White Box Implementations of Dedicated Ciphers on the ARM NEON Architecture

Ricardo Dahab[1], **Julio López**[1], **Félix Carvalho Rodrigues**[1], **Hayato Fujii**[1], **Giuliano Sider**[1], **Ana Clara Serpa**[1]

[1]Institute of Computing – University of Campinas (Unicamp)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brazil

{rdahab,jlopez,felix.rodrigues,hayato.fujii}@ic.unicamp.br,
{ra146271,ra165880}@students.ic.unicamp.br

Abstract. *Modern computer environments such as smartphones are increasingly susceptible to malware, a cause of concern regarding their trustworthiness. Nevertheless, cryptographic algorithms are still necessary in such environments, which raises the need for a secure white-box design and implementation of such algorithms. Dedicated white box block ciphers are encryption algorithms designed to operate in untrusted environments. In this paper we present a fast vectorized implementation of two families of dedicated white-box block ciphers, SPACE [Bogdanov and Isobe 2015] and WEM [Cho et al. 2017], for the ARM Cortex-53 processor, using the NEON vector instruction set. To the best of our knowledge, the implementations outlined in this paper currently have the best reported performance for these dedicated ciphers in the white box context.*

1. Introduction

In the realm of cryptography, the traditional *black box threat model* assumes that the end points of a communication channel are secure, and only the channel itself is vulnerable to attackers. With the increased reliance on mobile computer systems such as smartphones, this model no longer captures all viable threats encountered in practice. In a *white box threat model*, an attacker is assumed to have complete access to the full implementation and the execution environment of a cryptographic algorithm.

White box cryptography concerns the design and secure implementation of cryptographic algorithms running in untrusted environments. The concept was introduced in 2003 [Chow et al. 2003], exemplified in a software implementation of the AES cipher which attempts to obfuscate a secret key. This proposed implementation was shown to be susceptible to practical attacks [Billet et al. 2005], and every new proposed white box implementation of the AES [Karroumi 2011] was also successfully attacked [Lepoint et al. 2014, Bos et al. 2017].

These attacks have prompted efforts to design new symmetric ciphers, which take into account white-box-model threats from the start. Most of current proposals focus not only on ensuring that discovering the protected secret key is infeasible, but also on mitigating possible *code lifting* attacks, in which the attacker extracts the cryptographic implementation itself, to use it as an effectively larger key, duplicating the functionality of the cipher.

Bogdanov et al. [Bogdanov and Isobe 2015] introduced the SPACE family of dedicated block ciphers, with a focus on using proven standard cryptographic primitives to

guarantee their security. In another proposal [Cho et al. 2017], the WEM family of ciphers is presented, based on an Even-Mansour scheme, where the secret key layers are replaced by secret incompressible S-boxes. Existing performance measures of these ciphers cannot be directly compared since different computer platforms were used. In addition, there is no ARM implementation of these ciphers in the public domain.

In this paper we present results regarding the performance of the SPACE and the WEM family of ciphers, particularly on the software implementation of such ciphers using the ARMv8 platform and its cryptographic instructions.

1.1. Related Work and Organization

There have been several proposals of new white box block ciphers other than SPACE and WEM, but few provide reliable performance comparisons between one another.

The first dedicated white box cipher was proposed in 2014 [Biryukov et al. 2014]. Its design was based on an *ASASA* structure, and its security relied on the hardness of decomposing its layers. Subsequent cryptanalysis revealed possible vulnerabilities in this structure [Biryukov and Khovratovich 2015]. The SPNbox [Bogdanov et al. 2016] family of block ciphers is presented as an evolution of the SPACE design, replacing its Feistel structure by a Substitution-Permutation network with incompressible S-boxes. The WhiteBlock block cipher [Fouque et al. 2016] is similar to the WEM design as it uses a standard block cipher as a public permutation between each S-box layer, differing mostly on how the S-box layer is constructed. Its main contribution is in providing a more rigorous proof of its security goals when compared to other dedicated ciphers.

In the context of ARMv8 cryptographic implementations, most works focus on standard block ciphers and modes of operation, such as the AES and the GCM [Gouvêa and López 2015].

In Section 2, some preliminary concepts such as incompressibility are introduced. Section 3.1 gives an overview of the SPACE family of block ciphers, while Section 3.2 describes the WEM family of block ciphers. Section 4 details the implementation aspects, while Section 5 presents the performance comparisons between the implemented ciphers.

