Hybrid WBC: Secure and Efficient White-Box Encryption Schemes

Jihoon Cho[1], Kyu Young Choi[1], Orr Dunkelman[2], Nathan Keller[3], Dukjae Moon[1], and Aviya Vaidberg[3]

[1] Security Research Group, Samsung SDS, Inc., Republic of Korea
{jihoon1.cho,ky12.choi,dukjae.moon}@samsung.com
[2] Computer Science Department, University of Haifa, Israel
orrd@cs.haifa.ac.il
[3] Department of Mathematics, Bar-Ilan University, Israel
nkeller@math.biu.ac.il,aviya.v5@gmail.com

Abstract. White-box cryptography aims at providing security against an adversary that has access to the encryption process. Numerous white-box encryption schemes were proposed since the introduction of white-box cryptography by Chow et al. in 2002. However, most of them are slow, and thus, can be used in practice only to protect very small amounts of information, such as encryption keys.

In this paper we present a new threat model for white-box cryptography which corresponds to the practical abilities of the adversary in a wide range of applications. Furthermore, we study design criteria for white-box primitives that are important from the industry point of view. Finally, we propose a class of new primitives that combine a white-box algorithm with a standard block cipher to obtain white-box protection for encrypting long messages, with high security and reasonable performance.

1 Introduction

The standard threat model considered in secret-key cryptography is the *black-box* model, in which the endpoints of communication channels are assumed to be secure, and thus, an adversary may only obtain (plaintext,ciphertext) pairs but no information on the encryption process itself (see Figure 1(a)). In 1996, Kocher [14] introduced the *gray-box* model, in which the adversary may obtain *side-channel* information on the encryption process, such as execution time, power consumption and electromagnetic radiation (see Figure 1(b)). In 2002, Chow et al. [5,6] introduced the *white-box* model, in which the adversary is accessible to the entire information on the encryption process, and can even change parts of it at will (see Figure 1(c)).

The range of applications in which the white-box threat model corresponds to the practical abilities of the adversary is already extensive and continues to grow rapidly. One example is the Digital Rights Management (DRM) realm, where the legitimate user (who, of course, has full access to the encryption

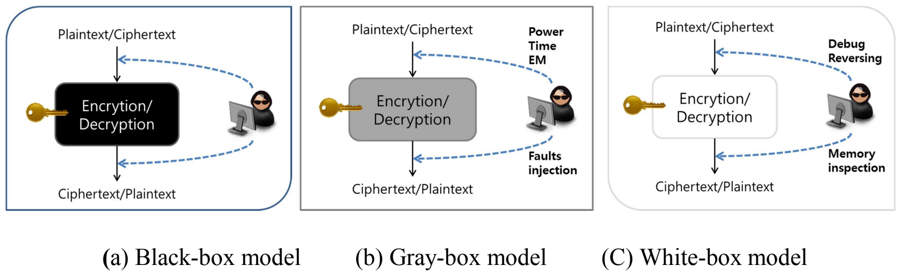

Fig. 1. Attack models

process), may be adversarial. Another example is resource-constrained Internet-of-Things (IoT) devices applied in an insecure environment (like RFID tags on the products in a supermarket). Yet another example is smartphones and public cloud services. While certain security-critical services in such devices are provided with support of hardware security features, such as 'secure element' or TrustZone in mobile devices or 'hardware security modules' in the cloud, most services are implemented as software operating within Rich OS. The main reasons for that are low cost, development efficiency and complicated ecosystems. As a result, the cryptographic implementations are vulnerable to a wide variety of attacks in which the adversary has 'white-box' capabilities.

The ever-growing range of applications where the white-box threat model is relevant necessitates devising secure and efficient solutions for white-box cryptography. And indeed, numerous white-box primitives were proposed since the introduction of white-box cryptography in 2002. These primitives can be roughly divided into two classes.

The first (and more common) class includes algorithms which take an existing block cipher (usually AES or DES), and use various methods to 'obfuscate' the encryption process, so that a white-box adversary will not be able to extract the secret key. Pioneered by Chow et al. [5,6], this approach was followed by quite a few designers. The more common way to 'fortify' the encryption process is using large tables and *random encodings*, as proposed in [5,6]. This way was followed in [13,17,18,22,26,27]. Another way is using algebraic equations and polynomials (e.g., [4]).

An obvious advantage of these designs is their relation to the original ciphers, which makes transition to the white-box primitive and compatibility with other systems much easier. Unfortunately, most of these designs were broken by practical attacks a short time after their presentation (see [1,7,8,11,12,15,16,23,25]), and the remaining ones are very recent and have not been subjected to extensive cryptanalytic efforts yet. Another disadvantage of the designs in this class is their performance – all of them are orders of magnitude slower than the 'black-box' primitives they are based upon (see, e.g., [24]) and also require a significant amount of memory (see Table 1).

