

White-Box Implementation to Advantage DRM

Antonius Cahya Prihandoko[#], Hossein Ghodosi^{*}, Bruce Litow^{*}

[#]*Department of Information System, University of Jember, Jl. Kalimantan 37, Jember, 68121, Indonesia*
E-mail: antoniuscp.ilkom@unej.ac.id

^{*}*Department of Information Technology, James Cook University, Townsville, QLD 4811, Australia*
E-mail: hossein.ghodosi/bruce.litow@jcu.edu.au

Abstract— Digital Rights Management (DRM) is a popular approach for secure content distribution. Typically, DRM encrypts the content before delivers it. Most DRM applications use secure algorithms to protect content. However, executing these algorithms in an insecure environment may allow adversaries to compromise the system and obtain the key. To withstand such attack, algorithm implementation is modified in such a way to make the implementation unintelligible, namely obfuscation approach. White-box cryptography (WBC) is an obfuscation technique intended to protect secret keys from being disclosed in a software implementation using a fully transparent methodology. This mechanism is appropriate for DRM applications and able to enhance security for the content provider. However, DRM is required to provide a balanced protection for the content provider and users. We construct a protocol on implementing WBC to improve DRM system; The system does not only provide security for the content provider but also preserves privacy for users.

Keywords— digital rights management; content distribution system; obfuscation; white-box cryptography; security; privacy

I. INTRODUCTION

Secure delivery is urgently required in a content distribution system to guarantee that only authorized users can access the protected content and use it properly. Digital Rights Management (DRM) is a popular approach for this security requirement. Under this system, content is typically sent in an encrypted form along with the license associated with it. At the users' side, an application processes the license by means of a rights expression manager (REM), authenticates the users and decrypts the content using the corresponding decryption routine. The application can be implemented in hardware, such as in a set-top box for typical pay TV systems, or in software on the users' PC.

Trusted media players in most DRM applications contain the decryption key. The key must be kept secret and inaccessible to users, as finding the key would allow someone to decrypt and access content without restriction, thus defeating DRM protection. Unfortunately, trusted media players are often running on an untrusted platform. Although encryption algorithms used by most DRM applications, such as the data encryption standard (DES) and the advanced

encryption standard (AES), are believed to be secure, executing them in an insecure environment may allow adversaries to compromise the system and obtain information about the decryption key [1]-[3]. Therefore, the protection scenario must be set to prevent the extraction of the key, even when the application is executed in an insecure environment.

Keeping the decryption key from being accessible to the users is a major challenge for the DRM systems. An approach to this problem is by applying obfuscation techniques. In these techniques, the implementation of the encryption algorithm is modified so that it would be unintelligible. Two common techniques for such an obfuscation approach are code obfuscation and white-box cryptography (WBC).

Code obfuscation is intended to protect software implementations. In this technique, the program (code) used to implement the algorithm is rewritten in such a way that certain characteristics of the original program are hidden and unintelligible. Theoretically, a probabilistic algorithm O is an obfuscator of a program P if it computes the same function as P , and anything that can be efficiently computed

from $\mathcal{Q}(P)$, can be efficiently computed given oracle access to P [4]. However, obfuscation is hard to achieve. Both positive and negative results that determine possibility and impossibility, respectively, of code obfuscation, were investigated [5]-[12]. While general purpose obfuscators are currently impossible, obfuscators for simple functions may exist.

In practice, obfuscation of a program (code) is applied to the variables used in the program. Variable names are scrambled, and data that was stored in a single variable is split into multiple variables and recombined at execution time. This mechanism makes a code difficult for human to understand, and thus effective for hiding an algorithm and protecting the code, but not for any encryption key used by the code. Additionally, code obfuscation is often integrated with code flattening. In code flattening, extra paths are introduced into the program structure. This technique makes the program difficult to analyze. However, it can be reverse engineered and, thus, fails to achieve the main goal of code obfuscation.

Theoretical researches on code obfuscation are many, but fewer on implementations. Most of the researches produce conceptual decisions whether or not the obfuscator of a particular program exists according to a certain definition. However, none provides a real example on how to obfuscate a program if such an obfuscator provable exists. Practical obfuscation, on the other hand, has a less theoretical foundation. Because of the lack of a bridge that connects theoretical and practical aspects, code obfuscation is less applicable in the DRM implementation.

