PERFORMANCE IMPROVEMENT OF WHITE-BOX ALGORITHMS USING LIGHTWEIGHT COMPONENTS

Abstract. Protecting secret keys from malicious observers is a major problem for cryptographic algorithms in untrusted environments. White-box cryptography suggests hiding the key in the cipher code with an appropriate method such that extraction of the key becomes impossible in the white-box settings. The key is generally embedded into the confusion layer with suitable methods. One of them is using encoding techniques. Nevertheless, many encoding methods are vulnerable to algebraic attacks and side-channel analysis. Another is the space hardness concept, which creates large lookup tables that cannot be easily extracted from the device. In (M,Z)-space hard algorithms, the secret key is embedded in large tables created as a substitution box with a suitable block cipher. So the key extraction problem in the white-box settings turns into a key recovery problem in the black-box case. One of the main issues in (M,Z)-space hard algorithms is accelerating the running time of the black-box/white-box implementation. In this study, we aim to use the advantage of the efficiency of lightweight components to speed up the diffusion layer of white-box algorithms without decreasing the security size. Therefore, we compare the linear layer of NIST Lightweight Standardization candidates for efficiency and suitability to white-box settings in existing space hard ciphers. The performance results of the algorithms are compared with WARX and SPNbox-32. According to the results, using the lightweight components in the diffusion layer accelerates the performance of white-box algorithms by at least eight times.

1. Introduction

Products used in an untrusted environment are vulnerable to capturing encryption keys by a malicious observer, as the observer has the ability to gain access to the cryptographic algorithms and the encryption keys. From DRM similar products and cloud servers to endpoint users such as mobile phones, laptops, or lightweight devices require protection against third parties. White-box cryptography suggests software protection using an appropriate method to hide the key in the algorithm phases. The key is generally embedded into the confusion layer with suitable methods.

First white-box algorithms, white-box DES [9] in 2002 and white-box AES [10] in 2003 were proposed by Chow et al. The secret key was embedded into the Sbox by transforming the algorithm layers into lookup tables with internal and external encodings. The suggested encoding methods to prevent key extraction were broken by the algebraic attacks [3, 18]. Some other white-box AES variants were proposed [30, 20], but they were also broken [11, 24].

A dedicated white-box block cipher based on ASASA structure was proposed by Biryukov et al. [4]. The paper defines weak white-box security and proposes a memory-hard white-box block cipher against code lifting attacks. Unfortunately,

Key words and phrases. White-box Cryptography, Space-hard ciphers, Lightweight components, Efficiency.

key recovery attacks were applied in [16, 27] against the ASASA structure. Even if the structure was broken, the proposed methods inspired space-hard approaches for white-box algorithms.

After DES, AES, and ASASA white-box block cipher designs, a new dedicated algorithm called space-hard ciphers has been proposed by Bogdanov and Isobe [5, 6]. In the new structure, large lookup tables, constructed with a small block cipher, are used to embed secret keys. The constructed tables are used as a nonlinear layer in the algorithm. With this approach, key extraction in white-box settings becomes a key recovery problem in the black-box case. Against code lifting attacks, *(M,Z)-space hardness* is defined such that the algorithm provides Z-bit security until the size of the leakage from the code (table) is reached M bits.

Two main issues exist for (M,Z)-space hard white-box algorithms. The first one is updating the lookup tables after a particular leakage to provide security against code lifting attacks. When the leak limit is reached, the tables must be updated, either on the device or by remotely loading [21] from the server.

The other issue is accelerating the black-box and the white-box implementations. The encryption/decryption process of the space-hard ciphers is mainly based on Feistel [5, 15, 23, 22] and SPN [6, 21, 26] structures. When examining SPN-based white-box implementations, the most time-consuming part is the linear layer with the MDS matrix, which consists of matrix multiplication and modular reduction. Hence, speeding up the white-box implementations depend on the linear layer. Among the (M,Z)-space hard ciphers, SPNbox and Yoroi use the advantage of the SIMD to gain acceleration in the linear layer. At the same time, WARX improves the performance by decreasing the round number of the algorithm through a random MDS matrix in the linear layer.

Lightweight algorithms are resource constraint designs with heavyweight security and fewer computational requirements. In this study, we considered using those lightweight components in the linear layer of space-hard white-box structures to speed up the encryption/decryption process without reducing the security level. With this purpose, NIST Lightweight Standardization [17] candidates are discussed to find alternative approaches to MDS matrices for the linear layer in the white-box settings.