Implementation	Size	Best attack
Chow et al. [5]	773 KB	2^{22} [15]
Karroumi [13]	752 KB	2^{22} [15]
Xiao-Lai [26]	20 MB	2^{32} [8,19]
Bringer et al. [4]	568 MB	2^{16} [7]

Table 1. Comparison of previous white-box AES variants.

The second class includes new block ciphers designed especially with white-box protection in mind. Usually such designs are based on *key-dependent components*, such that even if a white-box adversary can recover the full dictionary of such a component, he still cannot use this knowledge to recover the secret key. Recent designs of this class include the ASASA family [2] and the SPACE family [3]. An important advantage of these designs is their better performance and higher security (though, some of them were also broken, see [9,10,20]). On the other hand, transition from existing designs to the entirely new ciphers is not an easy task, and so, quite often commercial users will be reluctant to make such a major change in the design.

Our goal in this paper is to propose a class of new primitives which, on the one hand, provide strong security with respect to a white-box adversary, and on the other hand, are convenient for practical use – meaning that the performance is reasonable and that transition from currently used primitives to the new primitives is relatively easy.

To this end, we first consider the threat model of white-box cryptography. It is clearly unreasonable to assume that the adversary has an unlimited access to the whole encryption process, as in this case, there is no difference whatsoever between the legitimate users and the adversary (and so, no secret-key cryptography is possible). Instead, most of the previous papers on white-box cryptography (e.g., [5,6]) assume that there are parts of the encryption process that can be protected from the white-box adversary (e.g., *external encodings* in Chow et al.'s schemes). However, this assumption is problematic in scenarios where the entire primitive must be implemented in software. We assume, instead, that the adversary has a one-time 'white-box' access to the system, and after that, his capabilities are equal to that of a 'classical' black-box adversary. This model corresponds to a one-time compromise of a system employed by multiple users, when the security goal is to minimize the damage of compromise. We discuss the model in detail and describe practical scenarios in which it is relevant.

We then discuss the design criteria *from the point of view of industry*, aiming at balancing between strong security, reasonable performance, and good compatibility with the currently deployed ciphers.

Finally, we present the new class of primitives, called *Hybrid WBC* schemes. In Hybrid WBC, only the first block of the message is encrypted using a white-box block cipher, while all remaining blocks are encrypted using a 'classical' cipher, like the AES. The core of the scheme is a mode of operation which ensures that although only a single block is encrypted by a white-box scheme, the entire message gets white-box protection in our threat model.[4] As almost all blocks of the message are encrypted using a 'classical' cipher, the new primitives are reasonably fast, and transition from a 'classical' cipher to the new scheme is relatively easy. As per security, based on our preliminary analysis we anticipate that if the adversary has access to at most 2^{64} messages of length at most 2^{64} blocks each, then any attack on the scheme would require either memory or time complexity of at least 2^{80}, which is clearly sufficient for any practical purpose.

2 Practical Requirements and Design Strategy

In this section we analyze the requirements of a white-box encryption scheme from the industry point of view, and then describe our design strategy according to these requirements.

2.1 Security requirements – the threat model

Unlike the classical black-box model, in white-box cryptography the abilities of the adversary are not clearly defined, and different threat models are implicitly used by different authors. The basic intuition is that the adversary can 'do everything', but of course, this cannot be assumed as then no secret can be kept from the adversary whatsoever.

The papers of Chow et al. [5,6] implicitly assume that there is a part of the encryption process, called *external encoding*, which is performed outside of the encryption device and cannot be accessed by the white-box adversary. Such an assumption is not realistic in scenarios where the entire encryption process is implemented in software.

Instead, we propose the following threat model, which is relevant in a wide variety in realistic scenarios. Assume that the same white-box encryption scheme is used in many devices, with at most a small difference between them (e.g., a unique identification number that is used in the encryption process). Further, assume that the adversary can mount an 'expensive' white-box attack on at most a few devices (e.g., by purchasing them and then analyzing in depth), and he is willing to break the encryption of *all other devices*. Formally, we assume that the adversary has a white-box access to several devices from the family and only black-box access to all devices in the family. Using the white-box access, the adversary can obtain full information on the devices he took control of. His goal is to break the encryption schemes of all other devices. Thus, the security goal in this model can be thought of as *minimizing the damage from one-time compromise*.