2. Preliminaries

A symmetric encryption scheme is a tuple $\mathcal{E} = (\mathcal{K}, \mathcal{M}, \mathcal{C}, G, E, D)$, where $\mathcal{K}, \mathcal{M}$ and $\mathcal{C}$ are the set of possible keys, plaintexts (messages) and ciphertexts, respectively, while G, E and D are the functions for key generation, encryption and decryption, respectively. For any $k \in \mathcal{K}, m \in \mathcal{I}$, $D(E(m,k),k)$ must be equal to m. Note that we alternatively use $E(m)$ or $E_k(m)$ for denoting encryption (similarly for decryption), when the context is clear. A white box compiler $C_{\mathcal{E}}$, takes a symmetric encryption scheme $\mathcal{E}$, a key $k \in K$, a nonce r (optionally) and returns a compiled white box program $C_{\mathcal{E}}(k,r) = [E_k]$ (respectively $[D_k]$ for the decryption).

Any secure compiler must strive for the main security goal of *unbreakability*: given a program $[E_k]$, the key k embedded in $[E_k]$ must not be discovered efficiently by any adversary. Once such goal is obtained, additional security goals may be addressed. Among the most important, is the notion of *incompressibility*, which aims to mitigate *code lifting* attacks. In [Bogdanov and Isobe 2015], the authors use the term *space hardness* to refer to such goal. They define weak and strong space hardness. In the weak space

hardness security notion, an adversary must not be able to encrypt or decrypt messages at will with less than M bits of the compiled cipher's code, while in the strong security notion, the adversary must not encrypt or decrypt any messages even when having access to M bits of the code. All ciphers studied in this work use the concept of weak space hardness as their main incompressibility guarantee; we present a formal definition of this notion below.

Definition 1 (Weak $(M, Z)-$space hardness [Bogdanov and Isobe 2015].)**.** *Given an encryption scheme $\mathcal{E}$, a white box compiler $C_{\mathcal{E}}$ is weakly $(M, Z)-$space hard if it is infeasible for an adversary $\mathcal{A}$ to encrypt (or decrypt) a randomly drawn plaintext (or ciphertext) with probability greater than 2^{-Z}, given access to M bits from $[E_k]$ (or $[D_k]$).*

Note that the dedicated white box ciphers contemplated in this paper (and all current proposals on the literature) focus solely on these two security goals, and thus do not achieve other possible notions such as one-wayness and traceability.

3. Dedicated Ciphers

In this section, we describe two families of dedicated white-box block ciphers. We chose to implement the SPACE and WEM family of ciphers due to their clear difference in approach. While the SPACE design uses a single lookup table in a Feistel structure, the WEM proposes a design with numerous smaller S-boxes, each implemented as a separate lookup table. Other possible ciphers, such as the WhiteBlock, have a similar design to the WEM family, where S-box layers alternate with permutation layers.

3.1. SPACE

The SPACE family of block ciphers was one of the first proposed dedicated ciphers [Bogdanov and Isobe 2015]. It adopts a conservative design, ensuring that its key recovery and extraction security relies on the security of an underlying standard block cipher in the black box model. It uses a Feistel network construction, where a round function is applied to a portion of n_{in} bits of the full 128-bit input. The value of n_{in} determines the different instantiations of the cipher. The designers recommend instantiations with n_{in} equal to $8, 16, 24$ or 32.

In a SPACE-n_{in} encryption, the state of the cipher is updated by applying the Feistel round function to the first n_{in}-bit line of the state, and XORing the $(n - n_{in})$-bit output with the other lines, where n is the block size. The result is then concatenated with the first line. This is repeated for a number R of rounds. Let $l = \frac{n}{n_{in}}$ and $n_{out} = n - n_{in}$. The transformation for round $r = 0, 1, \ldots, R$ is given by the expression $X^{r+1} = (F_{n_{in}}(x_0^r) \oplus r \oplus (x_1^r||x_2^r|| \ldots ||x_{l-1}^r))||x_0^r$, where $||$ is the concatenation operator and $F_{n_{in}} : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$ is the Feistel function computing $F_{n_{in}}(x) = msb_{n_{out}}(E_k(0||x))$. Here, $msb_i(x)$ denotes the i most significant bits of x and E_k represents a standard block cipher (e.g. AES-128).