This study examined NIST Lightweight competition finalists and second round candidates to evaluate suitable designs for the (M,Z)-space hard algorithms. The lookup table was used as a substitution box in the nonlinear layer, while the linear component of the lightweight design was used as a linear layer. The tables were created using the WARX method for 16-bit word size algorithms and the SPNbox method for 32-bit word size algorithms. We suggested fixing the security size to $M \cdot 2^{-keysize}$ bits to more precisely calculate the round numbers of the algorithms based on the recommended security level. The running time of the white-box schemes was compared with the algorithms WARX and SPNbox-32. We observed lightweight components in white-box settings to be faster than WARX and SPNbox-32, with appropriate round numbers and security sizes.

The design approaches and algorithm specifications of NIST Lightweight Standardization candidates are discussed in Section 2. Security notions of the white-box settings and details of round number computation are stated in Section 3. Performance comparisons of the algorithms are given in Section 4.

2. White-box Implementations with the Lightweight Components

Ten finalists and 19 second round candidates of the NIST Lightweight competition are examined by compatibility of the white-box cryptography. Since the structure of lookup tables is based on 16-bit/32-bit word size, algorithms other than word size 16-bit or 32-bit are discarded. The diffusion of the white-box conversions is calculated with the strict avalanche criteria (SAC) to check the reliability of the algorithms.

According to SAC results in Section 3.3, the linear layers of Sparkle [1] from finalists, Spook-Shadow [2] and Saturnin [7] among the second round candidates provide required diffusion in the white-box settings. Saturnin [7] is an SPN-based lightweight algorithm. It uses MDS matrices in the linear layer. Sparkle and Shadow algorithms are permutation constructs with no key scheduling or key addition layer. Sparkle is based on the ARX design [13] and uses Feistel structure in the linear layer. Shadow with LS design has an additional diffusion layer.

The lookup table used as a substitution box in a white-box context can be created using small-block ciphers [6, 26]. Round keys of the small-block cipher are generated using a trusted XOF function with a secret key, and all values in the $2^{wordsize}$ space are encrypted with the cipher. The generated table is used as a substitution box to ensure nonlinearity in the white-box implementation. In any case, the round keys were generated using the extendable output function SHAKE [14].

Round numbers of the white-box algorithms were computed according to the code lifting security criteria in Section 3.2. The running time performance of 16-bit designs was compared with WARX, and 32-bit designs were compared with SPNbox-32.

3. Security

The white-box security of an algorithm is evaluated by its resistance to key extraction from the lookup table and by its inability to use the table as a large key outside of the white-box environment called code lifting [5].

3.1. **Key Extraction Security.**

In the space-hard ciphers, extracting the secret key from the white-box structure turns into recovering the key from the lookup table in black-box settings [5] since the secret key is embedded into the lookup table. Therefore, the algorithm is as resistant to key extraction attacks as the reliability of the table-creation method. The key extraction security of the white-box implementations relies on the security of the table creation methods of WARX [26] and SPNbox [6].

3.2. **Code Lifting Security.**

Resistance to code-lifting attacks is a crucial security measurement for space-hard ciphers since lookup tables are used as a large key in the design. Therefore, incompressible tables are needed to limit leakage of the code (table) and prevent code lifting attacks [12]. Code lifting security is defining with weak (M,Z)-space hardness [5].

Definition 3.1 (Weak (M,Z)-space hardness [5])**.** A white-box block cipher is called weak (M,Z)-space hard if it is not possible to encrypt/decrypt a randomly selected text with a probability greater than 2^{-Z} until the size of the leakage from the code (table) is reached to M bits.

When computing the probability of success in the encryption/decryption process in a white-box context, we include correctly guessing the corresponding entry of the table if the entry is not located in the leaked part of the lookup table, as in [26]. Let T represent all table sizes in the memory, and M represents the size of the leaked part of the table. We can generalize the success probability as

$$p = (\frac{M}{T} + (\frac{1}{T-M}) \cdot (1 - \frac{M}{T}))^{r \cdot t} \tag{3.1}$$

where r represents the round number, and t represents the table lookup number for one round. If the corresponding entry is in the leaked part of the lookup table, the probability of being correct is 1. The probability of encountering such entries is at most $\frac{M}{T}$, depending on the leak size. If the corresponding value is not in the leaked part, the probability of correctly guessing the value is $\frac{1}{T-M}$. The probability of encountering such an entry is $1 - \frac{M}{T}$.

