IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2016, accepted July 31, 2016, date of publication September 28, 2016, date of current version October 31, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2604289

Protecting Encrypted Signature Functions Against Intrusions on Computing Devices by Obfuscation

YANG SHI, (Member, IEEE), JINGXUAN HAN, HONGFEI FAN, QINPEI ZHAO, AND QIN LIU
School of Software Engineering, Tongji University, Shanghai 200092, China

Corresponding author: Q. Liu (sse508lab@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61202382 and Grant 61503286 and in part by the Fundamental Research Funds for the Central Universities.

ABSTRACT Digital signature schemes are widely used to protect information integrity in computer communications. However, conventional digital signature schemes are secure only in normal attack contexts. Technically, these schemes assume that the signing algorithm implementation is running in a perfectly secure platform protected from various kinds of attacks and intrusions. To complement existing studies, this paper studies how to securely generate identity-based signatures, key-insulated signatures, and fuzzy identity-based signatures in a more austere attack context, such as on a computing device that is potentially controlled by an attacker. We use program obfuscation for achieving a higher security level. Concretely, we give three specialized signature functions—encrypted identity-based signature, encrypted key-insulated signature, and encrypted fuzzy identity-based signature, and then propose an obfuscator for the three encrypted signature functions. The efficiency of the proposed obfuscator is theoretically analyzed, and the correctness and security are proved. Finally, we present experimental results that show the proposed scheme is efficient. As a result, the obfuscated implementations of these encrypted signature functions can be applied to many protocols and enhance their security.

INDEX TERMS Encrypted signature, identity-based signature, key-insulated signature, program obfuscation, software protection.

I. INTRODUCTION

Traditionally, in studies on computer system and communication security, it is assumed that the end-points of the network or communication channel are trusted and that the applications are executed in a secure context. In this traditional model, the attacker has, at most, access to the input/output behavior of the program because he/she is only able to manipulate or eavesdrop on the message between the trusted parties.

However, in practice, a program is sometimes executed on a platform that is not fully trustable. This could be the result of a physical capture of the computing device, e.g., a lost mobile phone or tablet, or an unattended sensor node. System vulnerabilities may also give rise to this issue because an attacker can intrude the system and even acquire the highest privileges as a root user or system administrator. With the fast development of Advanced Persistent Threats, such events happen even on ''well-protected'' systems such as the core systems of information technology companies (e.g., RSA and Google) or the nuclear industry. ''Side-channel attacks,'' such as power analysis, timing analysis, electromagnetic emanation attacks, fault injection, and acoustic analysis, is another category of threats that exceed the traditional attack model. Moreover, some new vulnerabilities, such as the ''SSL Heartbleed'' which was disclosed in April 2014 in the widely-used OpenSSL library, could also enable hackers to extract information from a remote host. These threats go beyond the extent of side-channel attacks because the extraction of cryptographic keys stored in the memory of a remote host is possible if the attackers are sufficiently ''lucky.''

Under these threats, in addition to the functionalities of the security modules of a system, we should focus on defending the implementations of security-sensitive programs from attackers that are assumed to have partial or even full access privileges for the software implementation and control of the operation system. Digital signature schemes allow a signer to sign a message with his/her private signing key, and any other party can verify that the message originates from the signer and has not been modified in any way. If the secret signing key of a digital signature scheme or signature related security scheme is obtained by a hacker, the security of the scheme will be severely damaged or even completely lost. Even in a key-evolvement cryptosystem, the security will

be lost, at least within a certain time span. As discussed above, with the privilege of access or even control on the computing platform, an attacker may do so simply by dumping memory and recovering high-entropy data those are likely to be a key. The attacker can also do so by dynamically tracing the execution of the code to identify the memory containing the key.

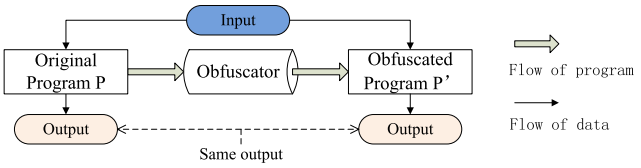

FIGURE 1. Principle of obfuscation.

Fortunately, a signing algorithm can be protected from these threats if it is obfuscated. Program obfuscation is a useful technique to enhance the security of an implementation of a program because it is capable of making the program ''unintelligible'' while preserving its computational functionality. As shown in Fig. 1, an obfuscator is a specialized ''compiler'' that performs obfuscating transformations [1]. Generally speaking, an obfuscator may change the identifiers, data, control flow, and even the class hierarchy of a program. Thus, it would be difficult for an attacker to analyze or break an obfuscated program.

General purpose commercial obfuscation tools, such as Zelix KlassMaster,[1] Shield4J,[2] and Allatori,[3] are not intended to and cannot be used to protect extra-sensitive information such as keys. They cannot achieve a sufficiently high level of security from attacks on software implementations. Moreover, standard digital signature schemes are hard to obfuscate from the perspective of cryptography. It was even proved in [2] that digital signature schemes cannot be securely obfuscated under certain assumptions. Therefore, we construct some signature-related schemes/algorithms that are obfuscatable (from the strong security perspective of provable cryptography) to provide building blocks for security solutions and applications. Inspired by prior works such as [3]–[5], we selected encrypted signatures as the object of our research. In addition, motivated by the widely used identity-based (ID-based) cryptography, we focus on finding a specialized obfuscator for an ID-based signature-related function to protect the secret signing key at the beginning. Note that although there are some advances in protecting secret keys of ID-based *encryption-related* schemes [6], [7], protecting signing keys of ID-based *signature-related* schemes by obfuscation with high computational efficiency and provable security is still demanded.

An obfuscatable encrypted fuzzy ID-based signing (EFIBS) algorithm is first provided. We then derive an obfuscatable encrypted ID-based signing (EIBS) algorithm and an obfuscatable encrypted key-insulated (KI) signing (EKIS) algorithm from the EFIBS because ID-based cryptography and KI cryptography are widely-used in security solutions. Based on the carefully designed algorithms, an (unified) obfuscator for the three algorithms is provided. The main contribution from a theoretical perspective is the design of an obfuscatable EFIBS and the proof of the obfuscator's security. From a practical perspective, the main contribution is the derived obfuscatable EIBS and EKIS for building security solutions and applications.

The rest of this article is organized as follows. Section II presents the main contributions: obfuscatable EFIBS, EIBS, and EKIS functions and the obfuscator. Section III presents a theoretical analysis on the efficiency, correctness (preserving computational functionality), and ''Virtual Black-box Property'' (VBP) security of the obfuscator. Section IV provides experimental results of performance testing. Section V compares the result of this paper to other studies on obfuscation for cryptographic purposes. Finally, we conclude with a discussion of our findings in Section VI.

II. OBFUSCATABLE EFIBS, EIBS, AND EKIS FUNCTIONS AND THE OBFUSCATOR

A. BACKGROUND AND OVERVIEW

Study on obfuscation consists of two main streams. One was initiated by Collberg *et al*. [8], [9] from the perspective of easy-to-achieve but relatively weak protection on the security, privacy, and intellectual property of software. The results of this stream are widely used in the practice of software engineering and many obfuscation tools (obfuscators) are available. The other one was initiated by Barak *et al*. [10] from the perspective of providing strong protection with strong mathematical basis. The main theoretic result of this stream is that, under the security definition called the VBP, general-purpose obfuscation is impossible. This theoretic result is expanded in a number of studies, such as the impossibility of approximate obfuscation in [11], the impossibility of obfuscation with auxiliary input in [2] and the impossibility of best possible obfuscation in [12].

Fortunately, a number of obfuscatable concrete security-related functions and corresponding obfuscators [3]–[5], [13]–[20] have been constructed since 2007. The security definition used by these findings is the ''Average Case VBP'' (ACVBP), which is slightly weaker than the original VBP. The main advantage of these functions and obfuscators is that they are sufficiently efficient to be applied in practice and at the same time, provably secure against the severe threats mentioned above.

This paper falls in within the scope of this approach and introduces three new obfuscatable functions. Concretely, in this subsection, four security schemes are described as building blocks for obfuscatable encrypted signature functions. The first one is a fuzzy ID-based signature (FIBS) scheme proposed in [21]. Based on the FIBS scheme, we derive an

[1] http://www.zelix.com/klassmaster/index.html
[2] http://shield4j.com/
[3] http://www.allatori.com/

ID-based signature (IBS) scheme and a KI signature (KIS) scheme as the second and the third schemes. The last one is a group homomorphic encryption (GHE) scheme proposed in [22]. The descriptions of some of the algorithms in the FIBS and GHE schemes are slightly modified for convenience. Note that this is a nontrivial task to put the schemes in collaboration because the encryption and signing algorithms should be "compatible" to build the obfuscatable functions. Moreover, it is easy to derive IBS/KIS from FIBS. However, it is not so easy to find a suitable IBS/KIS scheme to acquire obfuscatable EIBS/EKIS. Therefore, the proposed IBS/KIS scheme plays an important role in this study. In Fig. 2, we sketch out the relationship and novelty of the schemes and algorithms. The number of the relevant section is noted in parentheses.

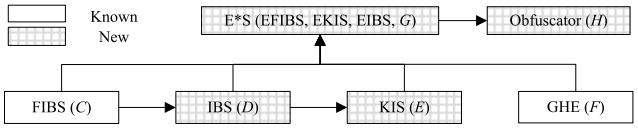

FIGURE 2. Overview of Section 2.

B. GENERAL SETTINGS AND COMPUTATIONAL ASSUMPTIONS

The general setting with algebraic structures and algorithms in this paper is as follows: p is a prime number, and G and G_T are two multiplicative cyclic groups of order p. Further, we have a bilinear map $\widehat{e} : G \times G \rightarrow G_T$ with the following three properties.

- Bilinearity: $\widehat{e}(u^a, v^b) = \widehat{e}(u, v)^{ab}$ where $u, v \in G$ and $a, b \in Z_p^*$.
- Non-degeneracy: $\exists u \in G \wedge \exists v \in G$ s.t. $\widehat{e}(u, v) \neq 1$.
- Computability: For all $u, v \in G$, it is efficient to compute $\widehat{e}(u, v)$.

The main computational assumption in this study is the l-linear decisional Diffie–Hellman (l-linear for short) assumption (when $l \geq 2$). This assumption asserts that there exists no probabilistic polynomial time (PPT) algorithm that is capable of solving the l-linear problem with non-negligible probability. The l-linear problem in G is defined as follows: Let $g_1, \cdots, g_l, h, H \in_\$ G$ and $r_1, \cdots, r_l \in_\$ Z_p^*$. Given $g_1, \cdots, g_l, g_1^{r_1}, \cdots, g_l^{r_l}, h$ and H as input, output "yes" if $h^{r_1+\cdots r_l} = H$ and "no" otherwise.

