Computer Security Symposium 2012
30 October – 1 November 2012

サインディクリプション方式の構成法について

満保 雅浩 †

† 金沢大学
920-1192 石川県金沢市角間町

あらまし サインディクリプション方式では，暗号文を復号する際に同意しなくてはいけない条件が暗号文の作成者により設定されており，その条件文への署名を生成することにより，復号が可能となる．著者の知る限り，現在までに，サインディクリプション方式の一般的な構成法は知られていない．本論文では，ID ベース暗号からサインディクリプション方式を構成する方法を示し，この構成に活用可能な ID ベース暗号の性質について議論する．また，この性質を満たす Waters ID ベース暗号を用いた具体的な構成法も示す．

On the Construction of Signdecryption Schemes

Masahiro Mambo†

†Kanazawa University
Kakuma, Kanazawa 920-1192, JAPAN

Abstract Signdecryption is a promising cryptographic primitive which allows an encryptor of a message to impose a decryptor to sign some document of encryptor's selection while performing the decryption. To the author's knowledge, there is no known general construction for the signdecryption scheme so far. In this paper, we present a general construction of signdecryption schemes from identity-based encryption (IBE) schemes and discuss properties required for the underlying IBE schemes. We also show a concrete example of the signdecryption scheme based on the Waters IBE scheme possessing such a property.

1 Introduction

Signdecryption [6] is a promising cryptographic primitive which allows an encryptor of a message to impose a decryptor to sign some document in order to perform the decryption. The encryptor selects the document, information appended to the ciphertext, and in some situation it can be viewed as agreement on the condition of the posession and/or the use of the message corresponding to the ciphertext. In this sense we call the document "condition" and express it by a notation *Cond*. Such a scheme to enforce signature generation in decryption has several applications such as secure software package delivery. To use software package, a user is usually requested to agree upon license agreement. A software package encrypted by a signdecryption scheme guarantees both of user's agreement by enforcing signature generation and access control by encrypting the software package without an interaction between a software vendor and a user.

To the author's knowledge, the notion of signdecryption schemes was introduced in [6]. In addition to the definition of signdecryption scheme, the paper showed a concrete signdecryption scheme called `BasicSigndecrypt`, which combines the Boneh-Franklin identity-based encryption scheme [1] and the Boneh-Lynn-Shacham short signature scheme [2] and also makes use of the approach for the Gentry certificate-based encryption scheme [5], and mentioned the extension to `FullSigndecrypt` using Fujisaki-

Okamoto transformation [4]. However, no general construction of signdecryption schemes is known so far.

In this paper, we present a general construction of signdecryption schemes from identity-based encryption (IBE) schemes and discuss what kind of properties are necessary for the underlying IBE scheme. We also show that `BasicSigndecrypt` does not fully succeed in enforcing signature generation. Based on the observation, we define a relaxed form of signdecryption called semi-signdecryption and examine a method to positively use the semi-signdecryption scheme such as `BasicSign−decrypt`.

2 Preliminaries

2.1 Signdecryption Schemes

We give definition of signdecryption schemes which is slightly modified from the one in [6]. A signdecryption scheme Γ consists of algorithms, Γ = (`Setup`, `KeyGen`, `KeyGenSig`, `Encrypt`, `Signdecrypt`, `Sign`, `Veri`, `Decrypt`):

- $(param, param_{sig}) \leftarrow$ `Setup`(1^k): The setup algorithm `Setup` takes as input a security parameter k and outputs a parameter $param$ for the whole signdecryption scheme and a parameter $param_{sig}$ for decryptor's signature scheme. Both of these parameters may include the security parameter 1^k.
- $(sk_R, pk_R) \leftarrow$ `KeyGen`$(param, param_{sig})$: The receiver's key generation algorithm `KeyGen` takes as input the parameter $param$ and the parameter $param_{sig}$ for decryptor's signature scheme and outputs secret and public keys (sk_R, pk_R), where $sk_R = (sk_{R,enc}, sk_{R,sig})$ and $pk_R = (pk_{R,enc}, pk_{R,sig})$. The receiver's secret and public keys for encryption are denoted by $sk_{R,enc}$ and $pk_{R,enc}$, respectively, and the receiver's signing key and the verification key for signature are denoted by $sk_{R,sig}$ and $pk_{R,sig}$, respectively. The key generation algorithm `KeyGenSig` for the signature scheme is used for generating $sk_{R,sig}$ and $pk_{R,sig}$: $(sk_{R,sig}, pk_{R,sig}) \leftarrow$ `KeyGenSig`$(param_{sig})$.
- $(C, Cond) \leftarrow$ `Encrypt`$(pk_R, param, param_{sig}, m, Cond)$: The encryption algorithm `Encrypt` for the signdecryption scheme takes as input the receiver's public key $pk_R = (pk_{R,enc}, pk_{R,sig})$, the parameter $param$, the parameter $param_{sig}$ for receiver's signature scheme, a message m and a condition $Cond$ and outputs a signdecryption ciphertext C together with the condition $Cond$.
- (m, σ) or $\perp \leftarrow$ `Signdecrypt`$(sk_R, pk_R, param, param_{sig}, C, Cond)$: The signdecryption algorithm `Signdecrypt` takes as input the receiver's secret and public keys $sk_R = (sk_{R,enc}, sk_{R,sig})$ and $pk_R = (pk_{R,enc}, pk_{R,sig})$, the parameter $param$, the parameter $param_{sig}$ for receiver's signature scheme, the ciphertext C and the condition $Cond$ and outputs a pair of a correct message m and a signature σ on $Cond$ or a special symbol $\perp$ by executing the following operations:

  1) $\sigma \leftarrow$ `Sign`$(sk_{R,sig}, param_{sig}, Cond)$: The signing algorithm `Sign` takes as input the receiver's secret key $sk_{R,sig}$, the parameter $param_{sig}$ for receiver's signature scheme and the condition $Cond$ and outputs a signature σ.

  2) `Decrypt`$(sk_{R,enc}, pk_R, param, param_{sig}, \sigma, C, Cond)$

  $$= \begin{cases} m & (\texttt{Veri}(pk_{R,sig}, param_{sig}, \sigma, \\ & Cond) = \texttt{T}), \\ \perp & (\texttt{Otherwise}) : \end{cases}$$

  The decryption algorithm `Decrypt` takes as input the receiver's secret and public keys $sk_{R,enc}$ and pk_R, the parameter $param$, the parameter $param_{sig}$ for receiver's signature scheme, the signature and the condition $Cond$ and outputs a message m from C if the verification algorithm `Veri` taking $pk_{R,sig}, param_{sig}, \sigma$ and $Cond$ as input outputs a symbol `T` indicating a true signature and a special symbol $\perp$, otherwise.

  The output together with the signature σ on $Cond$ are used as an output of `Signde−crypt` without any modification. Note that although `Decrypt` is included as one of algorithms of Γ, `Decrypt` can be viewed as

a part of Signdecrypt and one can omit to explicitly express it as a separate algorithm. The verification algorithm Veri can be executed either implicitly or explicitly. In the implicit verification the IBE decryption algorithm Decrypt is executed without executing Veri while Decrypt is executed after the execution of Veri in the explicit verification.

When the output $(C, cond)$ of $\texttt{Encrypt}(pk_{R,enc}, pk_{R,sig}, param, param_{sig}, m, Cond)$ is fed into Signdecrypt as a part of the input, the algorithm $\texttt{Signdecrypt}(sk_{R,enc}, sk_{R,sig}, param, param_{sig}, C, Cond)$ should output m in the message space. Likewise, $\texttt{Veri}(pk_{R,sig}, param_{sig}, \texttt{Sign}(sk_{R,sig}, param_{sig}, Cond), Cond) = \texttt{T}$ for any $Cond$ in the message space.

2.2 BasicSigndecrypt

BasicSigndecrypt is shown in [6] as an example of signdecryption scheme. The following scheme is a slightly modified version based on the definition in 2.1.

BasicSigndecrypt

$(param, param_{sig}) \leftarrow \texttt{Setup}(1^k)$:

On input 1^k, generate two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Pick an arbitrary generator $g \in \mathbb{G}_1$. Choose cryptographic hash functions $H_1 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some integer n and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$.

Set $param = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, H_1, H_2)$ and $param_{sig} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, H_2)$. The message space and the space for the condition are $\mathcal{M} = \{0,1\}^n$ and $\mathcal{COND} = \{0,1\}^*$, respectively.

$(sk_R, pk_R) \leftarrow \texttt{KeyGen}(param, param_{sig})$:

On input $(param, param_{sig})$, run KeyGenSig on input $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, H_2)$ to generate a public key $v \in \mathbb{G}_1$ by $v \leftarrow g^s$ after randomly selecting a secret key $s \in \mathbb{Z}_q$. The decryptor's key pair for the signature scheme is $(sk_{R,sig}, pk_{R,sig}) = (s, v)$.