White-box cryptography is an obfuscation technique that is often used in the DRM applications. Though having a less theoretical model than code obfuscation, WBC reflects more the reality. Of the protection techniques applied in the DRM implementations, white-box cryptography is suggested to be the most effective protection in the DRM applications [1]. Currently, WBC is being used in real-world applications. Several commercial companies such as Microsoft, Apple, and Sony have announced or have shown to deploy white-box techniques. Although there are many cryptanalysis techniques have been published, so far in a real-world product, there has been no white-box implementation that has suffered from a key extraction attack. Practically, breaking white-box implementations is hard and time-consuming [3]. The attacks are very dependent on the construction of the white-box implementation and the properties of the underlying cipher [3]. Therefore, broadly applicable attacks are difficult to deploy.

The main goal of this study is to implement the WBC concept to advantage DRM. As a protection system, DRM tends to put a great emphasis on content provider's rights and often neglects users' privacy. However, customers' satisfaction is an important factor in a content distribution system. Therefore, DRM is required to provide a balanced protection between the content provider's right and the users' privacy [13]. It means that DRM should not merely focus on achieving the security for the content provider, but

also on preserving the privacy for the users. We construct a white-box implementation protocol that enables DRM to provide a balanced protection for the content provider and the users.

A. Problem Statement

Typical DRM systems for content distribution (see Fig. 1) consist of four parties: content provider, distributor, clearing house and consumer or user [14]. The content provider is a content holder and wants to sell the content for profits. First of all, the content provider needs to encrypt the content for security purposes. The provider then passes the protected content to the distributor and the corresponding license to the clearinghouse. A distributor is typically a web server running an online shop. The distributor makes the protected content available on the web server and enables users to download it.

To be able to decrypt and use the downloaded content, a user needs to acquire an appropriate license from the clearinghouse. The user has to register his profile, provide details of the purchased content, and then do the payment. After all these steps are properly processed, the clearinghouse releases a license containing usage rules and a key to decrypt the content.

Most DRM systems make the protected digital content available on their distributor servers. Users can obtain the protected content from the distributor channel and then request a license containing the decryption key from the clearinghouse. Downloading content from the distributor's channel does not seriously threaten either content provider's security or users' privacy. Users may download content anonymously (and even freely) so that their identity would not be connected to the content they choose. However, without obtaining an appropriate decryption key, users cannot unlock the protected content.

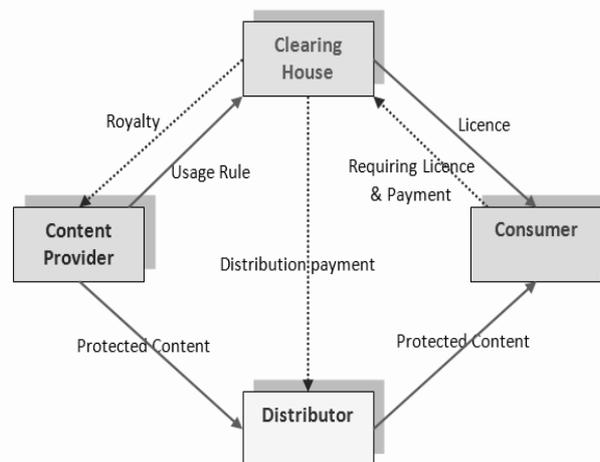


Fig. 1 Typical DRM system

In contrast, acquiring a license from the clearinghouse causes serious concerns over security and privacy. From the content provider perspective, this mechanism may give a threat to his security. If an eavesdropper steals licenses when

a user requests them from the clearinghouse, a great revenue will be lost. From the users' perspective, the mechanism emerges a threat to their privacy. Personal information they submit to the clearinghouse is not guaranteed to be kept secret, as the clearinghouse may send the users' data and viewing detail to marketing agencies. The users expect that they have their privacy protected and are able to access digital content anonymously. Indeed, in many applications, security and privacy need to be equally protected [15]. We utilize white-box implementation to overcome the security and privacy problems in this model.

The rest of this paper is organized as follows. Section II describes the Material and Method. Section III provides Result and Discussion. Finally, section IV draws concluding remarks.

II. MATERIAL AND METHOD

White-box cryptography (WBC) is intended to implement cryptographic primitives. The intention is to protect secret keys from being disclosed in a software implementation. The protection is done in such a way even when the platform on which the application is executed is subject to the control of potentially hostile end-users.

The term of "white-box" relates to the attack model that is applied to examine the security of this protection mechanism. Unlike the traditional cryptographic threat model, black-box, which assumes that attackers can only observe the input and output of the algorithm, the white-box model assumes that the attackers have full control over the whole operation and can freely observe dynamic code execution. Despite providing a fully transparent methodology, WBC integrates the cipher in such a way that does not reveal the secret key. This mechanism is appropriate for DRM applications which are often executed in an insecure environment.