Before presenting the concrete schemes and algorithms, frequently used symbols are listed in TABLE 1.

TABLE 1. Frequently used symbols.

Symbol	Description
G , G_T	Cyclic groups of prime order p
$\hat{e}$	A bilinear map from $G \times G$ to G_T
K_ω	A signing key (in FIBS)
N	The number of signers (in IBS)
ω , ω'	Identities (in FIBS)
pub	System parameters and public values
(n, d)	A pair of integers (threshold)
$\Delta_{i,[1,n+1]}(\cdot)$	The Lagrange coefficient
ID , ID'	Identities (in IBS)
K_{ID}	A signing key (in IBS)
T	The number of time periods (in KIS)
K_t	A signing key of time period t (in KIS)
PK	A signer's public key (in KIS)
SK	A signer's private key (in KIS)
PP	A set of public parameters ($PP \in pub$)
C	Ciphertext
M	Message
MK	The master key
m	The length of message (in bits)
S	A signature
PK_e	A receiver's encryption key
SK_e	A receiver's decryption key

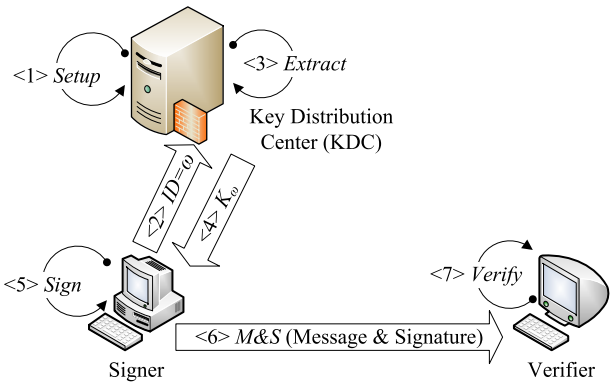

FIGURE 3. Working principle of a FIBS scheme.

C. FIBS SCHEME

The FIBS scheme enables a user with identity ω to generate a signature that can be verified with identity ω' if and only if ω and ω' are within a given distance that is judged by some metric. The FIBS scheme consists of four algorithms, that is, Setup, Extract, Sign, and Verify. The usage of the algorithms is shown in Fig. 3 and their descriptions are shown in Fig. 4.

1. Setup (Algorithm 1): The setup algorithm takes the system-wide general settings as input. The algorithm outputs MK (the master key) and pub (a tuple consisting of public values and system parameters). Let $Gen[G]$ denote all the generators of G. Note that a pair of integers (n, d) is the threshold value of the underlying secret sharing scheme.
2. Extract (Algorithm 2): Suppose that we wish to generate a signing key for a user whose identity is ω, where ω is a subset of $[1, n]$. The extract algorithm takes pub, MK, and ω as input. It outputs a corresponding private signing key K_ω.

Algorithm 1. $FIBS.Setup\left(n,p,d,\hat{e},G,G_T\right)$

Begin

$g \xleftarrow{\$} Gen[G]$

$y,z',z_1,\cdots,z_m \xleftarrow{\$} Z_p$

$g_1 \leftarrow g^y$

$g_2,t_1,\cdots,t_{n+1} \xleftarrow{\$} G$

$A \leftarrow \hat{e}(g_1, g_2) \in G_T$

$v' \leftarrow g^{z'}, v_1 \leftarrow g^{z_1},\cdots,v_m \leftarrow g^{z_m}$

$PP \leftarrow \left(g, g_1,g_2,t_1,\cdots,t_{n+1},v', v_1,\cdots, v_m, A\right)$

$pub \leftarrow \left(n,p,d,\hat{e},G,G_T,PP\right)$

$MK \leftarrow y$

$output \quad \left(pub,MK\right)$

End

Algorithm 2. $FIBS.Extract(pub,MK,\omega)$

Begin

$\left(n,p,d,\hat{e},G,G_T,PP\right) \leftarrow pub$

$\left(g, g_1,g_2,t_1,\cdots,t_{n+1},v', v_1,\cdots, v_m, A\right) \leftarrow PP$

$q(\cdot) \leftarrow GenRandPoly\left(d,MK\right)$

For each $i \in \omega$ Do :

$r_i \xleftarrow{\$} Z_p$

$d_i \leftarrow g^{-r_i}$

$D_i \leftarrow g_2^{q(i)} T\left(i\right)^{r_i}$

EndFor

$output \quad K_\omega = \left\{\left\{D_i \middle| i \in \omega\right\},\left\{d_i \middle| i \in \omega\right\}\right\}$

End

Algorithm 3. $FIBS.Sign(pub,K_\omega,M)$

Begin

$\left(n,p,d,\hat{e},G,G_T,PP\right) \leftarrow pub$

$\left(g, g_1,g_2,t_1,\cdots,t_{n+1},v', v_1,\cdots, v_m, A\right) \leftarrow PP$

For each $i \in \omega$: $s_i \xleftarrow{\$} Z_p$

$(\mu_1,\cdots,\mu_m) \leftarrow M$

$$S_1 \leftarrow \left\{ D_i \cdot \left(v' \prod_{j=1}^{m} v_j^{\mu_j} \right)^{s_i} \middle| i \in \omega \right\}$$

$S_2 \leftarrow \left\{d_i \middle| i \in \omega\right\}$

$S_3 \leftarrow \left\{g^{-s_i} \middle| i \in \omega\right\}$

$output \quad S = \left(S_1,S_2,S_3\right)$

End

Algorithm 4. $FIBS.Verify(pub,\omega',S,M)$

Begin

$\left(\left\{S_{1,i} \middle| i \in \omega\right\},\left\{S_{2,i} \middle| i \in \omega\right\},\left\{S_{3,i} \middle| i \in \omega\right\}\right) \leftarrow S$

Choose an arbitrary d-element subset λ of $\omega' \cap \omega$

$\left(n,p,d,\hat{e},G,G_T,PP\right) \leftarrow pub$

$\left(g, g_1,g_2,t_1,\cdots,t_{n+1},v', v_1,\cdots, v_m, A\right) \leftarrow PP$

$(\mu_1,\cdots,\mu_m) \leftarrow M$

$$\Pi \leftarrow \prod_{i \in \lambda} \left[\hat{e}\left(S_{1,i}, g\right) \cdot \hat{e}\left(S_{2,i}, T\left(i\right)\right) \cdot \hat{e}\left(S_{3,i}, v' \prod_{j=1}^{m} v_j^{\mu_j} \right) \right]^{\Delta_{i,\lambda}(0)}$$

IF $\left(\Pi = A\right)$ $output$ 1

Else $output$ 0

End

FIGURE 4. Algorithms of the FIBS scheme.

In Algorithm 2, $GenRandPoly(d, y)$ is used to generate a random polynomial $q(\cdot)$ such that the degree of $q(\cdot)$ is $d-1$ and $q(0) = y$, and T is given by

$$T(x) = g_2^x \prod_{i \in [1,n+1]} t_i^{\Delta_{i,[1,n+1]}(x)}, \tag{1}$$

where $[1, n+1] = \{1, 2, \cdots, n+1\}$ and the Lagrange coefficient is given by $\Delta_{i,[1,n+1]}(x) = \prod_{w \in [1,n+1], w \neq i} (x-w)/(i-w)$.

3. Sign (Algorithm 3): To sign a message $M = (\mu_1, \cdots, \mu_m) \in \{0,1\}^m$, the signing algorithm takes pub, K_ω, and the message M as input and outputs a signature $S = (S_1, S_2, S_3)$.
4. Verify (Algorithm 4): To verify a signature $S = (S_1, S_2, S_3)$ with regard to an identity ω' where $|\omega' \cap \omega| \geq d$, the verification algorithm takes pub, S, and M as input. The algorithm outputs 1 if the signature is valid, and outputs 0 otherwise.

D. IBS SCHEME

By modifying the FIBS scheme, an IBS scheme is obtained. Without loss of generosity, we suppose that the set of all possible ID values of signers is a consecutive subset of $\mathbb{N}$ that starts from 1. Let N be the number of signers; the set is $\{1, \cdots, N\}$. The IBS scheme consists of four algorithms, i.e., Setup, Extract, Sign, and Verify, as shown in Fig. 5. The usage of the algorithms of the IBS scheme is similar to the usage of the corresponding FIBS algorithms, as illustrated in Fig. 3.

1. Setup (Algorithm 5): The algorithm generates pub (a tuple consisting of public values and system parameters) and the master key MK. The algorithm is implemented by calling the $Setup$ algorithm of the FIBS scheme.
2. Extract (Algorithm 6): Suppose ID is the identity of a user. The extract algorithm takes pub, MK, and ID as input, and outputs a corresponding private signing key K_{ID} for the user.
3. Sign (Algorithm 7): To sign a message $M = (\mu_1, \cdots, \mu_m) \in \{0,1\}^m$, the signing algorithm takes pub, K_{ID}, and M as input and outputs a signature S.
4. Verify (Algorithm 8): To verify a signature S with regard to an identity ID', the verification algorithm takes pub, ID', S, and a message M as input and outputs 1 or 0 if the signature is valid or invalid, respectively.

E. KIS SCHEME

In this section, we propose a perfectly KIS scheme supporting random-access key updates. The "perfectly KI" is equivalent to (T–1, T)–KI, and "random-access key updates"

Algorithm 5. $IBS.Setup(N, p, \hat{e}, G, G_T)$ Begin $(pub, MK) \leftarrow FIBS.Setup(N, p, 1, \hat{e}, G, G_T)$ *output* (pub, MK) End	**Algorithm** 6. $IBS.Extract(pub, MK, ID)$ Begin $\omega \leftarrow \{ID\}$ $K_{ID} \leftarrow FIBS.Extract(pub, MK, \omega)$ *output* K_{ID} End
Algorithm 7. $IBS.Sign(pub, K_{ID}, M)$ Begin $(S_1, S_2, S_3) \leftarrow FIBS.Sign(pub, K_{ID}, M)$ *output* $S = (S_1, S_2, S_3)$ End	**Algorithm** 8. $IBS.Verify(pub, ID', S, M)$ Begin $\omega' \leftarrow ID'$; $b \leftarrow FIBS.Verify(pub, \omega', S, M)$ *output* b End

FIGURE 5. Algorithms of the IBS scheme.