Our threat model is well suited for IoT environment. IoT devices are usually manufactured in a production line simply assembling flash memories with

[4] We note that similar approaches to ours were pursued in [21,28]. However, as we show below, the primitives presented in these works are completely insecure in our model.

the same binary programmed including cryptographic keys, i.e. the same cryptographic keys are shared across multiple devices. This is because it would be quite expensive to embed separate keys into each device either in production lines or by consumers; additional key-embedding process and related key management, as well as adding UX layers to IoT devices, generally require considerable cost. In such an IoT environment, an adversary may implement the white-box attack for a single device, and try to compromise the whole system using the obtained key or any critical information, using capabilities from the conventional black-box model.

We note that this threat model does not fit for *all* applications of white-box cryptography. However, it seems relevant in sufficiently many scenarios for being considered specifically.

2.2 Performance and cost requirements

While industry accepts the need in strong security of the algorithms, it is often the case that practical efficiency considerations are prioritized by commercial users over security considerations. Hence, if we want to design a primitive that will be employed in practice, we should take into account the main practical requirements from the industry point of view.

The main two design criteria we concentrate on are the following:

Reasonable performance. Previously suggested white-box algorithms except the SPACE family are 12 to 55 times slower than AES. White-box primitives have thus been used to protect relatively small sizes of data. For this reason, in the architecture of DRM service, for example, a content encryption key (CEK) is typically encrypted using a white-box primitive. We aim at using the white-box primitive to protect large amounts of data, and so, the encryption speed must be reasonably fast – ideally, almost as fast as the AES.

Low cost. The new architecture should be designed so as to minimize the modification of the existing development or manufacturing process related to cryptographic implementations. Interestingly, this may be the most important factor for commercial adoption in reality.

To summarize, the practical requirements of the proposed architecture are as follows:

1. **Security.** The new primitive should minimize the damage from one-time compromise. That is, any compromised information in the white-box attack should not give any additional advantage to an adversary in subsequent attacks in the black-box model.
2. **Performance.** The new primitive should protect an arbitrary size of data, providing reasonable performance.
3. **Cost.** The new primitive should be designed to minimize the modification of the existing development or manufacturing process related to cryptographic implementations.

2.3 Design strategies

The practical requirements listed above lead to the following design considerations.

First, if we use a white-box algorithm to encrypt each block of the message then the performance of the resulting encryption scheme is the same as that of the white-box algorithm. For most of the currently existing white-box algorithms, this means that the scheme is very slow. Moreover, even for the SPACE family whose members are not so slow, standard 'software obfuscation techniques' aimed at protecting the security of the running code, make the encryption process much slower, and thus too slow for our purposes.

As a result, it is desirable to use the white-box algorithm to encrypt only part of the message blocks, and encrypt most blocks with a 'classical' algorithm. We will follow this strategy, and aim at reducing the number of calls to the white-box algorithm as much as possible.

Second, almost all existing solutions for data protection in data communication such as SSL, TLS and SSH are based on a shared secret (e.g. session key). Designers of some solutions for data communication want to apply this session key in white-box encryption with minimum modification of their cryptographic implementation. However, they cannot use this key directly in a white-box scheme since the initiation of a white-box algorithm is slow and in general is separate from running environment. In addition, in many cases users request a certificate algorithm to be used in their implementation. Hence, we aim at applying a session key directly in the components of our scheme, except the white-box algorithm.

Third, the most effective way to minimize the damage from one-time compromise is to encrypt each message by a one-time key which is protected by white-box algorithms. However, managing these one-time keys is a big burden and existing key exchange protocols do not provide a one-time session key. Thus, we will encrypt the nonce by a white-box algorithm and use it in the encryption process as a replacement for a one-time key.

2.4 Previous work

The idea of combining a white-box primitive with a standard primitive in order to improve performance was already proposed in [21,28], under the name *composition modes*. However, both proposals are completely insecure with respect to our threat model, as described below.

In 2010, Park et al. [21] proposed a novel method of using white-box cryptography to enhance performance. The design makes use of the PCBC mode of operation with dual keys as input to white-box-AES and standard AES, where the white-box primitive encrypts only one block of the plaintext. Clearly, the scheme of [21] runs as fast as AES for a large size of data. However, with the AES key obtained by the white-box attack, an adversary is easily able to decrypt any other ciphertext blocks encrypted using the same keys which precedes the data block encrypted using a white-box primitive.

