Authenticated Encryption with SPECK

Chase Manny

University of Maryland, College Park

cmanny@terpmail.umd.edu

24 October 2013

Abstract

In this paper, we provide performance measures for software implementations of the NSA-designed Speck128 block cipher together with various existing authenticated encryption modes. We investigated Speck128 using GCM, CCM, EAX, OCB3, COPA, and PAEAD-1, and we briefly discuss performance advantages and disadvantages of each mode. Our results indicate that Speck128 is capable of performing extremely fast authenticated encryption, as fast as 3.4 cycles/byte on a modern x86-based 64-bit processor.

This work was done by the author as a summer intern at the National Security Agency in the Gifted and Talented STEM Program (formerly called the High School Gifted and Talented Program). The author would like to thank Dr. Louis Wingers, one of the designers of Speck, for his mentorship.

Authenticated Encryption with SPECK

Chase Manny

University of Maryland, College Park

cmanny@terpmail.umd.edu

24 October 2013

1. Introduction

Standard block cipher algorithms encrypt data and thereby make it private, but by themselves do nothing to ensure that data hasn't been tampered with. In order to provide data integrity, a standard approach is to use an *authenticated encryption* (AE) mode together with an established block cipher. The goal of this paper is to consider the performance of existing AE modes, when paired with the recently proposed Speck block cipher algorithm [1].

AE modes typically also provide the user the option of not encrypting portions of the data (the "associated data"), but still authenticating that data. Such modes provide for *authenticated encryption with associated data* (AEAD). All of the modes we consider in this paper are AEAD modes.

One of the most widely used authenticated encryption modes is GCM, and it has reasonably good performance (at least in relatively unconstrained environments) and is both an ISO and NIST standard. Moreover, many recent high-end processors by Intel and AMD now provide built-in support for GCM via the carryless multiply instruction. Together with the AES-NI instructions, fast authenticated encryption (with associated data) is supported on the majority of desktop processors by Intel and AMD.* Nevertheless, many processors in current use lack fast AES and GCM support and existing legacy software may not take advantage of the hardware support even if it is available. In such cases, a

*Some future ARM processors will also have built-in support for AES.

fast block cipher like SPECK128 [1], combined with GCM or some other authenticated encryption mode, might be desirable. Moreover, highly constrained lightweight applications (e.g., RFID, sensor networks) can benefit from a fast lightweight authenticated encryption mode combined with a fast lightweight block cipher like SPECK.

There is ongoing interest in developing authenticated encryption methods which improve upon the AES + GCM combination in terms of either security, performance in some application space, or robustness.* For applications on highly constrained devices, particularly hardware devices, there have been efforts to design lightweight hybrid algorithms which simultaneously perform both encryption and authentication. Two examples of this type are HUMMINGBIRD-2 [5] and, more recently, ALE [4]. It is difficult to design such hybrid algorithms. Many of them, including HUMMINGBIRD (the precursor of HUMMINGBIRD-2) and ALE, have been cryptographically broken (see [12], [7]). Moreover, in many lightweight applications, a block cipher might be used to authenticate devices, not data, and there is a fairly large overhead which may not be eliminated when using hybrid designs. Perhaps a better approach would be to design a lightweight mode which could *optionally* be used with a lightweight block cipher when data authentication is required.

In the remainder of this paper, we present performance data for SPECK128 + M, where M is one of the authenticated encryption modes GCM, CCM, EAX, OCB3, COPA, or PAEAD-1. We chose to consider modes that were NIST and ISO standards,† but also included COPA and PAEAD-1, which we found interesting because of their simplicity, performance, unpatented status and/or suitability for use in lightweight applications.

2. THE SPECK BLOCK CIPHER

SPECK [1] is a family of block ciphers designed by the National Security Agency for use in highly constrained environments. The block ciphers come in a range of block and key sizes and can be used on a variety of hardware and software

*See, for example, recent research from the DIAC workshops and the ongoing CAESAR competition, competitions.cr.yp.to/caesar.html.

†Technically, OCB2 is an ISO standard, not OCB3.

based platforms. For the purposes of this paper, we only consider Speck128/128, the Speck block cipher with a 128-bit block and key size. This is a natural choice since the authentication modes that we consider are typically specified for use with 128-bit block ciphers.