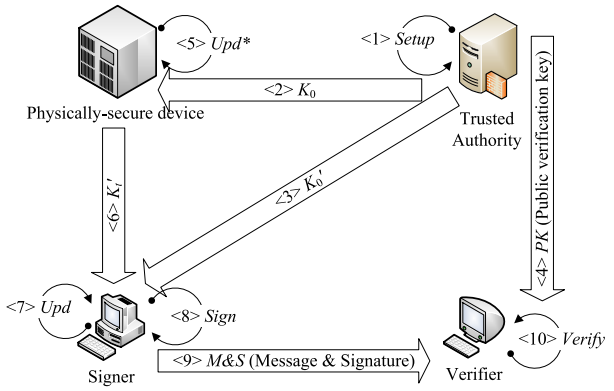

FIGURE 6. The working principle of a KIS scheme.

means a scheme can update SK_i to SK_j in one "step" for any i and j. Details of the two terminologies can be found in [23] and [24], respectively. Note that the signing key of the current period is not used in the update algorithms (i.e., the device key-update and user key-update algorithms), so we omit the current signing key in the list of parameters for the update algorithms for simplicity. The KIS scheme consists of five algorithms, i.e., Setup, Upd, Upd∗, Sign, and Verify. The usage of these algorithms is illustrated in Fig. 6.

1. Setup (Algorithm 9): Suppose that $T \in \mathbb{N}$ is the number of time periods. The algorithm is implemented by calling the *Setup* algorithm of the FIBS scheme. The algorithm proceeds as follows and outputs a public key PK (a tuple also includes the system parameters and public values) and the corresponding master secret key SK. Note that system parameters and public values are also stored in the public key PK, so we use PK as a part of the input of the key-update algorithms and signing algorithm of the scheme.
2. Device key-update $Upd*$ (Algorithm 10): In fact, the algorithm can directly generate the signing key for a user of an arbitrary time period $t \leq T$. The algorithm takes PK, SK, and t as input. It outputs a corresponding "partial" signing key K'_t for time period t. Note that the value of the partial key equals that of the corresponding secret key in this scheme, as found in the user key-update algorithm below.
3. User key-update Upd (Algorithm 11): This algorithm only serves to fulfill the syntax of a KIS scheme. The algorithm directly generates the signing key for a user for an arbitrary time period $t \leq T$. The algorithm takes PK and K'_t as input. It outputs a corresponding private signing key K_t.
4. Sign (Algorithm 12): To sign a message $M = (\mu_1, \cdots, \mu_m) \in \{0, 1\}^m$, the signing algorithm takes PK, the private signing key K_t for time period t, and M as input. It outputs a signature S.
5. Verify (Algorithm 13): To verify a signature S against a time period t', the verification algorithm takes PK, t', S, and a message M as input, and outputs 1 or 0 if the signature is valid or invalid, respectively.

The descriptions of these algorithms are given in Fig. 7.

F. GHE SCHEME

The GHE scheme consists of three algorithms, i.e., Algorithm 14, *EKGen* (encryption key generation algorithm), Algorithm 15, *Enc* (encryption algorithm), and Algorithm 16, *Dec* (decryption algorithm). l is a parameter used to denote the number of elements in the cryptographic key. For a given l, we write the scheme as $GHE_{[l]}$. In fact, $GHE_{[1]}$ is the ElGamal encryption scheme [25], and $GHE_{[2]}$ is the linear encryption (LE) scheme introduced in [26]. Because the above three signature schemes require that the decisional Diffie–Hellman (i.e., 1-linear) problem can be efficiently solved, we suggest that $l \geq 2$ in this paper and assume that the l-linear problem is hard when $l \geq 2$. In the rest of this paper, we omit subscript l if it is not necessary. *EKGen* takes the public parameters *pub* as input and outputs a key pair (PK_e, SK_e). *Enc* takes the public encryption key PK_e, and a (encoded) plaintext $M \in G$ as input. The public parameters *pub* are used as default system settings. *Enc* encrypts M and then outputs the ciphertext C. *Dec* decrypts the ciphertext into a plaintext. The three algorithms of $GHE_{[l]}$ are described in Fig. 8.

Remark 1: For simplicity, $GHE_{[l]}.Enc_{PK_e}(\cdot)$ is used to denote encryption with a encryption key PK_e.

Algorithm 9. $KIS.Setup\left(T, p, \hat{e}, G, G_T\right)$
Begin
$(PK, SK) \leftarrow FIBS.Setup\left(T, p, 1, \hat{e}, G, G_T\right)$; *output* (PK, SK)
End

Algorithm 10. $KIS.Upd*(PK, SK, t)$
Begin
$\omega \leftarrow \{t\}$; $K_t' \leftarrow FIBS.Extract(PK, SK, \omega)$
output K_t'
End

Algorithm 11. $KIS.Upd(PK, K_t')$
Begin
output $K_t = K_t'$
End

Algorithm 12. $KIS.Sign(PK, K_t, M)$
Begin
$(S_1, S_2, S_3) \leftarrow FIBS.Sign(PK, K_t, M)$
output $S = (S_1, S_2, S_3)$
End

Algorithm 13. $KIS.Verify(PK, t', S, M)$
Begin
$\omega' \leftarrow \{t'\}$
$FIBS.Verify(PK, \omega', S, M)$
End

FIGURE 7. Algorithms of the KIS scheme.

Algorithm 14. $GHE_{[l]}.EKGen(pub)$
Begin
$\left(n, p, d, \hat{e}, G, G_T, PP\right) \leftarrow pub$; $(g, g_1, g_2, t_1, \cdots, t_{n+1}, v', v_1, \cdots, v_m, A) \leftarrow PP$
For $(k = 1$ to $l)$: $\{SK_{e,k} \xleftarrow{\$} Z_p$; $PK_{e,j} \leftarrow g^{SK_{e,j}}\}$
$PK_e \leftarrow (PK_{e,1}, \cdots, PK_{e,l}), SK_e \leftarrow (SK_{e,1}, \cdots, SK_{e,l})$; *output* (PK_e, SK_e)
End

Algorithm 15. $GHE_{[l]}.Enc_{PK_e}(M)$
Begin
$\left(n, p, d, \hat{e}, G, G_T, PP\right) \leftarrow pub$
$(g, g_1, g_2, t_1, \cdots, t_{n+1}, v', v_1, \cdots, v_m, A) \leftarrow PP$
$x_1, \cdots, x_l \xleftarrow{\$} Z_p$
$(PK_{e,1}, \cdots, PK_{e,l}) \leftarrow PK_e$
$C \leftarrow \left(PK_{e,1}^{x_1}, \cdots, PK_{e,l}^{x_l}, g^{\sum_{k=1}^{l} x_k} \cdot M\right)$
output C
End

Algorithm 16. $GHE_{[l]}.Dec_{SK_e}(C)$
Begin
$\left(n, p, d, \hat{e}, G, G_T, PP\right) \leftarrow pub$
$(C_1, \cdots, C_l, C_{l+1}) \leftarrow C$
$(SK_{e,1}, \cdots, SK_{e,l}) \leftarrow SK_e$
$M \leftarrow C_{l+1} \cdot \left(\prod_{k=1}^{l} C_k^{\frac{1}{SK_{e,k}}}\right)^{-1}$
output M
End

FIGURE 8. Algorithms of the GHE scheme.

Similarly, $GHE_{[l]}.Dec_{SK_e}(\cdot)$ is used to denote decryption with a decryption key SK_e.

G. ENCRYPTED SIGNATURE FUNCTIONS

In this section, we introduce three obfuscatable encrypted signature functions with regard to the above three digital signature schemes. The three functions are the EIBS, EKIS, and EFIBS. Let $* \in \{IB, KI, FIB\}$. The encrypted signature generation function $E*S_{pub,sk,PK_e}$ (Algorithm 17) works as shown in Fig. 9, with respect to a tuple *pub* that consists of public values and system parameters, i.e., the first element of the output of $*S.Setup$, a signing key *sk* of the $*S$ scheme, an encryption key PK_e of the GHE scheme.

A typical application scenario of the function is illustrated in Fig. 10. This workflow can be further adapted into various security protocols because "sign and encrypt" is a widely used composition of asymmetric cryptographic functionalities.

Algorithm 17. $E*S_{pub,sk,PK_e}(M)$
Begin
IF$(M = NULL)$ *output* (pub, PK_e)
Else $(S_1, S_2, S_3) \leftarrow *S.Sign(pub, sk, M)$
output : $\left(\{GHE.Enc_{PK_e}(S_1^{(i)}) \mid i \in \omega\}, \{GHE.Enc_{PK_e}(S_2^{(i)}) \mid i \in \omega\}, \{GHE.Enc_{PK_e}(S_3^{(i)}) \mid i \in \omega\}\right)$
End

FIGURE 9. E*S function.

H. OBFUSCATOR

A specialized obfuscator for the EFIBS function is presented in this section. According to the consistency of parameters

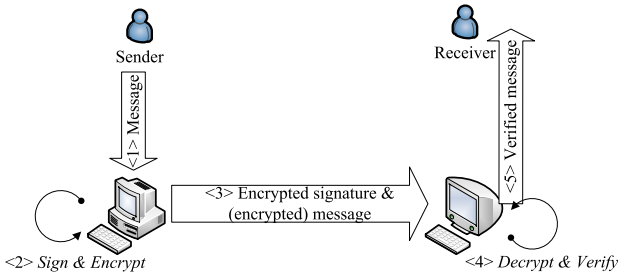

FIGURE 10. Sign-then-encrypt (encrypted signing) and decrypt-then-verify.

and flows of corresponding algorithms of the proposed IBS and KIS schemes, it is clear that an obfuscator for the EFIBS function is also capable of obfuscating the EIBS and EKIS functions.

We propose an obfuscator Obf_{EFIBS} for C_{pub,sk,PK_e} that implements the EFIBS function $EFIBS_{pub,sk,PK_e}$ shown in Fig. 11 as pseudo code. Intuitively, the obfuscator utilizes a special property of the carefully designed EFIBS function, i.e., the encryption applied on the signature and the signing key is interchangeable. Hence, we can use the encrypted signing key (in the obfuscated implementation) to generate a special signature that is the same as the encrypted signature.