The basic notion of white-box implementations (see Fig. 2) is to rewrite a key so that all information related to the key is hidden. External encoding can be used so that the encryption and decryption software require encoded inputs, and produce encoded outputs. This encoding mechanism can be done by replacing the encryption function E_k with the composition $E'_k = G \circ E_k \circ F^{-1}$. Input encoding function F and output decoding function G^{-1} must not be on the same platform that computes E'_k so that the white-box implementation cannot be used to compute E_k . This means that encoding input and decoding output have to be kept secret. At this point, white box implementation cannot stand alone; it should be used in conjunction with other techniques to provide protection against key recovery attacks [16]. Although this scenario is not standard, such an approach is useful for many DRM implementations.

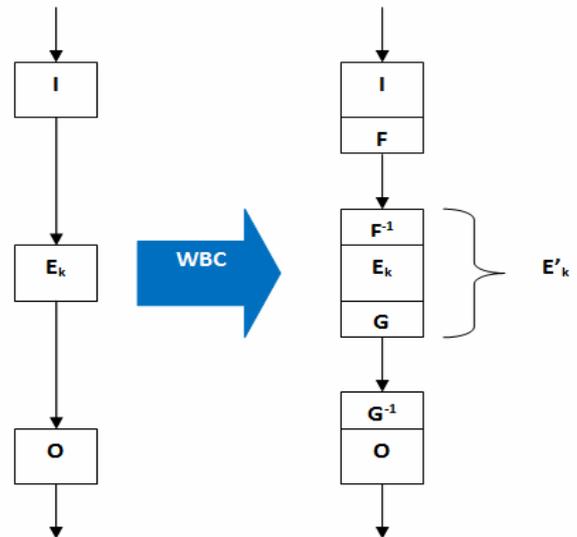


Fig. 2 Basic notion of white-box implementation [3]

A. Security and Feasibility of White-Box Cryptography

Security of white-box implementation is relative; there is no system that is absolutely secure. A system is secure relative to a security model which may depend on an adversary's goal and the resources that can be accessed by the adversary [16]. In the white-box scenario, it is much more difficult to determine the resources of an attacker as they are endless. The best effort in such an implementation is to prevent all known relevant threats in an effective way. A secure protection, for example, can be achieved by combining the effect of the secret key with some implementation specific data using a mathematical operation that is extremely hard to invert [17]. This mechanism allows constructing a system that operates similarly to the asymmetric encryption algorithm, with a performance level close to the symmetric algorithm [18]. The security also depends on the implementation --- a strong cryptographic algorithm is not necessary for a poor implementation.

Despite the robustness of practical white-box implementations, performance, memory size and security are still the main concerns for current applications. Low performance and high-consumed memory size limit the application of WBC, especially for mobile devices.

Although no attack on commercial white-box implementations has been found, it does not exclude the possibility of successful attacks in future. Additionally, the effectiveness of the white-box implementation is limited when an attacker can observe the execution of the DRM program. Therefore, it is not enough to only protect an application against key extraction; the application must also be hard to invert. Furthermore, adding hardware protection is extremely effective, but costly.

B. Constructed Model

To implement WBC in the DRM applications, we propose to employ smart cards. A smart card contains an embedded microprocessor so that it can be used not only to store data but also to process the data [19]. Since a smart card carries both processing power and information, it does not need access to the remote database at the time of a transaction. A smart card may contain programs and mobile databases that can be modified, updated or deleted through embedded program functions. The microprocessor is also used for security purposes. Data are never directly available to the external applications as the microprocessor controls data handling and memory access according to a given set of conditions.

The white-box implementation works by obfuscating the original encryption and decryption keys. Suppose E is the encryption function. Two random functions F and G are generated to obfuscate E . Instead of using $E(X)$, cipher text X' is computed using diffused function $G^{-1}(GEF^{-1})F(X)$. Composition (GEF^{-1}) is called internal function, while G^{-1} and F are external functions. The provider then passes the protected content along with the internal function to the distributor and corresponding external functions to the smart cards manufacturer. The smart card is used to store the external keys and the original encryption key, compute their inverse and do external encoding-decoding. The manufacturer produces smart cards and delivers them to the distributor that will make them available to purchase.

A model of the smart card needs to be defined to make the scheme works. In this model, a smart card memory has two parts: accessible area and inaccessible area. User's device can only communicate to the former part but cannot approach the latter. Data structure and mobile database are stored in the inaccessible area. In this case, external functions G^{-1} and F must be inaccessible. These items are only accessible by external encoding and decoding mechanisms defined in the accessible area. The inverses of G^{-1} and F are computed prior to encoding input and decoding output of the internal decryption, respectively. The manufacturer then sends the created smart cards to the distributor who then makes the corresponding protected content available online.