The maximum achievable security for a white-box algorithm is calculated by considering the leaked size of the lookup table [15, 8, 26] and limited to $keysize - log_2(T)$ bits. Since $\frac{M}{T}$ is considered a small rate, the level of security is generalized to the leak of the entire table. However, we think it is more convenient to take the security level as $keysize - log_2(M)$ bits for more precise calculations on round numbers. The round number may be smaller than the desired number assuming the entire table is leaked. If the leak limit is exceeded, the lookup table must be updated to provide the recommended size of security. Round numbers of the white-box implementations are calculated according to the Equation (3.2).

$$(\frac{M}{T} + (\frac{1}{T-M}) \cdot (1 - \frac{M}{T}))^{rt} = M \cdot 2^{-keysize} \tag{3.2}$$

Similar to [5], we assumed that space-hardness is fixed to the leakage of $\frac{1}{4}$ of the lookup table. As the specified leak size increases, the number of rounds of the white-box algorithm increases to provide the expected security level. For fixed leak size, if an algorithm offers higher security than others with the same number of lookup table entries in one round, the algorithm has more rounds than the others. The expected security level and the number of rounds required for each algorithm are given in Table 2.

3.3. **Diffusion Criteria.** The strict avalanche criteria [29] is used to measure the diffusion property of the white-box conversions. The experiments were performed on 100000 random samples. For each sample, i-th bit of the input was complemented, and its effect on each output bit was examined, respectively. If the j output bit was changed then the entry of $M_{i,j}$ was increased by one. The mean μ value of binomial distribution of M is computed as

$$\mu = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} M_{i,j} \tag{3.3}$$

where n represents the block length of the cipher.

According to the SAC test results in Table 1, the white-box Saturnin is supplied diffusion property after two rounds. The white-box Sparkle needs four rounds

to diffuse every bit of the output, while the white-box Shadow supplies diffusion property after the first round.

TABLE 1. SAC test results with 100000 random samples.

Algorithm	(n, round)	Min. Value	Max. Value	μ
Saturnin	(256,2)	49285	50792	50000
Sparkle	(256,4)	49264	50697	50000
Spook-Shadow	(384,1)	49306	50757	49999

4. PERFORMANCE RESULTS

The running time of the white-box implementations was compared with the white-box algorithms WARX and SPNbox-32. WARX code using Givaro library [25] for finite field calculations was taken from GitHub [19] while we implemented SPNbox-32 without using a library. Reference codes of the lightweight algorithms were pulled from the NIST GitHub repository [28], and the codes were adapted to the white-box context. The algorithms run on randomly generated 3072 bytes messages with 100000 cycles and -O2 optimization on a laptop equipped with x86-64 architecture and a 2.80 GHz Intel Core i7 CPU. The performance results for white-box and black-box implementations are given in Table 2 and Table 3.

White-box adaptation of Saturnin has been compared to WARX. According to performance results in Table 2, white-box Saturnin is almost nine times faster than the WARX, with a maximum achievable security level (MAS) of 242-bit. White-box Saturnin's code size is 1.3KB larger than WARX, while the memory usage of WARX is almost twelve times higher than white-box Saturnin's.

Based on performance comparisons for 32-bit word size algorithms, white-box Sparkle and white-box Shadow are faster than SPNbox-32. White-box Sparkle is ten times faster than SPNbox-32 with 226-bit security, while white-box Shadow is eight times faster than SPNbox-32 with a 354-bit maximum achievable security level. White-box implementation of Sparkle is %22 faster than Shadow. Although the memory usage of the algorithms is the same, SPNbox-32 has the smallest code size.

TABLE 2. Performance results of the white-box implementation.

Algorithm	Key Size (bit)	Round	MAS (bit)	WBI in Cycle (per byte)	Memory Usage	Code Size (KB)
WARX	128	7	114	304	852.5 KB	4.8
Saturnin	256	8	242	33	72 KB	6.1
SPNbox-32	128	10	98	942	16 GB	2.2
Sparkle	256	15	226	93	16 GB	5
Spook-Shadow	384	15	354	120	16 GB	6.6

The black-box Saturnin is %15 faster than the WARX. Similar to the white-box implementation, Saturnin's code size is larger than WARX, while WARX's memory usage is almost 12 times higher than Saturnin's. The black-box Sparkle is %1 faster than Shadow. Nevertheless, both of the black-box implementations are slower than

SPNbox-32. Memory usage of the black-box implementations is the same. When the code sizes of the algorithms are compared, the smallest one is SPNbox-32.

TABLE 3. Performance results of the black-box implementation.