The usage of the proposed obfuscator is shown in Fig. 12. The obfuscator takes an implementation of the EFIBS function C_{pub,sk,PK_e} as input. It outputs an obfuscated implementation R_{pub,z,PK_e} preserving the functionality of C_{pub,sk,PK_e}. The obfuscated implementation R_{pub,z,PK_e} generates encrypted signatures just as C_{pub,sk,PK_e} does. However, a hacker cannot extract the signing key from R_{pub,z,PK_e} as in the original implementation C_{pub,sk,PK_e} because the key is

Algorithm 18. $\text{Obf}_{EFIDS}\left(C_{pub,sk,PK_e}\right)$

Begin

$\left(PK_{e,1},\cdots,PK_{e,l}\right)\leftarrow PK_e;\ \left(\left\{K_1^{(i)}\middle|i\in\omega\right\},\left\{K_2^{(i)}\middle|i\in\omega\right\}\right)\leftarrow sk;\ \left(n,p,d,\hat{e},G,G_T,PP\right)\leftarrow pub$

For each $i\in\omega$ Do

$x_{i,1},\cdots,x_{i,l}\xleftarrow{\$}Z_p;\ y_{i,1},\cdots,y_{i,l}\xleftarrow{\$}Z_p;\ \overline{K_1^{(i)}}\leftarrow g^{\sum_{k=1}^{l}x_{i,k}}\cdot K_1^{(i)};\overline{K_2^{(i)}}\leftarrow g^{\sum_{k=1}^{l}y_{i,k}}\cdot K_2^{(i)}$

For $(k=1$ to $l)$ Do

$CX_{i,k}\leftarrow PK_{e,k}^{\ x_{i,k}};CY_{i,k}\leftarrow PK_{e,k}^{\ y_{i,k}}$

EndFor

$CX_i\leftarrow(CX_{i,1},\cdots,CX_{i,l});CY_i\leftarrow(CY_{i,1},\cdots,CY_{i,l})$

EndFor

$z\leftarrow\left\{\left(CX_i,CY_i,\overline{K_1^{(i)}},\overline{K_2^{(i)}}\right)\middle|i\in\omega\right\}$

Construct and output an obfuscated implementation R_{pub,z,PK_e} that works as follows on an input message M:

IF$(M=NULL)$

output $\left(pub,PK_e\right)$

Else

$\left(\mu_1,\cdots,\mu_m\right)\leftarrow M;\ \left(PK_{e,1},\cdots,PK_{e,l}\right)\leftarrow PK_e$

$\left\{\left(CX_i,CY_i,\overline{K_1^{(i)}},\overline{K_2^{(i)}}\right)\middle|i\in\omega\right\}\leftarrow z$

$\left(n,\hat{e},d,G,G_T,PP\right)\leftarrow pub$

For each $i\in\omega$ Do

$s_i\xleftarrow{\$}Z_p;$

$u_1^{(i,1)},\cdots,u_1^{(i,l)}\xleftarrow{\$}Z_p;u_2^{(i,1)},\cdots,u_2^{(i,l)}\xleftarrow{\$}Z_p;u_3^{(i,1)},\cdots,u_3^{(i,l)}\xleftarrow{\$}Z_p$

$\overline{\sigma_1^{(i)}}\leftarrow g^{\sum_{k=1}^{l}u_1^{(i,k)}}\cdot\overline{K_1^{(i)}}\cdot\left(v'\prod_{j=1}^{m}v_j^{\mu_j}\right)^{s_i};\overline{\sigma_2^{(i)}}\leftarrow g^{\sum_{k=1}^{l}u_2^{(i,k)}}\cdot\overline{K_2^{(i)}};\overline{\sigma_3^{(i)}}\leftarrow g^{-s_i+\sum_{k=1}^{l}u_3^{(i,k)}}$

$\left(CX_{i,1},\cdots,CX_{i,l}\right)\leftarrow CX_i;\left(CY_{i,1},\cdots,CY_{i,l}\right)\leftarrow CY_i$

For $(k=1$ to $l)$ Do

$U_1^{(i,k)}\leftarrow CX_{i,k}\cdot PK_{e,k}^{\ u_1^{(i,k)}};U_2^{(i,k)}\leftarrow CY_{i,k}\cdot PK_{e,k}^{\ u_2^{(i,k)}};U_3^{(i,k)}\leftarrow PK_{e,k}^{\ u_3^{(i,k)}}$

EndFor

$U_1^{(i)}\leftarrow\left(U_1^{(i,1)},\cdots,U_1^{(i,l)}\right);U_2^{(i)}\leftarrow\left(U_2^{(i,1)},\cdots,U_2^{(i,l)}\right);U_3^{(i)}\leftarrow\left(U_3^{(i,1)},\cdots,U_3^{(i,l)}\right)$

EndFor

$\overline{\sigma_1}\leftarrow\left\{\left(U_1^{(i)},\overline{\sigma_1^{(i)}}\right)\middle|i\in\omega\right\};\overline{\sigma_2}\leftarrow\left\{\left(U_2^{(i)},\overline{\sigma_2^{(i)}}\right)\middle|i\in\omega\right\};\overline{\sigma_3}\leftarrow\left\{\left(U_3^{(i)},\overline{\sigma_3^{(i)}}\right)\middle|i\in\omega\right\}$

output $\left(\overline{\sigma_1},\overline{\sigma_2},\overline{\sigma_3}\right)$

End

FIGURE 11. Obfuscator (pseudo code).

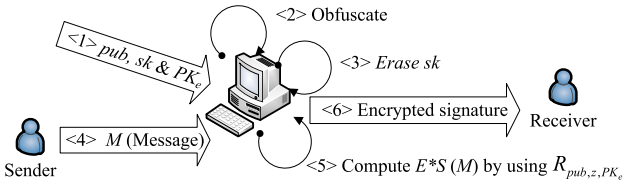

FIGURE 12. Obfuscation.

skillfully hidden in R_{pub,z,PK_e}. Note that it is not necessary to feed the whole implementation C_{pub,sk,PK_e} into the obfuscator in practice. A set of arguments that consists of pub, sk, and PK_e is sufficient for generating an obfuscated implementation.

III. THEORETICAL ANALYSIS OF THE OBFUSCATOR

In this section, a theoretical analysis of the efficiency, correctness, and security of our proposed obfuscator is provided. Because the EIBS and EKIS functions are in fact specific instances of the EFIBS function, we only focus on the case of EFIBS.

A. EFFICIENCY ANALYSIS

The numbers of various operations required to generate/run an obfuscated implementation are listed in TABLE 2. In this table, "Rand" denotes the operation that randomly selects element in Z_p, "Add" denotes addition in Z_p, "Mult" and "Exp" denote multiplication and exponentiation in G, respectively, and "Neg" denotes an operation that generates an additive inverse in Z_p. It is shown in the table that the obfuscator and obfuscated implementation have high efficiency in general because all the operations can be performed efficiently. Furthermore, in concrete applications, usually a small value of l, such as 2 or 3, are used.

TABLE 2. Efficiency of functions and algorithms (as the number of operations).

Operation		Algorithm 17. $E*S_{pub,sk,PK_e}$	Algorithm 18. Obf_{EFIDS}	Obfuscated implementation R_{pub,z,PK_e}
In Z_p	Rand	$\lvert\omega\rvert\cdot(3l+1)$	$2\cdot\lvert\omega\rvert\cdot l$	$\lvert\omega\rvert\cdot(3l+1)$
	Add	$3\cdot\lvert\omega\rvert\cdot(l-1)$	$2\cdot(\lvert\omega\rvert-1)\cdot l$	$\lvert\omega\rvert\cdot(3l-2)$
	Neg	$\lvert\omega\rvert$		$\lvert\omega\rvert$
In G	Mult	$\lvert\omega\rvert\cdot(m+4)+1$	$2\cdot\lvert\omega\rvert$	$\lvert\omega\rvert\cdot(2\cdot l+m+2)$
	Exp	$\lvert\omega\rvert\cdot(3\cdot l+4+m)$ *	$\lvert\omega\rvert\cdot(2+l)$	$\lvert\omega\rvert\cdot(3\cdot l+4+m)$ *

$*$ Note that in practice, the "m" in the last row should be 0 because m exponentiation operations in computing, i.e., $\prod_{j=1}^{m} v_j^{\mu_j}$, can be replaced by $\prod_{j=1}^{m} x_j$, where $x_j = v_j$ if $\mu_j = 1$, and otherwise $x_j = 1$.

B. PRESERVING FUNCTIONALITY

Let $\mathbb{C} = \{\mathbb{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of probabilistic functions and Obf be an obfuscator. The correctness of the obfuscator requires that, on any $C \in \mathbb{C}_\lambda$, with overwhelming probability, Obf(C) behaves almost identically to C for all inputs. Technically, the property is called "Preserving Functionality," as given by Definition 1 [3], [13].

Definition 1 Preserving Functionality (Correctness): A PPT machine Obf is an obfuscator for $\mathbb{C}$ if there exists a negligible function $negl(\lambda)$ for any input length λ for which the following holds:

$$\forall C \in \mathbb{C}_\lambda, \quad \Pr_{\text{coins of Obf}}\left[\begin{array}{l} C' \leftarrow \mathrm{Obf}(C): \\ \forall x \in \{0,1\}^\lambda, \\ Diff(C(x),\ C'(x)) \le negl(\lambda) \end{array}\right] \ge 1\text{-}negl(\lambda). \tag{2}$$

Note that the probability is taken over the random coins used by Obf, and *Diff* denotes the statistical distance between $C'(x)$ and $C(x)$.

Theorem 1 (Preserving Functionality): Consider any $C_{pub,sk,PK_e} \in C_\lambda$ and let $R_{pub,z,PK_e} = \mathrm{Obf}_{EFIBS}(C_{pub,sk,PK_e})$. On each possible input, the output distributions of C_{pub,sk,PK_e} and R_{pub,z,PK_e} are identical.

Proof: Let $PK_e = (g^a, g^b)$. On an arbitrary non-empty message M, the output of C_{pub,sk,PK_e} is

$$(c_1, c_2, c_3) = \begin{pmatrix} \{GHE_{[l]}.Enc_{PK_e}(S_1^{(i)}) | i \in \omega\}, \\ \{GHE_{[l]}.Enc_{PK_e}(S_2^{(i)}) | i \in \omega\}, \\ \{GHE_{[l]}.Enc_{PK_e}(S_3^{(i)}) | i \in \omega\} \end{pmatrix}, \tag{3}$$

where

$$\begin{aligned} &\left(\{(S_1^{(i)}) | i \in \omega\}, \{(S_2^{(i)}) | i \in \omega\}, \{(S_3^{(i)}) | i \in \omega\}\right) \\ &\quad = (S_1, S_2, S_3) = FIBS.Sign\,(M, K_\omega)\,. \end{aligned} \tag{4}$$

Let

$$K_\omega = \{\{D_i |\, i \in \omega\}, \{d_i |\, i \in \omega\}\} \tag{5}$$

and

$$M = (\mu_1 \ldots \mu_m) \in \{0,1\}^m. \tag{6}$$

For any $i \in \omega$, we have

$$c_1^{(i)} = \begin{pmatrix} PK_{e,1}^{x'_{i,1}}, \cdots, PK_{e,l}^{x'_{i,l}}, \\ g^{\sum_{k=1}^{l} x'_{i,k}} \cdot D_i \cdot \left(v' \cdot \prod_{j=1}^{m} v_j^{\mu_j}\right)^{s'_i} \end{pmatrix}, \tag{7}$$

$$c_2^{(i)} = \left(PK_{e,1}^{x''_{i,1}}, \cdots, PK_{e,l}^{x''_{i,l}}, g^{\sum_{k=1}^{l} x''_{i,k}} \cdot d_i\right), \tag{8}$$

$$c_3^{(i)} = \left(PK_{e,1}^{x'''_{i,1}}, \cdots, PK_{e,l}^{x'''_{i,l}}, g^{-s'_i + \sum_{k=1}^{l} x'''_{i,k}}\right), \tag{9}$$

where $x'_{i,1}, \cdots, x'_{i,l}, x''_{i,1}, \cdots, x''_{i,l}, x'''_{i,1}, \cdots, x'''_{i,l}, s'_i$ are uniformly randomly selected from Z_p.