A user can download the chosen content from the distributor server and purchases the corresponding smart card. To unlock the protected content, the user's device must be connected to a compatible card reader. User's device has two functions: runs the internal decryption and plays the decrypted content. The decryption protocol involves the communication between user's device and the smart card (see Fig. 3).

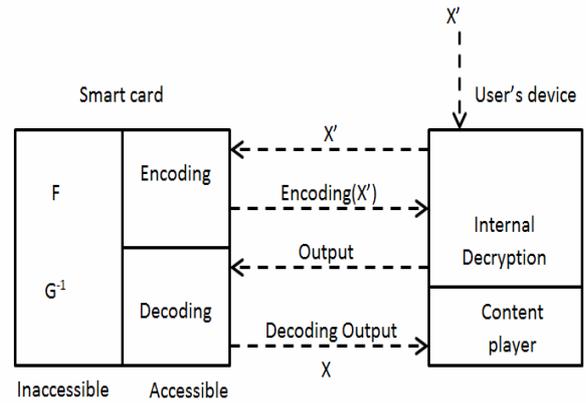


Fig. 3 Decryption protocol

- Upon receiving the encrypted content X' , user's device records its internal key and passes it to the smart card.
- The smart card encodes the cipher text X' using function G , and send $G(X')$ as an input for the internal decryption mechanism in the user's device.
- The device uses the inverse of the internal function to decrypt the input and send the output $FE^{-1}G(G(X'))$ back to the smart card.
- The smart card then decodes this output using function F^{-1} to obtain the content X . The content X is now playable to the user's device.

III. RESULT AND DISCUSSION

A. Analysis of Security and Privacy

Our mechanism improves the DRM model for content distribution (see Fig. 4). Instead of employing a clearing house, the system involves a smart card manufacturer. This mechanism makes the system more efficient. Users can obtain content and its corresponding smart card from one party, i.e. the distributor. Assuming the smart card is a tamper-proof device, security and privacy of the scheme can be analyzed as follow.

Theorem 1. Security

Assuming that the smart card is a tamper-proof device, the mechanism achieves security for the content provider.

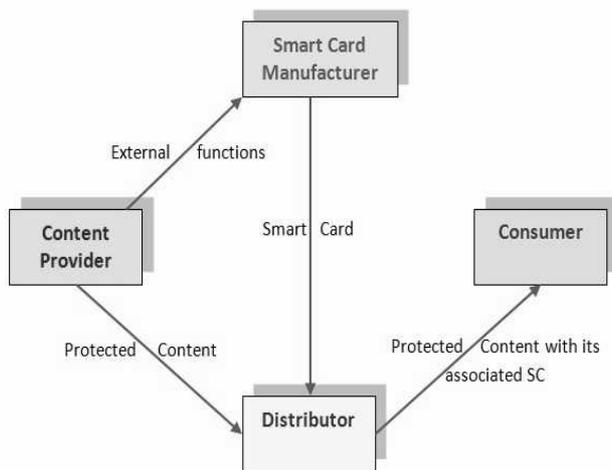


Fig. 4 Advantaged DRM system

Proof: In this mechanism, digital content is protected, and the decryption key is hidden behind the obfuscated encryption function. The essential functions (external functions) that can be used to reveal such the secret key are stored in the inaccessible smart card's memory. Knowing only the internal function is not adequate to unlock the content, as the internal decryption mechanism has to collaborate with external encoding-decoding operations which are undertaken inside the smart card. The smart card is only allocated to the user who has made the payment for it. Therefore, the user can only access the content corresponding to the smart card he purchased.

Theorem 2. Privacy

Assuming that the smart card is a tamper-proof device, the mechanism preserves the privacy of the users.

Proof: To be able for unlocking the downloaded content, a user does not need to provide his personal data for the license. Instead, he needs to purchase the corresponding smart card anonymously. The content and its associated smart card will not be connected to the user's identity. Therefore, the user can privately consume the content and, thus, his privacy is protected.

B. Applications of the White-Box Implementation Scheme

The advantage of a smart card helps white-box implementation to achieve security for the content provider and preserve privacy for the users of a content distribution system. The white-box implementation scheme can be applied in both off-line and online business scenarios.

1) *Off-Line Content Distribution Scenario:* Content provider encrypts digital content and passes the protected content to the distributor and all usage rules to the smart card manufacturer. The usage rules contain the external keys (G-1 and F), the original encryption key (E), all mechanisms on how and when these function can be retrieved, and external encoding-decoding function that has to be performed by the

corresponding smart card. Once smart cards are completed, the manufacturer sends them to the distributor.