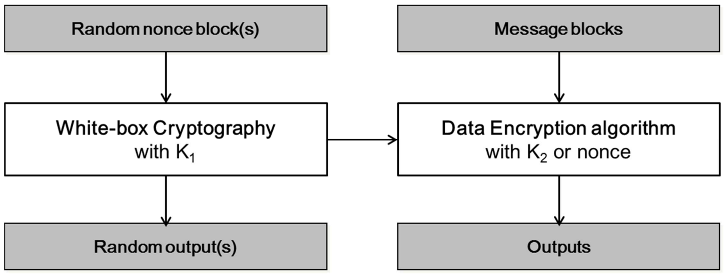
Random nonce block(s)
Message blocks
White-box Cryptography with K_1
Data Encryption algorithm with K_2 or nonce
Random output(s)
Outputs

Fig. 2. The general structure of our primitives

In 2012, Yoo et al. [28] proposed another type of composition mode, using the CBC and OFB modes of operation. In the specific construction using CBC mode, a random value is generated and then XORed with the first block of plaintext before the AES encryption. The random value is also encrypted using a white-box primitive, and concatenated with ciphertext blocks. With the AES key compromised by the white-box attack, an adversary is able to decrypt any other ciphertext blocks encrypted using the same keys in the black-box attack model, except only the first data block.

Thus, both proposals are completely insecure in our one-time-compromise model.

3 The New Primitives

In this section we propose secure and efficient encryption schemes under the white-box model, which we call *Hybrid White-box schemes* (abbreviated H-WBC).

3.1 General structure and security goals

The general structure of our primitives is outlined in Figure 2. It uses two separate keys – one for a white-box primitive and another for a 'classical' encryption algorithm (e.g., AES), where the white-box algorithm is only used for encryption of a nonce (e.g. initial vector (IV) or a counter) while the classical algorithm is used for encryption of plaintexts. The keys K_1 and K_2 are assumed to be permanent and may be shared by many devices, while the nonce in changed in every encryption session.

We restrict the use of our scheme to encrypting messages of length at most 2^{64} blocks in a single session (i.e. without rekeying). Obviously, this amount is sufficient for any practical purpose. Furthermore, as common in nonce-based algorithms, we do not allow re-use of the nonce. (This requirement has no real practical implication unless the number of encrypted messages is extremely large).

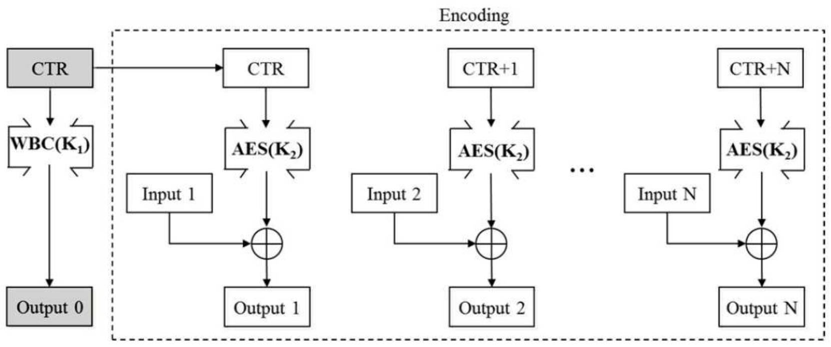

Fig. 3. CTR-WBC: A hybrid white-box scheme using the CTR mode of operation

The security level we aim at is data complexity of 2^{64} and memory and time complexities of 2^{80}. That is, any white-box attack that can recover the secret key K_1, or distinguish our scheme from random, or recover part of the plaintext in a non-compromised session, should require either more than 2^{64} messages, or more than 2^{80} time or more than 2^{80} memory.

Note that in each compromised session, the adversary can recover the full plaintext/ciphertext, as well as the key K_2 and the nonce (since only K_1 is white-box protected[5]). However, as the random nonce acts as a one-time key for this structure, mere knowledge of the nonce does not help to attack other sessions. (To be precise, it provides the adversary with a single plaintext/ciphertext pair for the white-box algorithm. However, this algorithm is assumed to be white-box secure, meaning that even a white-box adversary cannot recover the key K_1 with at most 2^{64} plaintext/ciphertext pairs, 2^{80} time and memory[6]).

3.2 Two basic examples

In this subsection we present two basic examples of our constructions. Both *are not secure* in our model, but serve as a basis for the schemes we present below.

The first scheme, called CTR-WBC and presented in Figure 3, is almost the same as the standard CTR mode of operation using the AES block cipher, the only difference being that a counter CTR is encrypted using a white-box primitive (e.g., white-box-AES or a member of the SPACE family).

[5] Alternatively, one may assume that the key K_1 is not white-box protected, but rather user-dependent. In such a case, the adversary can recover a few K_1 values by white-box attacks, but then his goal is to break the encryption of devices that use a different value of K_1.

[6] It may be possible to lift the whole binary of the white-box algorithm and then run it in a simulator, in which case thie white-box primitive itself is acts as a key. We assume that the system has a counter-measure against such code-lifting attacks, e.g. an additional coding scheme, node-locking techniques, etc.