The round function for Speck128 is the key-dependent map $R_k\colon GF(2)^{64} \times GF(2)^{64} \to GF(2)^{64} \times GF(2)^{64}$ defined by

$$R_k(x, y) = ((S^{-8}(x) + y) \oplus k, S^3(y) \oplus (S^{-8}(x) + y) \oplus k),$$

where S^{-8} corresponds to a right rotation by 8 bits and S^3 a left rotation by 3 bits, both on 64-bit words. Speck128/128 steps 32 times and uses 32 64-bit round keys derived using the Speck128 key schedule. The key schedule is also very simple but isn't necessary to understand for the purposes of this paper. Eight parallel encryptions under Speck128/128 can be performed using the 128-bit SSE registers on an Intel Xeon E5640 processor (see Appendix), and a rate of 2.6 cycles/byte can be achieved.

The high speed of Speck makes it a good choice when fast encryption is necessary and high-speed AES implementations are either not available or not desired. For example, in an application where a high-end server needs to send encrypted data to many low-end, highly constrained devices (e.g., RFID, sensor networks), AES might not be the best solution if there are strict size or power requirements. Since Speck was designed for such environments it might be preferred on such devices and would then be required on any back end server communicating with them.

3. Attributes of Authenticated Encryption Modes

When deciding on an authenticated encryption mode, there are various attributes to consider. An AE mode can be either *on-line* or *off-line*, based on whether or not encryption can be performed without knowing the final bit length of the data to be encrypted. Most, but not all, AE modes require a *nonce*, i.e., a variable that must be changed for each message encrypted with a given key; nonce reuse can lead to security issues. Existing AE modes are either *one-pass* or *two-pass*, depending on how many calls of the block cipher are required to perform authenticated encryption. If the encryption of a data block can be

performed simultaneously with the block before it, then an AE mode is said to be *parallelizable*. Parallelizable modes can offer greatly increased throughput, if the resources are available that allow for encryptions to be performed in parallel.

Table 3.1 summarizes the characteristics of each of the six modes we analyzed.

	on-line	one pass	associated data	nonce free	parallelizable	patent free
GCM	✔		✔		✔	✔
CCM			✔			✔
EAX	✔		✔			✔
OCB3	✔	✔	✔		✔	
COPA	✔		✔	✔	✔	✔
PAEAD-1	✔	✔	✔		✔	✔

Figure 3.1: Authentication mode attributes

4. Authentication Modes

In this section, we investigate performance aspects of the authenticated encryption modes GCM, CCM, EAX, OCB3, COPA, and PAEAD-1 using Speck128 as the underlying block cipher. We provide a brief description of the mode and discuss possible performance advantages and disadvantages. We also speculate about the performance of the given mode in highly constrained environments, though we do not provide any direct evidence to support our speculations, i.e., we haven't actually implemented the mode on a constrained device. Our implementations were all done in C using GCC version 4.5.1 on an Intel Xeon E5640 processor.

4.1. GCM

GCM is a NIST and ISO standardized AEAD mode using counter mode for encryption and hashing over a Galois field for authentication [10]. It was published in 2004 and became an industry standard because of its flexibity of usage, high speed, lack of patent encumberance and standardization by NIST in 2007.

Software implementations of GCM can be adjusted to provide for fast authenticated encryption at the expense of large code size, or for smaller code size at the expense of speed. In hardware or software, however, especially in highly constrained environments, GCM consumes a lot of resources. Each block of data requires two block cipher calls for the encryption/authentication process, and so GCM is not among the fastest AE modes. However, its high degree of parallelism can make it competetive in environments where parallelism is possible.

In our implementation of GCM + Speck, the performance was very poor due to our inefficient GHASH computation. Indeed, our implementation ran in 28.4 cycles/byte. This was mostly due to our poor implementation and is not representative of GCM performance. In Figure 5.1, we provide a good faith estimate based on the performance data found in [9] which suggests that GCM + Speck could run at a rate of 5.4 cycles/byte for a very good (non AES-NI) implementation.

4.2. CCM

CCM [14] is a NIST and ISO standardized AEAD mode which uses counter mode for encryption and CBC-MAC mode for authentication. It was created as an unpatented alternative to the patented OCB mode that had been recently introduced by Rogaway, Bellare and Black [11]. CCM requires two block cipher calls per block of encrypted data and one per block of associated data. Due to its use of CBC-MAC mode, the authentication of the data is not parallelizable, although the counter mode used for encryption is. CCM is not on-line, as the length of the data must be known before the CBC-MAC computation can begin.

In our implementation of CCM + Speck, we acheived an authenticated encryption rate of 9.7 cycles/byte. CCM and the related EAX were among the slowest of the modes that we tested, mostly because they required two passes and did not allow for parallelization.