The distributor is typically an off-line retailer. The distributed content could be digital movies or songs. The retailer wraps the protected content (stored in mass storage devices such as CD's or DVDs) along with its corresponding smart card and then makes them available to purchase. Users can purchase this package anonymously from the shop. This means that the distributor will not record the users' identity nor connect it to the purchased item. Thus, the users can privately play back the content on their smart card equipped players or computers connected with a smart card reader.

2) *Online Content Distribution Scenario:* In the context of an online content distribution scenario, the role of smart cards could be filled by a secure distributor server. The external and the original encryption keys are stored securely in the server. The distributor provides the protected content online and available to download. Users could download the protected content anonymously (and also maybe freely), but they cannot unlock the content unless they purchase passing codes. Two pass codes have to be used to unlock protected content. These codes are the outputs of the external encoding and decoding mechanisms which are undertaken inside the distributor server.

To keep anonymity, the purchasing *passing codes* can be done using an *electronic cash* scheme [20]. Before requesting *passing codes*, a user has to purchase adequate unit tokens from the distributor. The tokens can be used to purchase multiple items. Assuming that a user has downloaded protected content, the online content decryption is then performed through protocol 1. With this scenario, users can do an online transaction anonymously. Thus, while the content provider can securely distribute the protected assets, the users' privacy is also preserved.

The term of privacy protection in these applications, especially in the off-line scenario, is confined to the fact that the users' identity is not officially recorded nor connected to the purchased item. However, in real practice, an off-line retailer will know who is purchasing which item, as the customer directly comes to the shop. The privacy can be perfectly protected if the content provider has a package containing N protected items and the user is allowed to opt K out of these N items. In this case, the user must not be able to access more than K items, and the content provider must not be able to determine which items are selected by the user. This scenario is known as the *oblivious transfer* concept.

Protocol 1

- At the time of acquiring the passing codes, the user has to submit the downloaded content's ID (it could be the serial number) and an adequate amount of tokens.
- After verifying the payment, the distributor server performs external encoding according to content's ID submitted by the user. This process outputs the first passing code.

- This passing code enables user's player to partially decrypt the content. The user then sends the output of this decryption to the server.
- The server performs external decoding based on the partial decryption's output submitted by the user. This decoding process yields the second passing code.
- Finally, the second passing code allows the user's player to fully decrypt and play the content.

The illustration of these applications can be used to ease engineering technology students to connect mathematical concepts and technological applications. To achieve this goal, the illustration can be integrated in a relevant effort like in [21].

C. Comparative Evaluation

This section compares our proposed White-box implementation scheme to existing obfuscation approach literature. The literature includes two common techniques --- code obfuscation and white-box cryptography.

Code obfuscation is intended to protect software implementation. Theoretical study of the software protection was initiated by Goldreich and Ostrovsky [22], who provided a hardware-based theoretical treatment. This study motivated the emergence of code obfuscation ideas. The first contribution for a formalization of code obfuscation was provided by Hada [23], who presented a notion of obfuscation based on the simulation paradigm for zero knowledge. However, the formal definition of obfuscation was initiated by Barak *et al.* [4]. According to their definition, a probabilistic algorithm O is an obfuscator of a program P if it satisfies:

- **Functionality.** $O(P)$ is a program that computes the same function as P .
- **Virtual Black Box Property (VBBP).** Anything that can be efficiently computed from $O(P)$ can be efficiently computed given oracle access to P .

Obfuscation, however, is hard to achieve. Barak *et al.* [4] showed there exist some predicates that can be efficiently computed when having access to an obfuscated implementation $O(f)$, but, given oracle access to f , no efficient algorithm can compute the predicate much better than by random guessing. As a result, a generic obfuscator, i.e. an obfuscator that protect any given program, does not exist.

The first positive results in code obfuscation referred to the set of point functions as the obfuscatable family [5]. A point function can be obfuscated by random oracles because the output of a random oracle hides all information about the input that produced it. The use of random oracles for obfuscation was motivated by the expectation that given access to an idealized building block, it would be feasible to obfuscate some functions. However, the existence of the idealized block allows the construction of a natural class of functions that are impossible to obfuscate and programmable random oracles, in practice, are difficult to realize [6].

Moreover, under cryptographic assumptions, obfuscators of point functions with multi bit output can be constructed without a random oracle [9]. A best-possible obfuscation may not hide all information [6]. An obfuscated code may leak as little information as any other code, meaning that any information that is not hidden by the obfuscated code is also not hidden by another program with the same size and functionality.