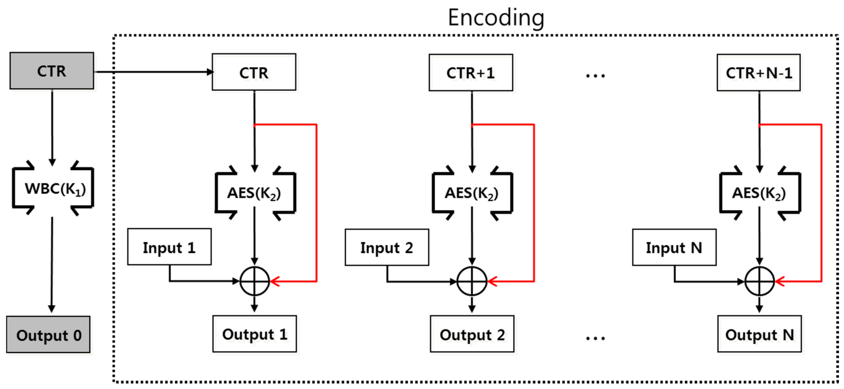

Fig. 4. F-CTR-WBC: Modified CTR-WBC with a feed-forward operation

This scheme is completely insecure in the white-box model. Indeed, an adversary can use a white-box access to a single session to recover K_2, and then for each non-compromised session, if he knows the value $Input_i \oplus Output_i$ even for a single value i, he can compute $CTR + (i-1) = AES^{-1}_{K_2}(Input_i \oplus Output_i)$, and thus, find CTR. Then, as the adversary knows K_2, he can decrypt the entire message. That is, given a single compromised session, the adversary can instantly decrypt any other session if he only knows $Input_i \oplus Output_i$ for a single block (which is usually the situation in practice, due to usage of common file headers etc.).

The second scheme, called F-CTR-WBC and presented in Figure 4, is similar to the first one. The only additional element is a feed-forward operation aimed at thwarting the trivial attack on the CTR-WBC scheme presented above.

If the adversary knows K_2 and $Input_i \oplus Output_i$, he cannot recover CTR directly, as we have $Input_i \oplus Output_i = (CTR + i - 1) \oplus AES_{K_2}(CTR + i - 1)$, and the function $x \mapsto x \oplus AES_K(x)$ is presumably hard to invert even if K is known. However, if for some non-compromised session the adversary knows the values $Input_i \oplus Output_i$ for D blocks $i_1, i_2, \ldots, i_D$, he can mount the following time-memory tradeoff attack.

1. Recover the value of K_2 by a white-box access to a single session.
2. Prepare a table of $2^{128}/D$ values of the form $(x, x \oplus AES_{K_2}(x))$, sorted by the second coordinate.
3. For each $j = 1, 2, \ldots, D$, check whether $Input_{i_j} \oplus Output_{i_j}$ appears in the table. If it appears and corresponds to some value x, guess that $CTR = x - i_j + 1$, and check the guess against one of the other known $Input_i \oplus Output_i$ values.

It is clear that the attack succeeds with a non-negligible probability once $Input_{i_j} \oplus Output_{i_j}$ appears in the table, and the latter occurs with probability $1 - 1/e$ by a birthday paradox argument.

In the ‘worst case’, where the encrypted message has 2^{64} blocks and the adversary knows $Input_i \oplus Output_i$ for all of them, the attack allows to distinguish F-CTR-WBC from a random scheme (and even recover the rest of the message if not *all* the message is known) with data complexity of a single message, and time and memory complexities of 2^{64}, which are below our security bound.

3.3 The new Hybrid White-box schemes

In this subsection we present two new hybrid white-box schemes, which – according to our preliminary analysis – are secure in the white-box model.

First scheme: F-CTR-WBC with a double block length The first scheme is similar to F-CTR-WBC described above, with the difference that the block length is increased to 256 bits, and 256-bit block white-box and classical block ciphers are used instead of 128-bit block ones. For example, we may use Rijndael-256 instead of AES-128, resulting in a scheme which is only about 1.3 times slower than that with AES-128. (As mentioned above, the performance of the white-box algorithm has only a small effect on the performance of the hybrid scheme, as it is used to encrypt only a single block of the message).

The security of this scheme in the black-box model stems from the security of the widely used CTR-AES mode of operation. One structural attack on the CTR mode of operation that may be considered uses the fact that if for two counters CTR, CTR' and for some j we have $CTR = CTR' + j$, then for all $k \geq 0$, the encryption processes starting with CTR, CTR' (respectively) satisfy $Input_k \oplus Output_k = Input'_{j+k} \oplus Output'_{j+k}$. This can be easily recognized by an adversary and used to decrypt unknown parts of the message. However, as the counter is updated sequentially, and there are at most 2^{64} different counters with messages of length up to 2^{64} each, the probability of such a collision for a 256-bit block size is negligible.