Figure 1a shows the Feistel round structure, while Figure 1b details an instantiation using AES-128 as the round function. The dashed line encloses the computation of the round function, which is performed by a lookup table in the white box environment.

In the white box implementation of the cipher for a fixed key k, the round cipher function is implemented as a $\{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$ lookup table. As a result, the key

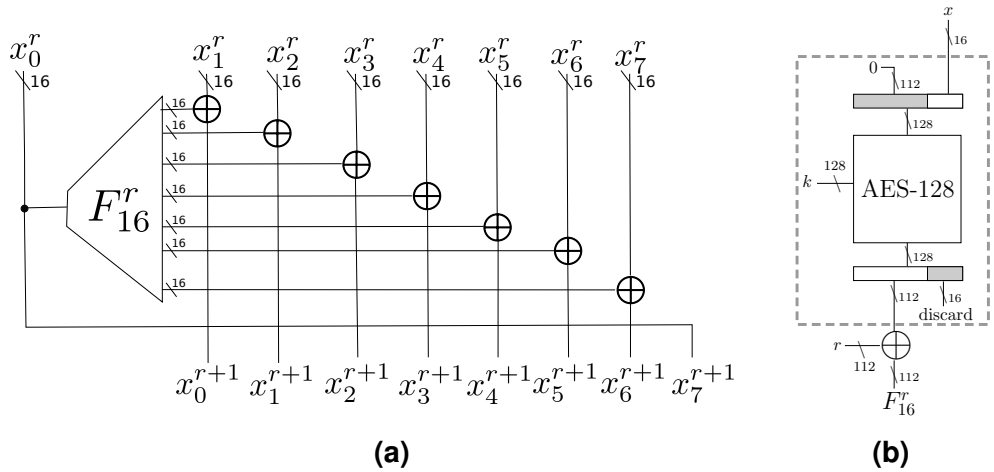

Figure 1. In (a), the round r transformation for SPACE-16, operating on $(x_0^r, \ldots, x_7^r)$. In (b), the round r Feistel function for SPACE-16, F_{16}^r, is shown instantiated with AES-128 as the block cipher E_k.

k becomes enmeshed with the pseudo-random output of the block cipher into a lookup table of size $2^{n_{in}} \cdot n_{out}$ bits. As a consequence, the table size varies greatly depending on n_{in}. For SPACE-8, a total of 2^8 entries of 14 bytes are needed, while for SPACE-32 the lookup table requires $2^{32} \times 12$ bytes.

The recommended number of rounds differs depending on n_{in}. For SPACE-8, the authors recommend at least 300 rounds, while the recommended number for the other sizes is at least 128 rounds. This elevated number of rounds is necessary to ensure that linear and differential cryptanalysis are infeasible, as well as compression attacks. The difficulty of mounting a compression attack is related to the space-hardness of the cipher. With access to a portion M of the table entries, an attacker is only able to successfully encrypt a plaintext with probability $(\frac{M}{2^{n_{in}}})^R$. This incompressibility makes code lifting more difficult for an attacker in the bounded-retrieval attack model [Bogdanov et al. 2016], particularly for larger instances of SPACE, such as SPACE-32, where the larger lookup table makes code lifting much harder.

3.2. WEM

The White Box Even-Mansour (WEM) [Cho et al. 2017] family of specialized white box block ciphers is based on the well known iterated Even-Mansour construction. The encryption function is defined as $EM_{k_0,k_1}(x) = k_1 \oplus P(k_0 \oplus x)$, for $x \in \{0,1\}^n$, where k_0 and k_1 are independent n-bit secret keys, P is an n-to-n permutation and x is an n-bit plaintext. A known-plaintext attack on this construction has $\Omega(2/n)$ workload [Dunkelman et al. 2012], defining the security level of the EM cipher to be $2^{2/n}$. To improve it, iterations (or rounds) of the construction are proposed. For a variant of this cipher using the same key in all rounds (i. e., $k_0 = k_1 = \ldots = k_t$), no attack faster than $2^n/n$ is known, even for a 2-round EM (2EM).