Other positive results of code obfuscation were also applied to cryptographic primitives. First of all, the simulation-based obfuscation [11], which allows obfuscating point function, converting secret-key cryptography into public-key cryptography and transforming message authentication codes (MAC). The obfuscation of an indistinguishability under chosen plaintext attack (IND-CPA) secure symmetric encryption scheme results in an IND-CPA secure asymmetric scheme. Similar results hold for the obfuscation of MAC algorithms into digital signature schemes. However, these results do not apply to indistinguishability under chosen ciphertext attack (IND-CCA) secure schemes. Another positive obfuscation for traditional cryptography was applied to widely used re-encryption functionality [12]. This functionality takes a ciphertext for message m encrypted under a party's public key and transforms it into a ciphertext for the same message encrypted with the other party's public key. Overall, code obfuscation is less applicable in the DRM applications than white-box cryptography.

The first introduced white-box implementations were applied to the DES and AES [24], [25]. The Advance Encryption Standard (AES) consists of N_r rounds; where $N_r = 10$ for AES-128. A basic round has four parts: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. An `AddRoundKey` operation occurs before the first round, and the `MixColumns` operations are omitted in the final round.

The general notion of the white box AES implementation [25] is to merge several steps of the cipher into a network of lookup tables and obfuscate the results using random input-output encoding. First of all, the partial evaluation technique is deployed to hide the key. The key is integrated into the `SubBytes` transformation by constructing 160 8×8 (i.e. 8-bit in, 8-bit out) lookup tables $T_{i,j}^r$.

$$T_{i,j}^r = S(x \oplus k_{i,j}^r) \text{ for } i = 0, \dots, 3; j = 0, \dots, 3; r = 1, \dots, 9 \quad (1)$$

$$T_{i,j}^{10} = S(x \oplus k_{i,j}^{10}) \oplus k_{i,j-i}^{11} \text{ for } i = 0, \dots, 3; j = 0, \dots, 3 \quad (2)$$

Here S is the AES S-box, and $k_{i,j}^r$ is the AES subkey byte in i -th row and j -th column at round r . These T -boxes comprise the `SubBytes` step with the previous round's `AddRoundKey` step. Each output of the T -box and the `ShiftRows` step contributes to 4 bytes of the state array after the `MixColumns` step. This contribution can be described by a 32×8 submatrix MC_i of the 32×32 bit matrix MC representing `MixColumns`. The entire function can be described as a 32×8 bit lookup table. This lookup table needs to be obfuscated with 4-bit nibble encodings. To add to the diffusion, 8×8 affine mixing bijections are

inserted before T_{ij}^r and a 32×32 affine bijections MB is inserted after the $MixColumns$. The mixing bijection is canceled out by the preceding MC (round $r - 1$), which includes the inverse. Hence, the inversion step is diffused over several lookup tables. To cancel out the effect of the MB mixing bijection, an additional lookup table is implemented. Finally, the external input and output encodings are implemented. A 128×128 mixing bijection is inserted prior to the first T -box and a different 128×128 mixing bijection after the last T -box computation. Thus instead of $AES_k, N_O \circ M_G \circ AES_k \circ M_F \circ N_I$ is implemented. M_F and M_G are the affine input and output respectively, and N_I and N_O are the input and output non-linear nibble encodings respectively.

Comparison of standard AES and white-box AES (see Fig. 5) shows that to obfuscate the AES algorithm, each of its parts is diffused into several lookup tables. The sequences of lookup tables S1 to S4, MC1 to MC4 and A1 to A4 are functionally equivalent to $SubBytes$, $MixColumns$, and $AddRoundKey$, respectively. The $Shift$ operation, however, is not undertaken as a lookup table due to its nature. White-box AES implementation provides a secure protection of a secret key in a cryptographic module but has a slow performance. The size and performance comparisons between standard AES and white-box AES implementation are presented in Table 1.

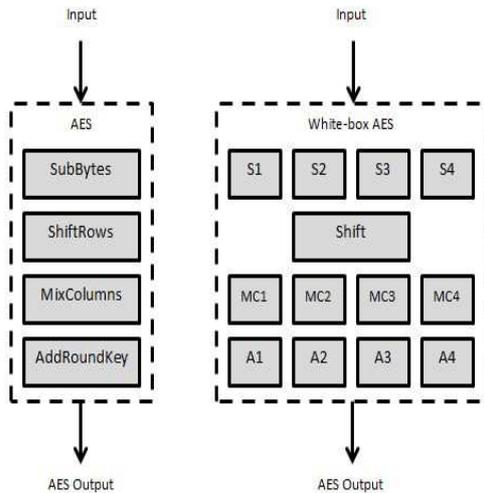


Fig. 5 Comparison of Standard AES and White-box AES [1]

TABLE 1
SIZE AND PERFORMANCE COMPARISON [25], [26]

Description	Standard AES	White-box AES
Memory size	4352 bytes	770048 bytes
Operation	300 lookups and xors	3104 lookups
Encrypt 1 MB data	< 0.5 seconds	> 3 seconds

The white-box AES implementation [25] has been broken using an algebraic cryptanalysis technique [27], with the worst time complexity of 2^{30} . Nevertheless, the implementation can still provide an effective key protection by increasing the attacks' time complexity. One technique for

increasing such complexity is Medusa [28], a software tamper resistance technique which makes a binary program code tamper resistant by incorporating the code into the key of a white-box implementation. This technique increases the integrity of the white-box implementation; if an adversary modified the implementation, the results would be wrong. As a result, the attack time complexity can be much higher.