For the same message, R_{pub,z,PK_e} outputs

$$R_{pub,z,PK_e}(M) = \left(\{\overline{\sigma_{1,i}} | i \in \omega\}, \{\overline{\sigma_{2,i}} | i \in \omega\}, \{\overline{\sigma_{3,i}} | i \in \omega\}\right). \tag{10}$$

Hence, for any $i \in \omega$, we have:

$$\overline{\sigma_{1,i}} = (U_1^{(i)}, \overline{\sigma_1^{(i)}}) = \begin{pmatrix} PK_{e,1}^{x_{i,1}+u_1^{(i,1)}}, \cdots, PK_{e,l}^{x_{i,l}+u_1^{(i,l)}}, \\ g^{\sum_{k=1}^{l}\left(u_1^{(i,k)}+x_{i,k}\right)} \cdot K_1^{(i)} \cdot \left(v' \prod_{j=1}^{m} v_j^{\mu_j}\right)^{s_i} \end{pmatrix}, \tag{11}$$

$$\overline{\sigma_{2,i}} = \left(U_2^{(i)}, \overline{\sigma_2^{(i)}}\right) = \left(\begin{array}{c} PK_{e,1}^{y_{i,1}+u_2^{(i,1)}}, \cdots, PK_{e,l}^{y_{i,l}+u_2^{(i,l)}}, \\ g^{\sum_{k=1}^{l} y_{i,k}+u_2^{(i,k)}} \cdot K_2^{(i)} \end{array}\right), \tag{12}$$

$$\begin{aligned}\overline{\sigma_{3,i}} &= \left(U_3^{(i)}, \overline{\sigma_3^{(i)}}\right) = \left(U_3^{(i,1)}, \cdots, U_3^{(i,1)}, \overline{\sigma_3^{(i)}}\right) \\ &= \left(PK_{e,1}^{u_3^{(i,1)}}, \cdots, PK_{e,l}^{u_3^{(i,l)}}, g^{-s_i+\sum_{k=1}^{l} u_3^{(i,k)}}\right),\end{aligned} \tag{13}$$

where $u_1^{(i,k)}, x_{i,k}, u_2^{(i,k)}, y_{i,k}, u_3^{(i,k)}, s_i, 1 \leq k \leq l$ are uniformly random elements of Z_p.

It is clear that (c_1, c_2, c_3) and the output of $R_{pub,z,PK_e}(M)$ are identically distributed. This completes the proof.

C. ACVBP

In this subsection, we study the security of the proposed obfuscator with respect to the ACVBP. ACVBP was introduced by Hohenberger *et al.* [13], and it was extended to ACVBP with respect to Dependent Oracles (DOs) by Hada [3]. This extension permits distinguishers to have sampling access not only to $\ll C \gg$ but also to a set of oracles that are dependent on C.

Remark 2: For a Turing machine TM, its black-box access to a probabilistic function C can be categorized into "oracle access" and "sampling access." The former means that TM is allowed to set both regular inputs and random inputs. This is denoted by TM^C, as is conventional. Sampling access means that TM is only permitted to set the regular input. This is denoted by $TM^{\ll C \gg}$.

Definition 2: An obfuscator Obf for $\mathbb{C}$ satisfies the ACVBP with respect to DOs $\mathcal{T}$ if the following condition holds: There exists a simulator S (PPT oracle machine) such that, for every PPT oracle machine $\mathcal{D}$ (distinguisher), every polynomial f, all sufficiently large $\lambda \in \mathbb{N}$, and every $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\left|\begin{array}{l} \Pr\left[\begin{array}{l} C \leftarrow \mathbb{C}_\lambda; C' \leftarrow \mathrm{Obf}(C); \\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C) \gg}(C', z) \end{array} : b = 1\right] \\ -\Pr\left[\begin{array}{l} C \leftarrow \mathbb{C}_\lambda; C'' \leftarrow S^{\ll C \gg}(1^\lambda, z); \\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C) \gg}(C'', z) \end{array} : b = 1\right] \end{array}\right| < \frac{1}{f(\lambda)}, \tag{14}$$

where $\mathcal{D}^{\ll C, \mathcal{T}(C) \gg}$ means that, in addition to C, $\mathcal{D}$ also has sampling access to all oracles in the set $\mathcal{T}(C)$.

When obfuscating normal encrypted signature functions, $\mathcal{T}(C)$ is commonly assigned to the signing function, such as in [3]–[5], [14], [16], and [17]. However, we should consider collision attacks from some corrupted users against the proposed obfuscator. Therefore, we suppose that an adversary against the obfuscator is capable of obtaining the signing key of corrupted users, excepting the user who generates the obfuscated implementation as a challenge, that is, the adversary can access the *Extract* oracle on any corrupted user's identity. Because there are certain restrictions on the contents of queries, the set of oracle-restrictions pairs dependent on C is defined as $\mathcal{R}(C)$. For instance, in this paper, we define that $\mathcal{R}(C) = \{(FIBS.Extract_{MK}, id{\approx}\mathrm{ID})\}$, where $id{\approx}\mathrm{ID}$ denotes $|\mathrm{ID} \cap id| \geq d$. Conventionally, we omit the braces in the description of the set when $|\mathcal{R}(C)| = 1$, and the restrictions are denoted as superscripts, e.g., $FIBS.Extract_{MK}^{[id{\approx}\mathrm{ID}]}$.

Accordingly, we use the extended version of ACVBP with respect to DOs, i.e., ACVBP with respect to DOs and Restricted Dependent Oracles (RDOs), which is proposed in [27] as follows.

Definition 3: An obfuscator Obf for $\mathbb{C}$ satisfies the ACVBP with respect to DO set $\mathcal{T}$ and RDO set $\mathcal{R}$ if and only if there exists a simulator S (PPT oracle machine) such that, for any distinguisher $\mathcal{D}$ (PPT oracle machine), all sufficiently large numbers $\lambda \in \mathbb{N}$, any polynomial f, and every $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\left|\begin{array}{l} \Pr\left[\begin{array}{l} C \leftarrow \mathbb{C}_\lambda; C' \leftarrow \mathrm{Obf}(C); \\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C) \gg}(C', z) \end{array} : b = 1\right] \\ -\Pr\left[\begin{array}{l} C \leftarrow \mathbb{C}_\lambda; C'' \leftarrow S^{\ll C \gg}(1^\lambda, z); \\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C) \gg}(C'', z) \end{array} : b = 1\right] \end{array}\right| < \frac{1}{f(\lambda)}, \tag{15}$$

where $\mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C) \gg}$ means that $\mathcal{D}$ has sampling access to C, all the oracles that are contained in $\mathcal{T}(C)$, and the restricted oracles that are contained in $\mathcal{R}(C)$.

For ID-based crypto-systems, ACVBP with respect to DOs and RDOs is stronger than ACVBP with respect to DOs because the adversary's extra accesses to RDOs are permitted in the former case. Similarly, ACVBP with respect to DOs is stronger than the standard ACVBP. In **Theorem 2**, we prove that the proposed obfuscator satisfies ACVBP with respect to DOs and RDOs, therefore no adversary is capable of distinguishing a real obfuscated implementation and the fake implementation that is generated by a simulator who does not know the signing key. Clearly, the theorem implies that a hacker cannot extract the signing key from an obfuscated implementation. Otherwise, the hacker could use the key to distinguish the real and fake implementations.

Theorem 2: The obfuscator Obf_{EFIBS} for the proposed EFIBS functionality satisfies ACVBP with respect to DO $\mathcal{T}(C) = FIBS.Sign_{sk_{\mathrm{ID}}}$ and RDO $\mathcal{R}(C) = FIBS.Extract_{MK}^{[id{\approx}\mathrm{ID}]}$, where $C = EFIBS_{pub,sk_{\mathrm{ID}},PK_e}$.

Proof: Suppose that $\mathrm{ID} = \omega$. The probabilities that a distinguisher $\mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C) \gg}$ outputs 1 given the real and simulated distributions are defined respectively as (16) and (17), as shown at the bottom of the next page, where the simulator *Sim* works as shown in Fig. 13.

For contradiction, assume that there exists a distinguisher $\mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C) \gg}$ that can distinguish between C' and C'' with a non-negligible probability. Without loss of generality, we suppose that $\Pr_{Nick} - \Pr_{Junk} = \delta > 0$.

We then construct a pair of adversaries, $(\mathcal{A}, \mathcal{B})$, which breaks the ciphertext indistinguishability of the GHE scheme as follows.

Using algorithm $\mathcal{A}.Init$ and playing the security game with $\mathcal{B}$ and $\mathcal{D}$, as shown in Fig. 14, $\mathcal{A}$ produces a pair of plaintext $\langle sk_{\mathrm{ID}}, sk' \rangle$, public settings, and system parameters of the GHE scheme. Suppose that the system parameters were generated honestly, that is, the adversary $\mathcal{A}$ does not intentionally

$Sim^{\ll C(\ )\gg}(\ )$
$(pub, PK_e) \leftarrow C(\ );\ (n, p, d, \hat{e}, G, G_T, PP) \leftarrow pub;$
$(PK_{e,1}, \cdots, PK_{e,l}) \leftarrow PK_e$
For each $i \in \omega$ Do
  $x_{i,1}, \cdots, x_{i,l} \xleftarrow{\$} Z_p; y_{i,1}, \cdots, y_{i,l} \xleftarrow{\$} Z_p$
  $K_1^{(i)\prime}, K_2^{(i)\prime} \xleftarrow{\$} G; K_1^{(i)\prime} \leftarrow g^{\sum_{k=1}^{l} x_{i,k}} \cdot K_1^{(i)\prime}$
  $K_2^{(i)\prime} \leftarrow g^{\sum_{k=1}^{l} y_{i,k}} \cdot K_2^{(i)\prime}$
  For ($k = 1$ to l) Do
    $CX_{i,k} \leftarrow PK_{e,k}{}^{x_{i,k}}; CY_{i,k} \leftarrow PK_{e,k}{}^{y_{i,k}}$
  EndFor
  $CX_i \leftarrow (CX_{i,1}, \cdots, CX_{i,l}); CY_i \leftarrow (CY_{i,1}, \cdots, CY_{i,l})$
EndFor
$Junk \leftarrow \left\{ \left(CX_i, CY_i, \overline{K_1^{(i)\prime}}, \overline{K_2^{(i)\prime}} \right) \middle| i \in \omega \right\}$
output $R_{pub,Junk,PK_e}()$ that works the same as $R_{pub,z,PK_e}()$

FIGURE 13. The simulator.