The WEM algorithm replaces key additions using incompressible, key dependent S-boxes. Several parameters are used to make the algorithm more flexible: n denotes the block size of the cipher, m denotes the size of the incompressible S-box, requiring that $m|n$. For the iterated white-box EM constructions we use the following notation: t is the

number of rounds, E is an underlying block cipher (e. g., AES) and d is the number of times that the underlying block cipher is applied using a key composed of all-zero bits. Figure 2a illustrates WEM when $n = 128$, $m = 16$, $t = 2$, $E =$ AES-128, $d = 5$. The main secret key k is embedded into the S-boxes. To generate them, a secure environment must be used to execute two steps: a long sequence of random bits is generated depending on the secret key k; then, the sequence is used as a way to provide random numbers to a shuffler (such as the Fisher Yates algorithm [Knuth 1998]) which permutes a sequence $(0, 1, \ldots, 2^m - 1)$ to form an m-to-m S-box. Figure 2b illustrates the shuffling method.

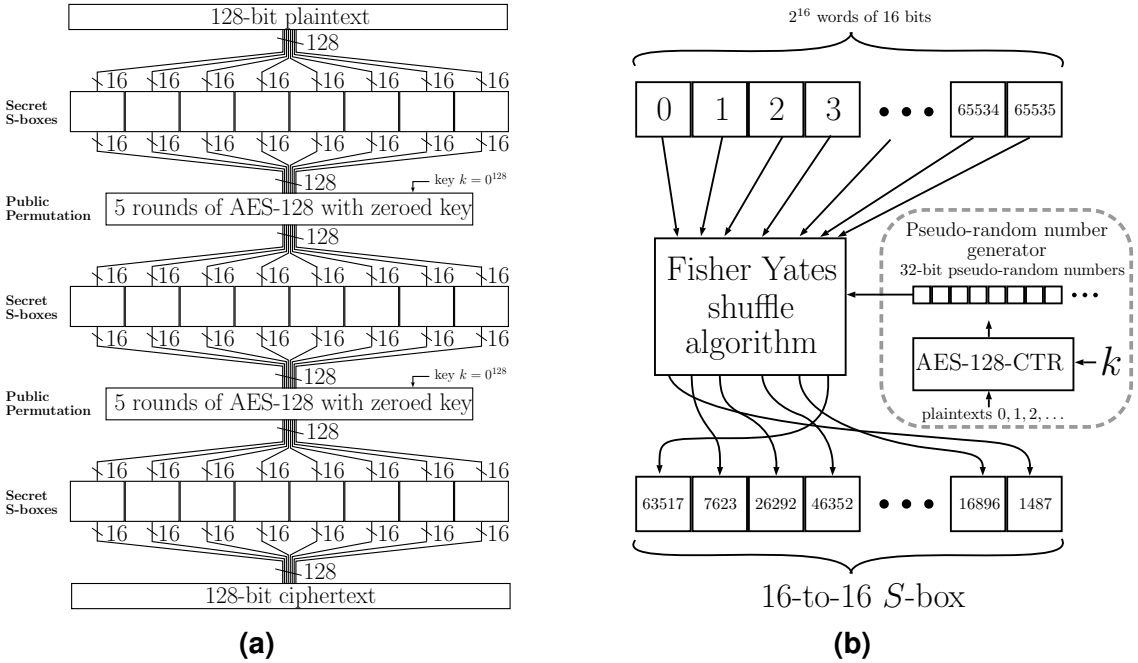

Figure 2. The full instance WEM(128, 16, 2, AES-128, 5) is shown in (a), while the 16-to-16 S-box generation is shown in (b).

The performance and the security of this cipher in the black box model are based in its EM construction. Literature on the iterated EM construction suggests that, in the black box model, the security level is close to 2^n [Chen et al. 2014]. In the white-box model, the workload to extract the key to generate the S-boxes is reduced to breaking the CTR mode of operation. Even if an adversary extracts all the S-boxes and reverses the shuffling, only a few pairs of plaintext and ciphertext will be known, which are not useful for recovering the original secret key.

The authors of WEM recommend using the WEM16 cipher, which is WEM using parameters $n = 128$, $m = 16$, $t = 12$, $E =$AES-128 and $d = 5$. This instance requires 104 S-boxes, each having 2^{16} entries of 16 bits. As an alternative smaller sized instance, the WEM8 cipher using the parameters $(128, 8, 12,$AES-$128, 5)$ can also be utilized.

4. Implementation

In this paper, we focus on the popular 64-bit ARMv8-A architecture, which is implemented by the Cortex-A series of processors. More specifically, our implementation requires processors with support for ARMv8 NEON instructions, as well as the optional ARMv8 Cryptographic Extension. This extension adds new instructions that accelerate

the Advanced Encryption Standard (AES) encryption and decryption, among other cryptographic primitives.