In response to the algebraic cryptanalysis, a new construction beyond the lookup table strategy, called perturbations strategy [29], was introduced. This strategy is an improvement of the traceable block cipher scheme [30]. However, this construction was proven to be insecure by De Mulder *et al.* [31], and there has been no improvement made for the perturbations strategy up to now.

The most recent improved scheme is proposed by Yoo *et al.* [26]. To address the performance problems, Yoo *et al.* combined white-box AES and standard AES; they used a white-box AES only once in the beginning, and then applied standard AES to the rest of the scheme. They claimed that their scheme has the same performance as the standard AES and is robust to withstand white-box attack. This claim, however, needs to be criticized. If white-box is only implemented in the first round and the subsequent rounds remain in standard AES, the scheme will be exposed to white-box attacks as the standard one as additions with round key information can easily be distinguished. The scheme may be secure to withstand black-box attacks but may fail to preserve security against software attacks.

Our proposed white-box implementation scheme directly adopts the basic notion presented in Fig. 2. For security purposes, the content key is obfuscated by a composition of internal and external keys which are stored separately. The external keys are essential --- without these keys, the content decryption key is hard to reconstruct. To keep the external keys secret, they are stored in an inaccessible area of a smart card's memory. With this mechanism, our proposed scheme requires a simpler computation than many white-box implementations proposed in the literature. The security of our scheme relies on the physical security of smart cards. To preserve users' privacy, the smart card can be purchased anonymously. This mechanism makes the scheme achieve content provider's security and users' privacy simultaneously.

IV. CONCLUSION

DRM is mainly employed by the content provider to combat piracy. Not surprisingly, the DRM systems put more attention to the content provider's security and often neglect the users' privacy. However, the users' privacy needs to be fairly protected as the users' satisfaction is important in a content distribution system. Additionally, the significant threat to both security and privacy may emerge in typical DRM model.

White-box cryptography (WBC) is suggested to be the most effective protection technique in the DRM applications. Despite having a transparent methodology, WBC integrates the cipher in such a way that does not reveal the decryption key. This mechanism is appropriate to protect software

implementation, even when the software is executed in an insecure environment. WBC is potential to enhance security for the content provider.

Our approach integrates white-box implementation and smart card application to advantage DRM. The smart card handles the external decoding which is a part of the white-box implementation and essential to hide the secret decryption key. The communication protocol between the smart card and the user's device enhances the content provider's security. Moreover, the use of smart card in the content distribution system also allows the users to consume the content anonymously and, thus, their privacy is protected. Our advantaged DRM model provides a balanced protection for the content provider and users.

ACKNOWLEDGMENT

This paper is a revised and extended version of our paper entitled *Obfuscation and WBC: Endeavour for Securing Encryption in the DRM Context*, presented at the 2013 International Conference on Computer Science and Information Technology (CSIT-2013) on June 16-18, 2013 at Yogyakarta, Indonesia.