Algorithm	Key Size (bit)	Round	BBI in Cycle (per byte)	Memory Usage (KB)	Code Size (KB)
WARX	128	7	498	852.5	6.1
Saturnin	256	8	422	72	10.9
SPNbox-32	128	10	73695	72	3.2
Sparkle	256	15	115079	72	9.7
Spook-Shadow	384	15	116355	72	12.3

Table 4 states different leak sizes for 114-bit and 98-bit expected security levels. According to the table, there is an inverse relationship between the block size of the algorithm and the table leak size at the same security level. Because the Shadow block size is bigger than others, more table leaks can occur at the same security level. The table leak size of SPNbox is calculated as $T/2^{2.45}$ for the 98-bit security level since the table leak is not included when calculating the round number in [6].

TABLE 4. Table leakage size for 2^{-114} and 2^{-98} success probability.

Algorithm	Security Size (bit)	Table Size (T)	Leakage Size (bit)
WARX	114	128 KB	$T/2^2$
Saturnin	114	128 KB	$T/2^{0.89}$
SPNbox-32	98	16 GB	$T/2^{2.45}$
Sparkle	98	16 GB	$T/2^{0.82}$
Spook-Shadow	98	16 GB	$T/2^{0.54}$

5. CONCLUSION

This study examined lightweight designs for efficiency and suitability to white-box settings. With this approach, linear layers of the appropriate algorithms from the NIST Lightweight Standardization candidates were adapted to the white-box settings to speed up the runtime of the white-box/black-box implementations according to WARX and SPNbox-32. The probability of correctly guessing the unknown part of the lookup table was included in the computing round number of the algorithm. In order to make more accurate calculations, the security size of the algorithms was taken as $keysize - log_2(M)$ bits. According to the performance results, all white-box transformations are faster than (M, Z)-space hard algorithms WARX and SPNbox-32 without decreasing the white-box security level. The use of lightweight components in white-box settings enables reasonably fast algorithm designs.

References

1. Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, Amir Moradi, and Rezaei Shahmirzadi, *Schwaemm and esch: Lightweight authenticated encryption and hashing using the sparkle permutation family*, (2021), https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/sparkle-spec-final.pdf.
2. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaétan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaétan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, and Friedrich Wiemer, *Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher*, (2019), https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf.
3. Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi, *Cryptanalysis of a white box aes implementation*, Selected Areas in Cryptography (Berlin, Heidelberg) (Helena Handschuh and M. Anwar Hasan, eds.), Springer Berlin Heidelberg, 2005, pp. 227–240.
4. Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich, *Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key (extended abstract)*, Advances in Cryptology – ASIACRYPT 2014 (Berlin, Heidelberg) (Palash Sarkar and Tetsu Iwata, eds.), Springer Berlin Heidelberg, 2014, pp. 63–84.
5. Andrey Bogdanov and Takanori Isobe, *White-box cryptography revisited: Space-hard ciphers*, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA), CCS '15, Association for Computing Machinery, 2015, p. 1058–1069.
6. Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser, *Towards practical whitebox cryptography: Optimizing efficiency and space hardness*, Advances in Cryptology – ASIACRYPT 2016 (Berlin, Heidelberg) (Jung Hee Cheon and Tsuyoshi Takagi, eds.), Springer Berlin Heidelberg, 2016, pp. 126–158.
7. Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher, *Saturnin: a suite of lightweight symmetric algorithms for post-quantum security*, (2019), https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/saturnin-spec-round2.pdf.
8. Jihoon Cho, Kyu Young Choi, Itai Dinur, Orr Dunkelman, Nathan Keller, Dukjae Moon, and Aviya Veidberg, *Wem: A new family of white-box block ciphers based on the even-mansour construction*, Topics in Cryptology – CT-RSA 2017 (Cham) (Helena Handschuh, ed.), Springer International Publishing, 2017, pp. 293–308.
9. Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot, *A white-box des implementation for drm applications*, Digital Rights Management (Berlin, Heidelberg) (Joan Feigenbaum, ed.), Springer Berlin Heidelberg, 2003, pp. 1–15.
10. Stanley Chow, Philip Eisen, Harold Johnson, and Paul C. Van Oorschot, *White-box cryptography and an aes implementation*, Selected Areas in Cryptography (Berlin, Heidelberg) (Kaisa Nyberg and Howard Heys, eds.), Springer Berlin Heidelberg, 2003, pp. 250–270.
11. Yoni De Mulder, Peter Roelse, and Bart Preneel, *Cryptanalysis of the xiao–lai white-box aes implementation*, International conference on selected areas in cryptography, Springer, 2012, pp. 34–49.
12. Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain, *White-box security notions for symmetric encryption schemes*, Selected Areas in Cryptography – SAC 2013 (Berlin, Heidelberg) (Tanja Lange, Kristin Lauter, and Petr Lisoněk, eds.), Springer Berlin Heidelberg, 2014, pp. 247–264.
13. Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov, *Design strategies for arx with provable bounds: Sparx and lax*, Advances in Cryptology – ASIACRYPT 2016 (Berlin, Heidelberg) (Jung Hee Cheon and Tsuyoshi Takagi, eds.), Springer Berlin Heidelberg, 2016, pp. 484–513.
14. Morris J Dworkin et al., *Sha-3 standard: Permutation-based hash and extendable-output functions*, (2015).
15. Pierre-Alain Fouque, Pierre Karpman, Paul Kirchner, and Brice Minaud, *Efficient and provable white-box primitives*, Advances in Cryptology – ASIACRYPT 2016 (Berlin, Heidelberg) (Jung Hee Cheon and Tsuyoshi Takagi, eds.), Springer Berlin Heidelberg, 2016, pp. 159–188.