For the SPACE family of dedicated ciphers, we implemented SPACE$-(8, 300)$ and SPACE$-(16, 128)$ in the white box context, where the first parameter indicates the input size of the Feistel round function, and the second is the number of rounds. The round function was transformed into a lookup table computing a full AES-128 encryption of 8 and 16 bits for SPACE-8 and SPACE-16, respectively.

For the WEM family of ciphers, we implemented two versions, WEM-16 and WEM-8, only differing in their S-box size: while WEM-8 has sixteen 8-to-8 S-boxes, WEM-16 has eight 16-to-16 S-boxes. Both ciphers use a block cipher of 128 bits and 12 outer rounds. We used 5 rounds of the AES-128 block cipher with zero key as the public permutation applied in each round, for a total of $12 \times 5 = 60$ AES-128 rounds for an encryption operation. We implemented the AES-128 standard cipher using the ARMv8 Cryptographic Extension instructions, and thus operated as much as possible in the NEON registers environment.

For comparison purposes, we also implemented an unoptimized white box version of the AES-128 cipher (WB-AES), based on its original proposal [Chow et al. 2003] without external encodings, as well as our own black box ARMv8 NEON implementation of the AES-128 block cipher (named BB-AES).

In Table 1, we compare the minimum theoretical size of each implementation and its code size in our implementation. Note that these sizes refer only to the encryption functionality of each cipher. All dedicated ciphers except for the WEM-8 cipher have their (weak) space hardness presented in their respective papers. For the WEM-8 cipher, the space hardness was calculated using the analysis present in Section 4.2 of Cho et al.'s paper.

Table 1. Space hardness and total table size of implemented ciphers.

Cipher	Rounds	Total tables size (bytes)	Space Hard. (bytes)	Code size
SPACE-8	300	$2^8 \times 15 = 3840$	$(960, 128)$-hard	4164
SPACE-16	128	$2^{16} \times 14 = 917504$	$(229376, 128)$-hard	1048644
WEM-8	12	$2^8 \times 16 \times 13 = 53248$	$(241, 117)$-hard	53912
WEM-16	12	$2^{16} \times 8 \times 13 \times 2 = 13631488$	$(29737, 112)$-hard	13631828
WB-AES	10	770048	—	832304
BB-AES	10	—	—	88

5. Results

In this section we present the performance measures of all implemented dedicated ciphers. Tests were executed on a machine with a Cortex-A53-based quad-core processor clocked at 1.15GHz. This CPU features 32KiB of L1 cache memory for instructions and 32KiB of L1 cache memory for data for each of its cores, plus 512KiB of L2 cache memory. This board is equipped with with 2GB of RAM. On the software side, the Linux kernel version 3.10.107 is used. The ciphers were implemented in C, using NEON intrinsics where applicable, and compiled with `gcc` version 5.4.0 with the `-O3` flag enabled.

For each white box cipher, we compiled 16 different instantiations and encrypted 256 blocks of size 128 bits. In Table 2, we present the average number of cycles

per byte (cpb) for our implementations and (where applicable) the measurements already present in the literature. For SPACE, the authors of the original measurements present in the last column used a Samsung Exynos 7420 CPU with a 2.1GHz Cortex-A57 core [Bogdanov et al. 2016] and, for WEM16, in a 32-bit Intel Broadway platform clocked at 2.4GHz [Cho et al. 2017].

Table 2. Number of table lookups and measured performance (cycles per byte) of the implemented dedicated ciphers.

Cipher	Table lookups	Average performance (cpb)	Original performance (cpb)
SPACE-8	300	248.29	409.57 [Bogdanov et al. 2016]
SPACE-16	128	138.66	377.51 [Bogdanov et al. 2016]
WEM-8	208	64.11	—
WEM-16	104	55.78	96.8 [Cho et al. 2017]
WB-AES	2976	15503.21	—
BB-AES	—	2.5	—

While the SPACE cipher has a much greater space hardness than the WEM family of ciphers, it comes at a great performance cost. For both SPACE-8 and SPACE-16, the performance is dominated by the number of table lookups executed, whereas for the WEM family of ciphers, the public permutation can be greatly optimized on specific hardware platforms. This indicates that the development of dedicated ciphers which are able to maintain a good space hardness guarantee, while relying less on lookup tables, might bridge the still significant gap between industry practice and the academic literature.