The output of KeyGen is $(sk_R, pk_R) = ((\Lambda, s), (v, v))$.

$(C, Cond) \leftarrow \texttt{Encrypt}(pk_R, param, param_{sig}, m, Cond)$:

To encrypt $m \in \mathcal{M}$ under the signdecryption condition $Cond \in \mathcal{COND}$, a sender computes $\texttt{Encrypt}((v, v), param, param_{sig}, m, Cond)$ by executing as follows. Compute $h \leftarrow H_2(Cond) \in \mathbb{G}_1$. Select a random $r \in \mathbb{Z}_q$. Set a ciphertext C as $C = (g^r, m \oplus H_1(\hat{e}(h, v)^r))$.

(m, σ) or $(\tilde{m}, \tilde{\sigma})(\neq (m, \sigma)) \leftarrow \texttt{Signdecrypt}(sk_R, sk_R, param, C, Cond)$:

To decrypt $C = (U, V)$, a decryptor computes $\texttt{Signdecrypt}((\Lambda, s), (v, v), param, C, Cond)$ by executing the following steps.

1. $\sigma \leftarrow \texttt{Sign}(s, param, Cond)$:
   Compute $h \leftarrow H_2(Cond)$. Generates a signature σ by $\sigma \leftarrow h^s \in \mathbb{G}_1$. Output of $\texttt{Sign}(s, param, Cond)$ is $\sigma = h^s$.
2. m or $\tilde{m}(\neq m) \leftarrow \texttt{Decrypt}(\Lambda, (v, v), param, \sigma, C, Cond)$:
   The decryptor computes $\texttt{Decrypt}(\Lambda, (v, v), param, \sigma, C, Cond)$ as follows. $m \leftarrow V \oplus H_1(\hat{e}(\sigma, U))$.

Output of Decrypt together with σ is the output (m, σ) of Signdecrypt.

2.3 Identity-Based Encryption Schemes

An identity-based encryption (IBE) scheme Π consists of algorithms, $\Pi = (\texttt{Setup}_{ibe}, \texttt{Extract}, \texttt{Encrypt}_{ibe}, \texttt{Decrypt}_{ibe})$:

- $(param_{ibe}, msk) \leftarrow \texttt{Setup}_{ibe}(1^k)$: The setup algorithm $\texttt{Setup}_{ibe}$ takes as input a security parameter k and outputs a system parameter $param_{ibe}$ and a master secret key msk. The system parameter $param_{ibe}$ usually contains a master public key mpk corresponding to msk. If we want to clearly describe such relationship, we use the notation $(param_{ibe}^{mpk}, mpk)$ in place of $param_{ibe}$. $param_{ibe}^{mpk}$ is obtained by excluding mpk out of $param_{ibe}$.
- $sk_{ID} \leftarrow \texttt{Extract}(param_{ibe}, msk, ID)$: The key extraction algorithm Extract takes as input the system parameter $param_{ibe}$, the master secret key msk and an arbitrary $ID \in \{0,1\}^*$ and outputs a private decryption key sk_{ID} corresponding to ID.

- $C_{ibe} \leftarrow \texttt{Encrypt}_{ibe}(param_{ibe}, ID, m)$: The encryption algorithm `Encrypt` for the IBE scheme takes as input the IBE parameter $param_{ibe}$ and a message m and outputs an IBE ciphertext C_{ibe}.
- m or $\perp \leftarrow \texttt{Decrypt}_{ibe}(param_{ibe}, C_{ibe}, sk_{ID})$: The decryption algorithm `Decrypt` takes as input the IBE parameter $param_{ibe}$, the IBE ciphertext C_{ibe} and the decryption key sk_{ID} corresponding to ID and outputs m or a special symbol $\perp$. The special symbol $\perp$ is output if the ciphertext C_{ibe} does not pass ciphertext integrity check. Note that a wrong value $\tilde{m} \neq m$ is output instead of $\perp$ if we deal with an IBE scheme not incorporating any ciphertext integrity check.

2.4 Waters IBE and IBS

Waters proposed an identity-based encryption (IBE) scheme and an identity-based signature (IBS) scheme [7], latter of which is converted from the former in a straightforward way. The Waters IBE secure against chosen-ciphertext attack (CCA) can be converted from the following Waters BasicIBE by the Canetti-Halevi-Katz conversion technique [3].