CCM may be a good choice in highly constrained hardware or software environments, if speed is not the main concern, and if the size of the message is known in advance. The fact that CCM is not parallelizable may be irrelevant for some applications. Both counter mode and CBC-MAC require very little overhead from block to block; in particular, they don't require complex data formatting or expensive mask computation. In [4], a small hardware implementation of CCM requires about 1040 GE. The smallest implementation of Speck in hardware requires 1280 GE (see [1]) so a rough estimate is that CCM + Speck would require about 2320 GE. Finally, despite the shortcomings of CCM, it is free to use and is a NIST standard.

4.3. EAX

EAX is an ISO standardized AEAD mode using counter mode for encryption and a variation of the CMAC mode, called OMAC, for authentication [2]. It was designed to be a more flexible alternative to the popular CCM mode. Through its implementation of counter and OMAC modes, it retains many of the properties of CCM while fixing some of its problems (e.g., EAX is on-line). This comes at the expense of further complication to the overall design. Like CCM, EAX requires two block cipher calls per block of encrypted data and one per block of associated data, giving it a very similar rate of authenticated encryption.

In our EAX implementation using Speck, we found it had performance very similar to CCM, at around 9.7 cycles/byte. If code size or area are important, EAX is probably not as well-suited as CCM for lightweight hardware or software environments due to its greater overhead from the addition of a tweak and message padding. However, by using OMAC instead of CMAC, EAX makes it possible to do on-line encryption and to pre-process constant associated data, and this makes it preferable to CCM in unconstrained environments. Like CCM, it is free of licensing restrictions.

4.4. OCB3

OCB3 [8] is an improvement to the ISO standardized OCB2 authenticated encryption mode. OCB3 provides for encryption and authentication with a single call to the block cipher, nearly doubling the average rate of encryption of two-pass modes like CCM or EAX. OCB3 is also fully parallelizable, giving it even greater performance benefits.

Our implementation of OCB3 + Speck was able to achieve a very fast authenticated encryption rate of 3.5 cycles/byte, much higher than for any two-pass mode. Krovetz and Rogaway [9] state that an optimized implementation of OCB3 may have an overhead as low as 0.3 cycles/byte which would give the combination of OCB3 + Speck a rate of around 2.9 cycles/byte, improving our results. However, OCB3 has some drawbacks. In highly constrained environments, OCB3 is likely to use significant memory or code on software platforms, and to require a significant amount of area in hardware. The most significant issue with OCB3 is not with the mode itself, but with the fact that it is encumbered by licensing restrictions.* This appears to be the primary reason for its slow adoption.

4.5. COPA

COPA [3] is a recent AEAD mode introduced at the *Summer Design and Security of Cryptographic Functions, Algorithms, and Devices Workshop* in Albena, Bulgaria.† It is advertised as a nonce-free, parallelizable AE mode that requires two block cipher calls per block of encrypted data and one per block of associated data. Designed to be flexible, it allows for pre-processing of constant associated data and for on-line encryption. While COPA is not a standardized mode of authenticated encryption, we found it interesting because of its lack of a nonce and simple structure.

We could not find a formal paper describing the complete details of COPA, and so our implementation of COPA + Speck is based on what we could deduce

*Rogaway has recently relaxed, but not entirely eliminated, licensing restrictions associated with OCB3.

†See www.cosic.esat.kuleuven.be/summer_school_albena/

about COPA from the diagram presented at the summer program. This implementation was able to achieve a relatively high rate of 5.9 cycles/byte, putting it between CCM/EAX and the one-pass mode OCB3 (and PAE, described below). It could potentially be suitable for high performance, lightweight hardware and software use, depending on how efficiently the masks can be generated.

4.6. PAE

PAE [13] is a relatively new family of one-pass AEAD modes. PAEAD-1 (the variant of PAE used for our timing data) is an AEAD mode that aims to achieve performance similar to OCB3 while using less memory and avoiding OCB3's licensing restrictions. It is parallelizable, on-line, and able to achieve encryption and authentication with only one block cipher call per block of data. The PAE family also provides the user flexiblity in the inclusion of associated data and allows memory to be optimized either for encryption or for decryption. PAEAD-1 is the variant optimized for encryption; it requires only the block cipher encryption algorithm to achieve authentication, but necessitates encryption and decryption for verification.

In our SPECK implementation of PAEAD-1, we were able to achieve very fast authenticated encryption at 3.4 cycles/byte, which was the best of the modes we tested. PAEAD-1 has the advantage that it can compute its masks on the fly using an LFSR, rather than relying on stored values computed during pre-processing. This potentially makes PAEAD-1 attractive for lightweight applications.