Although our results are not directly comparable to the ones reported in previous literature, the achieved results indicate that such dedicated ciphers are suitable for optimizations which explore NEON vector instructions. As possible research directions, a larger number of these dedicated ciphers should be efficiently implemented and compared, especially the SPNbox white box cipher, which reports better performance than both SPACE and WEM ciphers. Furthermore, none of these ciphers were compared when taking in consideration modes of operation, and thus some designs might reveal better integration and performance in this context. Finally, testing bigger sized ciphers such as SPACE-32 and a version of WEM with 32-to-32 S-boxes might lead to new optimization options.

6. Acknowledgments

This research was partially supported by Samsung Eletrônica da Amazônia Ldda., through the "White Box Cryptography" project, within the scope of the Informatics Law No. 8248/91.

References

Billet, O., Gilbert, H., and Ech-Chatbi, C. (2005). Cryptanalysis of a white box aes implementation. In Handschuh, H. and Hasan, M. A., editors, *Selected Areas in Cryptography*, pages 227–240, Berlin, Heidelberg. Springer Berlin Heidelberg.

Biryukov, A., Bouillaguet, C., and Khovratovich, D. (2014). Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key (extended abstract). In Sarkar, P. and Iwata, T., editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 63–84, Berlin, Heidelberg. Springer Berlin Heidelberg.

Biryukov, A. and Khovratovich, D. (2015). Decomposition attack on SASASASAS. Cryptology ePrint Archive, Report 2015/646. https://eprint.iacr.org/2015/646.

Bogdanov, A. and Isobe, T. (2015). White-box cryptography revisited: Space-hard ciphers. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1058–1069, New York, NY, USA. ACM.

Bogdanov, A., Isobe, T., and Tischhauser, E. (2016). Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In Cheon, J. H. and Takagi, T., editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 126–158, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bos, J. W., Hubain, C., Michiels, W., Mune, C., Gonzalez, E. S., and Teuwen, P. (2017). White-box cryptography: Don't forget about grey box attacks. Cryptology ePrint Archive, Report 2017/355. https://eprint.iacr.org/2017/355.

Chen, S., Lampe, R., Lee, J., Seurin, Y., and Steinberger, J. P. (2014). Minimizing the two-round even-mansour cipher. In *CRYPTO (1)*, volume 8616 of *Lecture Notes in Computer Science*, pages 39–56. Springer.

Cho, J., Choi, K. Y., Dinur, I., Dunkelman, O., Keller, N., Moon, D., and Veidberg, A. (2017). Wem: A new family of white-box block ciphers based on the even-mansour construction. In Handschuh, H., editor, *Topics in Cryptology – CT-RSA 2017*, pages 293–308, Cham. Springer International Publishing.

Chow, S., Eisen, P., Johnson, H., and Van Oorschot, P. C. (2003). White-box cryptography and an aes implementation. In Nyberg, K. and Heys, H., editors, *Selected Areas in Cryptography*, pages 250–270, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dunkelman, O., Keller, N., and Shamir, A. (2012). Minimalism in cryptography: The even-mansour scheme revisited. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer.

Fouque, P.-A., Karpman, P., Kirchner, P., and Minaud, B. (2016). Efficient and provable white-box primitives. In Cheon, J. H. and Takagi, T., editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 159–188, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gouvêa, C. P. L. and López, J. (2015). Implementing gcm on armv8. In Nyberg, K., editor, *Topics in Cryptology — CT-RSA 2015*, pages 167–180, Cham. Springer International Publishing.

Karroumi, M. (2011). Protecting white-box aes with dual ciphers. In Rhee, K.-H. and Nyang, D., editors, *Information Security and Cryptology - ICISC 2010*, pages 278–291, Berlin, Heidelberg. Springer Berlin Heidelberg.

Knuth, D. E. (1998). *The art of computer programming, Volume II: Seminumerical Algorithms, 3rd Edition*. Addison-Wesley.

Lepoint, T., Rivain, M., De Mulder, Y., Roelse, P., and Preneel, B. (2014). Two attacks on a white-box aes implementation. In Lange, T., Lauter, K., and Lisoněk, P., editors, *Selected Areas in Cryptography – SAC 2013*, pages 265–285, Berlin, Heidelberg. Springer Berlin Heidelberg.