`WatersBasicIBE`

$(param_{ibe}, msk) \leftarrow \texttt{Setup}_{ibe}(1^k)$:

On input 1^k, generate two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order q and an admissible pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Pick randomly an arbitrary generator $g \in \mathbb{G}$ and $g_2 \in \mathbb{G}$, select a random value $\alpha \in \mathbb{Z}_q$ and compute $g_1 = g^\alpha$. Additionally, a random value $u' \in \mathbb{G}$ and a random n-length vector $(u_1, \ldots, u_n) \in \mathbb{G}^n$ Choose a cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ for some integer n.

Set $param_{ibe} = (\mathbb{G},\ \mathbb{G}_T,\ \hat{e},\ g,\ g_1,\ g_2,\ u',\ (u_1, \ldots, u_n),\ H)$. The message space is $\mathcal{M} = \mathbb{G}$. Set $msk = g_2^\alpha \in \mathbb{G}$.

$sk_{ID} \leftarrow \texttt{Extract}(param_{ibe}, msk, ID)$:

To compute private decryption key sk_{ID} for $ID \in \{0,1\}^*$, the key extraction algorithm `Extract` computes $(v_1, \ldots, v_n) \leftarrow H(ID)$ at first and selects a random value $r \in \mathbb{Z}_q$. Then `Extract` outputs $sk_{ID} = (sk_{ID,1}, sk_{ID,2}) = \left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r,\ g^r\right)$.

$C_{ibe} \leftarrow \texttt{Encrypt}_{ibe}(param_{ibe}, ID, m)$:

To encrypt $m \in \mathbb{G}$ under ID the IBE encryption algorithm $\texttt{Encrypt}_{ibe}$ computes $(v_1, \ldots, v_n) \leftarrow H(ID)$ at first and selets a rand value $t \in \mathbb{Z}_q$. Then $\texttt{Encrypt}_{ibe}$ outputs an IBE ciphertext $C_{ibe} = (C_{ibe,1}, C_{ibe,2}, C_{ibe,3}) = \left(\hat{e}(g_1, g_2)^t m, g^t, \left(u' \prod_{i:v_i=1} u_i\right)^t\right)$.

m or $\perp \leftarrow \texttt{Decrypt}_{ibe}(param_{ibe}, C_{ibe}, sk_{ID})$:

To decrypt $C_{ibe} = (C_{ibe,1}, C_{ibe,2}, C_{ibe,3})$ a decryptor with a pair of secret decryption keys $sk_{ID} = (sk_{ID,1}, sk_{ID,2})$ computes

$$C_{ibe,1} \frac{\hat{e}\left(sk_{ID,2}, C_{ibe,3}\right)}{\hat{e}\left(sk_{ID,1}, C_{ibe,1}\right)} = m.$$

`WatersIBS`

$((param_{sig}, pk_{sig}), sk_{sig}) \leftarrow \texttt{Setup}_{sig}(1^k)$:

$\texttt{Setup}_{sig}$ is the same as $\texttt{Setup}_{ibe}$ and the public parameter and public key $(param_{sig}, pk_{R,sig})$ is $param_{ibe} = ((\mathbb{G},\ \mathbb{G}_T,\ \hat{e},\ g,\ g_2,\ u', (u_1, \ldots, u_n),\ H), g_1)$. The message space is $\mathcal{M} = \{0,1\}^*$. A secret signing key sk_{sig} of signer is $sk_{sig} = msk = g_2^\alpha \in \mathbb{G}$.

$\sigma \leftarrow \texttt{Signing}(param_{sig}, pk_{sig}, sk_{sig}, m)$:

To sign m a signer with a secret key sk_{sig} computes $(m_1, \ldots, m_n) \leftarrow H(m)$ at first and selects a random value $r \in \mathbb{Z}_q$. Then `Signing` outputs a signature $\sigma = (\sigma_1,\ \sigma_2) = \left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r, g^r\right)$.

`T` or `F` $\leftarrow \texttt{Veri}(param_{sig}, pk_{sig}, \sigma, m)$:

To verify a signature $\sigma = (\sigma_1, \sigma_2)$ on message m a verifier computes $(m_1, \ldots, m_n) \leftarrow H(m)$ at first and then checks

$$\frac{\hat{e}\left(\sigma_1, g\right)}{\hat{e}\left(\sigma_2, u' \prod_{i:m_i=1} u_i\right)} \stackrel{?}{=} \hat{e}(g_1, g_2).$$

3 Semi-signdecryption schemes

In the definition of signdecryption given in [6] a signature is not demanded as a part of the output of `Signdecrypt`. In this context the original `BasicSigndectypt` in [6] is not required to output a signature. Also its signature verfication is done implicitly. Note that a slightly modified version in Sect. 2.2 needs to output (m, σ) by following the definition in Sect. 2.1. The original `BasicSigndectypt` in

[6] has a problem such that a legitimate decryptor can decrypt ciphertexts without generating a signature σ on $Cond$ as follows.