5. PERFORMANCE RESULTS

Table 5.1 shows timing data for implementations of the AE modes we considered, using SPECK128 as the underlying block cipher. To obtain the numbers in the table, we applied the algorithm in question to a long (65536 byte) message which had no associated data*. All timing data was obtained using an Intel Xeon E5640 64-bit processor, and our code, written in C, is not fully optimized,

*Inclusion of associated data will tend to improve timing estimates

leaving some room for improvement. However, these numbers are an accurate estimate of the performance of each combination.

mode	GCM	CCM	EAX	OCB3	COPA	PAEAD-1
rate	(5.4)	9.7	9.7	3.5	5.9	3.4

Figure 5.1: Encryption/authentication rate in cycles/byte for the combination SPECK128 + mode. The GCM figure is an estimate.

There are a variety of considerations that go into the choice of an authenticated encryption mode, and there is no single mode of the six we considered that we would claim is the best for all situations. We conclude with a summary of the strengths and weaknesses of each.

Although it has a larger code size and worse performance than some other modes tested, GCM benefits from being both NIST and ISO standardized, and from the fact that it's widely deployed. EAX is a clear improvement to the older CCM mode in high performance environments, but it is not NIST standardized, and it is likely to require greater overhead and achieve poorer performance in constrained environments. The simplicity and low overhead of CCM could make it effective in highly constrained environments, but it requires two block cipher calls per block, and the message length must be known before the CBC-MAC can begin. While OCB3 generally has very high performance, it requires relatively large code size and computational overhead for the creation of the masks, and this can affect its suitability for constrained environments. Additionally, its licensing restrictions have limited its adoption. COPA is interesting because it eliminates the need for a nonce, and because it performs quite well for a two-pass mode. However, it is not yet fully specified in the literature, and it is slow when compared to the one-pass modes tested.

All the AE modes we analyzed provide authenticated encryption with performance adequate for most requirements. We were particularly interested, however, in finding an AE mode that, combined with SPECK, could potentially achieve high performance in constrained environments. Of the modes we analyzed, the PAE family seems the most promising for a practical implementation in a lightweight setting. These modes require only one block cipher call per

block, use a relatively small amount of memory, and have no licensing restrictions. They are also flexible, allowing for the algorithm to be tuned to the application. Therefore, the combination SPECK + PAE seems to offer superior performance for lightweight applications. Because our current results remain somewhat speculative, we encourage further research in this area.

REFERENCES

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, *The* SIMON *and* SPECK *Families of Lightweight Block Ciphers*, Cryptology ePrint Archive, Report 2013/404, 2013. 1, 2, 6

[2] M. Bellare, P. Rogaway, D. Wagner, *The EAX Mode of Operation*, Fast Software Encryption '04, Lecture Notes in Computer Science, 2004. 6

[3] A. Bogdanov, *COPA: Parallelizable single-pass nonce-free AE*, Summer Design and Security of Cryptographic Functions, Algorithms, and Devices, Authenticated Encryption, 2013. 7

[4] A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser, *ALE: AES-based lightweight authenticated encryption*, in FSE'13, to appear, 2013. 2, 6

[5] D. Engels, M. Saarinen, P. Schweitzer, and E. Smith, *The Hummingbird-2 Lightweight Authenticated Encryption Algorithm*, Cryptology ePrint Archive, Report 2011/126, 2011. 2

[6] E. Käsper and P. Schwabe, *Faster and timing-attack resistant AES-GCM*, CHES 2009, LNCS 5757, Springer, 2009.

[7] D. Khovratovich and C. Rechberger, *The LOCAL attack: Cryptanalysis of the authenticated encryption scheme ALE*, Cryptology ePrint Archive, Report 2013/357, 2013. 2

[8] T. Krovetz, P. Rogaway, *The OCB Authenticated-Encryption Algorithm*, Internet Research Task Force, draft-irtf-cfrg-ocb-03, 2013. 7

[9] T. Krovetz, P. Rogaway, *The Software Performance of Authenticated-Encryption Modes*, Available at www.cs.ucdavis.edu/~rogaway/papers/index.html. 5, 7

[10] D. McGrew, J. Viega, *The Galois/Counter Mode of Operation (GCM)*, Available at http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html. 5

[11] P. Rogaway, M. Bellare, J. Black, *OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption*, ACM Transactions on Information and System Security (TISSEC), vol. 6, no. 3, 2003. 5

[12] M. Saarinen, *Cryptanalysis of Hummingbird-1*, Fast Software Encryption, Lecture Notes in Computer Science, Volume 6733, pp. 328-341, 2011. 2

[13] P. Sarkar, *Pseudo-Random Functions and Parallelizable Modes of Operations of a Block Cipher*, IEEE Transactions on Information Theory, Volume 56, Issue 8, 2010. 8