REFERENCES

- [1] R. Schultz. (2012, April) The Many Facades of DRM. *MISC HS 5 Magazine*. 58-64. Available: http://whiteboxcrypto.com/files/2012_misc_drm.pdf
- [2] B. Wyseur, "White-box Cryptography," PhD Thesis, Elektrotechniek-Esat, Katholieke Universiteit Leuven, Leuven, 2009.
- [3] B. Wyseur. (2012, April) White-Box Cryptography: Hiding Keys in Software. *MISC HS 5 Magazine*. 65-72. Available: http://whiteboxcrypto.com/files/2012_misc.pdf
- [4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (Im)possibility of Obfuscating Program," in *Advance in Cryptology - CRYPTO 2001: 21st Annual International Cryptology Conference*, Santa Barbara, California, USA, 2001, pp. 1-18.
- [5] B. Lynn, M. Prabhakaran, and A. Sahai, "Positive Results and Techniques for Obfuscation," in *International Conference on the Theory and Applications of Cryptographic Techniques -Advances in Cryptology - EUROCRYPT 2004*, Interlaken, Switzerland, 2004, pp. 20-39.
- [6] S. Goldwasser and G. N. Rothblum, "On Best-Possible Obfuscation," in *TCC 2007*, Amsterdam, The Netherland, 2007, pp. 194-213.
- [7] N. Bitansky and R. Canetti, "On Strong Simulation and Composable Point Obfuscation," in *CRYPTO 2010*, 2010, pp. 520-537.
- [8] N. Bitansky, R. Canetti, S. Goldwasser, S. Halevi, Y. T. Kalai, and G. N. Rothblum. (2011, 22 June). *Program Obfuscation with Leaky Hardware*. Available: <http://eprint.iacr.org/2011/660.pdf>
- [9] R. Canetti and R. R. Dakdouk, "Obfuscating Point Functions with Multibit Output," in *EUROCRYPT 2008*, Istanbul, Turkey, 2008, pp. 489-508.
- [10] R. Canetti, G. N. Rothblum, and M. Varia, "Obfuscation of Hyperplane Membership," in *TCC 2010*, 2010, pp. 72-89.
- [11] D. Hofheinz, J. Malone-Lee, and M. Stam, "Obfuscation for Cryptographic Purposes," in *TCC 2007*, Amsterdam, The Netherland, 2007, pp. 214-232.
- [12] S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan, "Securely Obfuscating Re-encryption," in *TCC 2007*, 2007, pp. 233-252.
- [13] A. C. Prihandoko, B. Litow, and H. Ghodosi, "DRM's Rights Protection Capability: A Review," in *The First International Conference on Computational Science and Information Management*, Medan, Indonesia, 2012, pp. 12-17.
- [14] Q. Liu, R. Safavi-Naini, and N. P. Sheppard, "Digital Rights Management for Content Distribution," presented at the Australian Information Security Workshop on ACSW Frontiers'03, 2003.
- [15] M. H. Shaikh and N. A. Ansari, "Examining a Norwegian Client's Response over Information Security and Privacy Policy," *International Journal on Advanced Science Engineering Information Technology*, vol. 5, pp. 165-169, 2015.
- [16] M. Joye, "On White-Box Cryptography," in *Security of Information and Networks*, 2008, pp. 7-12.
- [17] A. Saxena, B. Wyseur, and B. Preneel, "Towards Security Notions for White-Box Cryptography," presented at the Information Security ISC 2009, 2009.
- [18] SafeNet. (2012, 24 July). *Understanding White Box Cryptography*. Available: www.safenet-inc.com
- [19] Z. Chen. (2000). *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*.
- [20] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," in *CRYPTO'88*, 1988, pp. 319-327.
- [21] N. Bakri, T. S. Salleh, and Z. M. Zin, "Designing an Integrated Teaching and Learning of Mathematics and Image Processing in Engineering Technology," *International Journal on Advanced Science Engineering Information Technology*, vol. 6, pp. 548-552, 2016.
- [22] O. Goldreich and R. Ostrovsky, "Software Protection and Simulation on Oblivious RAMs," *Journal of the Association of Computing Machinery*, vol. 43, pp. 431-473, 1996.
- [23] S. Hada, "Zero-Knowledge and Code Obfuscation," in *ASIACRYPT 2000*, 2000, pp. 443-457.
- [24] S. Chow, P. Eisen, H. Johnson, and P. C. v. Oorschot, "A White-Box DES Implementation for DRM Applications," presented at the DRM 2002, 2003.
- [25] S. Chow, P. Eisen, H. Johnson, and P. C. v. Oorschot, "White-Box Cryptography and an AES Implementation," presented at the SAC 2002, 2003.
- [26] J. Yoo, H. Jeong, and D. Won, "A Method for Secure and Efficient Block Cipher using White-Box Cryptography," presented at the 6th International Conference on Ubiquitous Information Management and Communication, 2012.
- [27] O. Billet, H. Gilbert, and C. Wch-Chatbi, "Cryptanalysis of a White Box AES Implementation," presented at the SAC 2004, 2005.
- [28] W. Michiels and P. Gorissen, "Mechanism for Software Tamper Resistance: An Application of White-Box Cryptography," in *7th ACM Workshop on Digital Rights Management*, 2007, pp. 82-89.
- [29] J. Bringer, H. e. Chabanne, and E. Dottax, "Perturbing and Protecting a Traceable Block Cipher," in *the 10th Communications and Multimedia Security (CMS) 2006*, 2006, pp. 109-119.
- [30] O. Billet and H. Gilbert, "A Traceable Block Cipher," in *Advances in Cryptology - ASIACRYPT 2003*, Taipei, Taiwan, 2003, pp. 331-346.
- [31] Y. D. Mulder, B. Wyseur, and B. Preneel, "Cryptanalysis of a Pertubated White-Box AES Implementation," presented at the Progress in Cryptology - Indocrypt 2010, 2010.