> Given a ciphertext (U, V), a legitimate decryptor with knowledge s computes $h \leftarrow H_2(Cond)$ at first. Then the decryptor computes $m \leftarrow V \oplus H_1(\hat{e}(h, U^s))$.

Since $\hat{e}(\sigma, U) = \hat{e}(h^s, U) = \hat{e}(h, U^s)$, a correct output m of `BasicSigndectypt` in [6] can be derived by the computation described above.

Based on the observation about original `BasicSigndectypt`, we can define semi-signdecryption schemes as a relaxed form of signdecryption schemes. A semi-signdecryption scheme $\Gamma^{(semi)}$ consists of algorithms $\Gamma^{(semi)} =$ (`Setup`, `KeyGen`, `KeyGenSig`, `Encrypt`, `Signdecrypt`, `Sign`, `Veri`, `Decrypt`) as defined for the signdecryption scheme in Sect. 2.1 except that the output of `Signdecrypt` does not contain a signature. Furthermore, a legitimate decryptor of $\Gamma^{(semi)}$ can correctly decrypt ciphertexts not only by generating a signature on $Cond$ but also by other method. In contrast, a legitimate decryptor of a signdecryption scheme Γ can correctly decrypt ciphertext only by generating a signature on $Cond$.

As shown in the original `BasicSigndectypt`, semi-signdecryption schemes with implicit signature verification do not guarantee enforcement of signature generation on $Cond$. Even though semi-signdecryption schemes with implicit signature verification have such a flaw, they can be positively utilized with assistance of additional mechanisms. For example, in the original `BasicSigndectypt`, suppose we can add some hardware or software mechanism to hide a part U of the ciphertext (U, V) from a legitimate decryptor. Then the decryptor cannot do the computation described above to circumvent the signature generation.

4 Signdecryption from IBE

We show that signdecryption schemes can be constructed from IBE schemes. Suppose an IBE scheme $\Pi =$ ($\texttt{Setup}_{ibe}$, `Extract`, $\texttt{Encrypt}_{ibe}$, $\texttt{Decrypt}_{ibe}$) described in Sect. 2.3 is given. Then a signdecryption scheme $\Gamma =$ (`Setup`, `KeyGen`, `KeyGenSig`, `Encrypt`, `Signdecrypt`, `Sign`, `Veri`, `Decrypt`) can be constructed as follows:

- $(param_{ibe}^{mpk}, param_{ibe}^{mpk}) \leftarrow \texttt{Setup}(1^k)$: The setup algorithm `Setup` takes as input a security parameter k of the IBE scheme, executes $(param_{ibe}, msk) \leftarrow \texttt{Setup}_{ibe}(1^k)$ and outputs a parameter $param_{ibe}^{mpk}$ for the whole signdecryption scheme and a parameter $param_{ibe}^{mpk}$ for decryptor's signature scheme.

- $((\Lambda, msk), (mpk, mpk)) \leftarrow \texttt{KeyGen}(param_{ibe}^{mpk}, param_{ibe}^{mpk})$: The receiver's key generation algorithm `KeyGen` takes as input the parameters $(param_{ibe}^{mpk}, param_{ibe}^{mpk})$ and outputs secret and public keys $(sk_R, pk_R) = ((\Lambda, msk), (mpk, mpk))$ where Λ is an empty string.

  The key generation algorithm `KeyGenSig` for the signature scheme takes as input the security parameter $param_{ibe}^{mpk}$ of the IBE scheme and outputs a pair of signing and verification keys $(sk_{R,sig}, pk_{R,sig}) = (msk, mpk)$ which are computed during the execution of $\texttt{Setup}_{ibe}(1^k)$ in the setup algorithm $\texttt{Setup}(1^k)$: $(msk, mpk) \leftarrow \texttt{KeyGenSig}(param_{ibe}^{mpk})$.

  `KeyGen` also uses the output of $\texttt{Setup}_{ibe}(1^k)$ for determining a pair $(sk_{R,enc}, pk_{R,enc})$ of keys for encryption.