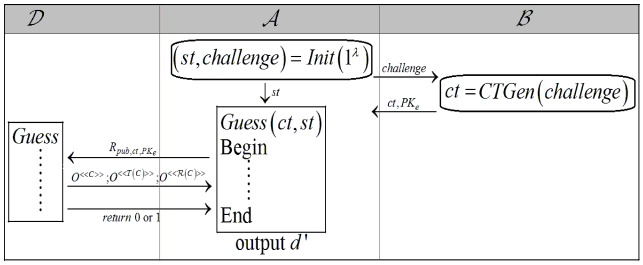

FIGURE 14. Security game.

set any "backdoor" in these parameters. Alternatively, we can specify that the parameters are setup by a trusted third party. Note that after the adversary $\mathcal{A}$ is initialized, ID, sk_{ID}, and *pub* are private "member variables" of $\mathcal{A}$. Algorithms $\mathcal{A}.OC$, $\mathcal{A}.OS$, and $\mathcal{A}.OE$ are used to reply on the $O^{\ll C\gg}$, $O^{\ll T(C)\gg}$, and $O^{\ll R(C)\gg}$ queries from $\mathcal{D}$, respectively. Further, $\mathcal{B}.CTGen$ is used to generate the challenge ciphertext and $\mathcal{A}.Guess$ is used to "guess" the value of d in $\mathcal{B}.CTGen$. The descriptions of the algorithms are shown in Fig. 14. Note that these algorithms only serve for this security proof and will never be put into practical use; hence, we ignore their numbering.

Note that if the random coin $d = 1$, then R_{pub,ct,PK_e} equals $R_{pub,Junk,PK_e}$ generated by the simulator S, otherwise, R_{pub,ct,PK_e} is a valid output of Obf_{EGS}.

The two probabilities $\Pr[d = 0 \wedge d' = 0]$ and $\Pr[d = 1 \wedge d' = 1]$ are calculated as follows, where $\mathcal{D}^*$ denotes $\mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C)\gg}$.

$$\begin{aligned}
&\Pr[d = 0 \wedge d' = 0] \\
&= \Pr[d = 0] \cdot \Pr\left[1 = \mathcal{D}^*(R_{pub,ct,PK_e}) | d = 0\right] \\
&\quad + \Pr[d = 0] \cdot \left((1/2) \cdot \Pr\left[1 \neq \mathcal{D}^*(R_{pub,ct,PK_e}) | d = 0\right]\right) \\
&= (1/2) \cdot \Pr_{Nice} + (1/2) \cdot \left((1/2) \cdot (1 - \Pr_{Nice})\right) \\
&= \left(\Pr_{Nice} + 1\right)/4 \qquad (18)
\end{aligned}$$

$$\begin{aligned}
&\Pr\left[d = 1 \wedge d' = 1\right] \\
&= \Pr\left[d = 1\right] \cdot \left((1/2) \cdot \Pr\left[1 \neq \mathcal{D}^*\left(R_{pub,ct,PK_e}\right) | d = 1\right]\right) \\
&= (1/2) \cdot (1/2) \cdot \left(1 - \Pr_{Junk}\right) \\
&= \left(1 - \Pr_{Junk}\right)/4 \qquad (19)
\end{aligned}$$

Finally, the advantage of $\mathcal{A}$ in the chosen-plaintext attack (CPA) against the GHE scheme is

$$\begin{aligned}
Adv_{\mathcal{A}}^{IND-CPA,GHE} &= 2 \cdot \Pr\left[d' = d\right] - 1 \\
&= 2 \cdot \left(\Pr\left[d' = 0 \wedge d = 0\right] + \Pr\left[d' = 1 \wedge d = 1\right]\right) - 1 \\
&= 2 \cdot \left(\frac{\Pr_{Nice} + 1}{4} + \frac{1 - \Pr_{Junk}}{4}\right) - 1 \\
&= 2 \cdot \left(\frac{1}{2} + \frac{\Pr_{Nice} - \Pr_{Junk}}{4}\right) - 1 \\
&= \frac{\Pr_{Nice} - \Pr_{Junk}}{4} = \frac{\delta}{2}. \qquad (20)
\end{aligned}$$

If $\delta = \Pr_{Nick} - \Pr_{Junk}$ is non-negligible, so is the advantage $Adv_{\mathcal{A}}^{IND-CPA,GHE}$. This contradicts the indistinguishability of ciphertexts against CPAs on the GHE scheme based on the l-linear assumption. This completes the proof.

IV. EXPERIMENTAL RESULTS

The proposed algorithms were implemented in Java by applying the Java Pairing-Based Cryptography Library [24]. The performance was tested on a PC with an Intel i7 CPU (@3.5 GHz). To report the experimental results compactly, we merge algorithms that are almost the same in software implementation, and consequently with almost the same efficiency, into a group. The seven algorithm groups are listed

$$\Pr_{Nick} = \Pr\left[\begin{array}{l} (pub, MK) \leftarrow FIBS.Setup\left(1^\lambda\right) \\ (PK_e, SK_e) \xleftarrow{\$} GHE.EKGen\ (pub) \\ sk_{\mathrm{ID}} = (K_1, K_2) \xleftarrow{\$} FIBS.Extract(pub, MK, \omega) \\ C' \leftarrow \mathrm{Obf}_{EFIBS}\ (C);\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C)\gg}\left(C'\right) \end{array} : b = 1\right], \qquad (16)$$

$$\Pr_{Junk} = \Pr\left[\begin{array}{l} (pub, MK) \leftarrow FIBS.Setup\left(1^\lambda\right) \\ (PK_e, SK_e) \xleftarrow{\$} GHE.EKGen\ (pub) \\ sk_{\mathrm{ID}} = (K_1, K_2) \xleftarrow{\$} FIBS.Extract(pub, MK, \omega) \\ C'' \leftarrow Sim^{\ll C()\gg}();\ b \leftarrow \mathcal{D}^{\ll C, \mathcal{T}(C), \mathcal{R}(C)\gg}\left(C''\right) \end{array} : b = 1\right], \qquad (17)$$

Algorithm $\mathcal{A}.Init(1^{\lambda})$
Begin
Generate system parameters $n, p, d, \hat{e}, G, G_T$
$(pub, MK) \leftarrow Setup(n, p, d, \hat{e}, G, G_T)$
$sk_{\mathrm{ID}} = K_{\omega} \xleftarrow{\$} Extract(pub, MK, \mathrm{ID})$
$sk' \xleftarrow{\$} G^{|\omega|}$
$challenge \leftarrow (sk_{\mathrm{ID}}, sk', n, \hat{e}, G, G_T, g)$
$st \leftarrow (pub, MK)$
output $(st, challenge)$
End

Algorithm $\mathcal{A}.OC(M)$
/* Compute $EFIBS_{pub, sk_{\mathrm{ID}}, PK_e}(M)$ */
Begin
output $C(M)$
End

Algorithm $\mathcal{A}.OS(M)$
Begin
output $Sign_{sk_{\mathrm{ID}}}(M)$
End

Algorithm $\mathcal{A}.OE(id)$
Begin
IF $(\mathrm{ID} \approx id)$ *output* $\perp$
Else *output* $Extract(pub, MK, id)$
End

Algorithm $\mathcal{A}.Guess(ct, st)$
Begin
$((n, p, d, \hat{e}, G, G_T, PP), MK) = (pub, MK) \leftarrow st$
Generate C according to the EGS functionality;
Generate R_{pub,ct,PK_e} that works the same as
R_{pub,z,PK_e} that is generated by Obf_{EGS};
IF $\left(1 = \mathcal{D}^{<<C,T(C),\mathcal{R}(C)>>}(R_{pub,ct,PK_e})\right)$ $d' \leftarrow 0$
Else $d' \xleftarrow{\$} \{0,1\}$
output d'
End

Algorithm $\mathcal{B}.CipherTextGen(challenge)$
Begin
$(sk, sk', n, \hat{e}, G, G_T, g) \leftarrow challenge$
$(PK_e, SK_e) \leftarrow GHE.KGen(n, \hat{e}, G, G_T, g)$
$d \xleftarrow{\$} \{0,1\}$
IF $(d = 0)$ $(K_1, K_2) \leftarrow sk$
Else $(K_1, K_2) \leftarrow sk'$
$ct \leftarrow \left(\left\{GHE.Enc_{PK_e}(K_1^{(i)}) \middle| i \in \omega\right\}, \left\{GHE.Enc_{PK_e}(K_2^{(i)}) \middle| i \in \omega\right\}\right)$
output ct, PK_e
End

FIGURE 15. Algorithms used in the security game.

in TABLE 3. The first column presents the unified identities of groups. Note that Alg.∗ (in this section) and Algorithm.∗ (in Section 2) do *not* denote the same object.

Fig. 16 shows that a positive correlation exists between the running time of Alg. 1 and the parameter m. When m is fixed, the running time increases with the value of n in the FIBS scheme (as well as N in the IBS scheme or T in the KIS scheme, where N stands for the number of users in the IBS scheme and T stands for the number of time periods in the KIS scheme). The increasing rate of time cost for Alg. 1, as shown in Fig. 17, converges at around 1.33 when n, N, or T become large (e.g., at 90). Obviously, the increasing rate of time is rarely sensitive to m when n, N, or T are greater than 40. This result indicates that although the running time of the setup algorithm (Alg. 1) increases linearly with the values of N or T, this increase is rather "slow." Therefore, even in an ID-based system with a large number of initial users or a KI system with a large number of time periods, the efficiency of Alg. 1 is still acceptable.

According to the descriptions of Algs. 2–7, parameter n takes little effect in these algorithms. Hence, we set

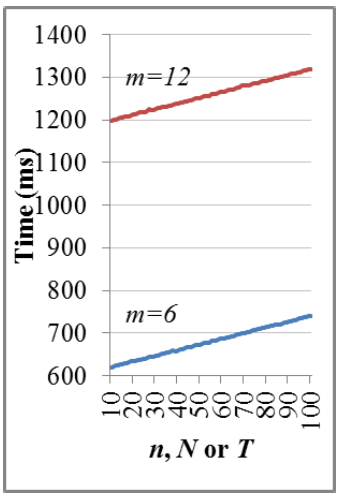

FIGURE 16. Running time of Alg. 1.