As for the white-box model, it is clear that the time-memory tradeoff attack described above can be applied also against this scheme. However, this time in order to get a match in the table, we need a table of size $2^{256}/D \geq 2^{128}$ (as the maximal possible amount of data is 2^{64} messages of 2^{64} blocks each), which means that the complexity of the attack is much higher than our security bound. In our preliminary analysis, we were not able to find any attack on this scheme better than this simple time-memory tradeoff attack.

Second scheme: UF-CTR-WBC The second scheme we propose, presented in Figure 5, is a bit more complex, using AES with feed-forward also in the counter update function. If the full AES is used in both layers of the scheme, it is at least two times slower than F-CTR-WBC with AES-128. However, as the upper layer is used mainly to reduce the relation between consecutive inputs to the second-layer AES and their relation to the initial CTR, it is actually sufficient to use 3-round AES-128 in the upper layer. As a result, this scheme

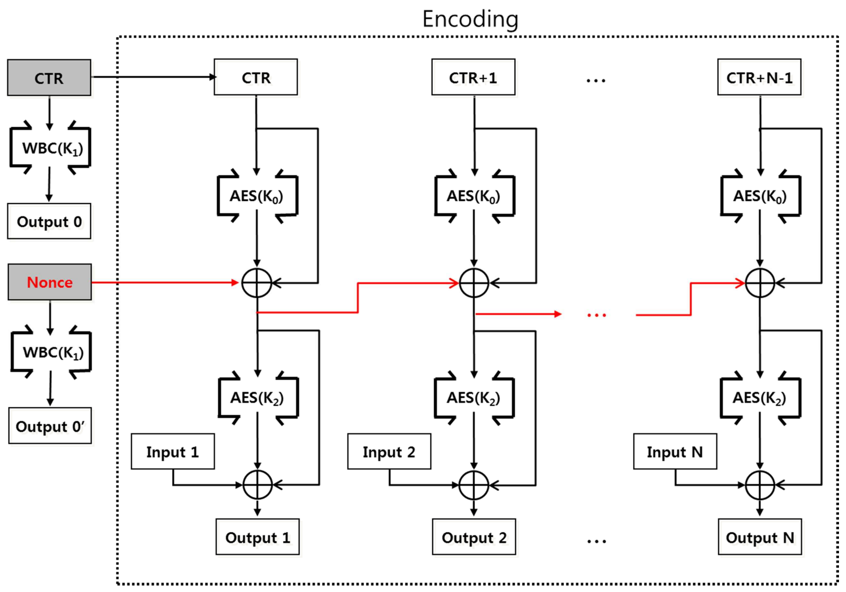

Fig. 5. UF-CTR-WBC: Two-layered CTR-WBC with feed-forwards

is roughly 1.3 times slower than F-CTR-WBC with AES, just like the previous scheme.

As in the previous case, the security of the scheme in the black-box model stems from the security of the widely used CTR-AES mode of operation.

The 'counter collision' attack presented above does not work against this scheme, as even if we have $CTR' = CTR + j$ for some j, the difference between the nonces makes the inputs to AES_{K_2} different, so such occurrence cannot be recognized in the input/output pairs. (An exception is where there is a simultaneous collision in the parts that come from the nonce. However, this already constitutes a 256-bit collision so any attack exploiting it has complexity of at least 2^{128}.)

Likewise, collision between two inputs of AES_{K_2} (which can be observed easily in the input/output pairs) does not help the adversary, since (unlike the case above) it does not imply a collision in the consecutive block. Even if the collision occurs between two consecutive inputs, this still cannot be exploited as the black-box attacker does not know the key K_0 of the first-layer AES.

In the white-box model, we found two attacks on this scheme.

The first attack uses a collision between two consecutive inputs of AES_{K_2}. Denote the input of AES_{K_2} in the encryption of the i'th block by y_i. If for some i we have $y_i = y_{i+1}$ (which can be easily recognized by checking that $Input_i \oplus Output_i = Input_{i+1} \oplus Output_{i+1}$), then by the counter update, we have $(CTR+i-1) \oplus AES_{K_0}(CTR+i-1) = 0$, which means that $CTR+i-1$ is a fixed point of AES_{K_0}. As the white-box adversary can recover K_0, he only has to find (possibly in advance) the fixed point of AES_K where the key is known. If full AES is used in the first layer, then this probably cannot be done much faster than the 2^{128} complexity of brute force. If only 3-round AES is used, this can be probably done rather easily. However, as the probability of a collision between two consecutive y_i values is 2^{-128}, the time complexity of the attack is anyway at least 2^{128}, which is much higher than our security bound.