- $(C_{ibe}, Cond) \leftarrow \texttt{Encrypt}((mpk, mpk), param_{ibe}^{mpk}, param_{ibe}^{mpk}, m, Cond)$: The encryption algorithm `Encrypt` for the signdecryption scheme takes as input the receiver's public key mpk, the parameter $param_{ibe}^{mpk}$ for the whole scheme, the parameter $param_{ibe}^{mpk}$ for receiver's signature scheme, a message m and a condition $Cond$, executes the IBE encryption $C_{ibe} \leftarrow \texttt{Encrypt}_{ibe}(param_{ibe}, Cond, m)$ by treating $Cond$ as an ID and outputs C_{ibe} as a signdecryption ciphertext together with the condition $Cond$.

- (m, sk_{Cond}) or $\perp \leftarrow \texttt{Signdecrypt}((\Lambda, msk), (mpk, mpk), param_{ibe}^{mpk}, param_{ibe}^{mpk}, C_{ibe}, Cond)$: The signdecryption algorithm `Signdecrypt` takes as input the receiver's

secret key msk, the parameter $param_{ibe}^{mpk}$ for the whole scheme, the parameter $param_{ibe}^{mpk}$ for receiver's signature scheme, the ciphertext C_{ibe} and the condition $Cond$ and outputs a correct message m or a special symbol $\perp$ by executing the following operations:

1) $sk_{Cond} \leftarrow \texttt{Sign}(msk, param_{ibe}^{mpk}, Cond)$: The signing algorithm `Sign` takes as input the receiver's secret key msk, the parameter $param_{ibe}^{mpk}$ for receiver's signature scheme and the condition $Cond$, executes the extraction of the IBE decryption key $sk_{Cond} \leftarrow \texttt{Extract}(param_{ibe}, msk, Cond)$ by treating $Cond$ as an ID and outputs sk_{Cond} as a signature σ on a message $Cond$.

2) $\texttt{Decrypt}(\Lambda, (mpk, mpk), param_{ibe}^{mpk}, param_{ibe}^{mpk}, sk_{Cond}, C_{ibe}, Cond)$

$$= \begin{cases} m & \left(\texttt{Veri}(mpk, param_{ibe}^{mpk}, sk_{Cond}, Cond) = \texttt{T}\right), \\ \perp & (\texttt{Otherwise}). \end{cases}$$

The decryption algorithm `Decrypt` takes as input the receiver's public key mpk, the parameter $param_{ibe}^{mpk}$, the parameter $param_{ibe}^{mpk}$ for receiver's signature scheme, the signature sk_{Cond} and the condition $Cond$, executes the IBE decryption m or $\perp \leftarrow \texttt{Decrypt}_{ibe}(param_{ibe}, C_{ibe}, sk_{Cond})$ and outputs a message m from C_{ibe} if the verification algorithm `Veri` taking mpk, $param_{ibe}^{mpk}$, sk_{Cond}, $Cond$ as input outputs a symbol `T` indicating a true signature and a special symbol $\perp$, otherwise. The output together with sk_{Cond} is used as output of `Signdecrypt` without any modification.

In the implicit verification the IBE decryption algorithm $\texttt{Decrypt}_{ibe}$ is executed without executing `Veri`.

Meanwhile, in the explicit verification, the verification algorithm `Veri` executes the probabilistic verification of the Naor-trans formation. That is, after a message m_{veri} is selected uniformly at random from the message space, it is encrypted by the IBE scheme as $C_{ibe,veri} \leftarrow \texttt{Encrypt}_{ibe}(param_{ibe}, Cond, m_{veri})$. Then the IBE decryption m'_{veri} or $\perp \leftarrow \texttt{Decrypt}_{ibe}(param_{ibe}, C_{ibe,veri}, sk_{Cond})$ is executed. `Veri` returns `T` if $m'_{veri} = m_{veri}$ and `F`, otherwise. Note that in these operations the IBE decryption is possible because $param_{ibe}$, which is equal to $(param_{ibe}^{mpk}, mpk)$, can be obtained from the input of `Veri` and also from the input of `Decrypt`.

5 Enforcing signature generation

As discussed in Sect. 3 not all IBE schemes can be used for the construction described in Sect. 4. If a decryptor of the IBE scheme, who is given the master secret key msk, succeeds in the IBE decryption without generating a signature on $Cond$, one can obtain only a semi-signdecryption scheme. Meanwhile, if the decryptor under the same condition succeeds in the IBE decryption only by generating a signature on $Cond$, the constructed scheme is a signdecryption scheme. An example of the latter type of IBE schemes is the Waters IBE scheme described in Sect. 2.4. Since the master secret key in his scheme is not α but g_2^α, it becomes difficult to decrypt ciphertexts without generating a signature on $Cond$.