$n = 100$ in the experiments on Algs. 2–7 for convenience. In the experiments to test the performance of the IBS and KIS schemes, we set $|\omega| = 1$ and provide the running times of

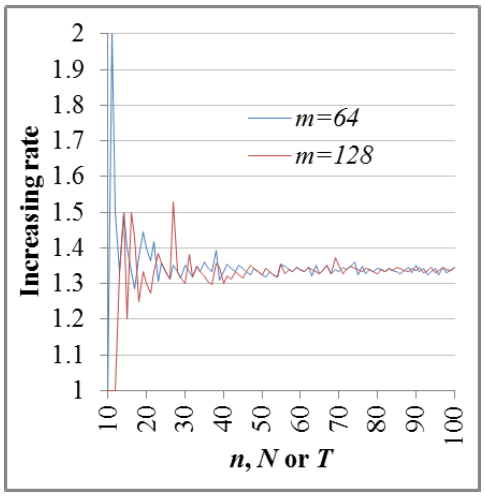

FIGURE 17. Increasing rate of time versus n, N or T of Alg. 1.

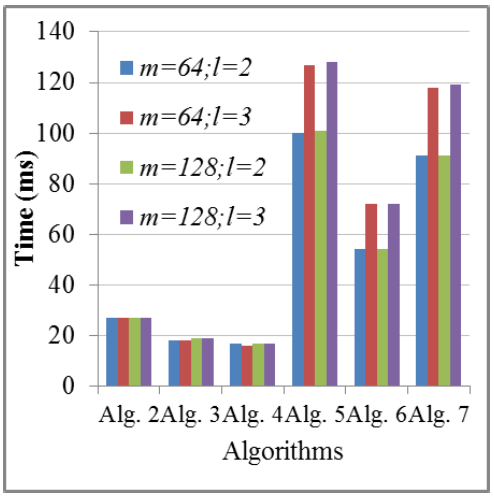

FIGURE 18. Running time of Algs. 2–7.

TABLE 3. Algorithm groups.

No.	Algorithms	Algorithm No (in Section 2)
Alg.1	*FIBS.Setup*; *IBS.Setup*; *KIS.Setup*.	1,5,9
Alg.2	*FIBS.Extract*; *IBS.Extract*; *KIS.Upd* *.	2,6,10
Alg.3	*FIBS.Sign*; *IBS.Sign*; *KIS.Sign*.	3,7,12
Alg.4	*FIBS.Verify*; *IBS.Verify*; *KIS.Verify*.	4,8,13
Alg.5	$EFIBS_{pub,sk,PK_e}$; $EIBS_{pub,sk,PK_e}$; $EKIS_{pub,sk,PK_e}$.	17 (We replace the * in $E*S_{pub,sk,PK_e}$ with " *FIB* ", " *IB* " and " *KI* ")
Alg.6	$\mathrm{Obf}_{EFIDS}\left(C_{pub,sk,PK_e}\right)$	18 (It can also be used to obfuscate EIBS and EKIS)
Alg.7	R_{pub,z,PK_e} (Obfuscated implementation of EFIBS/EIBS/EKIS)	N/A (It is embedded in the description of the obfuscator, i.e., Algorithm 18)

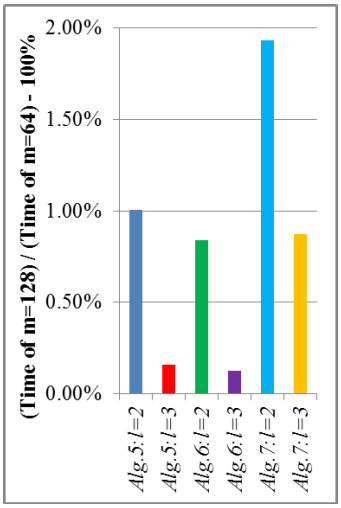

FIGURE 19. The average differences of running time with different m in Algs. 5–7 vary little.

the Algs. 2–7 for the different sets of parameters in Fig. 18. This figure indicates that parameter m has little effect on the performance of these algorithms. Parameter m is important because it denotes the size of the input message and, thereby, we further investigate the little effect of setting different m in Fig. 19. Setting the same parameter l and different $|\omega|$ values ranging from 10 to 100 in Algs. 5–7, the average running times for $m = 64$ and $m = 128$ vary very little, e.g., the running time for $m = 128$, $l = 3$ in Alg. 7 increases less than 0.02 times that with $m = 64$, $l = 3$. Based on this result, we can set m to be a fixed value, e.g., 64 or 128, when studying the performance of Algs. 5–7 in a variety of situations.

We display the running times of different $|\omega|$ for Algs. 5–7 (only in the FIBS scheme) with a fixed m ($m = 128$) in Fig. 20. These results lead to the conclusion that the running time linearly increases with increments of $|\omega|$. Note that $|\omega| = 1$ in both the KIS and IBS schemes, so the two schemes are omitted in the experiment corresponding to Fig. 20.

To compare the efficiency of using Alg. 5 (the original encrypted FIBS generation algorithm) and Alg. 7 (the obfuscated encrypted FIBS generation algorithm), we performed an experiment on their running time. As it is necessary to run the obfuscator (Alg. 6) once to obtain the implementation of Alg. 7, the running time of Alg. 7 should include the running time of Alg. 6 once. Thereby, the ratio of the running time of using Alg. 7 versus that of using Alg. 5 is shown in Fig. 21 with respect to the number of runs, calculated as follows:

$$\frac{\text{Time of run Alg.7} \times \text{times} + \text{Time of run Alg.6 once}}{\text{Time of run Alg.5} \times \text{times}} - 1,$$
$$l = 2, 3, \quad m = 128, \quad \omega = 1, 60. \tag{21}$$

Experimental results show that the ratio decreases while the number of runs increases. When $\omega = 60$, the increment in the running time of using Alg. 5 is larger than that of using Alg. 7, while the number of runs is greater than nine for $l = 3$ and seven for $l = 2$. When $\omega = 1$ (in the case of KIS or IBS), the increment in the running time of using Alg. 5 is larger than that of using Alg. 7, while the number of runs is larger than seven for $l = 3$ and five for $l = 2$. Based on this, we can conclude that the use of the obfuscated implementation (Alg. 7) is more time efficient in most scenarios.

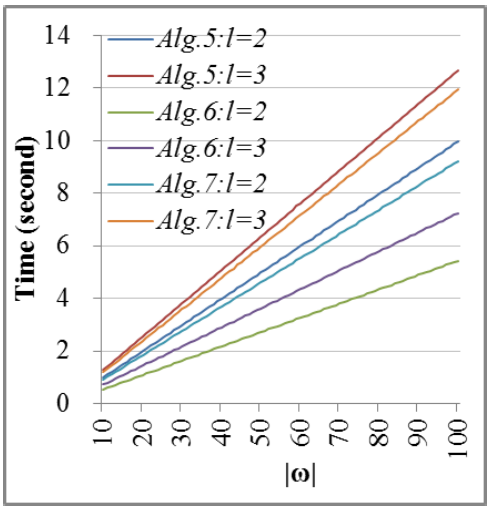

FIGURE 20. The running time of different $|\omega|$ for Algs. 5–7 with fixed m.

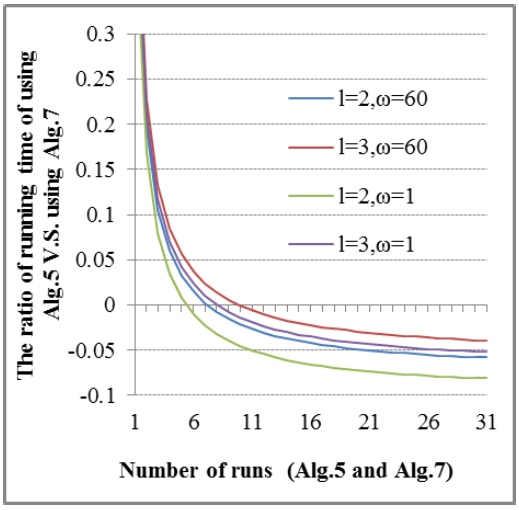

FIGURE 21. Ratio of the running time of using Alg. 7 versus that of using Alg. 5 with respect to the number of runs.

V. RELATED STUDIES AND COMPARISONS

A number of obfuscatable cryptographic functions and corresponding obfuscators for these functions with acceptable computational costs have been presented since 2007. TABLE 4 lists the functions and corresponding obfuscators to the best of our knowledge, where the last line is ours in this paper. As shown in TABLE 4, the proposed obfuscator is the first one for EIBS, EKIS and EFIBS. Furthermore, the proposed obfuscator and encrypted signature functions use distinct building blocks and complexity assumptions that are distinct from all previous studies. Note that an obfuscatable EIBS is sketched out as a by-product of the obfuscatable encrypted group signature and an EKIS is also derived in [27]. However, the authors of [27] did not provide a concrete description of the algorithms and a strict security analysis. Furthermore, even if an EIBS/EKIS is implemented following the idea in [27], compared with the proposed obfuscatable EIBS/EKIS, it would be much less efficient because composite order groups are used.

TABLE 4. A comparison of related studies.

Cryptographic function(s)	Year	Complexity assumptions[4]	Building block(s) or base scheme(s)
Re-encryption[13]	2007	DLIN	The linear encryption (LE) scheme[26].
Encrypted signature (ES)[3]	2010	DLIN,DBDH	The LE scheme[26], and Water's signature scheme[34].
Encrypted verifiable ES [4]	2011	DLIN,DBDH, Exponent l-weak DH	The LE scheme[26], and Water's signature scheme[34].
Functional re-encryption[15]	2012	SXDH	The re-encryption scheme[13].
Oblivious signature [14]	2012	DLIN,SDHI	The LE scheme[26], Water's signature [34], and Pedersen's VSS protocol[35].
Encrypted proxy signatures[17]	2013	DBDH,DLIN	The signature scheme in [36], and the LE scheme[26].
Encrypted blind signature[16]	2013	DH	Blind Schnorr's signature, and the LE scheme[26].
Re-encryption with keyword Search[18]	2013	DBDH	ElGamal encryption (a modified version).
Multi-use re-encryption[20]	2014	MDDH,DBDH	ElGamal encryption (a modified version).
Encrypted verifiably ES[5]	2014	CDH,DLIN,AgExt	The LE scheme[26], and Water's signature scheme[34].
Encrypted group signatures[27]	2015	SD,DLIN,HSDH	The LE scheme[26], and the group signature scheme [37].
Multi-recipient re-encryption[38]	2015	DLIN	The re-encryption scheme [13] and randomness reusing.
EFIBS EIBS EKIS	This paper	l-linear ($l \geq 2$)	The FIBS scheme[21], and the GHE scheme[22].