[14] D. Whiting, R. Housley, N. Ferguson, *Counter with CBC-MAC (CCM)*, Available at http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html. 5

6. Appendix

Below is the source code used for the parallel implementations of Speck128. This code performs eight parallel Speck128 encryptions. The 128-bit crypto-variable K is pre-expanded to generate 32, 64-bit round keys $k_0, \ldots, k_{31}$. These are stored in 32, 128-bit SSE registers $k[0] = (k_0, k_0)$, $k[1] = (k_1, k_1)$, ..., $k[31] = (k_{31}, k_{31})$. If the k_i are of type `ull`(unsigned long long) then, for example, $k[0]$ = `_mm_set_epi64x`(k_0, k_0). The array of *doubled* round keys, $k[i]$, of type `__m128i` is passed to the encryption function. The eight 128-bit plaintext and ciphertext blocks are stored in the arrays `pt[ ]` and `ct[ ]` of type `ull`. The first block of plaintext is `(pt[1],pt[0])`, the next is `(pt[3],pt[2])`, etc., and similarly for the ciphertext.

The code was compiled using GCC version 4.5.1 with the `-O3` option for speed. The `-msse4` option is also required. The resulting encryption rate is 2.6 cycles/byte on an Intel Xeon E5640 processor. The `x86intrin.h` header file also needs to be included.

```
#define ull unsigned long long
#define XOR _mm_xor_si128
#define SHFL _mm_shuffle_epi8
#define SET _mm_set_epi64x
#define STR _mm_store_si128
#define ADD _mm_add_epi64
#define SWAP _mm_set_epi64x(0x080f0e0d0c0b0a09LL,0x0007060504030201LL)
#define LOW _mm_unpacklo_epi64
#define HIGH _mm_unpackhi_epi64
#define SR _mm_srli_epi64
#define SL _mm_slli_epi64

#define R(X,Y,k) (X[0]=SHFL(X[0],SWAP), X[1]=SHFL(X[1],SWAP), X[2]=SHFL(X[2],SWAP),\
          X[3]=SHFL(X[3],SWAP), X[0]=ADD(X[0],Y[0]), X[1]=ADD(X[1],Y[1]),\
          X[2]=ADD(X[2],Y[2]),  X[3]=ADD(X[3],Y[3]),\
          X[0]=XOR(X[0],k), X[1]=XOR(X[1],k), X[2]=XOR(X[2],k), X[3]=XOR(X[3],k),\
          Z[0]=Y[0], Z[1]=Y[1], Z[2]=Y[2], Z[3]=Y[3],\
          Z[0]=SL(Z[0],3), Z[1]=SL(Z[1],3), Z[2]=SL(Z[2],3), Z[3]=SL(Z[3],3),\
          Y[0]=SR(Y[0],61), Y[1]=SR(Y[1],61), Y[2]=SR(Y[2],61), Y[3]=SR(Y[3],61),\
          Y[0]=XOR(Y[0],Z[0]), Y[1]=XOR(Y[1],Z[1]), Y[2]=XOR(Y[2],Z[2]),\
          Y[3]=XOR(Y[3],Z[3]), Y[0]=XOR(X[0],Y[0]), Y[1]=XOR(X[1],Y[1]),\
          Y[2]=XOR(X[2],Y[2]), Y[3]=XOR(X[3],Y[3]))

void Encrypt(ull pt[],ull ct[],__m128i k[])
{
  int i;
  __m128i X[4],Y[4],Z[4];

  X[0]=SET(pt[3],pt[1]);   Y[0]=SET(pt[2],pt[0]);
  X[1]=SET(pt[7],pt[5]);   Y[1]=SET(pt[6],pt[4]);
  X[2]=SET(pt[11],pt[9]);  Y[2]=SET(pt[10],pt[8]);
  X[3]=SET(pt[15],pt[13]); Y[3]=SET(pt[14],pt[12]);

  for(i=0;i<32;i++) R(X,Y,k[i]);

  STR((__m128i *)ct,LOW(Y[0],X[0]));
  STR((__m128i *)(ct+2),HIGH(Y[0],X[0]));

  STR((__m128i *)(ct+4),LOW(Y[1],X[1]));
  STR((__m128i *)(ct+6),HIGH(Y[1],X[1]));

  STR((__m128i *)(ct+8),LOW(Y[2],X[2]));
  STR((__m128i *)(ct+10),HIGH(Y[2],X[2]));

  STR((__m128i *)(ct+12),LOW(Y[3],X[3]));
  STR((__m128i *)(ct+14),HIGH(Y[3],X[3]));
}
```