As an example of signdecryption scheme based on such an IBE scheme, we show a signdecryption scheme based on `WatersBasicIBE` scheme. An important point of this construction is that a trusted center generates a secret random value α and gives a user only g_2^α as a secret signing key, which is the master secret key of the `WatersBasicIBE` scheme.

As in [6] we omit to describe security analysis including security model, both of which will be given in the full version.

`WatersBasicIBE&IBStoSigndecryption`

$(param_{ibe}^{mpk}, param_{ibe}^{mpk}) \leftarrow \texttt{Setup}(1^k)$:

On input 1^k, run $\texttt{Setup}_{ibe}$ of the WatersBasicIBE and set $param{=}param_{sig}{=}param_{ibe}^{mpk} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, g_2, u', (u_1, \ldots, u_n), H)$. The message space and the space for the condition are $\mathcal{M} = \mathbb{G}$ and $\mathcal{CON}\mathcal{D} = \{0,1\}^*$, respectively.

$(sk_R, pk_R) \leftarrow \texttt{KeyGen}(param_{ibe}^{mpk}, param_{ibe}^{mpk})$:

On input $(param_{ibe}^{mpk}, param_{ibe}^{mpk})$, the trusted

center runs `KeyGenSig` on input ($\mathbb{G}$, $\mathbb{G}_T$, $\hat{e}$, g, g_2, u', $(u_1, \ldots, u_n)$, H) to generate a public key $pk_{R,enc} = pk_{R,sig} = mpk = g_1 \in \mathbb{G}$ by $g_1 \leftarrow g^\alpha$ after randomly selecting a secret key $\alpha \in \mathbb{Z}_q$ of the trusted center and to generate a secret key $sk_{R,sig} = msk = g_2^\alpha \in \mathbb{G}$ of a receiver. The receiver's key pair for the signature scheme is $(sk_{R,sig}, pk_{R,sig}) = (g_2^\alpha, g_1)$.

The output of `KeyGen` is $(sk_R, pk_R) = ((\Lambda, g_2^\alpha), (g_1, g_1))$.

$(C, Cond) \leftarrow$ `Encrypt`$(pk_R, param_{ibe}^{mpk}, param_{ibe}^{mpk}, m, Cond)$:

To encrypt $m \in \mathcal{M}$ under the signdecryption condition $Cond \in \mathcal{CONF}$, a sender computes `Encrypt`$((g_1, g_1), param_{ibe}^{mpk}, param_{ibe}^{mpk}, m, Cond)$ by executing as follows. Compute $(v_1, \ldots, v_n) \leftarrow H(Cond) \in \mathbb{G}$. Select a random $t \in \mathbb{Z}_q$. Set a ciphertext C as $C = (C_1, C_2, C_3) = \left(\hat{e}(g_1, g_2)^t m, g^t, \left(u' \prod_{i:v_i=1} u_i\right)^t\right)$.

(m, sk_{Cond}) or $\perp \leftarrow$ `Signdecrypt`$(sk_R, pk_R, param_{ibe}^{mpk}, param_{ibe}^{mpk}, C, Cond)$:

To decrypt $C = (C_1, C_2, C_3)$, a decryptor computes `Signdecrypt`$((\Lambda, g_2^\alpha), (g_1, g_1), param_{ibe}^{mpk}, param_{ibe}^{mpk}, C, Cond)$ by executing the following steps.

1. $sk_{Cond} \leftarrow$ `Sign`$(g_2^\alpha, param_{ibe}^{mpk}, Cond)$:
   Compute $(v_1, \ldots, v_n) \leftarrow H(Cond)$. Generates a signature sk_{Cond} by $sk_{Cond} \leftarrow$ `Extract`$(param_{ibe}^{mpk}, g_2^\alpha, Cond)$, where $sk_{Cond} = (sk_{Cond,1}, sk_{Cond,2}) = \left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r, g^r\right) \in \mathbb{G}^2$. Output of `Sign`$(g_2^\alpha, param_{ibe}^{mpk}, Cond)$ is $sk_{Cond} = \left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r, g^r\right)$.