The second attack is a time-memory tradeoff attack. We assume that the adversary knows D consecutive values $z_j, z_{j+1}, \ldots, z_{j+D-1}$ of the form $z_i = Input_i \oplus Output_i$. The attack algorithm is as follows.

1. Recover the values of K_0, K_2 by a white-box access to a single session.
2. Prepare a table of $2^{128}/D^{1/3}$ values of the form $(x, x \oplus AES_{K_0}(x))$, sorted by the second coordinate.
3. Prepare a table of $2^{128}/D^{1/3}$ values of the form $(x, x \oplus AES_{K_2}(x))$, sorted by the second coordinate.
4. For each $j = 0, 1, \ldots, D-2$, check whether both z_j and z_{j+1} appear in the table of AES_{K_2}. For each such pair:
   (a) If z_j, z_{j+1} correspond to the values y_j, y_{j+1} respectively, guess that the output of the first layer in the $i+1$'th block is $y_j \oplus y_{j+1}$.
   (b) Check whether $y_j \oplus y_{j+1}$ appears in the table of AES_{K_0}. If it appears with x_{j+1}, guess that $CTR = x_{j+1} - j$, and check the guess against one of the other known $Input_i \oplus Output_i$ values. If no, discard the pair.

The probability that two consecutive values z_j, z_{j+1} appear in the table of AES_{K_2} is $D^{-2/3}$. Hence, it is expected that the D z_i values known to the adversary contain about $D^{1/3}$ pairs (z_j, z_{j+1}) that lead to a check in the table of AES_{K_0}. As the probability of a value to appear in that table is $D^{-1/3}$, a single match is expected to be found. The data complexity of the attack is $D/2^{64}$ messages of length 2^{64}, the memory complexity is $2^{128}/D^{1/3}$ and the time complexity is D. The total complexity is the lowest for $D = 2^{96}$, for which we get data complexity of 2^{32} messages and time and memory complexities of 2^{96}. While these are lower than 2^{128}, they are still much higher than our 2^{80} security bound.

We were not able to find stronger attacks on this scheme in the white-box model.

4 Conclusions

In this paper we formulated a new threat model for white-box cryptography, which corresponds to the realistic abilities of the adversary in a wide range of applications, including lightweight IoT applications running in a hostile environment. We proposed a new class of encryption schemes, called *hybrid WBC*, that allow obtaining strong security with respect to our threat model, along with reasonable performance (only 1.3 times slower than AES) and a relatively low cost of transition from 'classical' ciphers to the new primitive. We presented an initial analysis of our primitives, but of course external evaluation is needed in order to gain confidence in their security, before deploying them in practice.

