it maps into $\mathbb{Z}_p$.

The first VMAC paper [8] describes a hash function with 128-bit tags that has the following structure:

$$((\mathbb{Z}_{2^{64}})^{16})^N \xrightarrow{\text{NH}} ((\mathbb{Z}_{2^{128}})^2)^N \xrightarrow{\text{Round}} ((\mathbb{Z}_{2^{126}})^2)^N \xrightarrow{\text{Poly1271}} (\mathbb{Z}_p)^2 \xrightarrow{\text{Sum}} \mathbb{Z}_p$$

The second VMAC paper [3] recommends two instantiations of 64-bit VHASH with distinct keys. The main difference for the 64-bit version is that an additional step is need to map $\mathbb{Z}_p$ into a 64-bit result. Two injections, $\mathbb{Z}_p \to \mathbb{Z}_{2^{64}-2^{32}} \to \mathbb{Z}_q^2$, and a pseudo dot product over $\mathbb{Z}_q$ are used to obtain a 64-bit result. The 64-bit VHASH has the following structure:

$$((\mathbb{Z}_{2^{64}})^{16})^N \xrightarrow{\text{NH}} (\mathbb{Z}_{2^{128}})^N \xrightarrow{\text{Round}} (\mathbb{Z}_{2^{126}})^N \xrightarrow{\text{Poly1271}} \mathbb{Z}_p \xrightarrow{\text{Split}} (\mathbb{Z}_{2^{64}-2^{32}})^2 \xrightarrow{\text{Tiny NH}} \mathbb{Z}_q$$

The VMAC implementation [7] follows the second structure, except the two 64-bit VHASH instantiations have related keys.

The authors mention the need for larger keys, comparing the first and second version:

"On the negative side, defining VMAC-128 as two iterations of VMAC-64 introduces an additional 32-bytes of key, a separate polynomial computation and a third hashing stage, all of which are slowing influences."

The authors mention a version in their first paper which they claim does not require much more internal key.

"If an application needs collision probabilities less than those of VHASH, then VHASH could be applied to given messages twice, using a different key each time. Alternatively, Figure 3 gives a hash function VHASH-128 based on the same principles as VHASH, but producing 128-bit outputs without the need for significantly more internal key than VHASH."

It is unclear from their description if they mean to describe what is contained in the code, however they give no proof in any case. The implementation of the 128-bit version uses 48 bytes of additional key compared to the 64-bit version.

The lemmas we give in Section 2 can be composed to prove the security of the VMAC implementation [7] and both versions of VMAC [3, 8].

5. AES-VCM

5.1. **Key generation.** Our cipher follows the same general construction as AES-GCM, a universal hash function combined with AES-CTR.

An AES-VCM key is generated from an AES key. The internal AES-VCM key is composed of the (expanded) AES key and the hash function keys. Portions of the space of AES-CTR are reserved for the internal KDF, used for generating the keys of the universal hash function. We view the counter as a four-tuple of little-endian 32-bit integers. The NH keys are generated using the counters $\langle 0,0,0,0\rangle$, $\langle 0,0,0,1\rangle$, ..., $\langle 0,0,0,8\rangle$. The AES outputs of these counters are treated as pairs of 64-bit little-endian integers. The polynomial keys are generated as the AES outputs of the counter values $\langle 0,0,1,0\rangle$, $\langle 0,0,1,1\rangle$, treated as 128-bit little-endian integers, the high-order 3 bits of every 32-bit block are zeroed.

5.2. **Authenticated encryption.** A unique IV is required for each authenticated encryption. In the case of a 96-bit counter, an internal AES counter is formed using a little-endian 32-bit integer '1' for the low bits (recall '0' is reserved for the internal KDF) and the 96-bit IV for the high bits. IVs that are not 96-bits are handled similarly to AES-GCM, generated using an identically modified version of the underlying universal hash function. The low 32 bits of the internal AES counter is little-endian incremented modulo 2^{32} to obtain a new counter for each AES block.

Let A be associated data with bitlength n and P be plaintext data with bitlength m. Let m' be the least integer such that $m \leq 128m'$. Define 128-bit blocks P_i so that $P = (P_1, \ldots, P_{m'})$ with $P_{m'}$ zero-padded to 128-bits, if needed. Let Z_1 be the counter produced by the IV and let Z_2, ..., $Z_{m'+1}$ be next m' counter values. Define $C_i = P_i \oplus AES_K(Z_i)$ for $i = 1, \ldots, m'$ and set $C = (C_1, \ldots, C_{m'})\rceil_m$, where '$\rceil_m$' denotes truncation to m bits. We hash the value (A, C) to obtain T and set $T^* = T + AES_K(Z_{m'+1})$, computed modulo 2^{128}. The resulting ciphertext is C and tag is T^*.

This construction is very similar to AES-GCM, differing only in endianness, the distribution of the counter space, and the underlying hash function.

References

[1] Daniel J Bernstein. Polynomial evaluation and message authentication. 2007. `http://cr.yp.to/papers.html`.

[2] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. Umac: Fast and secure message authentication. In *Advances in CryptologyCRYPTO99*, pages 216–233. Springer, 1999.

[3] Wei Dai and Ted Krovetz. VHASH security. *IACR Cryptology ePrint Archive*. `https://eprint.iacr.org/2007/338/`.

[4] Bert den Boer. A simple and key-economical unconditional authentication scheme. *Journal of Computer Security*, 2(1):65–71, 1993.

[5] Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the Gbit/second rates. In *Fast Software Encryption*, pages 172–189. Springer, 1997.

[6] Thomas Johansson, Gregory Kabatianskii, and Ben Smeets. On the relation between a-codes and codes correcting independent errors. In *Advances in CryptologyEUROCRYPT93*, pages 1–11. Springer, 1994.

[7] Ted Krovetz. Fast cryptography. `http://www.fastcrypto.org/vmac/`.

[8] Ted Krovetz. Message authentication on 64-bit architectures. In *Selected Areas in Cryptography*, pages 327–341. Springer, 2007.

[9] Ted Krovetz. Patent-free authenticated-encryption as fast as OCB. In *Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications*, pages 459–461. Springer, 2007.

[10] David McGrew and John Viega. The Galois/counter mode of operation (GCM). *Submission to NIST*, 2004. `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf`.

[11] Victor Shoup. On fast and provably secure message authentication based on universal hashing. In *Advances in CryptologyCRYPTO96*, pages 313–328. Springer, 1996.

[12] Richard Taylor. An integrity check value algorithm for stream ciphers. In *Advances in CryptologyCrypto93*, pages 40–48. Springer, 1994.

[13] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

[14] Shmuel Winograd. A new algorithm for inner product. *Computers, IEEE Transactions on*, 100(7):693–694, 1968.

Google LLC. 1600 Amphitheatre Parkway Mountain View, CA 94043 USA
E-mail address, Ed Knapp: `edknapp@google.com`