Moreover, as we mentioned in the introduction, under security definitions that are weaker than the ACVBP (e.g., indistinguishable obfuscation [28]) or using non-standard security model (e.g., generic graded encoding model [29]), a variety of candidates of general-purpose obfuscators [28]–[30] have been proposed in recent years. However, there are two issues with these techniques. First, their security has been seriously challenged by recently developed cryptanalysis approaches such as those in [31]–[33]. Second, they have been implemented with heavyweight techniques that incur high computational cost and thus are somewhat impractical nowadays.

In summary, there is still no efficient approach for constructing a general-purpose obfuscator that satisfies ACVBP. Therefore, it is valuable to find obfuscatable EIBS, EKIS, and EFIBS functions as well as to design a corresponding secure and efficient obfuscator.

[4] Acronyms of complexity assumptions in Table 4 are explained as follows.

(i) AgExt: Aggregate Extraction
(ii) CDH: Computational Diffie–Hellman
(iii) DBDH: Decisional Bilinear Diffie–Hellman
(iv) DH: Diffie–Hellman
(v) DLIN: Decisional Linear
(vi) HSDH: Hidden Strong Diffie–Hellman
(vii) SD: Subgroup Decision
(viii) SDHI: Strong Diffie–Hellman Indistinguishability
(ix) SXDH: Symmetric External Diffie–Hellman

VI. CONCLUSIONS

To provide building blocks for security schemes and protocols that are partially running in austere security contexts such as a computing device that is potentially controlled by an attacker, we provided three obfuscatable cryptographic functions, that is, EFIBS, EIBS, and EKIS, and then provided an obfuscator for them. We proved that the proposed obfuscator is correct and secure, that is, the obfuscator preserves their functionalities and satisfies the ACVBP with respect to DOs and RDOs. Furthermore, experimental results show that the proposed algorithms are efficient. In future, we plan to apply these obfuscatable functions in practical security schemes and protocols for validation.

REFERENCES

[1] B. Barak *et al.*, "On the (im)possibility of obfuscating programs," *J. ACM*, vol. 59, no. 2, Apr. 2012, Art. no. 6.
[2] S. Goldwasser and Y. T. Kalai, "On the impossibility of obfuscation with auxiliary input," in *Proc. 46th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2005, pp. 553–562.
[3] S. Hada, "Secure obfuscation for encrypted signatures," in *Advances in Cryptology–EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2010, pp. 92–112.
[4] R. Cheng, B. Zhang, and F. Zhang, "Secure obfuscation of encrypted verifiable encrypted signatures," in *Provable Security*. Berlin, Germany: Springer-Verlag, 2011, pp. 188–203.
[5] R. Nishimaki and K. Xagawa, "Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES," *Designs, Codes Cryptograph.*, vol. 77, no. 1, pp. 61–98, 2015.
[6] Y. Watanabe and J. Shikata, "Identity-based hierarchical key-insulated encryption without random oracles," in *Proc. 19th IACR Int. Conf. Pract. Theory Public-Key Cryptograph.*, Taipei, Taiwan, 2016, pp. 255–279.
[7] J. Yu, R. Hao, H. Zhao, M. Shu, and J. Fan, "IRIBE: Intrusion-resilient identity-based encryption," *Inf. Sci.*, vol. 329, pp. 90–104, Feb. 2016.
[8] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Dept. Comput. Sci., Univ. Auckland, Auckland, New Zealand, Tech. Rep. 148, 1997.
[9] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation—Tools for software protection," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 735–746, Aug. 2002.
[10] B. Barak *et al.*, "On the (im)possibility of obfuscating programs," in *Advances in Cryptology–CRYPTO*. Berlin, Germany: Springer-Verlag, 2001, pp. 1–18.
[11] N. Bitansky and O. Paneth, "On the impossibility of approximate obfuscation and applications to resettable cryptography," in *Proc. 45th Annu. ACM Symp. Theory Comput.*, 2013, pp. 241–250.
[12] S. Goldwasser and G. N. Rothblum, "On best-possible obfuscation," *J. Cryptol.*, vol. 27, no. 3, pp. 480–505, 2014.
[13] S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan, "Securely obfuscating re-encryption," *J. Cryptol.*, vol. 24, no. 4, pp. 694–719, Oct. 2011.
[14] C. Li, Z. Yuan, and M. Mao, "Secure obfuscation of a two-step oblivious signature," in *Network Computing and Information Security*, vol. 345. Berlin, Germany: Springer-Verlag, 2012, pp. 680–688.
[15] N. Chandran, M. Chase, and V. Vaikuntanathan, "Functional re-encryption and collusion-resistant obfuscation," in *Theory of Cryptography*. Berlin, Germany: Springer-Verlag, 2012, pp. 404–421.
[16] X. Feng and Z. Yuan, "A secure obfuscator for encrypted blind signature functionality," in *Network and System Security*. Berlin, Germany: Springer-Verlag, 2014, pp. 311–322.
[17] X. Wei, Z. Yuan, X. Li, X. Feng, and J. Liu, "Secure obfuscation for tightly structure-preserving encrypted proxy signatures," in *Proc. 9th Int. Conf. Comput. Intell. Secur. (CIS)*, 2013, pp. 589–593.
[18] R. Cheng and F. Zhang, "Secure obfuscation of conditional re-encryption with keyword search," in *Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst. (INCoS)*, 2013, pp. 30–37.
[19] N. Chandran, M. Chase, F.-H. Liu, R. Nishimaki, and K. Xagawa, "Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices," in *Public-Key Cryptography–PKC*. Berlin, Germany: Springer-Verlag, 2014, pp. 95–112.
[20] R. Cheng and F. Zhang, "Obfuscation for multi-use re-encryption and its application in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 8, pp. 2170–2190, 2015.
[21] P. Yang, Z. Cao, and X. Dong, "Fuzzy identity based signature with applications to biometric authentication," *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 532–540, 2011.
[22] F. Armknecht, S. Katzenbeisser, and A. Peter, "Group homomorphic encryption: Characterizations, impossibility results, and applications," *Designs, Codes Cryptograph.*, vol. 67, no. 2, pp. 209–232, 2013.
[23] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Public Key Cryptography–PKC*. Berlin, Germany: Springer-Verlag, 2003, pp. 130–144.
[24] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Advances in Cryptology–EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2002, pp. 65–82.
[25] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[26] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology–CRYPTO*. Berlin, Germany: Springer-Verlag, 2004, pp. 41–55.
[27] Y. Shi, Q. Zhao, H. Fan, and Q. Liu, "Secure obfuscation for encrypted group signatures," *PLoS ONE*, vol. 10, no. 7, p. e0131550, 2015.
[28] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci. (Focs)*, Oct. 2013, pp. 40–49.
[29] Z. Brakerski and G. N. Rothblum, "Virtual black-box obfuscation for all circuits via generic graded encoding," in *Theory of Cryptography*. Berlin, Germany: Springer-Verlag, 2014, pp. 1–25.
[30] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai, "Protecting obfuscation against algebraic attacks," in *Advances in Cryptology–EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2014, pp. 221–238.
[31] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé, *Cryptanalysis of the Multilinear Map Over the Integers*. Berlin, Germany: Springer-Verlag, 2015, pp. 3–12.
[32] J.-S. Coron *et al.*, *Zeroizing Without Low-Level Zeroes: New MMAP Attacks and Their Limitations* (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 2015, pp. 247–266.
[33] B. Minaud and P.-A. Fouque, *Cryptanalysis of the New Multilinear Map Over the Integers*. (The International Association for Cryptologic Research (IACR)), USA, 2015.
[34] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology–EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2005, pp. 114–127.
[35] L. Chen and T. P. Pedersen, "New group signature schemes," in *Advances in Cryptology–EUROCRYPT*. Berlin, Germany: Springer-Verlag, 1995, pp. 171–181.
[36] D. Hofheinz and T. Jager, "Tightly secure signatures and public-key encryption," in *Advances in Cryptology–CRYPTO*. Berlin, Germany: Springer-Verlag, 2012, pp. 590–607.
[37] X. Boyen and B. Waters, "Full-domain subgroup hiding and constant-size group signatures," in *Public Key Cryptography–PKC*. Berlin, Germany: Springer-Verlag, 2007, pp. 1–15.
[38] Y. Shi, H. Fan, and G. Xiong, "Obfuscatable multi-recipient re-encryption for secure privacy-preserving personal health record services," *Technol. Health Care*, vol. 23, no. s1, pp. S139–S145, 2015.

YANG SHI (M'13) received the B.S. degree in electronic engineering from the Hefei University of Technology, China, in 1999, the M.S. degree in pattern recognition and intelligence systems from the Kunming University of Science and Technology, China, in 2002, and the Ph.D. degree in pattern recognition and intelligent systems from Tongji University, Shanghai, China, in 2005.

From 2005 to 2011, he was with Pudong CS&S Co., Ltd., Shanghai, China. Since 2012, he has been an Associate Professor with the School of Software Engineering, Tongji University. His research interests include cryptography, information security, and software protection.

JINGXUAN HAN received the B.S. degree in software engineering from Tongji University, Shanghai, China, in 2014.

Since 2014, he has been a Graduate Student with the School of Software Engineering, Tongji University, Shanghai, China. His current research is mainly focused on cryptography and information security.

HONGFEI FAN was born in Suzhou, China, in 1985. He received the B.Eng. degree in software engineering from Tongji University, China, in 2007, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2013.

Since 2014, he has been an Assistant Professor with the School of Software Engineering, Tongji University. His research interests include information security and software engineering.

QINPEI ZHAO received the B.S. degree in automation technology from Xi'dian University, Xi'an, China, in 2004, the M.S. degree in pattern recognition and intelligent systems in Shanghai Jiaotong University, Shanghai, China, in 2007, and the Ph.D. degree in computer science from the University of Eastern Finland in 2012.

From 2013, she was a Lecturer with the School of Software Engineering, Tongji University, Shanghai. Her research interest includes clustering algorithms and multimedia processing, location-based service and security, and privacy protection.

QIN LIU received the B.S. degree in industrial automation from the Dalian University of Technology and Science, China, in 1998, the M.S. degree in software engineering from Southampton Solent University, U.K., in 2001, and the PhD degree in software engineering from Northumbria University, Newcastle, U.K., in 2006. Since 2007, she has been with the School of Software Engineering, Tongji University, Shanghai, China. She is currently a Professor and the Executive Dean of the School of Software Engineering, Tongji University. Her research interests include software measurement, information security and privacy, data mining, and data analysis.