References

1. O. Billet, H. Gilbert, and C. Ech-Chatbi, *Cryptanalysis of a White Box AES Implementation*, proceedings of Selected Areas in Cryptography (SAC) 2004, volume 3357 of Lecture Notes in Computer Science, pp. 227-240. Springer, 2004.
2. A. Biryukov, C. Bouillaguet, and D. Khovratovich, *Cryptographic Schemes Based on the ASASA Structure: Black-Box, White-Box, and Public-key*, proceedings of ASIACRYPT 2014, volume 8873 of Lecture Notes in Computer Science, pp. 63–84, Springer, 2014.
3. A. Bogdanov and T. Isobe, *White-box Cryptography Revisited: Space-Hard Ciphers*, proceedings of Computer and Communications Security (CCS) 2015, pp. 1058–1069, ACM, 2015.
4. A. Bringer, H. Chabanne, and E. Dottax, *White Box Cryptography: Another Attempt*, IACR ePrint report 2006:468.
5. S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot, *White-Box Cryptography and an AES Implementation*, proceedings of Selected Areas in Cryptography (SAC) 2002, volume 2595 of Lecture Notes in Computer Science, pp. 250-270. Springer, 2002.
6. S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot, *A white-box DES implementation for DRM applications*, proceedings of Security and Privacy in Digital Rights Management (DRM) 2002, volume 2696 of Lecture Notes in Computer Science, pp. 1-15, Springer, 2002.
7. Y. de Mulder, B. Wyseur, and B. Preneel, *Cryptanalysis of a Perturbated White-Box AES Implementation*, proceedings of Indocrypt 2010, volume 6498 of Lecture Notes in Computer Science, pp. 292–310, Springer, 2010.
8. Y. De Mulder, P. Roelse, and B. Preneel, *Cryptanalysis of the Xiao–Lai White-Box AES Implementation*, proceedings of Selected Areas in Cryptography (SAC) 2012, pp. 34-49, Springer, 2013.
9. I. Dinur, O. Dunkelman, T. Kranz, and G. Leander, *Decomposing the ASASA Block Cipher Construction*, IACR ePrint report 2015:507.
10. H. Gilbert, J. Plût, and J. Treger, *Key-Recovery Attack on the ASASA Cryptosystem with Expanding S-Boxes*, proceedings of CRYPTO 2015 (1), volume 9215 of Lecture Notes in Computer Science, pp. 475–490, Springer, 2015.
11. L. Goubin, J. M. Masereel, and M. Quisquater, *Cryptanalysis of White Box DES Implementations*, proceedings of Selected Areas in Cryptography (SAC) 2007, volume 4876 of Lecture Notes in Computer Science, pp. 278–295, Springer, 2007.
12. M. Jacob, D. Boneh, and E. W. Felten, *Attacking an Obfuscated Cipher by Injecting Faults*, proceedings of Security and Privacy in Digital Rights Management (DRM) 2002, volume 2696 of Lecture Notes in Computer Science, pp. 16-31, Springer, 2002.
13. M. Karroumi, *Protecting White-Box AES with Dual Ciphers*, proceedings of Information Security and Cryptology (ICISC) 2010, volume 6829 of Lecture Notes in Computer Science, pp. 278-291. Springer, 2011.
14. P. C. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems*, proceedings of CRYPTO 1996, volume 1109 of Lecture Notes in Computer Science, pp. 104–113, Springer, 1996.
15. T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, and B. Preneel, *Two Attacks on a White-Box AES Implementation*, proceedings of Selected Areas in Cryptography (SAC) 2013, volume 8282 of Lecture Notes in Computer Science, pp. 265–285. Springer, 2014.

16. T. Lin and X. Lai, *Efficient attack to white-box SMS4 implementation*, Journal of Software, **24** (2013) pp. 2238-2249. (In Chinese)
17. H. E. Link and W. D. Neumann, *Clarifying Obfuscation: Improving the Security of White-Box DES*, proceedings of Information Technology: Coding and Computing (ITCC) 2005, pp. 679-684, IEEE Computer Society, 2005.
18. R. Luo, X. Lai, and R. You, *A new attempt of white-box AES implementation*, proceedings of Security, Pattern Analysis, and Cybernetics (SPAC) 2014, pp. 423–429, IEEE Computer Society, 2014.
19. W. Michiels, P. Gorissen, and H. D. L. Hollmann, *Cryptanalysis of a generic class of white-box implementations*, proceedings of Selected Areas in Cryptography (SAC), volume 5381 of Lecture Notes in Computer Science, pp. 414–428, Springer, 2009.
20. B. Minaud, P. Derbez, P.-A. Fouque, and P. Karpman, *Key-Recovery Attacks on ASASA*, proceedings of ASIACRYPT 2015 (2), volume 9453 of Lecture Notes in Computer Science, pp. 3–27, Springer, 2015.
21. J. Park, O. Yi, and J. Choi, *Methods for practical whitebox cryptography*, proceedings of Information and Communication Technology Convergence (ICTC) 2010, pp. 474–479, IEEE, 2010.
22. Y. Shi, W. Wei, and Z. He, *A Lightweight White-Box Symmetric Encryption Algorithm against Node Capture for WSNs*, Sensors **15(5)** (2015), pp. 11928–11952.
23. L. Tolhuizen, *Improved Cryptanalysis of an AES implementation*, proceedings of Symposium on Information Theory in the Benelux 2012.
24. B. Wyseur, *White-Box Cryptography*, Ph.D. thesis, Katholieke Universiteit Leuven, 2009.
25. B. Wyseur, W. Michiels, P. Gorissen, and B. Preneel, *Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings*, proceedings of Selected Areas in Cryptography (SAC) 2007, volume 4876 of Lecture Notes in Computer Science, pp. 264-277, Springer, 2007.
26. Y. Xiao and X. Lai, *A Secure Implementation of White-Box AES*, proceedings of Computer Science and its Applications (CSA 2009), pp. 1-6, IEEE, 2009.
27. Y. Xiao and X. Lai, *White-Box Cryptography and a White-Box Implementation of the SMS4 Algorithm*, proceedings of ChinaCrypt 2009, pp. 24-34. (In Chinese)
28. J. Yoo, H. Jeong, and D. Won, *A method for secure and an efficient block cipher using white-box cryptography*, proceedings of Ubiquitous Information Management and Communication (ICUIMC) 2012, No. 89, pp. 1–8, ACM, 2012.