16. Henri Gilbert, Jérôme Plût, and Joana Treger, *Key-recovery attack on the asasa cryptosystem with expanding s-boxes*, Advances in Cryptology – CRYPTO 2015 (Berlin, Heidelberg) (Rosario Gennaro and Matthew Robshaw, eds.), Springer Berlin Heidelberg, 2015, pp. 475–490.
17. Computer Security Resource Center Information Technology Laboratory, *Lightweight cryptography*, https://csrc.nist.gov/Projects/lightweight-cryptography.
18. Matthias Jacob, Dan Boneh, and Edward Felten, *Attacking an obfuscated cipher by injecting faults*, ACM Workshop on Digital Rights Management, Springer, 2002, pp. 16–31.
19. JunLiu9102, *Warx-project*, https://github.com/JunLiu9102/WARX-Project, 2021.
20. Mohamed Karroumi, *Protecting white-box aes with dual ciphers*, Information Security and Cryptology - ICISC 2010 (Berlin, Heidelberg) (Kyung-Hyune Rhee and DaeHun Nyang, eds.), Springer Berlin Heidelberg, 2011, pp. 278–291.
21. Yuji Koike and Takanori Isobe, *Yoroi: Updatable whitebox cryptography*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2021** (2021), no. 4, 587–617.
22. Yuji Koike, Kosei Sakamoto, Takuya Hayashi, and Takanori Isobe, *Galaxy: A family of stream-cipher-based space-hard ciphers*, Information Security and Privacy (Cham) (Joseph K. Liu and Hui Cui, eds.), Springer International Publishing, 2020, pp. 142–159.
23. Jihoon Kwon, Byeonghak Lee, Jooyoung Lee, and Dukjae Moon, *Fpl: White-box secure block cipher using parallel table look-ups*, Topics in Cryptology – CT-RSA 2020 (Cham) (Stanislaw Jarecki, ed.), Springer International Publishing, 2020, pp. 106–128.
24. Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel, *Two attacks on a white-box aes implementation*, Selected Areas in Cryptography – SAC 2013 (Berlin, Heidelberg) (Tanja Lange, Kristin Lauter, and Petr Lisoněk, eds.), Springer Berlin Heidelberg, 2014, pp. 265–285.
25. linbox team, *givaro*, https://github.com/linbox-team/givaro, 2021.
26. Jun Liu, Vincent Rijmen, Yupu Hu, Jie Chen, and Baocang Wang, *Warx: efficient white-box block cipher based on arx primitives and random mds matrix*, Science China Information Sciences (2021), 1869–1919, https://doi.org/10.1007/s11432-020-3105-1.
27. Brice Minaud, Patrick Derbez, Pierre-Alain Fouque, and Pierre Karpman, *Key-recovery attacks on asasa*, Journal of Cryptology, vol. 31, Springer, 2018, pp. 845–884.
28. usnistgov, *Lightweight-cryptography-benchmarking*, https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking, 2021.
29. A. F. Webster and S. E. Tavares, *On the design of s-boxes*, Advances in Cryptology — CRYPTO '85 Proceedings (Berlin, Heidelberg) (Hugh C. Williams, ed.), Springer Berlin Heidelberg, 1986, pp. 523–534.
30. Yaying Xiao and Xuejia Lai, *A secure implementation of white-box aes*, 2009 2nd International Conference on Computer Science and its Applications, IEEE, 2009, pp. 1–6.