2. m or $\perp \leftarrow$ `Decrypt`$(\Lambda, (g_1, g_1), param_{ibe}^{mpk}, sk_{Cond}, C, Cond)$:
   The decryptor computes `Decrypt`$(\Lambda, (g_1, g_1), param_{ibe}^{mpk}, sk_{Cond}, C, Cond)$ by
   $$C_1 \frac{\hat{e}\left(sk_{Cond,2}, C_3\right)}{\hat{e}\left(sk_{Cond,1}, C_1\right)} = m.$$
   If `Veri`$(pk_{R,sig}, param_{ibe}^{mpk}, sk_{Cond}, Cond)$ is explicitly executed, it is suffice to execute the verification of the Waters IBS: First, compute $(v_1, \ldots, v_n) \leftarrow H(Cond)$. Then check
   $$\frac{\hat{e}\left(sk_{Cond,1}, g\right)}{\hat{e}\left(sk_{Cond,2}, u' \prod_{i:v_i=1} u_i\right)} \stackrel{?}{=} \hat{e}(g_1, g_2).$$
   Output of `Decrypt` together with $\sigma = sk_{Cond}$ is the output (m, sk_{Cond}) of `Signdecrypt`.

The legitimate decryptor posessing $sk_{R,sig} = g_2^\alpha$ may try to avoid to generate a signature $\left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r, g^r\right) \in \mathbb{G}^2$ Since the decryption formula can be alternatively expressed as
$$C_1 \frac{\hat{e}\left(g, C_3\right)^r}{\hat{e}\left(g_2^\alpha, C_1\right) \hat{e}\left(\left(u' \prod_{i:v_i=1} u_i\right), C_1\right)^r} = m,$$
the decryptor may use each component of tuple $(g_2^r, \left(u' \prod_{i:v_i=1} u_i\right), g, r)$ for the decryption. Apparently, one can compute a correct signature $\left(g_2^\alpha \left(u' \prod_{i:v_i=1} u_i\right)^r, g^r\right)$ from the tuple. As long as a correct signature can be derived from the input tuple of decryption, we do not consider the decryption method as an alternative decryption method not requiring signature generation.

We should further note that the signdecryption scheme based on Waters IBE, `Waters-BasicIBE&IBStoSigndecryption`, has a key generation mechanism different from both public-key cryptography and identity based cryptography. Since the trusted center knows the secret key g_2^α of a user, `WatersBasicIBE&IBSto-Signdecryption` has a key escrow problem by the trusted center as exisiting in the identity based cryptography. However, unlike the identity based cryptography, the user's public key $g_1 = g^\alpha$ cannot be set as a fixed value like ID. In this sense, `BasicIBE&IBStoSigndecryption` seems to have negative properties of both cryptographies. So far, it seemed to be considered that this type of key generation mechanism has no merit and hasn't been studied. We can consider `BasicIBE&IBStoSigndecryption` as an example showing a merit of such a key generation mechanism.

6 Conclusion

We have studied signdecryption schemes. After giving a refinement of their definition we have shown that in `BasicSigndecrypt` [6], a legitimate decryptor is able to avoid to generate a signature on the condition specified by the encryptor. Based on the observation we

have introduced a notion semi-signdecryption, which is a relaxed form of signdecryption and can be used together with some additional protecting mechanism. Then we have proposed a general construction of signdecryption schemes from the IBE schemes. As an example of such signdecryptions schemes we have shown a signdecryption scheme constructed from Waters-BascisIBE and WatersIBS. Its security analysis will be given in the full version.

References

[1] Dan Boneh and Matthew Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. of Computing*, Vol.32, No.3, pp.586-615, 2003.

[2] Dan Boneh, Ben Lynn and Hovav Shacham, "Short Signatures from the Weil Pairing," *Journal of Cryptology*, Vol.17, pp.297-319, 2004.

[3] Ran Canetti, Shai Halevi and Jonathan Katz, "Chosen-Ciphertext Security from Identity-Based Encryption," C. Cachin and J. Camenisch (Eds.), LNCS 3027, *Advances in Cryptology -Eurocrypt 2004*, Springer-Verlag, pp.207-222, 2004.

[4] Eiichiro Fujisaki and Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes," M. Wiener (Ed.), LNCS 1666, *Advances in Cryptology -Crypto '99*, Springer-Verlag, pp.537-554, 1999.

[5] Craig Gentry, "Certificate-Based Encryption and the Certificate Revocation Problem," E. Biham (Ed.), LNCS 2656, *Advances in Cryptology -Eurocrypt 2003*, Springer-Verlag, pp.272-293, 2003.

[6] Masahiro Mambo, "On the Extension of Cryptographic Schemes with Conditions ∼Signdecryption∼," *Abstracts of the 29th Symposium on Cryptography and Information Security (SCIS 2012)*, 3A3-3, February, 2012.

[7] Brent Waters, "Efficient Identity-Based Encryption Without Random Oracles," R. Cramer (Ed.), LNCS 3494, *Advances in Cryptology -Eurocrypt 2005*, Springer-Verlag, pp.114-127, 2005.